



SQL リファレンス 第 2 巻
最終更新: 2009 年 4 月



SQL リファレンス 第 2 巻
最終更新: 2009 年 4 月

ご注意

本書および本書で紹介する製品をご使用になる前に、1033 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、www.ibm.com/shop/publications/order にある IBM Publications Center をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、www.ibm.com/planetwide にある IBM Directory of Worldwide Contacts をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC23-5862-02
DB2 Version 9.5
for Linux, UNIX, and Windows
SQL Reference, Volume 2
Updated April, 2009

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2009.3

© Copyright International Business Machines Corporation 1993, 2009.

目次

本書について	vii	コンバウンド SQL (動的)	221
本書の対象読者	vii	コンバウンド SQL (組み込み)	225
本書の構成	vii	コンバウンド SQL (プロシージャ)	229
構文図の見方	vii	CONNECT (タイプ 1)	238
本書の表記規則	ix	CONNECT (タイプ 2)	246
エラー条件	ix	CREATE ALIAS	254
強調表記規則	x	CREATE AUDIT POLICY	256
関連資料	x	CREATE BUFFERPOOL	259
ステートメント	1	CREATE DATABASE PARTITION GROUP	263
SQL ステートメント	1	CREATE EVENT MONITOR	265
SQL ステートメントの呼び出し方法	9	CREATE FUNCTION	287
SQL 制御ステートメントについて	14	CREATE FUNCTION (外部スカラー)	288
関数、メソッド、およびプロシージャの指定子	16	CREATE FUNCTION (外部表)	315
ALLOCATE CURSOR	21	CREATE FUNCTION (OLE DB 外部表)	335
ALTER AUDIT POLICY	22	CREATE FUNCTION (ソース派生またはテンプレ ト)	344
ALTER BUFFERPOOL	25	CREATE FUNCTION (SQL スカラー、表、または 行)	356
ALTER DATABASE PARTITION GROUP	28	CREATE FUNCTION MAPPING	367
ALTER DATABASE	32	CREATE HISTOGRAM TEMPLATE	371
ALTER FUNCTION	35	CREATE INDEX	373
ALTER HISTOGRAM TEMPLATE	38	CREATE INDEX EXTENSION	390
ALTER METHOD	39	CREATE METHOD	397
ALTER NICKNAME	40	CREATE NICKNAME	403
ALTER PROCEDURE (外部)	48	CREATE PROCEDURE	416
ALTER PROCEDURE (ソース派生)	51	CREATE PROCEDURE (外部)	416
ALTER PROCEDURE (SQL)	52	CREATE PROCEDURE (ソース派生)	432
ALTER SECURITY LABEL COMPONENT	53	CREATE PROCEDURE (SQL)	436
ALTER SECURITY POLICY	56	CREATE ROLE	443
ALTER SEQUENCE	60	CREATE SCHEMA	443
ALTER SERVER	64	CREATE SECURITY LABEL COMPONENT	446
ALTER SERVICE CLASS	67	CREATE SECURITY LABEL	449
ALTER TABLE	74	CREATE SECURITY POLICY	450
ALTER TABLESPACE	123	CREATE SEQUENCE	452
ALTER THRESHOLD	133	CREATE SERVICE CLASS	457
ALTER TRUSTED CONTEXT	140	CREATE SERVER	465
ALTER TYPE (構造化)	149	CREATE TABLE	469
ALTER USER MAPPING	156	CREATE TABLESPACE	542
ALTER VIEW	158	CREATE THRESHOLD	557
ALTER WORK ACTION SET	160	CREATE TRANSFORM	566
ALTER WORK CLASS SET	173	CREATE TRIGGER	570
ALTER WORKLOAD	178	CREATE TRUSTED CONTEXT	583
ALTER WRAPPER	185	CREATE TYPE (配列)	590
ALTER XSROBJECT	187	CREATE TYPE (特殊)	592
ASSOCIATE LOCATORS	188	CREATE TYPE (構造化)	598
AUDIT	190	CREATE TYPE MAPPING	624
BEGIN DECLARE SECTION	194	CREATE USER MAPPING	630
CALL	195	CREATE VARIABLE	632
CASE	201	CREATE VIEW	635
CLOSE	203	CREATE WORK ACTION SET	650
COMMENT	205	CREATE WORK CLASS SET	658
COMMIT	218		

CREATE WORKLOAD	663	REVOKE (データベース権限)	867
CREATE WRAPPER	672	REVOKE (免除)	871
DECLARE CURSOR	673	REVOKE (グローバル変数特権)	873
DECLARE GLOBAL TEMPORARY TABLE	680	REVOKE (索引特権)	875
DELETE	688	REVOKE (パッケージ特権)	877
DESCRIBE	695	REVOKE (ロール)	880
DESCRIBE INPUT	695	REVOKE (ルーチン特権)	882
DESCRIBE OUTPUT	698	REVOKE (スキーマ特権)	886
DISCONNECT	703	REVOKE (セキュリティ・ラベル)	888
DROP	705	REVOKE (シーケンス特権)	890
END DECLARE SECTION	740	REVOKE (サーバー特権)	892
EXECUTE IMMEDIATE	741	REVOKE (SETSESSIONUSER 特権)	894
EXECUTE	743	REVOKE (表スペース特権)	895
EXPLAIN	750	REVOKE (表、ビュー、またはニックネーム特権)	897
FETCH	755	REVOKE (ワークロード特権)	903
FLUSH EVENT MONITOR	758	REVOKE (XSR オブジェクト特権)	904
FLUSH OPTIMIZATION PROFILE CACHE	759	ROLLBACK	905
FLUSH PACKAGE CACHE	761	SAVEPOINT	908
FOR	761	SELECT INTO	911
FREE LOCATOR	764	SELECT	913
GET DIAGNOSTICS	765	SET COMPILATION ENVIRONMENT	913
GOTO	767	SET CONNECTION	914
GRANT (データベース権限)	768	SET CURRENT DECFLOAT ROUNDING MODE	915
GRANT (免除)	773	SET CURRENT DEFAULT TRANSFORM GROUP	917
GRANT (グローバル変数特権)	775	SET CURRENT DEGREE	918
GRANT (索引特権)	777	SET CURRENT EXPLAIN MODE	920
GRANT (パッケージ特権)	779	SET CURRENT EXPLAIN SNAPSHOT	923
GRANT (ロール)	782	SET CURRENT FEDERATED ASYNCHRONY	925
GRANT (ルーチン特権)	785	SET CURRENT IMPLICIT XMLPARSE OPTION	926
GRANT (スキーマ特権)	789	SET CURRENT ISOLATION	927
GRANT (セキュリティ・ラベル)	791	SET CURRENT LOCK TIMEOUT	928
GRANT (シーケンス特権)	794	SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	930
GRANT (サーバー特権)	796	SET CURRENT MDC ROLLOUT MODE	932
GRANT (SETSESSIONUSER 特権)	798	SET CURRENT OPTIMIZATION PROFILE	933
GRANT (表スペース特権)	800	SET CURRENT PACKAGE PATH	937
GRANT (表、ビュー、またはニックネーム特権)	802	SET CURRENT PACKAGESET	940
GRANT (ワークロード特権)	809	SET CURRENT QUERY OPTIMIZATION	942
GRANT (XSR オブジェクト特権)	811	SET CURRENT REFRESH AGE	944
IF	812	SET ENCRYPTION PASSWORD	946
INCLUDE	814	SET EVENT MONITOR STATE	947
INSERT	815	SET INTEGRITY	949
ITERATE	824	SET PASSTHRU	968
LEAVE	826	SET PATH	970
LOCK TABLE	827	SET ROLE	972
LOOP	828	SET SCHEMA	973
MERGE	830	SET SERVER OPTION	975
OPEN	839	SET SESSION AUTHORIZATION	977
PREPARE	844	SET 変数	979
REFRESH TABLE	853	SIGNAL	986
RELEASE (接続)	856	TRANSFER OWNERSHIP	989
RELEASE SAVEPOINT	858	UPDATE	1004
RENAME TABLESPACE	858	VALUES INTO	1015
RENAME	860	VALUES	1017
REPEAT	862	WHENEVER	1017
RESIGNAL	863	WHILE	1018
RETURN	865		

付録 A. DB2 技術情報の概説	1021
DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)	1022
DB2 の印刷資料の注文方法	1025
コマンド行プロセッサから SQL 状態ヘルプを表示する	1026
異なるバージョンの DB2 インフォメーション・センターへのアクセス	1026
DB2 インフォメーション・センターでの希望する言語でのトピックの表示	1026

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新	1027
DB2 チュートリアル	1029
DB2 トラブルシューティング情報	1030
ご利用条件	1030

付録 B. 特記事項	1033
-----------------------------	-------------

索引	1037
---------------------	-------------

本書について

SQL リファレンス (第 1 巻、第 2 巻) では、DB2[®] Database for Linux[®]、UNIX[®]、および Windows[®] によって使用される SQL 言語が定義されています。これには、次のものが含まれます。

- リレーショナル・データベースの概念、言語エレメント、関数、および照会の形式に関する情報 (第 1 巻)
- SQL ステートメントの構文およびセマンティクスに関する情報 (第 2 巻)

本書の対象読者

本書は構造化照会言語 (SQL) を使ってデータベースにアクセスするすべてのユーザーを対象としています。本書は主にプログラマーおよびデータベース管理者を対象としていますが、コマンド行プロセッサ (CLP) を通してデータベースにアクセスする方も利用することができます。

本書はチュートリアルではなく、解説書です。本書では、読者がアプリケーション・プログラムを作成することを想定しており、このためデータベース・マネージャーのすべての機能を説明しています。

本書の構成

SQL リファレンス 第 2 巻には、SQL ステートメントの構文およびセマンティクスに関する情報が含まれています。

- 『ステートメント』には、SQL プロシージャ・ステートメントを含むすべての SQL ステートメントの構文図、セマンティクスの説明、規則、および例があります。

構文図の見方

構文図の構造を理解するために、以下の情報を参考にできます。

構文図は、左から右、上から下に、線に沿って読みます。

記号  は、構文図の始まりを示します。

記号  は、構文が次の行に続くことを示します。

記号  は、構文が前の行から続いていることを示します。

記号  は、構文図の終わりを示します。

構文フラグメントは、記号  で始まり、記号  で終わります。

必須項目は、横線 (メインパス) 上に示されます。

構文図の見方



オプション項目は、メインパスの下に示されます。

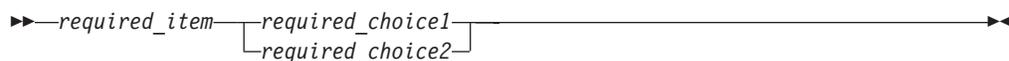


オプション項目をメインパスの上に示すこともありますが、それは構文図を見やすくするためであり、実行には関係しません。

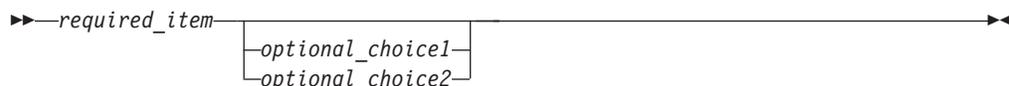


複数の項目からの選択が可能な場合、それらの項目を縦に並べて (スタックに) 示しています。

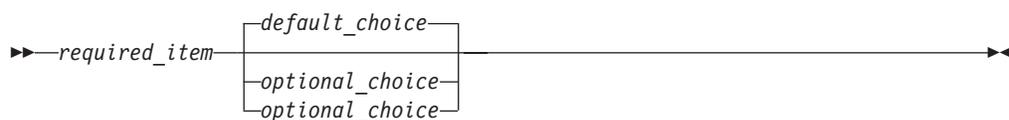
項目から 1 つを選択しなければならない場合、スタックの項目の 1 つはメインパス上に示されます。



項目から 1 つをオプションで選択できる場合、スタック全体がメインパスよりも下に示されます。



項目の 1 つがデフォルト値の場合、その項目はメインパスより上に示され、残りの選択項目はメインパスよりも下に示されます。



メインパスの上に、左へ戻る矢印がある場合には、項目を繰り返して指定できることを示しています。このような場合、繰り返す項目相互の間は、1 つ以上の空白で区切らなければなりません。



繰り返しの矢印にコンマが示されている場合は、繰り返し項目をコンマで区切らなければなりません。



スタックの上部の反復の矢印の記号は、そのスタックの中から複数の項目を選択できること、または 1 つの選択項目を繰り返して選択できることを示します。

キーワードは英大文字で示してあります (例: FROM)。示されているとおりに入力することが必要です。変数は英小文字で示しています (例: column-name)。このような変数は、構文にユーザーが指定する名前や値を示しています。

句読点、括弧、算術演算子、その他の記号が示されている場合には、それらを構文の一部として入力する必要があります。

1 つの変数が、構文を構成する大きいフラグメントを表すことがあります。たとえば次の図で、変数 `parameter-block` は、**parameter-block** というラベルの構文フラグメント全体を表します。



parameter-block:



「黒丸」 (●) ではさまれて隣接しているセグメントは、任意の順序で指定することができます。



上記の図は、`item2` と `item3` をどのような順序で指定しても構わないことを示しています。以下はいずれも有効です。

```
required_item item1 item2 item3 item4
required_item item1 item3 item2 item4
```

本書の表記規則

エラー条件

マニュアルの文章内では、エラーに関連する `SQLSTATE` を括弧に入れて表示することによって、エラー条件を示しています。以下に例を示します。

シグニチャーが重複していると、SQL エラー (`SQLSTATE 42723`) を戻します。

強調表記規則

本書では、以下の表記規則を使用しています。

太字	コマンド、キーワード、および名前がシステムによって事前定義されている他の項目を表します。
イタリック	以下のいずれかを示します。 <ul style="list-style-type: none"> • ユーザーが指定する必要がある名前または値 (変数) • 一般的な強調 • 新しい用語の紹介 • 他の情報源の参照
モノスペース	以下のいずれかを示します。 <ul style="list-style-type: none"> • ファイルおよびディレクトリー • コマンド・プロンプトまたはウィンドウでユーザーが入力する必要がある情報 • 特定のデータ値の例 • システムで表示されるものと類似したテキストの例 • システム・メッセージの例

関連資料

以下の資料は、アプリケーションを準備する際に役立つ可能性があります。

- データベース・アプリケーション開発の基礎
 - DB2 アプリケーション開発の概要を示します。これにはプラットフォーム前提条件、サポートされる開発ソフトウェア、およびサポートされているプログラミング API の利点と制約事項についてのガイダンスが含まれます。
- *DB2 for i5/OS SQL* リファレンス
 - この資料では、DB2 Query Manager and SQL Development Kit on System i[®] によってサポートされる SQL が定義されています。この資料にはシステム管理のタスク、データベース管理、アプリケーション・プログラミング、および操作のタスクに関する参照情報が含まれています。このマニュアルには、構文、使用上の注意、キーワード、および DB2 を実行する i5/OS[®] システム上で使用される各 SQL ステートメントの例が含まれます。
- *DB2 for z/OS SQL* リファレンス
 - この資料では、DB2 for z/OS[®] で使用される SQL を定義しています。この資料では、DB2 を実行する z/OS システムでの照会書式、SQL ステートメント、SQL プロシージャ・ステートメント、DB2 の制約事項、SQLCA、SQLDA、カタログ表、および SQL 予約語について説明しています。
- *DB2 Spatial Extender* ユーザーズ・ガイドおよびリファレンス
 - この資料では、地理情報システム (GIS) を作成および使用するアプリケーションの作成方法を説明しています。GIS の作成および使用には、データベースにリソースを提供すること、またデータの照会を行って位置、距離、および領域内の分布などの情報を取得することが含まれます。
- *IBM SQL* リファレンス

- この資料には、IBM のデータベース製品に関係したすべての共通 SQL エレメントを収録しています。この資料では、IBM® データベースを使用する移植可能プログラムを準備する際に参照できる、制約事項や規則について説明しています。このマニュアルでは、SQL 拡張機能、および各種の規格と製品 (SQL92E、XPG4-SQL、IBM-SQL、および IBM リレーショナル・データベース製品) 間における非互換性のリストを示しています。
- *American National Standard X3.135-1992, Database Language SQL*
 - SQL の ANSI 規格定義があります。
- *ISO/IEC 9075:1992, Database Language SQL*
 - SQL の 1992 ISO 標準定義があります
- *ISO/IEC 9075-2:2003, Information technology -- Database Languages -- SQL -- Part 2: Foundation (SQL/Foundation)*
 - SQL の 2003 ISO 標準定義の大部分がここにあります。
- *ISO/IEC 9075-4:2003, Information technology -- Database Languages -- SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
 - SQL プロシージャ制御ステートメントの 2003 ISO 標準定義があります。

ステートメント

SQL ステートメント

以下の表は、SQL ステートメントをタイプ別に分類して一覧で示しています。

- SQL スキーマ・ステートメント (表 1)
- SQL データ変更ステートメント (5 ページの表 2)
- SQL データ・ステートメント (5 ページの表 3)
- SQL トランザクション・ステートメント (6 ページの表 4)
- SQL 接続ステートメント (6 ページの表 5)
- SQL 動的ステートメント (6 ページの表 6)
- SQL セッション・ステートメント (7 ページの表 7)
- SQL 組み込みホスト言語ステートメント (8 ページの表 8)
- SQL 制御ステートメント (8 ページの表 9)

表 1. SQL スキーマ・ステートメント

SQL ステートメント	目的
22 ページの『ALTER AUDIT POLICY』	現行サーバーの監査ポリシーの定義を変更します。
25 ページの『ALTER BUFFERPOOL』	バッファ・プールの定義を変更します。
32 ページの『ALTER DATABASE』	自動ストレージ表スペースに使用されるパスのコレクションへの新規ストレージ・パスを追加します。
28 ページの『ALTER DATABASE PARTITION GROUP』	データベース・パーティション・グループの定義を変更します。
35 ページの『ALTER FUNCTION』	関数のプロパティを変更して既存の関数を変更します。
38 ページの『ALTER HISTOGRAM TEMPLATE』	1 つ以上のサービス・クラスまたは作業クラスのデフォルト・ヒストグラムをオーバーライドするために使用できるヒストグラムのタイプを記述するテンプレートを定義します。
39 ページの『ALTER METHOD』	メソッドに関連したメソッド本文を変更して既存のメソッドを変更します。
40 ページの『ALTER NICKNAME』	ニックネームの定義を変更します。
48 ページの『ALTER PROCEDURE (外部)』	プロシージャのプロパティを変更して既存の外部プロシージャを変更します。
51 ページの『ALTER PROCEDURE (ソース派生)』	ソース派生プロシージャの 1 つ以上のパラメーターのデータ・タイプを変更することによって、既存のソース派生プロシージャを変更します。
52 ページの『ALTER PROCEDURE (SQL)』	SQL のプロパティを変更して既存のプロシージャを変更します。
53 ページの『ALTER SECURITY LABEL COMPONENT』	セキュリティ・ラベル・コンポーネントを変更します。
56 ページの『ALTER SECURITY POLICY』	セキュリティ・ポリシーを変更します。
60 ページの『ALTER SEQUENCE』	シーケンスの定義を変更します。

SQL ステートメント

表 1. SQL スキーマ・ステートメント (続き)

SQL ステートメント	目的
64 ページの『ALTER SERVER』	フェデレーテッド・システム内のデータ・ソースの定義を変更します。
67 ページの『ALTER SERVICE CLASS』	サービス・クラスの定義を変更します。
74 ページの『ALTER TABLE』	表の定義を変更します。
123 ページの『ALTER TABLESPACE』	表スペースの定義を変更します。
133 ページの『ALTER THRESHOLD』	しきい値の定義を変更します。
140 ページの『ALTER TRUSTED CONTEXT』	現行サーバーのトラステッド・コンテキストの定義を変更します。
149 ページの『ALTER TYPE (構造化)』	構造化タイプの定義を変更します。
156 ページの『ALTER USER MAPPING』	ユーザー許可マッピングの定義を変更します。
158 ページの『ALTER VIEW』	参照タイプ列を変更して有効範囲を追加することによって、ビューの定義を変更します。
160 ページの『ALTER WORK ACTION SET』	作業アクション・セット内の作業アクションについて追加、変更、またはドロップを行います。
173 ページの『ALTER WORK CLASS SET』	作業クラス・セット内の作業クラスについて追加、変更、またはドロップを行います。
178 ページの『ALTER WORKLOAD』	ワークロードを変更します。
185 ページの『ALTER WRAPPER』	特定のタイプのデータ・ソースにアクセスするために (ラッパー・モジュールと一緒に) 使用されるオプションを更新します。
187 ページの『ALTER XSROBJECT』	特定の XML スキーマの分解サポートを使用可能または使用不可にします。
190 ページの『AUDIT』	現行サーバーの特定のデータベースまたはデータベース・オブジェクトで使用する監査ポリシーを決定します。
205 ページの『COMMENT』	オブジェクトの記述のコメントを置換または追加します。
254 ページの『CREATE ALIAS』	表、ビュー、または別の別名に対する別名を定義します。
256 ページの『CREATE AUDIT POLICY』	現行サーバーの監査ポリシーを定義します。
259 ページの『CREATE BUFFERPOOL』	新しいバッファ・プールを作成します。
263 ページの『CREATE DATABASE PARTITION GROUP』	データベース・パーティション・グループを定義します。
265 ページの『CREATE EVENT MONITOR』	モニターしたいデータベースのイベントを指定します。
287 ページの『CREATE FUNCTION』	ユーザー定義関数を登録します。
288 ページの『CREATE FUNCTION (外部スカラー)』	ユーザー定義外部スカラー関数を登録します。
315 ページの『CREATE FUNCTION (外部表)』	ユーザー定義外部表関数を登録します。
335 ページの『CREATE FUNCTION (OLE DB 外部表)』	ユーザー定義 OLE DB 外部表関数を登録します。
344 ページの『CREATE FUNCTION (ソース派生またはテンプレート)』	ユーザー定義ソース関数を登録します。
356 ページの『CREATE FUNCTION (SQL スカラー、表、または行)』	ユーザー定義 SQL 関数を登録および定義します。
367 ページの『CREATE FUNCTION MAPPING』	関数マッピングを定義します。

表 1. SQL スキーマ・ステートメント (続き)

SQL ステートメント	目的
371 ページの『CREATE HISTOGRAM TEMPLATE』	1 つ以上のサービス・クラスまたは作業クラスのデフォルト・ヒストグラムをオーバーライドするために使用できるヒストグラムのタイプを記述するテンプレートを定義します。
373 ページの『CREATE INDEX』	表の索引を定義します。
390 ページの『CREATE INDEX EXTENSION』	構造化タイプまたは特殊タイプ列のある表で、索引を使用するための拡張オブジェクトを定義します。
397 ページの『CREATE METHOD』	以前から定義されているメソッド指定にメソッド本体を関連付けます。
403 ページの『CREATE NICKNAME』	ニックネームを定義します。
416 ページの『CREATE PROCEDURE』	プロシージャを登録します。
416 ページの『CREATE PROCEDURE (外部)』	外部プロシージャを登録します。
432 ページの『CREATE PROCEDURE (ソース派生)』	別のプロシージャ (ソース・プロシージャ) を基にしたプロシージャ (ソース派生プロシージャ) を登録します。フェデレーテッド・システムにおいて、フェデレーテッド・プロシージャとは、サポートされているデータ・ソースにソース・プロシージャを持つソース派生プロシージャのことです。
436 ページの『CREATE PROCEDURE (SQL)』	SQL プロシージャを登録します。
443 ページの『CREATE ROLE』	現行サーバーのロールを定義します。
443 ページの『CREATE SCHEMA』	スキーマを定義します。
446 ページの『CREATE SECURITY LABEL COMPONENT』	セキュリティー・ポリシーの一環として使用されるコンポーネントを作成します。
449 ページの『CREATE SECURITY LABEL』	セキュリティー・ラベルを作成します。
450 ページの『CREATE SECURITY POLICY』	セキュリティー・ポリシーを作成します。
452 ページの『CREATE SEQUENCE』	シーケンスを定義します。
465 ページの『CREATE SERVER』	データ・ソースをフェデレーテッド・データベースへ定義します。
457 ページの『CREATE SERVICE CLASS』	サービス・クラスを定義します。
469 ページの『CREATE TABLE』	表を定義します。
542 ページの『CREATE TABLESPACE』	表スペースを定義します。
557 ページの『CREATE THRESHOLD』	しきい値を定義します。
566 ページの『CREATE TRANSFORM』	変換関数を定義します。
570 ページの『CREATE TRIGGER』	トリガーを定義します。
583 ページの『CREATE TRUSTED CONTEXT』	現行サーバーのトラステッド・コンテキストを定義します。
590 ページの『CREATE TYPE (配列)』	配列タイプを定義します。
592 ページの『CREATE TYPE (特殊)』	特殊データ・タイプを定義します。
598 ページの『CREATE TYPE (構造化)』	構造化データ・タイプを定義します。
624 ページの『CREATE TYPE MAPPING』	データ・タイプ間でのマッピングを定義します。
630 ページの『CREATE USER MAPPING』	ユーザー許可間でのマッピングを定義します。
632 ページの『CREATE VARIABLE』	グローバル変数を定義します。

SQL ステートメント

表 1. SQL スキーマ・ステートメント (続き)

SQL ステートメント	目的
635 ページの『CREATE VIEW』	1 つ以上の表、ビュー、あるいはニックネームのビューを定義します。
650 ページの『CREATE WORK ACTION SET』	作業アクション・セットおよびその中の作業アクションを定義します。
658 ページの『CREATE WORK CLASS SET』	作業クラス・セットを定義します。
663 ページの『CREATE WORKLOAD』	ワークロードを定義します。
672 ページの『CREATE WRAPPER』	ラッパーを登録します。
705 ページの『DROP』	データベース中のオブジェクトを削除します。
768 ページの『GRANT (データベース権限)』	データベース全体に対する権限を付与します。
773 ページの『GRANT (免除)』	指定されたラベル・ベースのアクセス制御 (LBAC) セキュリティー・ポリシーに対して、アクセス規則の免除を認可します。
775 ページの『GRANT (グローバル変数特権)』	作成されたグローバル変数に 1 つ以上の特権を付与します。
777 ページの『GRANT (索引特権)』	データベースの索引に対する CONTROL 特権を付与します。
779 ページの『GRANT (パッケージ特権)』	データベースのパッケージに対する特権を付与します。
782 ページの『GRANT (ロール)』	ロールを、ユーザー、グループ、またはその他のロールに付与します。
785 ページの『GRANT (ルーチン特権)』	ルーチン (関数、メソッド、またはプロシージャ) についての特権を付与します。
789 ページの『GRANT (スキーマ特権)』	スキーマについての特権を付与します。
791 ページの『GRANT (セキュリティ・ラベル)』	読み取りアクセス、書き込みアクセス、または読み取りアクセスと書き込みアクセスの両方に対する、ラベル・ベースのアクセス制御 (LBAC) セキュリティー・ラベルを認可します。
794 ページの『GRANT (シーケンス特権)』	シーケンスについての特権を付与します。
796 ページの『GRANT (サーバー特権)』	特定のデータ・ソースを照会するための特権を付与します。
798 ページの『GRANT (SETSESSIONUSER 特権)』	SET SESSION AUTHORIZATION ステートメントを使用する特権を認可します。
800 ページの『GRANT (表スペース特権)』	表スペースについての特権を付与します。
802 ページの『GRANT (表、ビュー、またはニックネーム特権)』	表、ビュー、およびニックネームについての特権を付与します。
809 ページの『GRANT (ワークロード特権)』	ワークロードに USAGE 特権を付与します。
811 ページの『GRANT (XSR オブジェクト特権)』	XSR オブジェクトに対する USAGE 特権を認可します。
853 ページの『REFRESH TABLE』	マテリアライズ照会表のデータをリフレッシュします。
860 ページの『RENAME』	既存の表の名前を変更します。
858 ページの『RENAME TABLESPACE』	既存の表スペースの名前を変更します。
867 ページの『REVOKE (データベース権限)』	データベース全体から権限を取り消します。
871 ページの『REVOKE (免除)』	指定されたラベル・ベースのアクセス制御 (LBAC) セキュリティー・ポリシーに対するアクセス規則の免除を取り消します。
873 ページの『REVOKE (グローバル変数特権)』	作成されたグローバル変数に対する 1 つ以上の特権を取り消します。

表 1. SQL スキーマ・ステートメント (続き)

SQL ステートメント	目的
875 ページの『REVOKE (索引特権)』	所定の索引に対する CONTROL 特権を取り消します。
877 ページの『REVOKE (パッケージ特権)』	データベースの所定のパッケージから特権を取り消します。
880 ページの『REVOKE (ロール)』	ロールを、ユーザー、グループ、またはその他のロールから取り消します。
882 ページの『REVOKE (ルーチン特権)』	ルーチン (関数、メソッド、またはプロシージャ) についての特権を取り消します。
886 ページの『REVOKE (スキーマ特権)』	スキーマの特権を取り消します。
888 ページの『REVOKE (セキュリティ・ラベル)』	読み取りアクセス、書き込みアクセス、または読み取りアクセスと書き込みアクセスの両方に対する、ラベル・ベースのアクセス制御 (LBAC) セキュリティ・ラベルを取り消します。
890 ページの『REVOKE (シーケンス特権)』	シーケンスについての特権を取り消します。
892 ページの『REVOKE (サーバー特権)』	特定のデータ・ソースを照会するための特権を取り消します。
894 ページの『REVOKE (SETSESSIONUSER 特権)』	SET SESSION AUTHORIZATION ステートメントを使用する特権を取り消します。
895 ページの『REVOKE (表スペース特権)』	指定した表スペースに対する USE 特権を取り消します。
897 ページの『REVOKE (表、ビュー、またはニックネーム特権)』	所定の表、ビュー、またはニックネームから特権を取り消します。
903 ページの『REVOKE (ワークロード特権)』	ワークロードに対する USAGE 特権を取り消します。
904 ページの『REVOKE (XSR オブジェクト特権)』	XSR オブジェクトに対する USAGE 特権を取り消します。
949 ページの『SET INTEGRITY』	SET INTEGRITY ペンディング状態を設定し、制約違反の有無についてデータを検査します。
989 ページの『TRANSFER OWNERSHIP』	データベース・オブジェクトの所有権を転送します。

表 2. SQL データ変更ステートメント

SQL ステートメント	目的
688 ページの『DELETE』	1 つ以上の行を表から削除します。
815 ページの『INSERT』	1 つ以上の行を表に挿入します。
830 ページの『MERGE』	ソース (表参照の結果) からのデータを使ってターゲット (表またはビュー) を更新します。
1004 ページの『UPDATE』	表の 1 つ以上の行に含まれる 1 つ以上の列の値を更新します。

表 3. SQL データ・ステートメント

SQL ステートメント	目的
21 ページの『ALLOCATE CURSOR』	結果セット・ロケータ変数で識別される結果セットにカーソルを割り当てます。
188 ページの『ASSOCIATE LOCATORS』	プロシージャから戻される結果セットごとの結果セット・ロケータ一値を入手します。
203 ページの『CLOSE』	カーソルをクローズします。
673 ページの『DECLARE CURSOR』	SQL カーソルを定義します。
688 ページの『DELETE』	1 つ以上の行を表から削除します。

SQL ステートメント

表 3. SQL データ・ステートメント (続き)

SQL ステートメント	目的
755 ページの『FETCH』	行の値をホスト変数に割り当てます。
758 ページの『FLUSH EVENT MONITOR』	イベント・モニターのアクティブな内部バッファを書き出します。
761 ページの『FLUSH PACKAGE CACHE』	現在パッケージ・キャッシュ内に存在する、キャッシュに入れられたすべての動的 SQL を除去します。
764 ページの『FREE LOCATOR』	ロケータ変数とその値の間の関連を除去します。
815 ページの『INSERT』	1 つ以上の行を表に挿入します。
827 ページの『LOCK TABLE』	並行処理による表の変更、または並行処理による表の使用のいずれかを禁止します。
830 ページの『MERGE』	ソース (表参照の結果) からのデータを使ってターゲット (表またはビュー) を更新します。
839 ページの『OPEN』	FETCH ステートメントが出された時に値を検索するのに使用するカーソルを準備します。
911 ページの『SELECT INTO』	1 行以内の結果表を指定し、その値をホスト変数に割り当てます。
979 ページの『SET 変数』	NEW 遷移変数に値を割り当てます。
1004 ページの『UPDATE』	表の 1 つ以上の行に含まれる 1 つ以上の列の値を更新します。
1015 ページの『VALUES INTO』	1 行以内の結果表を指定し、その値をホスト変数に割り当てます。

表 4. SQL トランザクション・ステートメント

SQL ステートメント	目的
218 ページの『COMMIT』	作業単位を終了し、その作業単位によって行われたデータベースの変更をコミットします。
858 ページの『RELEASE SAVEPOINT』	トランザクション内のセーブポイントを解放します。
905 ページの『ROLLBACK』	作業単位を終了し、その作業単位によって行われたデータベースの変更をバックアウトします。
908 ページの『SAVEPOINT』	トランザクション内にセーブポイントを設定します。

表 5. SQL 接続ステートメント

SQL ステートメント	目的
238 ページの『CONNECT (タイプ 1)』	リモート作業単位の規則に従って、アプリケーション・サーバーに接続します。
246 ページの『CONNECT (タイプ 2)』	アプリケーション制御の分散作業単位の規則に従って、アプリケーション・サーバーに接続します。
703 ページの『DISCONNECT』	アクティブな作業単位がない場合に 1 つ以上の接続を終了します。
856 ページの『RELEASE (接続)』	1 つ以上の接続を解放ペンディング状態にします。
914 ページの『SET CONNECTION』	接続の状態を休止状態から現行状態に変更し、指定した位置を現行サーバーにします。

表 6. SQL 動的ステートメント

SQL ステートメント	目的
695 ページの『DESCRIBE』	オブジェクトに関する情報を入手します。
695 ページの『DESCRIBE INPUT』	準備済みステートメントの入力パラメーター・マーカーについての情報を入手します。

表 6. SQL 動的ステートメント (続き)

SQL ステートメント	目的
698 ページの『DESCRIBE OUTPUT』	準備済みステートメントに関する情報、または準備済み SELECT ステートメント内の選択リスト列に関する情報を入手します。
743 ページの『EXECUTE』	準備済み SQL ステートメントを実行します。
741 ページの『EXECUTE IMMEDIATE』	SQL ステートメントを準備して実行します。
844 ページの『PREPARE』	SQL ステートメント (およびオプション・パラメーター) を準備し、実行できるようにします。

表 7. SQL セッション・ステートメント

SQL ステートメント	目的
680 ページの『DECLARE GLOBAL TEMPORARY TABLE』	グローバル一時表を定義します。
750 ページの『EXPLAIN』	選択したアクセス・プランについての情報をキャプチャーします。
913 ページの『SET COMPILATION ENVIRONMENT』	接続内の現行コンパイル環境を、デッドロック・イベント・モニターによって提供されるコンパイル環境に含まれている値と一致するように変更します。
915 ページの『SET CURRENT DECFLOAT ROUNDING MODE』	指定された丸めモードが CURRENT DECFLOAT ROUNDING MODE 特殊レジスターに現在設定されている値であることを検査します。
917 ページの『SET CURRENT DEFAULT TRANSFORM GROUP』	CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターの値を変更します。
918 ページの『SET CURRENT DEGREE』	CURRENT DEGREE 特殊レジスターの値を変更します。
920 ページの『SET CURRENT EXPLAIN MODE』	CURRENT EXPLAIN MODE 特殊レジスターの値を変更します。
923 ページの『SET CURRENT EXPLAIN SNAPSHOT』	CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値を変更します。
925 ページの『SET CURRENT FEDERATED ASYNCHRONY』	CURRENT FEDERATED ASYNCHRONY 特殊レジスターの値を変更します。
926 ページの『SET CURRENT IMPLICIT XMLPARSE OPTION』	CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの値を変更します。
927 ページの『SET CURRENT ISOLATION』	CURRENT ISOLATION 特殊レジスターの値を変更します。
928 ページの『SET CURRENT LOCK TIMEOUT』	CURRENT LOCK TIMEOUT 特殊レジスターの値を変更します。
930 ページの『SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION』	CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターの値を変更します。
932 ページの『SET CURRENT MDC ROLLOUT MODE』	CURRENT MDC ROLLOUT MODE 特殊レジスターの値を割り当てます。
933 ページの『SET CURRENT OPTIMIZATION PROFILE』	CURRENT OPTIMIZATION PROFILE 特殊レジスターの値を割り当てます。
937 ページの『SET CURRENT PACKAGE PATH』	CURRENT PACKAGE PATH 特殊レジスターの値を割り当てます。
940 ページの『SET CURRENT PACKAGESET』	パッケージ選択のためのスキーマ名を設定します。
942 ページの『SET CURRENT QUERY OPTIMIZATION』	CURRENT QUERY OPTIMIZATION 特殊レジスターの値を変更します。

SQL ステートメント

表7. SQL セッション・ステートメント (続き)

SQL ステートメント	目的
944 ページの『SET CURRENT REFRESH AGE』	CURRENT REFRESH AGE 特殊レジスターの値を変更します。
946 ページの『SET ENCRYPTION PASSWORD』	暗号化のためのパスワードを設定します。
947 ページの『SET EVENT MONITOR STATE』	イベント・モニターを活動化または非活動化します。
968 ページの『SET PASSTHRU』	データ・ソースのネイティブ SQL を、対象となるデータ・ソースへ直接にサブミットするため、セッションをオープンします。
970 ページの『SET PATH』	CURRENT PATH 特殊レジスターの値を変更します。
972 ページの『SET ROLE』	セッションの許可 ID が特定のロールのメンバーであることを確認します。
973 ページの『SET SCHEMA』	CURRENT SCHEMA 特殊レジスターの値を変更します。
975 ページの『SET SERVER OPTION』	サーバーのオプション設定をセットします。
977 ページの『SET SESSION AUTHORIZATION』	SESSION USER 特殊レジスターの値を変更します。

表8. SQL 組み込みホスト言語ステートメント

SQL ステートメント	目的
194 ページの『BEGIN DECLARE SECTION』	ホスト変数宣言セクションの始まりを示します。
740 ページの『END DECLARE SECTION』	ホスト変数宣言セクションの終わりを示します。
765 ページの『GET DIAGNOSTICS』	以前に実行した SQL ステートメントについての情報を得るために使用されます。
814 ページの『INCLUDE』	ソース・プログラムにコードまたは宣言を挿入します。
863 ページの『RESIGNAL』	エラーまたは警告条件を再通知するのに使用します。
986 ページの『SIGNAL』	エラーまたは警告条件を通知するのに使用します。
1017 ページの『WHENEVER』	SQL の戻りコードに基づいて実行するアクションを定義します。

表9. SQL 制御ステートメント

SQL ステートメント	目的
195 ページの『CALL』	プロシーチャーを呼び出します。
201 ページの『CASE』	複数の条件に基づいて実行パスを選択します。
221 ページの『コンパウンド SQL (動的)』	1 つ以上の他の SQL ステートメントを結合して 1 つの動的ブロックにします。
225 ページの『コンパウンド SQL (組み込み)』	1 つ以上の他の SQL ステートメントを結合して 1 つの実行可能ブロックにします。
229 ページの『コンパウンド SQL (プロシーチャー)』	別のステートメントを SQL プロシーチャーにグループ化します。
761 ページの『FOR』	表の行ごとに、ステートメントまたはステートメントのグループを実行します。
767 ページの『GOTO』	SQL プロシーチャー内のユーザー定義ラベルに分岐させるために使用されます。
812 ページの『IF』	条件の評価に基づいて実行パスを選択します。

表 9. SQL 制御ステートメント (続き)

SQL ステートメント	目的
824 ページの『ITERATE』	制御のフローがラベル付きループの最初に戻ります。
826 ページの『LEAVE』	プログラム制御をループまたはコンパウンド・ステートメントの外側に移動させます。
828 ページの『LOOP』	ステートメント、またはステートメントのグループの実行を繰り返します。
862 ページの『REPEAT』	検索条件が真になるまでステートメントまたはステートメントのグループを実行します。
863 ページの『RESIGNAL』	エラーまたは警告条件を再通知するのに使用します。
865 ページの『RETURN』	ルーチンから戻るのに使用します。
986 ページの『SIGNAL』	エラーまたは警告条件を通知するのに使用します。
1018 ページの『WHILE』	指定した条件が真である間、ステートメント、またはステートメントのグループの実行を繰り返します。

SQL ステートメントの呼び出し方法

SQL ステートメントは、実行可能と実行不能に分類されます。

実行可能ステートメントには、4 つの呼び出し方法があります。それらは次のとおりです。

- アプリケーション・プログラムに組み込む。
- SQL プロシージャに組み込む。
- 動的に準備して実行する。
- 対話式に発行する。

ステートメントによっては、これらのいくつかまたはすべての方式を使用することができます。(REXX™ に組み込んだステートメントは、動的に準備され実行されません。)

実行不能ステートメントは、アプリケーション・プログラムに組み込む方式だけが可能です。

別の SQL ステートメント構成は、select ステートメントです。select ステートメントには、3 つの呼び出し方法があります。それらは次のとおりです。

- DECLARE CURSOR に組み込んで、OPEN、FETCH および CLOSE によって暗黙的に実行する。(静的起動)
- 動的に準備し、DECLARE CURSOR で参照して、OPEN、FETCH および CLOSE によって暗黙的に実行する。(動的起動)
- 対話式に発行する。

アプリケーション・プログラムへのステートメントの組み込み

SQL ステートメントは、プリコンパイラにサブミットされるソース・プログラムに組み込むことができます。このようなステートメントは、プログラムに組み込まれていると言います。組み込みステートメントは、ホスト言語のステートメントが

SQL ステートメントの呼び出し方法

可能な位置であればそのプログラム内のどこにでも組み込むことができます。各組み込みステートメントの前には、キーワード `EXEC SQL` を付ける必要があります。

アプリケーション・プログラムに組み込まれた実行可能ステートメントは、ホスト言語ステートメントが実行されるたびに、そこに実行可能ステートメントが指定されていると同じ時点で実行されます。したがって、ループ内のステートメントは、ループが行われるたびに実行され、条件構文内のステートメントは、その条件が満たされた場合にのみ実行されます。

組み込みステートメントには、ホスト変数への参照を含むことができます。参照されるホスト変数は、以下のような 2 つの方法で使用することができます。

- 入力として使用する (ホスト変数の現行値がそのステートメントの実行に使用されます)。
- 出力として使用する (ホスト変数には、そのステートメントの実行結果として新しい値が割り当てられます)。

特に、式および述部の中のホスト変数に対する参照はすべて、変数の現行値により置き換えられます。つまり、変数は入力として使用されます。

すべての実行可能ステートメントの後で、必ず `SQL` 戻りコードのテストを行う必要があります。別の方法として、`WHENEVER` ステートメント (それ自体は実行不能) を使用して、組み込みステートメントの実行直後の制御の流れを変更することもできます。

データ操作言語 (DML) ステートメントで参照されるオブジェクトはすべて、ステートメントがデータベースにバインドされる時点で存在している必要があります。

組み込まれた実行不能ステートメントは、プリコンパイラーによってのみ処理されます。プリコンパイラーはステートメントにエラーを検出すると、それを報告します。このようなステートメントは、プログラムの実行時に処理されることはありません。したがって、このようなステートメントの後で `SQL` 戻りコードのテストを行ってはなりません。

`CREATE PROCEDURE` ステートメントの `SQL` プロシージャ本体にステートメントを組み込むことができます。このようなステートメントは、`SQL` プロシージャに組み込まれているとされます。 `SQL` ステートメントの説明でホスト変数が参照されるときはいつでも、ステートメントが `SQL` プロシージャに組み込まれていれば `SQL` 変数 を使用できます。

動的な準備と実行

アプリケーション・プログラムでは、ホスト変数に入った文字ストリングの形式の `SQL` ステートメントを動的に構築することができます。一般にステートメントは、プログラムが入手可能な何らかのデータから構築されます (たとえば、ワークステーションからの入力)。構成されたステートメント (`select` ステートメントではない) は、(組み込み) `PREPARE` ステートメントによって準備され、(組み込み) `EXECUTE` ステートメントによって実行することができます。あるいは、(組み込み) `EXECUTE IMMEDIATE` ステートメントを使用して、1 つのステップでステートメントを準備して実行することもできます。

動的に準備されるステートメントには、ホスト変数への参照が含まれてはなりません。パラメーター・マーカーは含めることができます。(パラメーター・マーカーの規則に関しては、『PREPARE』を参照してください。) 準備済みのステートメントが実行される時点で、パラメーター・マーカーは、実際には EXECUTE ステートメントで指定されたホスト変数の現行値に置き換えられます。一度準備したステートメントは、ホスト変数の他の値を用いて何回も実行することができます。パラメーター・マーカーは、EXECUTE IMMEDIATE ステートメントでは使用できません。

ステートメントが正しく実行されたか否かは、EXECUTE (または EXECUTE IMMEDIATE) ステートメントの実行後の SQLCA への SQL 戻りコードの設定値によって示されます。前述のように、SQL 戻りコードは必ず検査する必要があります。詳しくは、12 ページの『SQL 戻りコード (SQLCODE と SQLSTATE)』を参照してください。

select ステートメントの静的呼び出し

select ステートメントは、(実行不能) DECLARE CURSOR ステートメントの一部として含めることができます。このようなステートメントは、(組み込み) OPEN ステートメントによってカーソルがオープンされるたびに実行されます。カーソルがオープンされた後で、一連の FETCH ステートメントを実行することにより、結果表を一度に 1 つの行ずつ取り出すことができます。

このように使用する場合、select ステートメントにホスト変数への参照を含めることができます。これらの参照は、実際には、OPEN ステートメントを実行した時点での変数の値によって置き換えられます。

select ステートメントの動的呼び出し

アプリケーション・プログラムは、ホスト変数に入った文字ストリングの形式で、選択 (SELECT) ステートメントを動的に構築することができます。一般に、ステートメントはプログラムが入手可能な何らかのデータから構築されます (たとえば、ワークステーションから入手した照会)。このように構成されたステートメントは、(組み込み) PREPARE ステートメントによって実行の準備が行われ、(実行不能) DECLARE CURSOR ステートメントによって参照されます。このようなステートメントは、(組み込み) OPEN ステートメントによってカーソルがオープンされるたびに実行されます。カーソルがオープンされた後で、一連の FETCH ステートメントを実行することにより、結果表を一度に 1 つの行ずつ取り出すことができます。

このように使用する場合、select ステートメントにホスト変数への参照を含めることはできません。パラメーター・マーカーは含めることができます。パラメーター・マーカーは、実際には、OPEN ステートメントに指定されたホスト変数の値によって置き換えられます。

対話式呼び出し

ワークステーションから SQL ステートメントを入力する機能は、データベース・マネージャーのアーキテクチャーの一部です。この方法で入力されたステートメントは、「対話式に発行される」と呼ばれます。このようなステートメントは、アプリケーション・プログラムのコンテキストでのみ認識されるので、パラメーター・

SQL ステートメントの呼び出し方法

マーカーやホスト変数への参照を含まない実行可能ステートメントでなければなりません。

異なるホスト・システムで使用される SQL

SQL ステートメントの構文は、ホスト・システムの種類 (DB2 for z/OS、DB2 for System i、DB2 Database for Linux, UNIX, and Windows) によって微妙に異なります。アプリケーション内の SQL ステートメントが静的か動的にかかわらず、別のデータベース・ホスト・システムにアクセスするアプリケーションの場合は、SQL ステートメントとプリコンパイル/ BIND オプションが、アクセス先のデータベース・システムでサポートされるようにするのは重要なことです。

他のホスト・システムでの SQL ステートメントの使用についての詳細情報は、*DB2 for i5/OS SQL Reference* and the *DB2 for z/OS SQL* 解説書を参照してください。

SQL 戻りコード (SQLCODE と SQLSTATE)

実行可能な SQL ステートメントを含むアプリケーション・プログラムは、SQLCODE または SQLSTATE の値のいずれかを使用して、SQL ステートメントからの戻りコードを処理することができます。アプリケーションでこれらの値にアクセスするには、2つの方法があります。

- SQLCA と呼ばれる構造体を組み込む。SQLCA には SQLCODE という名前の整変数と、SQLSTATE という名前の文字ストリングが含まれています。REXX では、SQLCA は自動的に提供されます。他の言語では、INCLUDE SQLCA ステートメントを使用することによって、SQLCA を入手することができます。
- プリコンパイル・オプションとして LANGLEVEL SQL92E が指定されている場合は、プログラムの SQL 宣言セクションに SQLCODE または SQLSTATE という名前の変数を宣言することができます。これらの値がいずれも SQL 宣言セクションに宣言されていない場合は、プログラムの別のロケーションで SQLCODE という名の変数が宣言されているものと想定されます。LANGLEVEL SQL92E を使用する場合は、プログラムに INCLUDE SQLCA ステートメントがあってはなりません。

SQLCODE は、各 SQL ステートメントの実行後に、データベース・マネージャーによって設定されます。すべてのデータベース・マネージャーは、次のように ISO/ANSI SQL 標準規格に準拠しています。

- SQLCODE = 0 で SQLWARN0 がブランクの場合、実行は成功しました。
- SQLCODE = 100 の場合、“データが見つかりませんでした”。たとえば、カーソルが結果表の最後の行より後に設定されていたために、FETCH ステートメントからデータが戻されませんでした。
- SQLCODE > 0 で、100 ではない場合、実行は警告付きで成功しました。
- SQLCODE = 0 で SQLWARN0 = 'W' の場合、実行は成功しましたが、1つ以上の警告標識がセットされました。
- SQLCODE < 0 の場合、実行は不成功でした。

0 と 100 以外の SQLCODE の値の意味は、製品によって異なります。

SQLSTATE は、各 SQL ステートメントの実行後に、データベース・マネージャーによって設定されます。アプリケーション・プログラムは、SQLCODE ではなく、

SQLSTATE をテストすることによって SQL ステートメントの実行を検査することができます。SQLSTATE は、共通エラー条件に関する共通コードを示します。アプリケーション・プログラムが特定のエラーまたは特定クラスのエラーの有無をテストできます。コード体系は、IBM のどのデータベース・マネージャーでも同じであり、ISO/ANSI SQL92 標準規格に基づいています。

SQL コメント

静的 SQL ステートメントには、ホスト言語または SQL のコメントを含めることができます。動的 SQL ステートメントには、SQL コメントを含めることができます。SQL コメントには以下の 2 つのタイプがあります。

単純コメント

単純コメントは、2 つの連続するハイフン (--) で始まり、行末で終わります。

括弧で囲まれたコメント

括弧で囲まれたコメントは、/* で始まり、*/ で終わります。

単純コメントを使用する際には、以下の規則が適用されます。

- 2 つのハイフンが同一行にあることが必要で、その間にスペースを入れることはできません。
- 単純コメントは、スペースが有効な個所であればどこからでも開始できます (区切りトークンの中、または 'EXEC' と 'SQL' との間を除く)。
- 単純コメントは次の行へ続けることはできません。
- COBOL では、2 つのハイフンの前にスペースを 1 つ入れる必要があります。

括弧で囲まれたコメントを使用する際には、以下の規則が適用されます。

- /* が同一行にあることが必要で、その間にスペースを入れることはできません。
- */ が同一行にあることが必要で、その間にスペースを入れることはできません。
- 括弧で囲まれたコメントは、スペースが有効な個所であればどこからでも開始できます (区切りトークンの中、または 'EXEC' と 'SQL' との間を除く)。
- 括弧で囲まれたコメントは後続の行へ続けることができます。

例 1: この例は、ステートメントに単純コメントを組み込む方法を示しています。

```
CREATE VIEW PRJ_MAXPER      -- PROJECTS WITH MOST SUPPORT PERSONNEL
AS SELECT PROJNO, PROJNAME -- NUMBER AND NAME OF PROJECT
FROM PROJECT
WHERE DEPTNO = 'E21'      -- SYSTEMS SUPPORT DEPT CODE
AND PRSTAFF > 1
```

例 2: この例は、ステートメントに括弧で囲まれたコメントを組み込む方法を示しています。

```
CREATE VIEW PRJ_MAXPER      /* PROJECTS WITH MOST SUPPORT
                             PERSONNEL */
AS SELECT PROJNO, PROJNAME /* NUMBER AND NAME OF PROJECT */
FROM PROJECT
WHERE DEPTNO = 'E21'      /* SYSTEMS SUPPORT DEPT CODE */
AND PRSTAFF > 1
```

SQL 制御ステートメントについて

制御ステートメントとは、構造化プログラミング言語でプログラムを作成する方法と同様に構造化照会言語を使用できるようにする SQL ステートメントです。SQL 制御ステートメントには、論理の流れを制御し、変数を宣言して設定し、警告や例外を処理する機能が備えられています。ある SQL 制御ステートメントに他のネストされた SQL ステートメントが組み込まれていることもあります。ルーチンの本体、トリガー、または動的コンパウンド・ステートメントで SQL 制御ステートメントを使用できます。

SQL パラメーター、SQL 変数、およびグローバル変数の参照

式や変数を指定できる SQL プロシージャ・ステートメント中のどこでも、SQL パラメーター、SQL 変数、およびグローバル変数を参照できます。SQL ルーチン、SQL トリガー、または動的コンパウンド・ステートメント中にホスト変数を指定することはできません。ルーチン中のどこでも SQL パラメーターを参照でき、ルーチン名で修飾できます。SQL 変数が宣言されているコンパウンド・ステートメント中のどこでも SQL 変数を参照でき、そのコンパウンド・ステートメントの先頭にラベル名を指定して SQL 変数を修飾できます。グローバル変数は、式が決定論的である必要がない限り、どの式の中でも参照できます。以下のシナリオでは決定論的式が必要となり、グローバル変数は使用できません。

- チェック制約
- 生成列の定義
- 即時リフレッシュ MQT

SQL パラメーター、SQL 変数、およびグローバル変数はすべて NULL 可能と見なされます。SQL ルーチン中の SQL パラメーター、SQL 変数、またはグローバル変数の名前を、そのルーチン中で参照されている表やビューの列名と同じ名前にすることもできます。SQL 変数の名前を、同じルーチンで宣言されている別の SQL 変数名と同じ名前にすることもできます。このことは、2 つの SQL 変数が別々のコンパウンド・ステートメントで宣言される場合に生じる可能性があります。SQL 変数の宣言を含むコンパウンド・ステートメントは、この変数の有効範囲を決定します。詳しくは、『コンパウンド SQL (プロシージャ)』を参照してください。

SQL ルーチン中の SQL 変数または SQL パラメーターの名前を、特定の SQL ステートメント中で使用されている ID 名と同じ名前にすることもできます。名前を修飾しない場合は、以下の規則により、名前が ID、SQL パラメーター、または SQL 変数のいずれであるかが示されます。

- SET PATH および SET SCHEMA ステートメント中では、名前は SQL パラメーターまたは SQL 変数として検査されます。SQL 変数または SQL パラメーターとして検出されない場合は、ID として使用されます。
- CONNECT、DISCONNECT、RELEASE、および SET CONNECTION ステートメントでは、この名前は ID として使用されます。

同一の名前は明示的に修飾する必要があります。名前の修飾は、この名前が列、SQL 変数、SQL パラメーター、またはグローバル変数を参照するかどうかを明確に示します。名前を修飾しない場合、または、修飾されていても未確定な場合は、以下の規則により、名前が列、SQL 変数、SQL パラメーター、またはグローバル変数のいずれであるかが示されます。

- SQL ルーチン本体で指定されている表やビューが、そのルーチンの作成時に既存の場合は、名前は最初に列名として検査されます。列として検出されない場合は、次にコンパウンド・ステートメント内の SQL 変数として検査され、その後 SQL パラメーターとして検査され、そして最後に、グローバル変数として検査されます。
- 参照されている表やビューが、そのルーチンの作成時に存在しない場合は、名前は最初にコンパウンド・ステートメント内の SQL 変数として、次に SQL パラメーターとして、次にグローバル変数として検査されます。変数は、参照を含むコンパウンド・ステートメント内、またはそのようなコンパウンド・ステートメントがネストしているコンパウンド・ステートメント内で宣言しなければなりません。同一の有効範囲内に 2 つの SQL 変数があって、同一の名前を持つ場合 (この変数が別々のコンパウンド・ステートメント内で宣言されると生じる場合がある)、最も内側のコンパウンド・ステートメントで宣言された SQL 変数が使用されます。検出されない場合は、列であるとみなされます。

ラベルの参照

ラベルはほとんどの SQL プロシージャ・ステートメント上で指定できます。ラベルを定義するステートメントを含むコンパウンド・ステートメントは、このラベル名の有効範囲を決定します。ラベル名は、それが定義されるコンパウンド・ステートメント内で固有でなければなりません。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントに定義されたラベルも同様です (SQLSTATE 42734)。ラベルは、コンパウンド・ステートメント自体で指定したラベルと同一にしてはなりません (SQLSTATE 42734)。また、SQL プロシージャ・ステートメントを含むルーチン名と同一にしてはなりません (SQLSTATE 42734)。

ラベル名は、それが定義されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です。ラベルは SQL 変数名の修飾に使用することができます。また、GOTO、LEAVE、または ITERATE ステートメントのターゲットとして指定することもできます。

SQL 条件名の参照

SQL 条件名を、同じルーチンで宣言されている別の SQL 条件名と同じ名前にすることもできます。このことは、2 つの SQL 条件が別々のコンパウンド・ステートメントで宣言される場合に生じる可能性があります。SQL 条件名の宣言を含むコンパウンド・ステートメントは、この条件名の有効範囲を決定します。条件名は、それが宣言されるコンパウンド・ステートメント内で固有でなければなりません。ただし、そのようなコンパウンド・ステートメント内でネストされたコンパウンド・ステートメント内での宣言は例外です (SQLSTATE 42734)。条件名は、それが宣言されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です。条件名の参照がある場合、最も内側のコンパウンド・ステートメントで宣言された条件が、使用される条件です。詳しくは、『コンパウンド SQL (プロシージャ)』を参照してください。

SQL ステートメント名の参照

SQL ステートメント名を、同じルーチンで宣言されている別の SQL ステートメントと同じ名前にもすることもできます。このことは、2 つの SQL ステートメントが別々のコンパウンド・ステートメントで宣言される場合に生じる可能性があります。SQL ステートメント名の宣言を含むコンパウンド・ステートメントは、このステートメント名の有効範囲を決定します。ステートメント名は、それが宣言されるコンパウンド・ステートメント内で固有でなければなりません。ただし、そのようなコンパウンド・ステートメント内でネストされたコンパウンド・ステートメント内での宣言は例外です (SQLSTATE 42734)。ステートメント名は、それが宣言されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です。ステートメント名の参照がある場合、最も内側のコンパウンド・ステートメントで宣言されたステートメントが使用されるステートメントです。詳しくは、『コンパウンド SQL (プロシージャ)』を参照してください。

SQL カーソル名の参照

SQL カーソル名を、同じルーチンで宣言されている別の SQL カーソル名と同じ名前にもすることもできます。このことは、2 つの SQL カーソルが別々のコンパウンド・ステートメントで宣言される場合に生じる可能性があります。SQL カーソルの宣言を含むコンパウンド・ステートメントは、このカーソル名の有効範囲を決定します。カーソル名は、それが宣言されるコンパウンド・ステートメント内で固有でなければなりません。ただし、そのようなコンパウンド・ステートメント内でネストされたコンパウンド・ステートメント内での宣言は例外です (SQLSTATE 42734)。カーソル名は、それが宣言されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です。カーソル名の参照がある場合、最も内側のコンパウンド・ステートメントで宣言されたカーソルが使用されるカーソルです。詳しくは、『コンパウンド SQL (プロシージャ)』を参照してください。

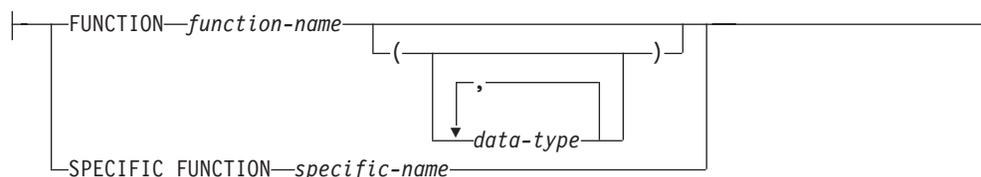
関数、メソッド、およびプロシージャの指定子

以下の節では、関数、メソッド、またはプロシージャを一意的に識別するために使用される構文フラグメントを記述します。

関数指定子

関数指定子は、単一の関数を一意的に識別します。一般的に関数指定子は、関数の DDL ステートメント (DROP または ALTER など) で使用されます。

function-designator:



FUNCTION *function-name*

特定の関数を指定します。 *function-name* という名前の関数インスタンスがスキーマ内に 1 つだけ存在している場合にのみ有効です。指定する関数には、任意の数のパラメーターを定義できます。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこの名前の関数が存在しない場合は、エラー (SQLSTATE 42704) になります。指定したスキーマまたは暗黙のスキーマに、この関数のインスタンスが複数存在する場合は、エラー (SQLSTATE 42725) になります。

FUNCTION *function-name* (*data-type*,...)

関数を固有に指定する関数シグニチャーを指定します。関数解決のアルゴリズムは使用されません。

function-name

関数の名前を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(data-type, ...)

値は、CREATE FUNCTION ステートメント上で (対応する位置に) 指定されたデータ・タイプに一致していなければなりません。データ・タイプの数、およびデータ・タイプを論理的に連結した値が、特定の関数インスタンスを識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索してタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE FUNCTION ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つ関数がない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC FUNCTION *specific-name*

関数の作成時に指定された名前、またはデフォルト値として与えられた名前を使

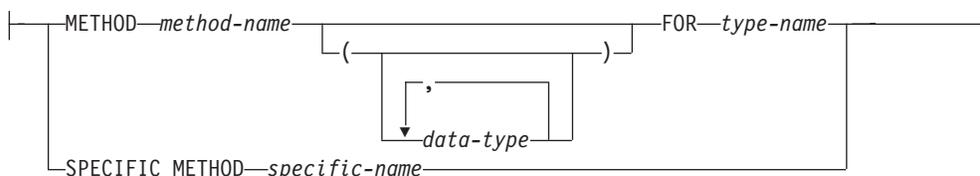
関数、メソッド、およびプロシーチャーの指定子

用して、特定のユーザー定義関数を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定関数のインスタンスを指定していなければなりません。そうでない場合、エラー (SQLSTATE 42704) になります。

メソッド指定子

メソッド指定子は、単一のメソッドを一意的に識別します。一般的にメソッド指定子は、メソッドの DDL ステートメント (DROP または ALTER など) で使用されます。

method-designator:



METHOD *method-name*

特定のメソッドを指定します。 *type-name* というサブジェクト・タイプの *method-name* という名前のメソッド・インスタンスが 1 つだけ存在している場合にのみ有効です。このように指定されたメソッドには、任意の数のパラメーターを定義できます。タイプに、指定された名前のメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。タイプに、そのメソッドのインスタンスが複数存在する場合も、エラーが戻されます (SQLSTATE 42725)。

METHOD *method-name* (*data-type*,...)

メソッドを一意に指定するメソッド・シグニチャーを指定します。メソッド解決のアルゴリズムは使用されません。

method-name

type-name タイプのメソッドの名前を指定します。

(*data-type*,...)

値は、CREATE TYPE ステートメント上で (対応する位置に) 指定されたデータ・タイプと一致していなければなりません。データ・タイプの数、およびデータ・タイプを論理的に連結した値が、特定のメソッド・インスタンスを識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索してタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

関数、メソッド、およびプロシーチャーの指定子

長さ、精度、または位取りをコーディングする場合、その値は、CREATE TYPE ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(n) のタイプは、 n に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つメソッドのタイプがない場合は、エラー (SQLSTATE 42883) になります。

FOR *type-name*

指定されたメソッドを関連付けるタイプを指定します。ここで指定される名前は、カタログにすでに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

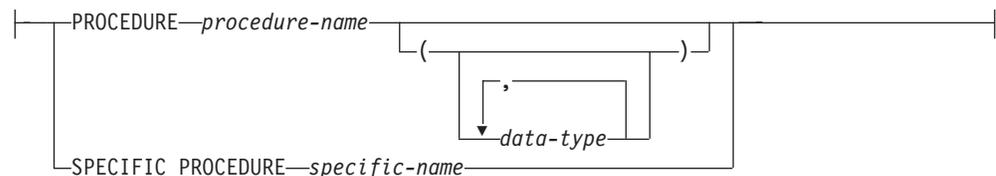
SPECIFIC METHOD *specific-name*

メソッドの作成時に指定された名前か、デフォルト値として与えられた名前を使用して、特定のメソッドを識別します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定のメソッド・インスタンスを指定していなければなりません。そうでない場合、エラー (SQLSTATE 42704) になります。

プロシーチャー指定子

プロシーチャー指定子は、単一のプロシーチャーを一意的に識別します。一般的にプロシーチャー指定子は、プロシーチャーの DDL ステートメント (DROP または ALTER など) で使用されます。

procedure-designator:



PROCEDURE *procedure-name*

特定のプロシーチャーを指定します。 *procedure-name* という名前のプロシーチャー・インスタンスがスキーマ内に 1 つだけ存在している場合にのみ有効です。指定するプロシーチャーには、任意の数のパラメーターを定義できます。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のない

関数、メソッド、およびプロシーチャーの指定子

オブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマに該当する名前のプロシーチャーが存在しない場合は、エラーが戻されず (SQLSTATE 42704)。指定したスキーマまたは暗黙のスキーマに、このプロシーチャーのインスタンスが複数存在する場合は、エラー (SQLSTATE 42725) になります。

PROCEDURE *procedure-name* (*data-type*,...)

プロシーチャーを一意的に固有に識別するプロシーチャー・シグニチャーを指定します。プロシーチャー解決のアルゴリズムは使用されません。

procedure-name

プロシーチャーの名前を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(*data-type*,...)

値は、CREATE PROCEDURE ステートメント上で (対応する位置に) 指定されたデータ・タイプに一致していなければなりません。データ・タイプの数、およびデータ・タイプを論理的に連結した値が、特定のプロシーチャー・インスタンスを識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索してタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE PROCEDURE ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つプロシーチャーがない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC PROCEDURE *specific-name*

プロシーチャーの作成時に指定された名前、またはデフォルト値として与えられた名前を使用して、特定のプロシーチャーを指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* に指定される名前は、指定したスキーマ

マまたは暗黙のスキーマに含まれる特定プロシーチャーのインスタンスを識別するものでなければなりません。それ以外の名前が指定された場合は、エラーが戻されます (SQLSTATE 42704)。

ALLOCATE CURSOR

ALLOCATE CURSOR ステートメントは、結果セット・ロケータ変数で識別される結果セットにカーソルを割り当てます。結果セット・ロケータ変数については、ASSOCIATE LOCATORS ステートメントの説明を参照してください。

呼び出し

このステートメントは、SQL プロシーチャーに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

許可

必要ありません。

構文

```
▶▶—ALLOCATE—cursor-name—CURSOR FOR RESULT SET—rs-locator-variable—▶▶
```

説明

cursor-name

カーソルの名前を指定します。ソース SQL プロシーチャーですでに宣言されているカーソルと同じ名前は使用しないでください (SQLSTATE 24502)。

CURSOR FOR RESULT SET *rs-locator-variable*

結果セット・ロケータ変数の宣言規則に従って、ソース SQL プロシーチャーで宣言されている結果セット・ロケータ変数の名前を指定します。SQL 変数の宣言について詳しくは、『コンパウンド SQL (プロシーチャー) ステートメント』を参照してください。

結果セット・ロケータ変数には、ASSOCIATE LOCATORS SQL ステートメントで戻された、有効な結果セット・ロケータ値を入れなければなりません (SQLSTATE 0F001)。

規則

- 割り当てカーソルを使用する際には、以下の規則が適用されます。
 - 割り当てカーソルは、OPEN ステートメントによってオープンすることはできません (SQLSTATE 24502)。
 - 割り当てカーソルは、位置指定 UPDATE または DELETE ステートメントでは使用できません (SQLSTATE 42828)。
 - 割り当てカーソルは、CLOSE ステートメントによってクローズできます。割り当てカーソルをクローズすると、関連付けられたカーソルがクローズされません。

ALLOCATE CURSOR

- 各結果セットに割り当てられるカーソルは 1 つだけです。
- 割り当てカーソルは、ロールバック操作、暗黙的クローズ、または明示的クローズが行われるまで継続します。
- コミット操作を行うと、割り当てカーソルで、WITH HOLD が定義されていないものが破棄されます。
- 割り当てカーソルを破棄すると、SQL プロシージャ内の関連付けられたカーソルがクローズされます。

例

以下の SQL プロシージャの例では、カーソル C1 を定義し、結果セット・ロケータ変数 LOC1、および SQL プロシージャによって戻される関連する結果セットに関連付けます。

```
ALLOCATE C1 CURSOR FOR RESULT SET LOC1;
```

ALTER AUDIT POLICY

ALTER AUDIT POLICY ステートメントは、現行サーバーの監査ポリシーの定義を変更します。

呼び出し

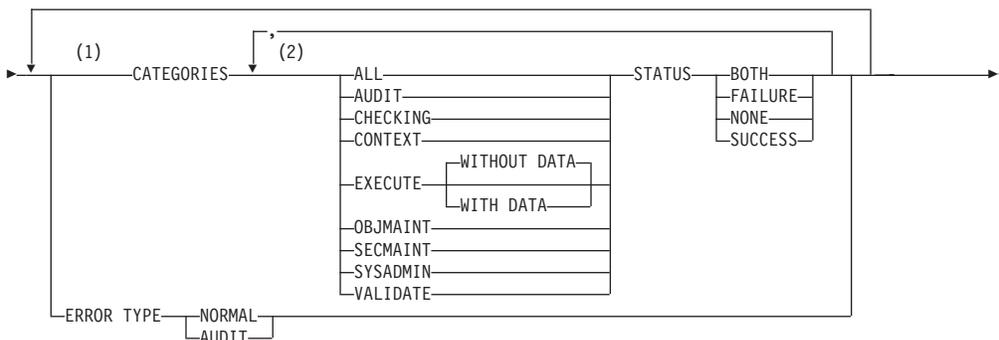
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

▶▶ALTER AUDIT POLICY—*policy-name*————▶▶



注:

- 1 CATEGORIES および ERROR TYPE 節はそれぞれ 1 度しか指定できません (SQLSTATE 42614)。
- 2 カテゴリーはそれぞれ 2 度以上指定できず (SQLSTATE 42614)、ALL が指定されている場合にはその他のカテゴリーを指定できません (SQLSTATE 42601)。

説明

policy-name

変更する監査ポリシーを指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。名前は、現行のサーバー上の既存の監査ポリシーを一意的に識別するものでなければなりません (SQLSTATE 42704)。

CATEGORIES

新規の状況値が指定される 1 つ以上の監査カテゴリーのリスト。ALL が指定されない場合、明示的に指定されないカテゴリーの STATUS は未変更のままとなります。

ALL

すべてのカテゴリーを同じ状況に設定します。EXECUTE カテゴリーは WITHOUT DATA です。

AUDIT

監査設定が変更されたとき、または監査ログにアクセスされたときにレコードを生成します。

CHECKING

データベース・オブジェクトまたは関数へのアクセス試行またはその操作試行の許可検査中にレコードを生成します。

CONTEXT

データベース操作が実行されたときの操作コンテキストを示すレコードを生成します。

EXECUTE

SQL ステートメントの実行を示すレコードを生成します。

WITHOUT DATA または WITH DATA

任意のホスト変数およびパラメーター・マーカーに指定される入力データ値を EXECUTE カテゴリーの一部としてログに記録するかどうかを指定します。

WITHOUT DATA

任意のホスト変数およびパラメーター・マーカーに指定される入力データ値は EXECUTE カテゴリーの一部としてログに記録されません。

WITH DATA

任意のホスト変数およびパラメーター・マーカーに指定される入力データ値は EXECUTE カテゴリーの一部としてログに記録されます。すべての入力値がログに記録されるわけではありません。特に、LOB、LONG、XML、および構造化タイプのパラメーターは

ALTER AUDIT POLICY

NULL 値となります。日付、時刻、およびタイム・スタンプの各フィールドは ISO 形式でログに記録されます。入力データ値はログに記録される前にデータベース・コード・ページに変換されます。コード・ページの変換に失敗してもエラーは戻されず、変換前のデータがログに記録されます。

OBJMAINT

データ・オブジェクトが作成またはドロップされたときにレコードを生成します。

SECMAINT

オブジェクト特権、データベース特権、または DBADM 権限が付与されたとき、または取り消されたときにレコードを生成します。データベース・マネージャのセキュリティ構成パラメーター **sysadm_group**、**sysctrl_group**、または **sysmaint_group** が変更されたときにもレコードが生成されます。

SYSADMIN

SYSADM、SYSMAINT、または SYSCTRL 権限を必要とする操作が実行されたときにレコードを生成します。

VALIDATE

ユーザーが認証されたとき、またはユーザーに関連したシステム・セキュリティ情報が取得されたときにレコードを生成します。

STATUS

指定されたカテゴリの状況を指定します。

BOTH

成功イベントと失敗イベントが監査されます。

FAILURE

失敗イベントのみが監査されます。

SUCCESS

成功イベントのみが監査されます。

NONE

このカテゴリのイベントは監査されません。

ERROR TYPE

監査エラーを戻すか、無視するかを指定します。

NORMAL

監査によって生成されたエラーはすべて無視され、実行される操作に関連したエラーの SQLCODE だけがアプリケーションに戻されます。

AUDIT

監査機能自体で発生したエラーを含む、すべてのエラーがアプリケーションに戻されます。

規則

- AUDIT 排他 SQL ステートメントの後は、COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。AUDIT 排他 SQL ステートメントは次のとおりです。

– AUDIT

- CREATE AUDIT POLICY、ALTER AUDIT POLICY、または DROP (AUDIT POLICY)
- ロールまたはトラステッド・コンテキストが監査ポリシーと関連付けられる場合は DROP (ROLE) または DROP (TRUSTED CONTEXT)
- AUDIT 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- データベース・パーティション全体を通じて、同時に実行できる非コミットの AUDIT 排他 SQL ステートメントは 1 つのみです。非コミットの AUDIT 排他 SQL ステートメントが実行されている場合、後続の AUDIT 排他 SQL ステートメントは、現行の AUDIT 排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。
- 変更される監査ポリシーが現在データベース・オブジェクトと関連付けられている場合、変更はそれによって影響を受けるアプリケーションの次の作業単位までは有効になりません。例えば、監査ポリシーがデータベースで使用されている場合、現行の作業単位はその COMMIT または ROLLBACK ステートメントが完了するまではポリシーに対する変更を認識しません。

例

DBAUDPRF という名前の監査ポリシーの SECMAINT、CHECKING、および VALIDATE カテゴリが成功と失敗の両方を監査するように変更します。

```
ALTER AUDIT POLICY DBAUDPRF
  CATEGORIES SECMAINT STATUS BOTH,
             CHECKING STATUS BOTH,
             VALIDATE STATUS BOTH
```

ALTER BUFFERPOOL

ALTER BUFFERPOOL ステートメントは、以下を行う場合に使用されます。

- すべてのデータベース・パーティション、あるいは 1 つのデータベース・パーティションのバッファ・プールのサイズを変更する。
- バッファ・プールの自動サイジングを使用可能あるいは使用不可に設定する。
- このバッファ・プール定義を新規のデータベース・パーティション・グループに追加する。
- ブロック・ベース入出力用のバッファ・プールのブロック域を変更する。

呼び出し

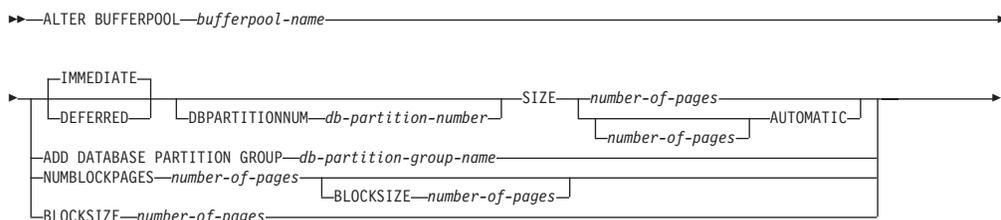
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

ALTER BUFFERPOOL

許可

このステートメントの許可 ID が持つ特権には、SYSCTRL または SYSADM 権限が含まれている必要があります。

構文



説明

bufferpool-name

バッファークラスタの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。バッファークラスタは、カタログで記述されている必要があります。

IMMEDIATE または DEFERRED

バッファークラスタのサイズが直ちに変更されるようにするかどうかを指示します。

IMMEDIATE

バッファークラスタのサイズが直ちに変更されます。メモリーを共用するデータベース内に新規スペースを割り振るのに十分な予約済みのスペースがない場合 (SQLSTATE 01657)、このステートメントは DEFERRED で実行されます。

DEFERRED

データベースが再活動化される時に (すべてのアプリケーションがデータベースから切断される必要があります)、バッファークラスタのサイズが変更されます。予約済みのメモリー・スペースは必要ありません。DB2 が、活動状態にあるときにシステムから必要なメモリーを割り振ります。

DBPARTITIONNUM *db-partition-number*

そのバッファークラスタのサイズを変更するデータベース・パーティションを指定します。例外エントリが、SYSCAT.BUFFERPOOLDBPARTITIONS システム・カタログ・ビューで作成されます。データベース・パーティションは、そのバッファークラスタのデータベース・パーティション・グループのいずれかに入っている必要があります (SQLSTATE 42729)。この節の指定がない場合、SYSCAT.BUFFERPOOLDBPARTITIONS に例外エントリを持つデータベース・パーティションを除く、すべてのデータベース・パーティションでバッファークラスタのサイズが変更されます。

SIZE

バッファークラスタの新規サイズを指定するか、このバッファークラスタのセルフチューニングを使用可能または使用不可に設定します。

number-of-pages

新規のバッファークラスタ・サイズに対応するページ数。バッファークラスタ

ルがすでにセルフチューニング・バッファーク・プールになっている場合に、**SIZE number-of-pages** 節を指定すると、この変更操作によりこのバッファーク・プールのセルフチューニングは使用不可になります。

AUTOMATIC

このバッファーク・プールのセルフチューニングを使用可能に設定します。データベース・マネージャーは、作業負荷の要件に応じてバッファーク・プールのサイズを調整します。AUTOMATIC が指定された場合、DBPARTITIONNUM 節は指定できません (SQLSTATE 42601)。

ADD DATABASE PARTITION GROUP db-partition-group-name

バッファーク・プール定義が適用されるデータベース・パーティション・グループのリストに、このデータベース・パーティション・グループを追加します。バッファーク・プールが定義されていないデータベース・パーティション・グループにあるデータベース・パーティションについては、バッファーク・プールに指定されているデフォルト・サイズを使用して、このデータベース・パーティションにバッファーク・プールが作成されます。db-partition-group-name 内の表スペースは、このバッファーク・プールを指定できます。データベース・パーティション・グループは、現在データベースに存在している必要があります (SQLSTATE 42704)。

NUMBLOCKPAGES number-of-pages

ブロック・ベース域に存在していなければならないページ数を指定します。ページ数は、バッファーク・プールのページ数の 98% より小さくしなければなりません (SQLSTATE 54052)。値 0 を指定すると、ブロック入出力は不可になります。使用されている NUMBLOCKPAGES の実際の値は、BLOCKSIZE の倍数になります。

BLOCKSIZE number-of-pages

ブロック内のページ数を指定します。ブロック・サイズの値は、2 から 256 でなければなりません (SQLSTATE 54053)。デフォルト値は 32 です。

注

- バッファーク・プール・サイズのみが動的に (直ちに) 変更可能です。他のすべての変更は据え置かれ、データベースが再活動化された後にこれらの変更は効力を持つようになります。
- このステートメントが据え置かれて実行 (Deferred) される場合には、次のことが当てはまります。バッファーク・プール定義はトランザクションで、コミット時にバッファーク・プール定義に対する変更がカタログ表に反映されますが、実際のバッファーク・プールに対する変更は、次回にデータベースが始動されるまでは有効になりません。それまではバッファーク・プールの現行の属性が存在し、その間バッファーク・プールには何の影響もありません。新しいデータベース・パーティション・グループの表スペースに作成された表は、デフォルトのバッファーク・プールを使用します。キーワードが適用される際のこのステートメントのデフォルトは、IMMEDIATE です。
- すべてのバッファーク・プールの合計と、その他のデータベース・マネージャーやアプリケーションの要件に合うように、マシンに十分な実メモリーが必要です。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。

ALTER BUFFERPOOL

- DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。

ALTER DATABASE PARTITION GROUP

ALTER DATABASE PARTITION GROUP ステートメントは、以下の目的で使用されます。

- データベース・パーティション・グループに 1 つ以上のデータベース・パーティションを追加する。
- データベース・パーティション・グループから 1 つ以上のデータベース・パーティションをドロップする。

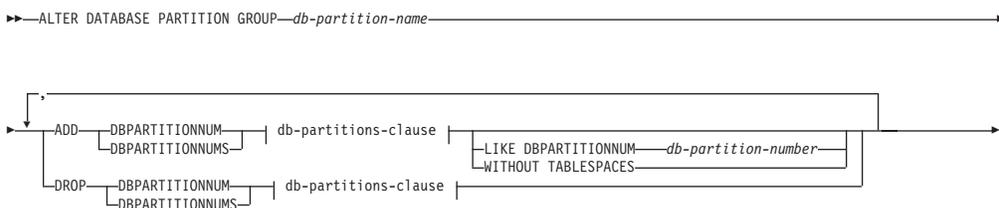
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合のみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID には、SYSCTRL 権限または SYSADM 権限がなければなりません。

構文



db-partitions-clause:



説明

db-partition-name

データベース・パーティション・グループの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。データベース・パーティション・グループはカタログに記述されている必要があります。 IBMCATGROUP および IBMTEMPGROUP は指定できません (SQLSTATE 42832)。

ADD DBPARTITIONNUM

データベース・パーティション・グループに特定の 1 つまたは複数のデータベース・パーティションを追加することを指定します。DBPARTITIONNUMS は DBPARTITIONNUM の同義語です。指定するデータベース・パーティションは、データベース・パーティション・グループにすでに定義済みであってはなりません (SQLSTATE 42728)。

DROP DBPARTITIONNUM

データベース・パーティション・グループから特定の 1 つまたは複数のデータベース・パーティションをドロップすることを指定します。

DBPARTITIONNUMS は DBPARTITIONNUM の同義語です。指定するデータベース・パーティションは、データベース・パーティション・グループにすでに定義されている必要があります (SQLSTATE 42729)。

db-partitions-clause

追加またはドロップする 1 つまたは複数のデータベース・パーティションを指定します。

db-partition-number1

特定のデータベース・パーティション番号を指定します。

TO *db-partition-number2*

データベース・パーティション番号の範囲を指定します。

db-partition-number2 の値は、*db-partition-number1* の値以上でなければなりません (SQLSTATE 428A9)。

LIKE DBPARTITIONNUM *db-partition-number*

データベース・パーティション・グループの既存の表スペースのコンテナが、指定した *db-partition-number* のコンテナと同じであることを指定します。指定するデータベース・パーティションは、このステートメントの前にデータベース・パーティション・グループに存在しており、同じステートメントの DROP DBPARTITIONNUM 節に含まれていないパーティションである必要があります。

自動ストレージを使用するよう定義されている表スペース (つまり、CREATE TABLESPACE ステートメントの MANAGED BY AUTOMATIC STORAGE 節を使用して作成されているか、それに対してまったく MANAGED BY 節が指定されていない表スペース) については、コンテナは必ずしも、指定のパーティションのものと一致するとは限りません。その代わりに、コンテナは、データベースに関連するストレージ・パスに基づいて、データベース・マネージャーによって自動的に割り当てられ、この結果として同じコンテナが使用される場合もあれば、使用されない場合もあります。各表スペースのサイズは、表スペース作成時に指定された初期サイズに基づいて決まり、指定のパーティション上の表スペースの現行サイズと一致しない場合もあります。

WITHOUT TABLESPACES

データベース・パーティション・グループの既存の表スペースのコンテナが、新規に追加された 1 つまたは複数のデータベース・パーティション上に作成されないことを指定します。*db-partitions-clause* を用いた ALTER TABLESPACE ステートメントを使用して、このデータベース・パーティション・グループに対して定義される表スペースで使用するコンテナを定義する必要があります。このオプションの指定がない場合、そのデータベース・パーティ

ALTER DATABASE PARTITION GROUP

ション・グループに対して表スペースが定義されるたびに、新たに追加されるデータベース・パーティションにデフォルトのコンテナが指定されます。

自動ストレージを使用するよう定義されている表スペース (つまり、CREATE TABLESPACE ステートメントの MANAGED BY AUTOMATIC STORAGE 節を使用して作成されているか、それに対してまったく MANAGED BY 節が指定されていない表スペース) については、このオプションは無視されます。こうした表スペースに関して、コンテナの作成を後に延ばすということではできません。コンテナは、データベースに関連するストレージ・パスに基づいて、データベース・マネージャーによって自動的に割り当てられます。各表スペースのサイズは、表スペース作成時に指定された初期サイズに基づいて決まります。

規則

- 番号によって指定するそれぞれのデータベース・パーティションは、db2nodes.cfg ファイルに定義されていなければなりません (SQLSTATE 42729)。
- db-partitions-clause にリストされる db-partition-number は、それぞれ固有のデータベース・パーティションに対する番号でなければなりません (SQLSTATE 42728)。
- 有効なデータベース・パーティション番号は、0 から 999 まで (0 と 999 を含む) です (SQLSTATE 42729)。
- 1 つのデータベース・パーティションを ADD と DROP の両方の節に指定することはできません (SQLSTATE 42728)。
- データベース・パーティション・グループには少なくとも 1 つのデータベース・パーティションが残っている必要があります。最後のデータベース・パーティションをデータベース・パーティション・グループからドロップすることはできません (SQLSTATE 428C0)。
- データベース・パーティションを追加する際に、LIKE DBPARTITIONNUM 節も WITHOUT TABLESPACES 節も指定されていない場合、デフォルト解釈により、データベース・パーティション・グループの既存のデータベース・パーティションの最も小さいデータベース・パーティション番号 (ここでは 2 とします) が使用され、LIKE DBPARTITIONNUM 2 が指定された場合と同様の処理が行われます。既存のデータベース・パーティションをデフォルト値として使用する場合、そのデータベース・パーティションではデータベース・パーティション・グループ内のすべての表スペースに対してコンテナが定義されている必要があります (SYSCAT.DBPARTITIONGROUPDEF の列 IN_USE が 'T' でない)。

注

- データベース・パーティションがデータベース・パーティション・グループに追加されると、そのデータベース・パーティションに対するカタログ項目が作成されます (SYSCAT.DBPARTITIONGROUPDEF を参照)。以下のいずれかの場合には、分散マップは直ちに更新され、新しいデータベース・パーティションが、そのデータベース・パーティションが分散マップにあることを示す標識 (IN_USE) を伴って組み込まれます。
 - データベース・パーティション・グループに表スペースが定義されていない、または
 - データベース・パーティション・グループに定義されている表スペースに表が定義されておらず、WITHOUT TABLESPACES 節が指定されていない

以下のいずれかの場合は、分散マップは変更されず、標識 (IN_USE) はそのデータベース・パーティションが分散マップに組み込まれていないことを示すように設定されます。

- データベース・パーティション・グループの表スペースに表が存在する、または
- データベース・パーティション・グループに表スペースが存在し、WITHOUT TABLESPACES 節が指定された (すべての表スペースが自動ストレージを使用するよう定義されている場合を除く。その場合、WITHOUT TABLESPACES 節は無視される)。

分散マップを変更するには、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用する必要があります。このコマンドは、任意のデータを再配分し、分散マップを変更し、標識を変更します。 WITHOUT TABLESPACES 節が指定された場合は、データを再配分する前に表スペース・コンテナを追加する必要があります。

- データベース・パーティションがデータベース・パーティション・グループからドロップされると、そのデータベース・パーティションのカatalog項目 (SYSCAT.DBPARTITIONGROUPDEF を参照) が更新されます。データベース・パーティション・グループに定義された表スペースに表が定義されていない場合、分散マップが直ちに変更され、ドロップされたデータベース・パーティションを除外し、データベース・パーティション・グループのそのデータベース・パーティションに関する項目がドロップされます。表が存在する場合は、分散マップは変更されず、標識 (IN_USE) はそのデータベース・パーティションがドロップを待機していることを示すように設定されます。 REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データを再配分し、データベース・パーティション・グループからそのデータベース・パーティションに関する項目をドロップする場合に、使用しなければなりません。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。
 - DBPARTITIONNUMS の代わりに NODES を指定できます。
 - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。

例

0、1、2、5、7、および 8 というデータベース・パーティションを持つ、6 つのパーティションのデータベースがあると想定します。2 つのデータベース・パーティション (3 と 6) をシステムに追加します。

- MAXGROUP という名前のデータベース・パーティション・グループに、データベース・パーティション 3 と 6 を追加し、データベース・パーティション 2 と同種の表スペース・コンテナを設定するとします。その場合、ステートメントは以下のようになります。

```
ALTER DATABASE PARTITION GROUP MAXGROUP
ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- データベース・パーティション 1 をドロップし、データベース・パーティション 6 をデータベース・パーティション・グループ MEDGROUP に追加するとしま

ALTER DATABASE PARTITION GROUP

す。ALTER TABLESPACE を使用して、データベース・パーティション 6 に対して別個に表スペース・コンテナを定義します。必要なステートメントは以下のようになります。

```
ALTER DATABASE PARTITION GROUP MEDGROUP
ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
DROP DBPARTITIONNUM(1)
```

ALTER DATABASE

ALTER DATABASE ステートメントは、自動ストレージ表スペースに使用されるパスのコレクションへの新規ストレージ・パスを追加します。自動ストレージ表スペースとは、自動ストレージを使用して作成した表スペースのことです。つまり、CREATE TABLESPACE ステートメントに MANAGED BY AUTOMATIC STORAGE 節を指定して作成した表スペース、または MANAGED BY 節を一切指定しないで作成した表スペースのことです。自動ストレージが使用可能なデータベースでは、その表スペースのコンテナおよびスペース管理特性を、データベース・マネージャーがすべて決定できます。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また対話式に出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合のみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SYSADM 権限か SYSCTRL 権限のいずれかが含まれている必要があります。

構文

```
▶▶ ALTER DATABASE database-name ADD STORAGE ON 'storage-location' ▶▶
```

説明

database-name

変更するデータベースの名前を指定するオプションの値。この値を指定する場合は、アプリケーションが現在接続しているデータベースの名前 (クライアントがカタログした別名ではない) と一致していなければなりません。そうでないとエラーが戻されます (SQLSTATE 42961)。

ADD STORAGE ON

自動ストレージ表スペースに使用されるストレージ・ロケーションのコレクションに、1 つ以上の新しいストレージ・ロケーションを追加するよう指定します。

'storage-location'

自動ストレージ表スペースのコンテナを作成する場所の絶対パスまたはドライブ名 (Windows オペレーティング・システムのみ) のいずれかを指定する文字列定数。

規則

- 自動ストレージがデータベースに定義されていない場合 (つまり、CREATE DATABASE コマンドで AUTOMATIC STORAGE NO 節を指定した場合) には、新規ストレージ・パスを追加できません (SQLSTATE 55060)。
- ストレージ・パスが存在し、アクセス可能でなければなりません (SQLSTATE 57019)。同様に、パーティション・データベース環境では、ストレージ・パスが存在し、各データベース・パーティションごとにアクセス可能でなければなりません (SQLSTATE 57019)。

注

- 新規ストレージ・パスを追加する際には、
 - 自動ストレージを使用する既存の REGULAR および LARGE 表スペースは、最初、これらの新規パスを使用しません。データベース・マネージャーは、スペース不足状態が発生した場合にのみ、これらのパスに新しい表スペース・コンテナを作成する可能性があります。
 - 自動ストレージを使用する既存の TEMPORARY 表スペースは、これらの新しいパスをすぐに使用する可能性があります。ADD STORAGE ON 操作時に TEMPORARY 表スペースが使用中でない場合には、これらのストレージ・パスに新規コンテナがすぐに作成される可能性があります。しかし TEMPORARY 表スペースが使用中の場合には、表スペースのユーザーがいなくなるまで、新規コンテナは作成されません。
- ADD STORAGE はログに記録される操作ですが、ロールフォワード操作中にそれが再実行されるかどうかは、データベースを復元した方法に左右されます。データベースに関連したストレージ・パスが復元操作によって再定義されない場合、ストレージ・パス定義を含むログ・レコードが再実行され、ログ・レコードに記述されたストレージ・パスがロールフォワード操作中に追加されます。ただし、復元操作中にストレージ・パスが再定義される場合、ストレージ・パスはセットアップ済みと見なされるため、ロールフォワード操作でこのログ・レコードが再実行されることはありません。この動作は高可用性災害時リカバリー (HADR) 環境にも当てはまります。スタンバイ・システムのセットアップ時にストレージ・パスが再定義された場合、ログ・レコードは再実行されません。
- 新規のデータベース・パーティションの追加中に、ADD STORAGE 節を使用してストレージ・パスをデータベースに追加しないでください。データベース・パーティションの追加中または追加後から、その新規パーティションの開始までの間に、新規のストレージ・パスを追加すると、新規のデータベース・パーティションにはその新規のストレージ・パスは反映されず、そのデータベース・パーティションは同期していない状態になります。
- データベース・パーティション上のストレージ・パス用のフリー・スペースの計算時には、ストレージ・パス内で以下のディレクトリまたはマウント・ポイントが存在するかどうかデータベース・マネージャーによって検査され、最初に見つかったものが使用されます。

ALTER DATABASE

```
<storage path>/<instance name>/NODE####/<database name>  
<storage path>/<instance name>/NODE####  
<storage path>/<instance name>  
<storage path>
```

ここで、

- <storage path> は、データベースに関連付けられたストレージ・パスです。
- <instance name> は、データベースが置かれているインスタンスです。
- NODE#### は、データベース・パーティション番号 (たとえば、NODE0000 または NODE0001) に対応します。
- <database name> は、データベースの名前です。

ファイル・システムは、ストレージ・パスの下の地点にマウントすることができます。すると、表スペース・コンテナに使用できる実際のフリー・スペース量が、ストレージ・パス・ディレクトリそのものに関連付けられている量と同量ではないことがデータベース・マネージャーで認識されます。

2 つの論理データベース・パーティションが 1 つの物理マシン上に存在し、1 つのストレージ・パス (/db2data) がある例について考察してみます。各データベース・パーティションはこのストレージ・パスを使用します。ただし、それぞれ独自のファイル・システム内で、各パーティションごとにデータを分離しようとしているとします。この場合、各パーティションごとに別々のファイル・システムを作成し、それを /db2data/<instance>/NODE#### にマウントすることができます。ストレージ・パス上にコンテナを作成し、フリー・スペースを判別するときは、データベース・マネージャーは /db2data のフリー・スペース情報を検索するのではなく、対応する /db2data/<instance>/NODE#### ディレクトリの情報を検索します。

- 一般的に、複数パーティション・データベースでは、どのパーティションにも、同じストレージ・パスを使用する必要があります。その例外の 1 つは、ストレージ・パス内でデータベース・パーティション式が使用される場合です。そうすることで、ストレージ・パスにデータベース・パーティション番号を反映することができ、その結果のパス名が各パーティションごとに異なることとなります。

データベース・パーティション式を示すには、引数 『 \$N 』 ([ブランク]\$N) を使用します。データベース・パーティション式は、ストレージ・パス内のどこでも使用できます。また、複数のデータベース・パーティション式を指定することもできます。データベース・パーティション式を終了するにはスペース文字を使用します。スペースの後ろのものはすべて、データベース・パーティション式が評価された後でストレージ・パスに付加されます。ストレージ・パスの中で、データベース・パーティション式の後ろにスペース文字がない場合は、ストリングの残りは式の一部であると見なされます。引数は、以下のいずれかの形式でのみ使用できます。

表 10. ストレージ・パスを作成するための引数： 演算子は左から右へと評価されます。データベース・パーティション番号は 10 であると想定します。

構文	例	値
[blank]\$N	" \$N"	10
[blank]\$N+[number]	" \$N+100"	110
[blank]\$N%[number]	" \$N%5" ^a	0

表 10. ストレージ・パスを作成するための引数 (続き): 演算子は左から右へと評価されます。データベース・パーティション番号は 10 であると想定します。

構文	例	値
[blank]\$N+[number]#[number]	" \$N+1#5"	1
[blank]\$N#[number]+[number]	" \$N#4+2"	4

^a % は係数演算子を表します。

例

例 1: 2 つのパスを /db2 ディレクトリの下に追加します (/db2/filesystem1 および /db2/filesystem2)。/filesystem3 という 3 つ目のパスは、現在接続中のデータベースに関連した自動ストレージ表スペースのスペースに追加します。

```
ALTER DATABASE ADD STORAGE ON '/db2/filesystem1', '/db2/filesystem2',
'/filesystem3'
```

例 2: ドライブ D および E を、SAMPLE データベースに関連した自動ストレージ表スペースのスペースに追加します。

```
ALTER DATABASE SAMPLE ADD STORAGE ON 'D:', 'E:¥'
```

例 3: ディレクトリ F:¥DB2DATA およびドライブ G を、現在接続中のデータベースに関連した自動ストレージ表スペースのスペースに追加します。

```
ALTER DATABASE ADD STORAGE ON 'F:¥DB2DATA', 'G:'
```

例 4: 各データベース・パーティション上のストレージ・パスを区別するために、データベース・パーティション式を使用するストレージ・パスを追加します。

```
ALTER DATABASE ADD STORAGE ON '/dataForPartition $N'
```

使用されるストレージ・パスは、データベース・パーティション 0 上では /dataForPartition0、データベース・パーティション 1 上では /dataForPartition1、などとなります。

ALTER FUNCTION

ALTER FUNCTION ステートメントは、既存の関数のプロパティを変更します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 関数のスキーマに対する ALTERIN 特権

ALTER FUNCTION

- SYSCAT.ROUTINES カタログ・ビューの OWNER 列に記録されているその関数の所有者
- SYSADM または DBADM 権限

関数の EXTERNAL NAME を変更するには、ステートメントの許可 ID の特権に、以下の特権の少なくとも 1 つが含まれている必要があります。

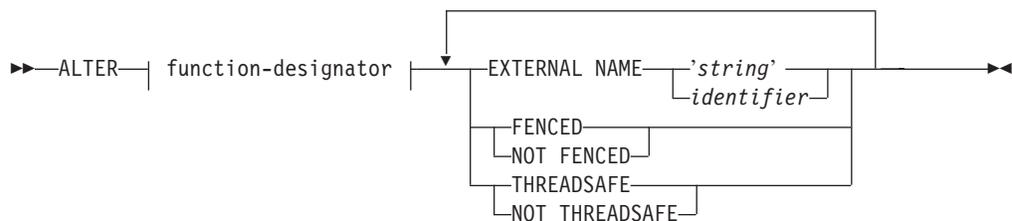
- データベースに対する CREATE_EXTERNAL_ROUTINE 権限
- SYSADM または DBADM 権限

fenced でない関数を変更するには、ステートメントの許可 ID の特権に以下の特権の少なくとも 1 つが含まれている必要があります。

- データベースに対する CREATE_NOT_FENCED_ROUTINE 権限
- SYSADM または DBADM 権限

fenced である関数を変更するには、さらに別の権限や特権は必要ありません。

構文



説明

function-designator

変更される関数を一意的に識別します。詳しくは、16 ページの『関数、メソッド、およびプロシーチャーの指定子』を参照してください。

EXTERNAL NAME 'string' または identifier

関数をインプリメントするユーザー作成コードの名前を指定します。このオプションは、外部関数を変更する際にのみ指定できます (SQLSTATE 42849)。

FENCED または NOT FENCED

関数をデータベース・マネージャーのオペレーティング環境のプロセスまたはアドレス・スペースで実行しても安全か (NOT FENCED)、そうでないか (FENCED) を指定します。多くの関数は、FENCED または NOT FENCED のどちらかで実行するように選択することができます。

関数が FENCED として登録されると、データベース・マネージャーは、その内部リソース (データ・バッファなど) を fenced して、その関数からアクセスされないようにします。一般に、FENCED として実行される関数は、NOT FENCED として実行されるものと同じようには実行されません。

注意:

適切にコード化、検討、および検査されていない関数に **NOT FENCED** を使用すると、**DB2** の保全性を損なう場合があります。**DB2** では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられています。ただし、**NOT FENCED** ユーザー定義関数が使用される場合には、完全な整合性を確保できません。

NOT THREADSAFE を宣言した関数は、**NOT FENCED** には変更できません (SQLSTATE 42613)。

関数が定義済みの **AS LOCATOR** の任意のパラメーターを有していて、**NO SQL** オプションが指定されていた場合には、この関数は **FENCED** には変更できません (SQLSTATE 42613)。

このオプションは **LANGUAGE OLE** 関数、**OLEDB** 関数、または **CLR** 関数を変更できません (SQLSTATE 42849)。

THREADSAFE または NOT THREADSAFE

関数を他のルーチンと同じプロセスで実行しても安全か (**THREADSAFE**)、そうでないか (**NOT THREADSAFE**) を指定します。

関数が **OLE** および **OLEDB** 以外の **LANGUAGE** で定義される場合:

- 関数が **THREADSAFE** に定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスで関数を呼び出すことができます。一般に、スレッド・セーフにするには、関数はどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。**FENCED** および **NOT FENCED** 関数の両方が **THREADSAFE** になることが可能です。
- 関数が **NOT THREADSAFE** と定義される場合には、データベース・マネージャーが関数を他のルーチンと同じプロセスで同時に呼び出すことは決してありません。**fenced** された関数だけが、**NOT THREADSAFE** になり得ます (SQLSTATE 42613)。

このオプションは **LANGUAGE OLE** 関数または **OLEDB** 関数を変更できません (SQLSTATE 42849)。

注

- **SYSIBM**、**SYSFUN**、または **SYSPROC** スキーマの関数は変更できません (SQLSTATE 42832)。
- **LANGUAGE SQL** で宣言した関数、ソース関数、またはテンプレート関数は変更できません (SQLSTATE 42917)。

例

関数 **MAIL()** は完全にテストされました。パフォーマンスを向上させるために、関数が **fenced** でないように変更します。

```
ALTER FUNCTION MAIL() NOT FENCED
```

ALTER HISTOGRAM TEMPLATE

ALTER HISTOGRAM TEMPLATE ステートメントは、1 つ以上のサービス・クラスまたは作業クラスのデフォルト・ヒストグラムをオーバーライドするために使用できるヒストグラムのタイプを記述するテンプレートを変更するために使用されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、 SYSADM または DBADM 権限が含まれている必要があります。

構文

```
▶▶ALTER HISTOGRAM TEMPLATE—template-name—HIGH BIN VALUE—bigint-constant—▶▶
```

説明

template-name

ヒストグラム・テンプレートの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。名前は、現行のサーバー上の既存のヒストグラム・テンプレートを識別するものでなければなりません (SQLSTATE 42704)。テンプレート名には デフォルト・システム・ヒストグラム・テンプレート SYSDEFAULTHISTOGRAM を指定できます。

HIGH BIN VALUE *bigint-constant*

最後から 2 番目の bin の上限値を指定します (最後の bin の値には上限がありません)。単位はヒストグラムの使い方によって異なります。最大値は 268 435 456 です。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。 WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)

- CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
- GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 全パーティションを通じて、同時に実行できる非コミットの WLM 排他 SQL ステートメントは 1 つのみです。非コミットの WLM 排他 SQL ステートメントが実行されている場合、後続の WLM 排他 SQL ステートメントは、現行の WLM 排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

LIFETIMETEMP という名前のヒストグラム・テンプレートの HIGH BIN VALUE を変更します。

```
ALTER HISTOGRAM TEMPLATE LIFETIMETEMP
HIGH BIN VALUE 90000
```

ALTER METHOD

ALTER METHOD ステートメントは、メソッドに関連付けたメソッド本体を変更して、既存のメソッドを変更します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する CREATE_EXTERNAL_ROUTINE 権限、および以下の少なくとも 1 つ。
 - タイプのスキーマに対する ALTERIN 特権
 - SYSCAT.DATATYPES カタログ・ビューの OWNER 列に記録されているそのタイプの所有者
- SYSADM または DBADM 権限

構文

```
▶▶ALTER | method-designator | EXTERNAL NAME | 'string' | identifier
```

ALTER METHOD

説明

method-designator

変更されるメソッドを一意的に識別します。詳しくは、16 ページの『関数、メソッド、およびプロシージャの指定子』を参照してください。

EXTERNAL NAME *'string'* または *identifier*

メソッドをインプリメントするユーザー作成コードの名前を指定します。このオプションは、外部メソッドを変更する際にのみ指定できます (SQLSTATE 42849)。

注

- SYSIBM、SYSFUN、または SYSPROC スキーマのメソッドは変更できません (SQLSTATE 42832)。
- LANGUAGE SQL として宣言されたメソッドは変更できません (SQLSTATE 42917)。
- LANGUAGE CLR として宣言されたメソッドは変更できません (SQLSTATE 42849)。
- 指定するメソッドは、変更する前に本体を持っている必要があります (SQLSTATE 42704)。

例

newaddresslib ライブラリーを使用するように、構造化タイプ ADDRESS_T の DISTANCE() メソッドを変更します。

```
ALTER METHOD DISTANCE()  
FOR TYPE ADDRESS_T  
EXTERNAL NAME 'newaddresslib!distance2'
```

ALTER NICKNAME

ALTER NICKNAME ステートメントは、データ・ソース・オブジェクト (表、ビュー、またはファイルなど) に関連したニックネーム情報を変更します。このステートメントは、次のようにして、フェデレーテッド・データベースに保管されている情報を変更します。

- データ・ソース・オブジェクトの列のローカル列名を変更する
- データ・ソース・オブジェクトの列のローカル・データ・タイプを変更する
- ニックネーム・オプションおよび列オプションを追加、設定、またはドロップする
- 主キーを追加またはドロップする
- 1 つ以上のユニーク制約、参照制約、またはチェック制約を追加またはドロップする
- 1 つ以上の参照制約属性またはチェック制約属性を変更する
- フェデレーテッド・サーバーでのデータのキャッシングを変更する

呼び出し

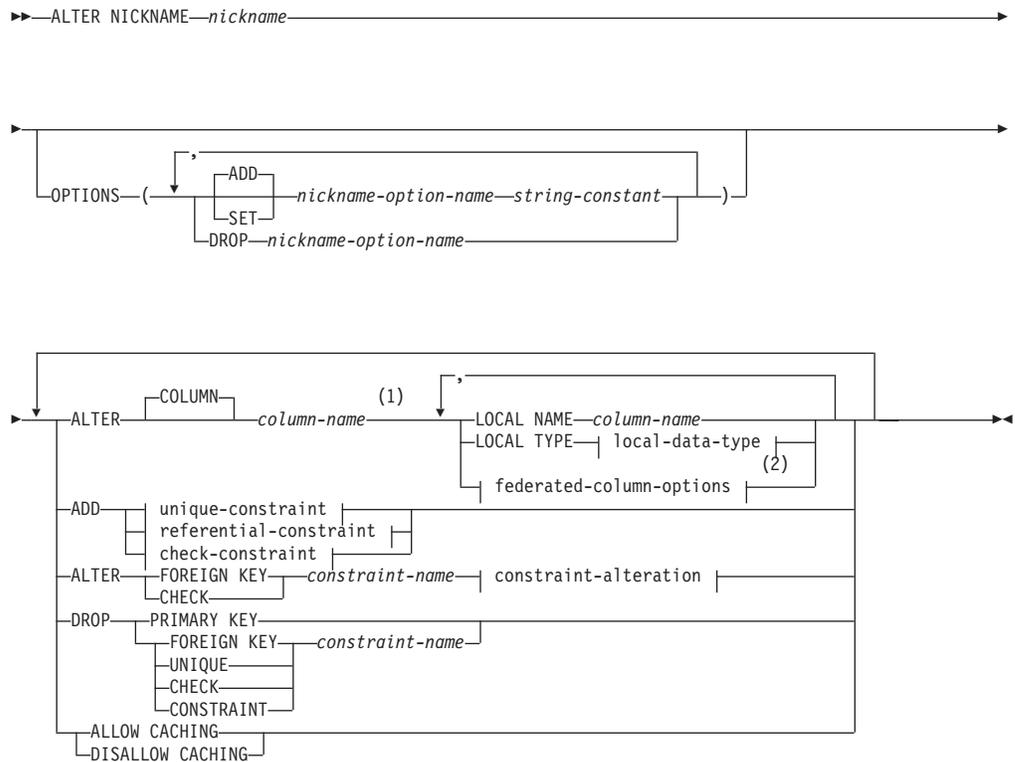
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

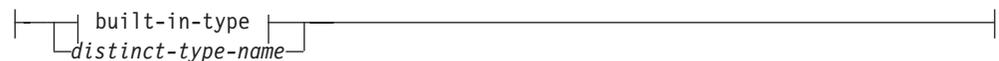
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- ステートメントで指定したニックネームに対する ALTER 特権
- ステートメントで指定したニックネームに対する CONTROL 特権
- スキーマに対する ALTERIN 特権 (ニックネームのスキーマ名が存在する場合)。
- SYSCAT.TABLES カタログ・ビューの OWNER 列に記録されているそのニックネームの所有者
- SYSADM または DBADM 権限

構文

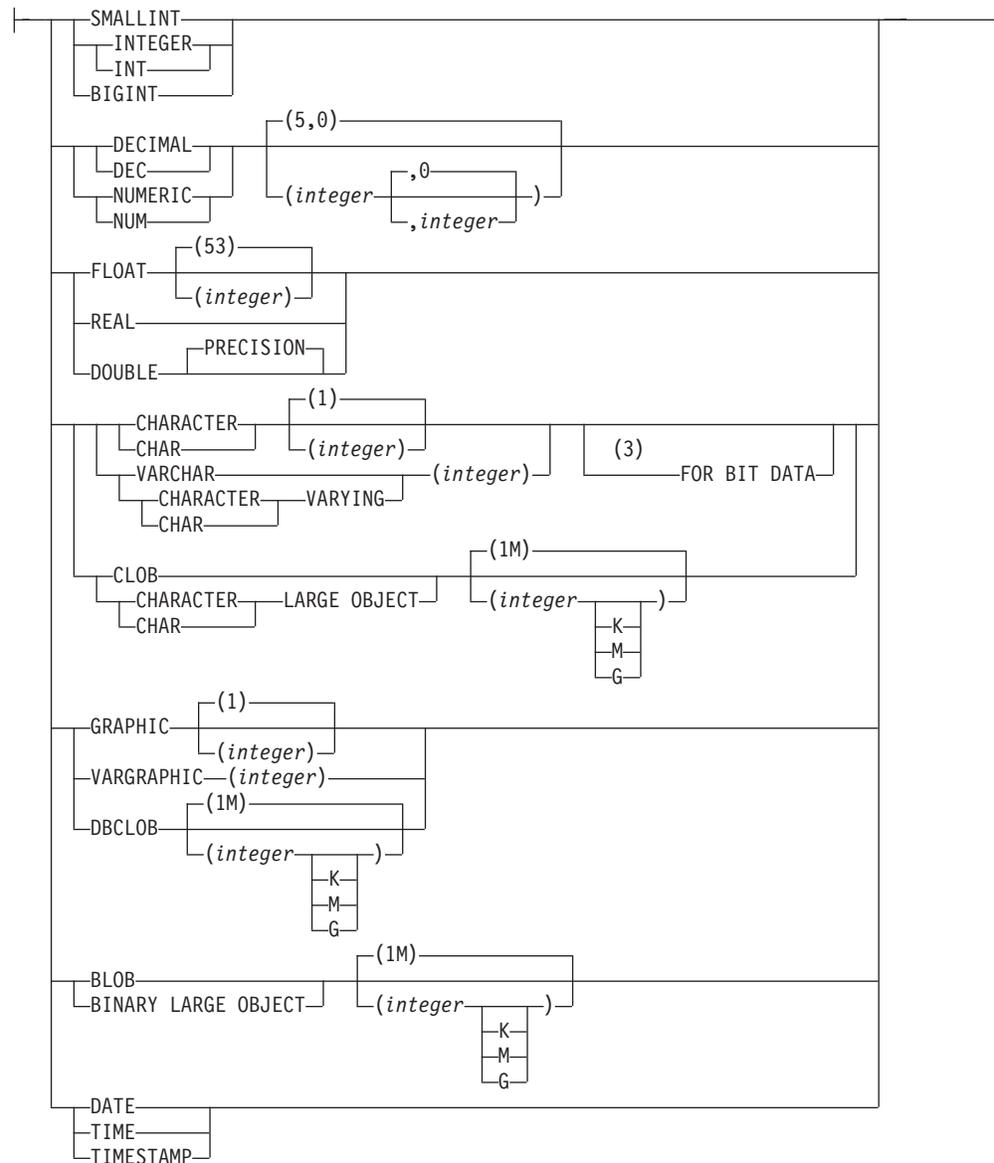


local-data-type:

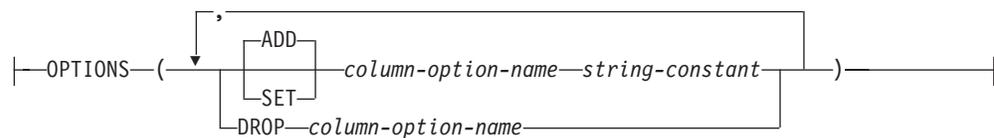


ALTER NICKNAME

built-in-type:



federated-column-options:



unique-constraint:



▶ constraint-attributes |-----|

referential-constraint:

|-----| FOREIGN KEY ((column-name))
 |-----|
 |-----| CONSTRAINT constraint-name |-----|

▶ references-clause |-----|

references-clause:

|-----| REFERENCES table-name
 |-----|
 |-----| nickname
 |-----| ((column-name))

▶ constraint-attributes |-----|

check-constraint:

|-----| CHECK (| check-condition |)
 |-----|
 |-----| CONSTRAINT constraint-name |-----|

▶ constraint-attributes |-----|

check-condition:

|-----| search-condition
 |-----| functional-dependency

functional-dependency:

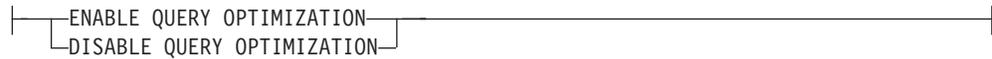
|-----| column-name DETERMINED BY column-name
 |-----|
 |-----| ((column-name)) |-----|
 |-----| ((column-name)) |-----|

constraint-attributes:

| ● NOT ENFORCED ● |-----|
 |-----| ENABLE QUERY OPTIMIZATION
 |-----| (4)
 |-----| DISABLE QUERY OPTIMIZATION

ALTER NICKNAME

constraint-alteration:



注:

- 1 ALTER COLUMN 節と、ADD、ALTER、または DROP インフォメーション制約節との両方を同じ ALTER NICKNAME ステートメント内に指定することはできません。
- 2 LOCAL NAME パラメーターまたは LOCAL TYPE パラメーター、あるいはその両方にフェデレーテッド列オプション (federated-column-options) 節を指定する必要がある場合、この federated-column-options 節は最後に指定しなければなりません。
- 3 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- 4 DISABLE QUERY OPTIMIZATION はユニーク制約または主キー制約をサポートしていません。

説明

nickname

変更される列を含むデータ・ソース・オブジェクト (表、ビュー、またはファイルなど) のニックネームを指定します。ニックネームはカタログに記述されている必要があります。

OPTIONS

ニックネームを変更したときに追加、設定、またはドロップされるニックネーム・オプションを指示します。

ADD

ニックネーム・オプションを追加します。

SET

ニックネーム・オプションの設定を変更します。

nickname-option-name

追加または設定するニックネーム・オプションを指定します。

string-constant

nickname-option-name の設定を、文字ストリング定数として指定します。

DROP *nickname-option-name*

ニックネーム・オプションをドロップします。

ALTER COLUMN *column-name*

変更する列を指定します。 *column-name* は、フェデレーテッド・サーバーでのデータ・ソースにある表またはビューの列の、現在の名前です。 *column-name* は、ニックネームの既存の列を指定するものでなければなりません (SQLSTATE 42703)。同一の ALTER NICKNAME ステートメントで、同じ列名を複数回は参照できません (SQLSTATE 42711)。

LOCAL NAME *column-name*

変更された列を参照するフェデレーテッド・サーバーによって指定される新しい

名前は *column-name* です。新しい名前を修飾したり、ニックネームの複数の列に対して同じ名前を使用することはできません (SQLSTATE 42711)。

LOCAL TYPE *local-data-type*

変更する列のデータ・タイプをマップする、新しいローカル・データ・タイプを指定します。新しいタイプは *local-data-type* に示されます。

一部のラッパーは、SQL データ・タイプのサブセットのみをサポートします。特定のデータ・タイプの説明は、CREATE TABLE ステートメントの説明を参照してください。

OPTIONS

COLUMN キーワードの後に指定した列のどの列オプションを追加、設定、またはドロップするか指定します。

ADD

列オプションを追加します。

SET

列オプションの設定を変更します。

column-option-name

追加または設定する列オプションを指定します。

string-constant

column-option-name の設定を、文字ストリング定数として指定します。

DROP *column-option-name*

列オプションをドロップします。

ADD *unique-constraint*

ユニーク制約を定義します。『CREATE NICKNAME』ステートメントの説明を参照してください。

ADD *referential-constraint*

参照制約を定義します。『CREATE NICKNAME』ステートメントの説明を参照してください。

ADD *check-constraint*

チェック制約を定義します。『CREATE NICKNAME』ステートメントの説明を参照してください。

ALTER FOREIGN KEY *constraint-name*

参照制約 *constraint-name* の制約属性を変更します。制約属性の説明については、「CREATE NICKNAME」ステートメントを参照してください。

constraint-name は既存の参照制約を指定する必要があります (SQLSTATE 42704)。

ALTER CHECK *constraint-name*

チェック制約 *constraint-name* の制約属性を変更します。 *constraint-name* には既存のチェック制約を指定する必要があります (SQLSTATE 42704)。

constraint-alteration

参照制約またはチェック制約に関連付けられた属性の変更のオプションを指定します。

ENABLE QUERY OPTIMIZATION

適切な状況下では、制約を照会最適化に使用することができます。

ALTER NICKNAME

DISABLE QUERY OPTIMIZATION

制約を照会の最適化に使用できません。

DROP PRIMARY KEY

主キーの定義、およびその主キーに従属するすべての参照制約をドロップします。ニックネームには主キーがなければなりません。

DROP FOREIGN KEY *constraint-name*

制約名が *constraint-name* の参照制約をドロップします。 *constraint-name* には、ニックネームに定義されている既存の参照制約を指定する必要があります。

DROP UNIQUE *constraint-name*

ユニーク制約 *constraint-name* の定義、およびこのユニーク制約に従属するすべての参照制約をドロップします。 *constraint-name* には、既存のユニーク制約を指定する必要があります。

DROP CHECK *constraint-name*

制約名が *constraint-name* のチェック制約をドロップします。 *constraint-name* には、ニックネームに定義されている既存のチェック制約を指定する必要があります。

DROP CONSTRAINT *constraint-name*

制約名が *constraint-name* の制約をドロップします。 *constraint-name* には、ニックネームに定義されている既存のチェック制約、参照制約、主キー、またはユニーク制約のいずれかを指定する必要があります。

ALLOW CACHING または DISALLOW CACHING

フェデレーテッド・サーバーで、このニックネームのデータをキャッシュに入れることができるかどうかを指定します。

ALLOW CACHING

フェデレーテッド・サーバーで、このニックネームのデータをキャッシュに入れることができるということを指定します。

DISALLOW CACHING

フェデレーテッド・サーバーで、このニックネームのデータをキャッシュに入れることができないということを指定します。マテリアライズ照会表の定義は、キャッシングを許可しないニックネームを参照できません。さらに、この節は、マテリアライズ照会表の定義の全選択で参照されるニックネームに対して指定できません (SQLSTATE 42917)。

規則

- ニックネームがビュー、SQL メソッド、または SQL 関数に使用されている場合、またはそれらにインフォメーション制約が定義されている場合、そのニックネーム内の列のローカル名やデータ・タイプを変更するために、ALTER NICKNAME ステートメントを使用することはできません (SQLSTATE 42893)。しかし、列オプション、ニックネーム・オプション、または通知制約を追加、設定、またはドロップするときには、このステートメントを使えます。
- ニックネームがマテリアライズ照会表定義によって参照されている場合、ローカル名、データ・タイプ、列オプション、またはニックネーム・オプションを変更するために、ALTER NICKNAME ステートメントを使用することはできません (SQLSTATE 42893)。さらに、キャッシングを使用不可にするためにこのステート

メントを使用することはできません (SQLSTATE 42917)。しかし、通知制約を追加、変更、またはドロップするときには、このステートメントを使えます。

- 列オプションは、同じ ALTER NICKNAME ステートメントに複数回指定することはできません (SQLSTATE 42853)。列オプションを使用可能にする、リセットする、あるいはドロップする場合、使用中の他の列オプションには影響はありません。
- リレーショナル・ニックネームの場合、所定の作業単位 (UOW) 内の ALTER NICKNAME ステートメントは、以下のいずれかの条件の下では処理できません (SQLSTATE 55007)。
 - このステートメントで参照されているニックネームには、同じ UOW 内でオープンされているカーソルがある。
 - このステートメントで参照されているニックネームに対して、同じ UOW 内ですでに INSERT、DELETE、または UPDATE ステートメントのいずれかが出されている。
- 非リレーショナル・ニックネームの場合、所定の作業単位 (UOW) 内の ALTER NICKNAME ステートメントは、以下の条件の下では処理できません (SQLSTATE 55007)。
 - このステートメントで参照されているニックネームには、同じ UOW 内でオープンされているカーソルがある。
 - このステートメントで参照されているニックネームは、同じ UOW 内の SELECT ステートメントですすでに参照されている。
 - このステートメントで参照されているニックネームに対して、同じ UOW 内ですでに INSERT、DELETE、または UPDATE ステートメントのいずれかが出されている。

注

- ALTER NICKNAME ステートメントを使用してニックネームの列のローカル名を変更する場合、その列に対する照会ではその名前を新しい名前参照する必要があります。
- 列のデータ・タイプのローカル仕様を変更すると、データベース・マネージャーは、その列に関して収集されたすべての統計 (HIGH2KEY、LOW2KEY、など) を無効にします。
- データ・ソース・オブジェクトが保護されたニックネームには DISALLOW CACHING 節を指定してください。これにより、ニックネームが使用されるたび、照会の実行時に適切な許可 ID のデータがデータ・ソースから戻されるようになります。

例

例 1: ニックネーム NICK1 は、T1 という DB2 for System i 表を参照します。また、COL1 はこの表の最初の列である C1 を示すローカル名です。C1 のローカル名を COL1 から NEWCOL に変更します。

```
ALTER NICKNAME NICK1
ALTER COLUMN COL1
LOCAL NAME NEWCOL
```

ALTER NICKNAME

例 2: ニックネーム EMPLOYEE は、EMP という DB2 for z/OS 表を参照します。また、SALARY はこの表の列の 1 つである、EMP_SAL を示すローカル名です。列のデータ・タイプ FLOAT は、ローカル・データ・タイプ DOUBLE にマップされます。FLOAT が DECIMAL (10, 5) にマップされるように、マッピングを変更します。

```
ALTER NICKNAME EMPLOYEE
ALTER COLUMN SALARY
LOCAL TYPE DECIMAL(10,5)
```

例 3: Oracle 表において、データ・タイプが VARCHAR の列には、後書きブランクがないことを示します。この表のニックネームは NICK2 で、この列のローカル名は COL1 です。

```
ALTER NICKNAME NICK2
ALTER COLUMN COL1
OPTIONS (ADD VARCHAR_NO_TRAILING_BLANKS 'Y')
```

例 4: ニックネーム DRUGDATA1 の表構造化ファイル drugdata1.txt の完全修飾パスを変更します。FILE_PATH ニックネーム・オプションを使用して、パスを現行値 '/user/pat/drugdata1.txt' から '/usr/kelly/data/drugdata1.txt' に変更します。

```
ALTER NICKNAME DRUGDATA1
OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

ALTER PROCEDURE (外部)

ALTER PROCEDURE (外部) ステートメントは、プロシージャのプロパティを変更して、既存の外部プロシージャを変更します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- プロシージャのスキーマに対する ALTERIN 特権
- SYSCAT.ROUTINES カタログ・ビューの OWNER 列に記録されているそのプロシージャの所有者
- SYSADM または DBADM 権限

プロシージャの EXTERNAL NAME を変更するには、ステートメントの許可 ID の特権に、以下の特権の少なくとも 1 つが含まれている必要があります。

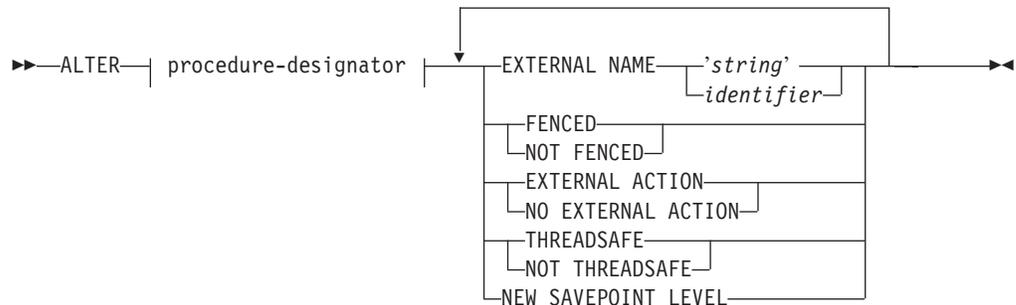
- データベースに対する CREATE_EXTERNAL_ROUTINE 権限
- SYSADM または DBADM 権限

fenced でないようにプロシージャを変更するには、ステートメントの許可 ID の特権に以下の特権の少なくとも 1 つが含まれている必要があります。

- データベースに対する CREATE_NOT_FENCED_ROUTINE 権限
- SYSADM または DBADM 権限

fenced であるようにプロシージャを変更するには、さらに別の権限や特権は必要ありません。

構文



説明

procedure-designator

変更するプロシージャを指定します。 *procedure-designator* は、現行サーバーに存在するプロシージャを示していなければなりません。プロシージャの所有者およびそのプロシージャに対するすべての特権は保存されます。詳しくは、16 ページの『関数、メソッド、およびプロシージャの指定子』を参照してください。

EXTERNAL NAME 'string' または identifier

プロシージャをインプリメントするユーザー作成コードの名前を指定します。

FENCED または NOT FENCED

プロシージャをデータベース・マネージャーのオペレーティング環境のプロセスまたはアドレス・スペースで実行しても安全か (NOT FENCED)、そうでないか (FENCED) を指定します。多くのプロシージャは、FENCED として実行するか NOT FENCED として実行するかの選択が可能です。

プロシージャが FENCED に変更されると、データベース・マネージャーは、その内部リソース (データ・バッファなど) を fenced して、そのプロシージャからアクセスされないようにします。一般に、FENCED として実行されるプロシージャは、NOT FENCED として実行されるものと同じようには実行されません。

注意:

適切にコード化、検討、および検査されていないプロシージャに対して NOT FENCED を使用すると、DB2 の健全性を損なう場合があります。DB2 では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられていますが、NOT FENCED ストアド・プロシージャが使用される場合には、完全な整合性を確保できません。

ALTER PROCEDURE (外部)

NOT THREADSAFE を宣言したプロシージャは、NOT FENCED には変更できません (SQLSTATE 42613)。

プロシージャが AS LOCATOR を定義した任意のパラメーターを有していて、NO SQL オプションも指定されている場合には、このプロシージャは FENCED には変更できません (SQLSTATE 42613)。

LANGUAGE OLE プロシージャまたは CLR プロシージャでは、このオプションを変更できません (SQLSTATE 42849)。

EXTERNAL ACTION または NO EXTERNAL ACTION

プロシージャが、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るか (EXTERNAL ACTION)、または取らないか (NO EXTERNAL ACTION) を指定します。NO EXTERNAL ACTION を指定した場合、プロシージャが外部に影響を与えないことを前提とした最適化を、システムは使用できます。

THREADSAFE または NOT THREADSAFE

プロシージャを他のルーチンと同じプロセスで実行しても安全か (THREADSAFE)、そうでないか (NOT THREADSAFE) を指定します。

プロシージャが OLE 以外の LANGUAGE で定義される場合:

- プロシージャが THREADSAFE として定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスにプロシージャを呼び出すことができます。一般に、スレッド・セーフになるには、プロシージャはどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。FENCED および NOT FENCED プロシージャの両方を THREADSAFE にすることができます。
- プロシージャが NOT THREADSAFE に定義される場合には、データベース・マネージャーは他のルーチンと同じプロセスにプロシージャを決して呼び出しません。fenced されたプロシージャだけが、NOT THREADSAFE になり得ます (SQLSTATE 42613)。

このオプションは LANGUAGE OLE プロシージャを変更できません (SQLSTATE 42849)。

NEW SAVEPOINT LEVEL

新規セーブポイント・レベルをプロシージャに対して作成することを指定します。セーブポイント・レベルは、任意のセーブポイント関連ステートメントの参照範囲と、セーブポイント名の比較および参照に使用されるネーム・スペースを参照します。

プロシージャのプロシージャ・レベルは NEW SAVEPOINT LEVEL にのみ変更可能です。

規則

- SYSIBM、SYSFUN、または SYSPROC スキーマ (SQLSTATE 42832) のプロシージャは変更できません。

例

プロシージャ `PARTS_ON_HAND()` が `fenced` でないように変更します。

```
ALTER PROCEDURE PARTS_ON_HAND() NOT FENCED
```

ALTER PROCEDURE (ソース派生)

`ALTER PROCEDURE` (ソース派生) ステートメントは、ソース・プロシージャの 1 つ以上のパラメーターのデータ・タイプを変更して、既存のソース・プロシージャを変更します。

呼び出し

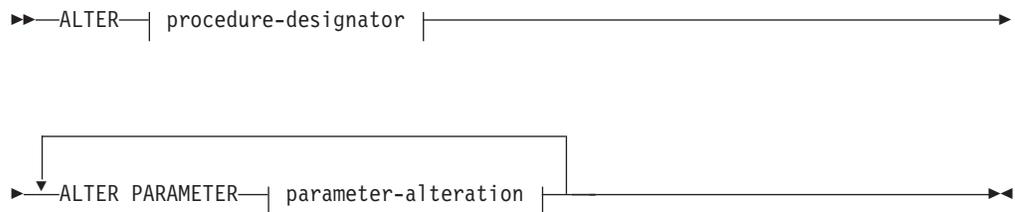
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 `DYNAMICRULES` の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- プロシージャのスキーマに対する `ALTERIN` 特権
- `SYSCAT.ROUTINES` カタログ・ビューの `OWNER` 列に記録されているそのプロシージャの所有者
- `SYSADM` または `DBADM` 権限

構文



parameter-alteration:

```
|—parameter-name—SET DATA TYPE—data-type—|
```

説明

procedure-designator

変更されるプロシージャを固有識別します。識別されるプロシージャは、ソース・プロシージャでなければなりません (SQLSTATE 42849)。詳しくは、16 ページの『関数、メソッド、およびプロシージャの指定子』を参照してください。

parameter-name

変更するパラメーター値を識別します。 *parameter-name* は、プロシージャの

ALTER PROCEDURE (ソース派生)

既存のパラメーターを指定しなければなりません (SQLSTATE 42703)。この名前には、同じ ALTER PROCEDURE ステートメントで変更されるパラメーターは指定できません (SQLSTATE 42713)。

data-type

パラメーターの新規ローカル・データ・タイプを指定します。CREATE TABLE ステートメントの *data-type* の定義で有効な SQL データ・タイプ仕様と省略形を指定することができます。BLOB、CLOB、DBCLOB、DECFLOAT、XML、REFERENCE、およびユーザー定義タイプはサポートされません (SQLSTATE 42815)。

例

フェデレーテッド・プロシージャー FEDEMPLOYEE が、'EMPLOYEE' というリモート Oracle プロシージャーに対して作成されていると想定します。SALARY という入力パラメーターのデータ・タイプは、DB2 では DOUBLE(8) にマップされます。このパラメーターのデータ・タイプを DECIMAL(5,2) に変更します。

```
ALTER PROCEDURE FEDEMPLOYEE
ALTER PARAMETER SALARY
SET DATA TYPE DECIMAL(5,2)
```

ALTER PROCEDURE (SQL)

ALTER PROCEDURE (SQL) ステートメントは、プロシージャーのプロパティを変更して、既存の SQL プロシージャーを変更します。

呼び出し

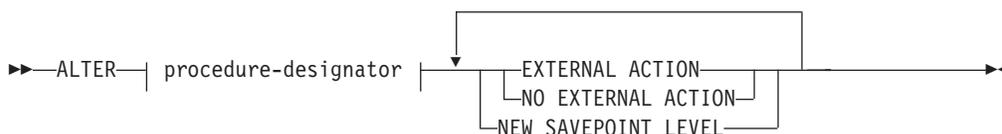
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- プロシージャーのスキーマに対する ALTERIN 特権
- SYSCAT.ROUTINES カタログ・ビューの OWNER 列に記録されているそのプロシージャーの所有者
- SYSADM または DBADM 権限

構文



説明

procedure-designator

変更するプロシージャを指定します。 *procedure-designator* は、現行サーバーに存在するプロシージャを示していなければなりません。プロシージャの所有者およびそのプロシージャに対するすべての特権は保存されます。詳しくは、16 ページの『関数、メソッド、およびプロシージャの指定子』を参照してください。

EXTERNAL ACTION または NO EXTERNAL ACTION

プロシージャが、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るか (EXTERNAL ACTION)、または取らないか (NO EXTERNAL ACTION) を指定します。 NO EXTERNAL ACTION を指定した場合、プロシージャが外部に影響を与えないことを前提とした最適化を、システムは使用できます。

NEW SAVEPOINT LEVEL

新規セーブポイント・レベルをプロシージャに対して作成することを指定します。セーブポイント・レベルは、任意のセーブポイント関連ステートメントの参照範囲と、セーブポイント名の比較および参照に使用されるネーム・スペースを参照します。

プロシージャのプロシージャ・レベルは NEW SAVEPOINT LEVEL にのみ変更可能です。

規則

- SYSIBM、SYSFUN、または SYSPROC スキーマ (SQLSTATE 42832) のプロシージャは変更できません。

例

MEDIAN_RESULT_SET プロシージャが外部アクションを持たないことを示すように変更します。

```
ALTER PROCEDURE MEDIAN_RESULT_SET(DOUBLE)
NO EXTERNAL ACTION
```

ALTER SECURITY LABEL COMPONENT

ALTER SECURITY LABEL COMPONENT ステートメントは、セキュリティー・レベル・コンポーネントを変更します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

ALTER SECURITY LABEL COMPONENT

構文

▶▶ ALTER SECURITY LABEL COMPONENT *component-name* | add-element-clause |▶▶▶▶

add-element-clause:

| ADD ELEMENT *string-constant* |
| array-element-clause |
| tree-element-clause |

array-element-clause:

| BEFORE *string-constant* |
| AFTER *string-constant* |

tree-element-clause:

| ROOT *string-constant* |
| UNDER *string-constant* |
| OVER *string-constant* |

説明

component-name

変更されるセキュリティー・ラベル・コンポーネントの名前を指定します。名前の指定されたコンポーネントは、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

ADD ELEMENT

セキュリティー・ラベル・コンポーネントに追加されるエレメントを指定します。 *array-element-clause* および *tree-element-clause* が指定されない場合、エレメントはセット・コンポーネントに追加されます。

string-constant

セキュリティー・ラベル・コンポーネント用の一連の有効値に追加されるストリング定数値。この値は、セキュリティー・ラベル・コンポーネント用の一連の有効値に含まれる他のいずれかの値と同じであることはできません (SQLSTATE 42713)。

BEFORE または AFTER

配列コンポーネントでは、セキュリティー・ラベル・コンポーネント用のエレメント値の順序付きセット内で、エレメントが追加される位置を指定します。

BEFORE

追加されるエレメントは、識別される既存のエレメントの直前にランク付けされます。

AFTER

追加されるエレメントは、識別される既存のエレメントの直後にランク付けされます。

string-constant

配列コンポーネント内にある既存の要素のストリング定数値を指定します (SQLSTATE 42704)。

ROOT または UNDER

ツリー・コンポーネントでは、セキュリティー・ラベル・コンポーネント用のノード・要素値のツリー構造内で、要素が追加される位置を指定します。

ROOT

追加される要素は、ツリーのルート・ノードであるとみなされます。

UNDER *string-constant*

追加される要素は、*string-constant* で識別される要素の直接の子です。*string-constant* 値は、ツリー・コンポーネント内に存在する要素でなければなりません (SQLSTATE 42704)。

OVER *string-constant,...*

追加される要素は、*string-constant* 値のリストで識別されるすべての要素の直接の子です。それぞれの *string-constant* 値は、ツリー・コンポーネント内に存在する要素でなければなりません (SQLSTATE 42704)。

規則

- エLEMENTの名前中で以下のいずれの文字も使用できません (SQLSTATE 42601):
 - 左括弧 - (
 - 右括弧 -)
 - コンマ - ,
 - コロン - :
- ELEMENT名は、32 バイト以下でなければなりません (SQLSTATE 42622)。
- セキュリティー・ラベル・コンポーネントがセットまたはツリーである場合、最大 64 個のELEMENTをそのコンポーネントの一部とすることができます。
- コンポーネントが配列である場合、タイプが配列のセキュリティー・ラベル・コンポーネントを作成するときに指定できるELEMENTの総数 (65 535) と、ELEMENTの総数が一致する配列に到達することもできないこともあります。DB2は、新しいELEMENTが追加されるインターバル内から、エンコード値を新しいELEMENTに割り当てます。ELEMENTを配列コンポーネントに追加するときに従うパターンに応じて、特定のインターバル内で割り当て可能な値の数は、複数のELEMENTがそのインターバル内に挿入された場合にはすぐに使い果たされることがあります。
- BEFORE および AFTER を指定するのは、配列のセキュリティー・ラベル・コンポーネントだけです (SQLSTATE 42613)。
- ROOT および UNDER を指定するのは、ツリーのセキュリティー・ラベル・コンポーネントだけです (SQLSTATE 42613)。

注

- セット・コンポーネントでは、セット内のELEMENTに順序はありません。

ALTER SECURITY LABEL COMPONENT

例

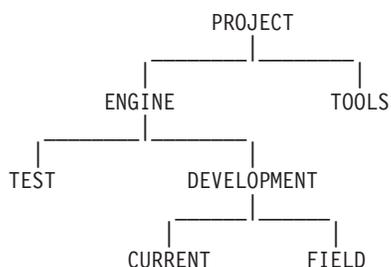
例 1: エlement「High classified」を、Element「Secret」および「Classified」の間の LEVEL セキュリティー・ラベル配列コンポーネントに追加します。

```
ALTER SECURITY LABEL COMPONENT LEVEL
ADD ELEMENT 'High classified' BEFORE 'Classified'
```

例 2: Element「Funding」を COMPARTMENTS セキュリティー・ラベルのセット・コンポーネントに追加します。

```
ALTER SECURITY LABEL COMPONENT COMPARTMENTS
ADD ELEMENT 'Funding'
```

例 3: Element「ENGINE」および「TOOLS」を、GROUPS セキュリティー・ラベルの配列コンポーネントに追加します。以下の図は、これらの新しいElementが配置される場所を示しています。



```
ALTER SECURITY LABEL COMPONENT GROUPS
ADD ELEMENT 'TOOLS' UNDER 'PROJECT'
```

```
ALTER SECURITY LABEL COMPONENT GROUPS
ADD ELEMENT 'ENGINE' UNDER 'PROJECT'
OVER 'TEST', 'DEVELOPMENT'
```

ALTER SECURITY POLICY

ALTER SECURITY POLICY ステートメントは、セキュリティー・ポリシーを変更します。

呼び出し

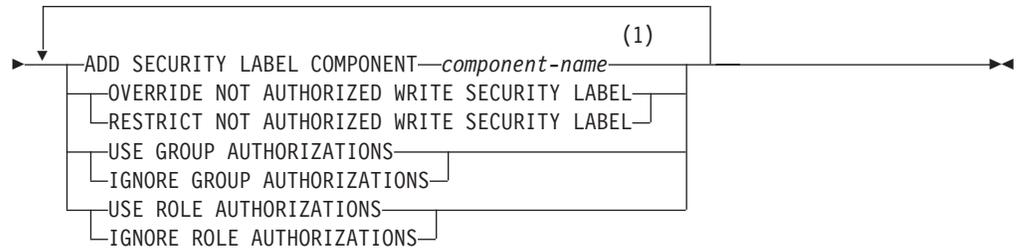
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

```
▶—ALTER SECURITY POLICY—security-policy-name—▶
```



注:

- 1 複数回指定できるのは ADD SECURITY LABEL COMPONENT 節だけです。

説明

security-policy-name

変更されるセキュリティ・ポリシーの名前を指定します。名前は、現行のサーバー上の既存のセキュリティ・ポリシーを識別するものでなければなりません (SQLSTATE 42710)。

ADD SECURITY LABEL COMPONENT *component-name*

セキュリティ・ラベル・コンポーネントをセキュリティ・ポリシーに追加します。セキュリティ・ポリシーに対して同じセキュリティ・コンポーネントを複数回指定してはなりません (SQLSTATE 42713)。セキュリティ・ポリシーが現在、表によって使用されている可能性はありません (SQLSTATE 42893)。

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL または **RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL**

このセキュリティ・ポリシーで保護されている表に対して発行された INSERT または UPDATE ステートメント内に明示的に指定されているセキュリティ・ラベルを書き込む許可をユーザーが持たない場合に取り処置を指定します。ユーザーのセキュリティ・ラベルおよび免除クレデンシャルによって、明示的に指定されたセキュリティ・ラベルを書き込むユーザー許可が判別されます。

OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

挿入または更新の操作での書き込みアクセスに対して、明示的に指定されているセキュリティ・ラベルではなく、ユーザーのセキュリティ・ラベルの値を使用することを指定します。

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

INSERT または UPDATE ステートメント内に置かれている明示的に指定されたセキュリティ・ラベルの書き込みをユーザーが許可されていない場合、挿入または更新の操作が失敗することを指示します (SQLSTATE 42519)。

USE GROUP AUTHORIZATION または **IGNORE GROUP AUTHORIZATION**

直接または間接的にグループに付与されたセキュリティ・ラベルおよび免除が、いずれかのアクセスを試行する際に考慮されるかどうかを指定します。

USE GROUP AUTHORIZATION

直接または間接的にグループに付与されたセキュリティ・ラベルまたは免除が、考慮されることを指定します。

IGNORE GROUP AUTHORIZATION

グループに付与されたセキュリティ・ラベルまたは免除が、考慮されないことを指定します。

USE ROLE AUTHORIZATION または IGNORE ROLE AUTHORIZATION

直接または間接的にロールに付与されたセキュリティ・ラベルおよび免除が、いずれかのアクセスを試行する際に考慮されるかどうかを指定します。

USE ROLE AUTHORIZATION

直接または間接的にロールに付与されたセキュリティ・ラベルまたは免除が、考慮されることを指定します。

IGNORE ROLE AUTHORIZATION

ロールに付与されたセキュリティ・ラベルまたは免除が、考慮されないことを指定します。

規則

- ユーザーが書き込みアクセス用のセキュリティ・ラベルを直接保持していない場合、以下の状態のときにエラーが戻されます (SQLSTATE 42519):
 - 行セキュリティ・ラベル列の値が、SQL ステートメントの一部として明示的に提供されていない
 - **OVERVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL** オプションがセキュリティ・ラベルに対して有効であり、ユーザーは設定されたセキュリティ・ポリシー・レベルではデータ・オブジェクトを書き込むことが許可されない

注

- 新規コンポーネントは、変更されたポリシーによって組み込まれた既存のセキュリティ・ラベル定義の最後に論理的に追加されます。このセキュリティ・ポリシー用に定義された既存のセキュリティ・ラベルは、定義の一部として新規コンポーネントを組み込むように、そしてこのコンポーネントの値にはエレメントを含まないように変更されます。
- **NOT AUTHORIZED WRITE SECURITY LABEL** を変更するときのキャッシュの無効化: **NOT AUTHORIZED WRITE SECURITY LABEL** を新しい値に変更すると、変更中のセキュリティ・ポリシーによって保護された表に依存する、キャッシュに入れられた動的または静的な SQL ステートメントが無効になります。
- **SESSION** 許可 ID はラベル・ベースのアクセス制御のフォーカス許可 ID なので、グループまたはグループを介してアクセス可能なロールに付与されたセキュリティ・ラベルは、静的 SQL を含むすべてのタイプの SQL ステートメントに関して考慮されることに適格となります。
- 読み取りまたは書き込みアクセスの試行時に関連付けられたグループまたはロールを持つユーザーに対して複数のセキュリティ・ラベルまたは免除が使用可能な場合、それらのセキュリティ・ラベルおよび免除は以下の規則に従って適格性が評価されます。
 - セキュリティ・ポリシーがロール権限だけを考慮の対象として使用可能にする場合、ユーザー許可 ID が直接または間接のメンバーであるロールに付与されたすべてのセキュリティ・ラベルおよび免除は考慮されます。ユーザー許

可 ID に関連付けられたグループを介してのみメンバーシップにアクセス可能なロールに付与されたセキュリティー・ラベルおよび免除は、考慮されません。

- セキュリティー・ポリシーがグループ権限だけを考慮の対象として使用可能にする場合、ユーザー許可 ID がグループに付与されたすべてのセキュリティー・ラベルおよび免除は考慮されます。ユーザー許可 ID に関連付けられたグループを介してのみメンバーシップにアクセス可能なロールに付与されたセキュリティー・ラベルおよび免除は、考慮されません。
- セキュリティー・ポリシーがグループ権限およびロール権限の両方を考慮の対象として使用可能にする場合、ユーザー許可 ID に関連付けられたグループを介して間接的にユーザーにアクセス可能なロールに付与されたセキュリティー・ラベルおよび免除は考慮されます。
- PUBLIC を介してのみユーザーにアクセス可能なロール権限は、常に考慮されません。
- アクセスを試行するときに複数のセキュリティー・ラベルが考慮の対象として適格である場合、各セキュリティー・ラベルに提供された値は個別のコンポーネント・レベルでマージされて、セキュリティー・ポリシーの各コンポーネントの部分で使用可能なすべての値の組み合わせを反映するセキュリティー・ラベルが形成されます。これは、アクセスの試行に使用されるセキュリティー・ラベル値です。

セキュリティー・ラベルを結合する仕組みは、コンポーネント・タイプごとに異なります。結果として生じるセキュリティー・ラベルのコンポーネントは、以下のとおりです。

- セット・コンポーネントには、適格なセキュリティー・ラベル内で検出されたすべての固有値の共用体が含まれています。
- 配列コンポーネントには、適格なセキュリティー・ラベル内で検出された最高順位のエレメントが含まれています。
- ツリー・コンポーネントには、適格なセキュリティー・ラベル内で検出されたすべての固有値の共用体が含まれています。
- アクセスの試行時に複数の免除が考慮の対象として適格である場合、検索されたすべての免除はアクセスの試行に適用されます。

例

例 1: DATA_ACCESS という名前のセキュリティー・ポリシーを変更して、REGION という名前の新規コンポーネントを追加します。

```
ALTER SECURITY POLICY DATA_ACCESS
ADD COMPONENT REGION
```

例 2: DATA_ACCESS という名前のセキュリティー・ポリシーを変更して、ロールに付与されたセキュリティー・ラベルを介するアクセスを許可します。

```
ALTER SECURITY POLICY DATA_ACCESS
USE ROLE AUTHORIZATIONS
```

例 3: セキュリティー・ポリシー内のグループ権限またはロール権限の設定値に応じて考慮される、適格なセキュリティー・ラベルを表示します。セキュリティー・

ALTER SECURITY POLICY

ポリシー SECUR_POL には、以下のエレメントから構成される、配列コンポーネントおよびセット・コンポーネントがあります。

Array = {TS, S, C, U}

Set = {A, B, X, Y}

以下のセキュリティー・ラベルが SECUR_POL 用に定義されます。

Security label L1 = C:A

Security label L2 = S:B

Security label L3 = TS:X

Security label L4 = U:Y

ユーザー Paul は、ロール R1 およびグループ G1 のメンバーです。グループ G1 は、ロール R2 のメンバーです。セキュリティー・ラベル L1 は、Paul に付与されます。セキュリティー・ラベル L2 は、ロール R1 に付与されます。セキュリティー・ラベル L3 は、グループ G1 に付与されます。セキュリティー・ラベル L4 は、ロール R2 に付与されます。以下の表は、セキュリティー・ポリシー SECUR_POL のさまざまな可能な設定値に応じて、Paul がアクセスを試行するときどのセキュリティー・ラベルが考慮されるかを示しています。

表 11. セキュリティー・ポリシー設定の関数として考慮されるセキュリティー・ラベル

	使用可能にされるロール	使用不可にされるロール
使用可能にされるグループ	L1, L2, L3, L4	L1, L3
使用不可にされるグループ	L1, L2	L1

以下の表は、セキュリティー・ポリシー SECUR_POL のさまざまな設定値に応じて、Paul がアクセスを試行するときのセキュリティー・ラベルの結合値を示しています。

表 12. セキュリティー・ポリシー設定の関数として結合されたセキュリティー・ラベル

	使用可能にされるロール	使用不可にされるロール
使用可能にされるグループ	TS:(A, B, X, Y)	TS:(A, X)
使用不可にされるグループ	S:(A, B)	C:A

ALTER SEQUENCE

ALTER SEQUENCE ステートメントを使用して、シーケンスを以下のように変更できます。

- シーケンスを再始動する
- 将来のシーケンス値の間の増分を変更する
- 最小値または最大値を設定または除去する
- キャッシュ済みシーケンス番号の数を変更する
- シーケンスが循環するかどうかを決定する属性を変更する
- 要求の順序でシーケンス番号が生成されるかどうかを変更する

呼び出し

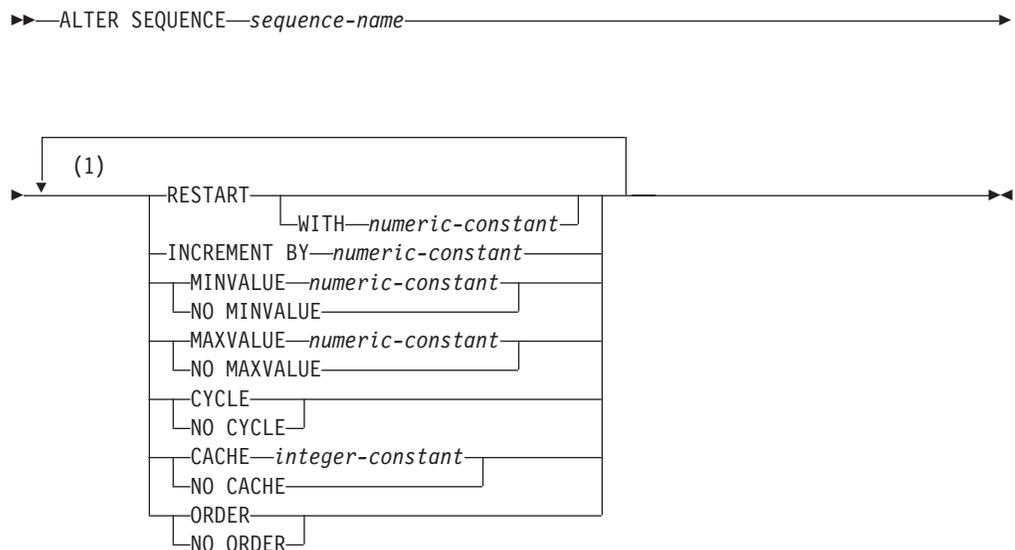
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 変更するシーケンスに対する ALTER 特権
- 暗黙的または明示的に指定されているスキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

構文



注:

- 1 同じ節を複数回指定することはできません。

説明

sequence-name

変更するシーケンスを識別します。この名前 (暗黙的または明示的スキーマ修飾子を含む) は、現行のサーバーに存在するシーケンスを固有に識別しなければなりません。この名前が示すシーケンスが、明示的または暗黙的に指定されたスキーマに存在しない場合、エラー (SQLSTATE 42704) が戻されます。

sequence-name には、システムが ID 列に対して生成したシーケンスを指定することはできません (SQLSTATE 428FB)。

RESTART

シーケンスを再始動します。 *numeric-constant* が指定されていない場合、シーケンスは、そのシーケンスを作成した CREATE SEQUENCE ステートメントに開始値として暗黙的または明示的に指定されている値で再始動されます。

ALTER SEQUENCE

WITH *numeric-constant*

指定した値でシーケンスを再始動します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、シーケンスに関連するデータ・タイプの列に割り当てられる正または負の値にすることができます (SQLSTATE 42815)。

INCREMENT BY *numeric-constant*

連続したシーケンス値のインターバルを指定します。この値として、長精度整数定数の値を超えない範囲で (SQLSTATE 42820)、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます (SQLSTATE 42815)。ただし、小数点の右側に非ゼロの数字が存在してはなりません (SQLSTATE 428FA)。

この値が負の場合、これは降順シーケンスです。この値が 0 の場合、または正の場合は、ALTER ステートメント以降は昇順になります。

MINVALUE または **NO MINVALUE**

降順シーケンスが値の生成を循環または停止する最小値、あるいは最大値に達した後、昇順シーケンスが循環する最小値を指定します。

MINVALUE *numeric-constant*

最小値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、シーケンスに関連するデータ・タイプの列に割り当てられる正または負の値にすることができます (SQLSTATE 42815) が、最大値以下でなければなりません (SQLSTATE 42815)。

NO MINVALUE

昇順シーケンスの場合、値は元の開始値です。降順シーケンスの場合、シーケンスに関連するデータ・タイプの最小値です。

MAXVALUE または **NO MAXVALUE**

昇順シーケンスが値の生成を循環または停止する最大値、あるいは最小値に達した後、降順シーケンスが循環する最大値を指定します。

MAXVALUE *numeric-constant*

最大値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、シーケンスに関連するデータ・タイプの列に割り当てられる正または負の値にすることができます (SQLSTATE 42815) が、最小値以上でなければなりません (SQLSTATE 42815)。

NO MAXVALUE

昇順シーケンスの場合、値はシーケンスに関連するデータ・タイプの最大値です。降順シーケンスの場合、値は最初の開始値です。

CYCLE または **NO CYCLE**

その最大値または最小値に達した後、シーケンスが値の生成を続行するかどうかを指定します。シーケンスが境界に達するのは、次の値が境界条件を正確に満たしたとき、またはその値を超えたときです。

CYCLE

最大値または最小値に達した後、このシーケンスについて値の生成を続行することを指定します。このオプションが使用されると、昇順シーケンスが最

大値に達した後、その最小値が生成されます。降順シーケンスが最小値に達した後、その最大値が生成されます。シーケンスの最大値および最小値は、循環に使用される範囲を決定します。

CYCLE が有効な場合、DB2 が重複するシーケンス値を生成する場合があります。

NO CYCLE

シーケンスの最大値または最小値に達した後、そのシーケンスについて値は生成されないことを指定します。

CACHE または NO CACHE

高速アクセスのため、事前割り振り値のいくつかをメモリーに保管するかどうかを指定します。これはパフォーマンスおよびチューニング・オプションです。

CACHE *integer-constant*

事前割り振りされ、メモリーに保管されるシーケンス値の最大数を指定します。値を事前割り振りしてキャッシュに保管しておくこと、シーケンス値を生成するとき、ログへの同期入出力が少なくなります。

システム障害が起こると、コミットされたステートメントで使用されていないキャッシュ済みシーケンス値はすべて失われます（使用されなくなります）。CACHE オプションに指定する値は、システム障害の際に失われても構わないシーケンス値の最大数です。

最小値は 2 です (SQLSTATE 42815)。

NO CACHE

シーケンスの値が事前割り振りされないよう指定します。システム障害、シャットダウン、またはデータベース非活動化の際、値が失われることはありません。このオプションが指定されると、シーケンスの値はキャッシュに保管されません。この場合、シーケンスの新しい値が要求されるたびに、ログに対して同期入出力が行われます。

ORDER または NO ORDER

要求の順序でシーケンス番号が生成されるかどうかを指定します。

ORDER

要求の順序でシーケンス番号が生成されるよう指定します。

NO ORDER

要求の順序でシーケンス番号を生成する必要がないことを指定します。

注

- 今後のシーケンス番号だけが ALTER SEQUENCE ステートメントによって影響を受けます。
- シーケンスのデータ・タイプは変更できません。代わりに、新しいシーケンスに目的のデータ・タイプを指定して、シーケンスをドロップおよび再作成してください。
- シーケンスが変更されると、キャッシュされている値はすべて失われます。
- シーケンスを再始動、または CYCLE に変更した後、以前にシーケンスによって生成された値と重複するシーケンス番号が生成される可能性があります。
- 互換性

ALTER SEQUENCE

- 以前のバージョンの DB2 との互換性と整合性:
 - コンマは、複数のシーケンス・オプションを分離するのに使用できます。
- 以下の構文もサポートされています。
 - NOMINVALUE、NOMAXVALUE、NOCYCLE、NOCACHE、および NOORDER

例

例 1 : 数値なしで **RESTART** を指定する理由として考えられるのは、シーケンスを **START WITH** 値にリセットすることです。この例では、1 から表の行数までの数値を生成し、一時表を使用して表に追加した列にその数値を挿入しています。以降使用する時には、すべての結果行に番号が付けられて結果が返されます。

```
ALTER SEQUENCE ORG_SEQ RESTART
SELECT NEXT VALUE FOR ORG_SEQ, ORG.* FROM ORG
```

ALTER SERVER

ALTER SERVER ステートメントは、以下の目的で使用されます。

- 特定のデータ・ソースの定義を変更する場合、またはデータ・ソースのカテゴリの定義を変更する場合。
- 特定のデータ・ソースの構成を変更する場合、またはデータ・ソースのカテゴリの構成を変更する場合 (この変更は、フェデレーテッド・データベースへ何回か接続する間、持続します)。

このステートメントでは、**SERVER** という語と、*server-* で始まるパラメーター名は、フェデレーテッド・システムでのデータ・ソースのみを指しています。そのようなシステムでのフェデレーテッド・サーバー、あるいは **DRDA**[®] アプリケーション・サーバーを指すわけではありません。

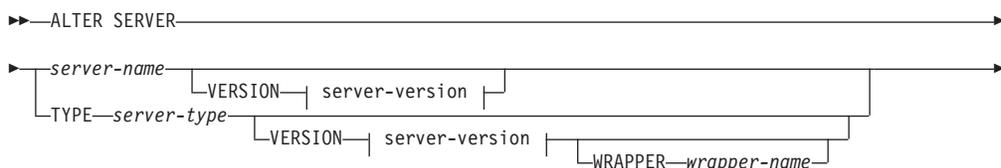
呼び出し

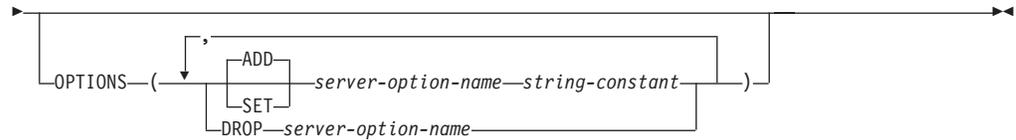
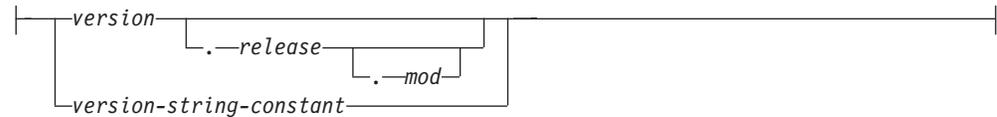
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 **SQL** ステートメントを使用して出すことができます。 **DYNAMICRULES** の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (**SQLSTATE** 42509)。

許可

このステートメントの許可 ID が持つ特権には、**SYSADM** または **DBADM** 権限が含まれている必要があります。

構文



**server-version:****説明***server-name*

要求された変更を適用する対象のデータ・ソースに関連するフェデレーテッド・サーバーの名前を指定します。このデータ・ソースは、カタログに記述されているものでなければなりません。

VERSION

server-name の後にある **VERSION** とそのパラメーターは、*server-name* に示されている新しいバージョンのデータ・ソースを指定します。

version

バージョン番号を指定します。値は整数でなければなりません。

release

version で示されたバージョンのリリース番号を指定します。値は整数でなければなりません。

mod

release で示されたリリースのモディフィケーション番号を指定します。値は整数でなければなりません。

version-string-constant

バージョンの正式名称を指定します。*version-string-constant* は単一値 (例えば、'8i') にすることができます。あるいは、*version*、*release*、そして該当する場合は *mod* を連結した値にすることができます (例えば、'8.0.3')。

TYPE *server-type*

要求された変更を適用する対象のデータ・ソースのタイプを指定します。

VERSION

server-type の後の **VERSION** とそのパラメーターでは、サーバー・オプションを使用可能にする、リセットする、あるいはドロップするときの対象となるデータ・ソースのバージョンを指定します。

WRAPPER *wrapper-name*

server-type および *server-version* に示されたタイプおよびバージョンのデータ・ソースと対話するために、フェデレーテッド・サーバーが使用するラッパーの名前を指定します。このラッパーは、カタログにリストされていなければなりません。

OPTIONS

server-name に示されたデータ・ソースに対して、あるいは *server-type* および関

ALTER SERVER

連パラメーターに示されたデータ・ソースのカテゴリに対して、どのサーバー・オプションを使用可能にする、リセットする、またはドロップするかを指定します。

ADD

サーバー・オプションを使用可能にします。

SET

サーバー・オプションの設定を変更します。

server-option-name

使用可能にする、あるいはリセットするサーバー・オプションを指定します。

string-constant

server-option-name の設定を、文字ストリング定数として指定します。

DROP *server-option-name*

サーバー・オプションをドロップします。

注

- サーバー・オプションは、同じ ALTER SERVER ステートメントに複数回指定することはできません (SQLSTATE 42853)。サーバー・オプションを使用可能にする、リセットする、あるいはドロップする場合、使用中の他のサーバー・オプションには影響はありません。
- 所定の作業単位 (UOW) 内の ALTER SERVER ステートメントは、以下のいずれかの条件の下では処理できません (SQLSTATE 55007)。
 - ステートメントが 1 つのデータ・ソースを参照していて、次のいずれかがすでに UOW に含まれている。
 - このデータ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント。
 - このデータ・ソース内の表またはビューのニックネーム上のオープン・カーソル。
 - このデータ・ソース内の表またはビューのニックネームに対して発行された INSERT、DELETE、または UPDATE ステートメント。
 - ステートメントがデータ・ソースのカテゴリ (たとえば、特定のタイプおよびバージョンのすべてのデータ・ソースなど) を参照しており、次のいずれかがすでに UOW に含まれている。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネームを参照する SELECT ステートメント。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネーム上のオープン・カーソル。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネームに対して発行された INSERT、DELETE、または UPDATE ステートメント。
- サーバー・オプションが、データ・ソースのタイプについてある値に設定され、このタイプのインスタンスについてはそれとは別の値に設定される場合、そのインスタンスの値については、前者の値は後者の値によってオーバーライドされます。例えば、サーバー・タイプ ORACLE について PLAN_HINTS を 'Y' に設定し、DELPHI という Oracle データ・ソースについては 'N' に設定したとしま

す。このような構成の場合、DELPHI 以外のすべての Oracle データ・ソースで、プランのヒントが使用可能になります。

- 前の alter add server オプション操作で使用可能になったデータ・ソースのカテゴリに対して、alter set オプションまたは alter drop server オプションのみを実行できます (SQLSTATE 42704)。
- サーバーのバージョンを変更するとき、DB2 は指定されたサーバー・バージョンがリモートのサーバー・バージョンと一致するかどうかを検査しません。正しくないサーバー・バージョンを指定すると、DB2 サーバー定義に属するニックネームにアクセスするとき、SQL エラーが生じることがあります。その可能性が最も高いのは、リモートのサーバー・バージョンよりも後のサーバー・バージョンを指定する場合です。その場合、サーバー定義に属するニックネームにアクセスすると、DB2 はリモート・サーバーが認識できない SQL を送信することがあります。

例

例 1: ID が未変更のままとなる場合に、Oracle 8.0.3 データ・ソースへ許可 ID がいつ送信されるかを確認します。さらに、ローカルのフェデレーテッド・サーバー CPU の速度がデータ・ソース CPU の 2 倍であるとして、オプティマイザーにこの統計を通知します。

```
ALTER SERVER
TYPE ORACLE
VERSION 8.0.3
OPTIONS
  (ADD FOLD_ID 'N',
   SET CPU_RATIO '2.0')
```

例 2: Documentum データ・ソース DCTM_SVR_ASIA がバージョン 4 に変更されていることを指示します。

```
ALTER SERVER DCTM_SVR_ASIA
VERSION 4
```

ALTER SERVICE CLASS

ALTER SERVICE CLASS ステートメントは、サービス・クラスの定義を変更します。

呼び出し

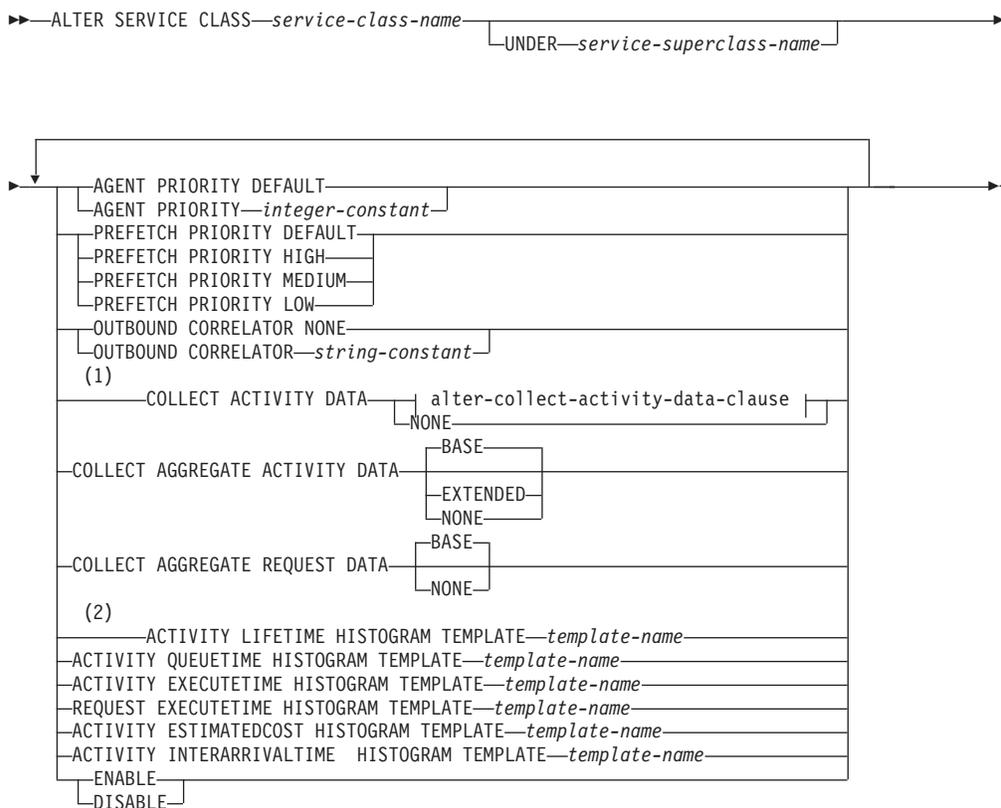
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

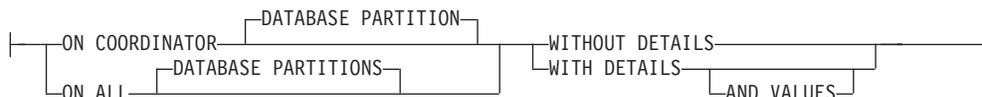
このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

ALTER SERVICE CLASS

構文



alter-collect-activity-data-clause:



注:

- 1 COLLECT 節は、サービス・サブクラスでのみ有効です。
- 2 HISTOGRAM TEMPLATE 節は、サービス・サブクラスのみで有効です。

説明

service-class-name

変更するサービス・クラスを識別します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。*service-class-name* は、データベースに存在しているサービス・クラスを指定していなければなりません (SQLSTATE 42704)。サービス・サブクラスを変更するためには、UNDER 節を使用して *service-superclass-name* が指定されている必要があります。

UNDER *service-superclass-name*

この節は、サービス・サブクラスの変更にのみ使用されます。*service-superclass-name* は、サービス・サブクラスのサービス・スーパークラスを識別します。識別されるサービス・スーパークラスは、データベースに存在していなければなりません (SQLSTATE 42704)。

AGENT PRIORITY DEFAULT または **AGENT PRIORITY** *integer-constant*

サービス・クラスで実行されるエージェントの相対的な (差分) オペレーティング・システム優先順位、または DB2 で実行されるスレッドの通常優先順位を指定します。デフォルト値は DEFAULT です。DEFAULT に設定されている場合、特別なアクションは実行されず、サービス・クラスのエージェントは、オペレーティング・システムがすべての DB2 スレッドをスケジュールに入れる場合の通常優先順位に従ってスケジュールされます。このパラメーターが DEFAULT 以外の値に設定されている場合、エージェントは、通常優先順位に、次のアクティビティの開始時点の AGENT PRIORITY を加えた値に等しい優先順位に設定されます。例えば、通常優先順位が 20 で AGENT PRIORITY が -10 に設定されている場合、サービス・クラスのエージェントの優先順位は $20 - 10 = 10$ に設定されます。

UNIX オペレーティング・システムおよび Linux の場合、有効な値は DEFAULT、および -20 から 20 までです (SQLSTATE 42615)。負の値は高い相対優先順位を示します。正の値は低い相対優先順位を示します。

Windows オペレーティング・システムの場合、有効値は DEFAULT および -6 から 6 です (SQLSTATE 42615)。負の値になるほど相対的に低い優先順位を示します。正の値になるほど相対的に高い優先順位を示します。

サービス・サブクラスの AGENT PRIORITY が DEFAULT の場合、その親のスーパークラスの AGENT PRIORITY 値が継承されます。デフォルト・サブクラスの AGENT PRIORITY は変更できません (SQLSTATE 5U032)。

OUTBOUND CORRELATOR が設定されている場合は、AGENT PRIORITY を DEFAULT に設定する必要があります (SQLSTATE 42613)。

注: AIX® では、サービス・クラスの中で AGENT PRIORITY を使用してエージェントにさらに高い相対優先順位を設定するには、インスタンス所有者に CAP_NUMA_ATTACH および CAP_PROPAGATE の能力がなければなりません。これらの能力を付与するには、root としてログオンし、以下のコマンドを実行します。

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

PREFETCH PRIORITY

このパラメーターは、サービス・クラス内のエージェントがプリフェッチ要求をサブミットできる優先順位を制御します。有効な値は HIGH、MEDIUM、LOW、または DEFAULT です (SQLSTATE 42615)。HIGH、MEDIUM、および LOW は、優先順位がそれぞれ高、中、および低い優先キューにプリフェッチ要求がサブミットされることを意味します。プリフェッチャーは、優先順位の高いものから低いものへ、順に優先キューを空にします。サービス・クラスのエージェントは、次のアクティビティの開始時に、PREFETCH PRIORITY レベルでプリフェッチ要求をサブミットします。プリフェッチ要求がサブミットされてから PREFETCH PRIORITY が変更された場合、要求優先順位は変更されません。デフォルト値は DEFAULT です。この設定は、内部的にサービス・スーパークラスの MEDIUM にマップされています。あるサービス・サブクラスについて DEFAULT が設定された場合、その親のスーパークラスの PREFETCH PRIORITY が継承されます。

デフォルト・サブクラスの PREFETCH PRIORITY は変更できません (SQLSTATE 5U032)。

OUTBOUND CORRELATOR NONE または OUTBOUND CORRELATOR

string-constant

このサービス・クラスのスレッドを外部ワークロード・マネージャー・サービス・クラスに関連付けるかどうかを指定します。

サービス・スーパークラスで OUTBOUND CORRELATOR が *string-constant* に設定されていて、サービス・サブクラスで OUTBOUND CORRELATOR NONE が設定されている場合、そのサービス・サブクラスは親の OUTBOUND CORRELATOR を継承します。AGENT PRIORITY が DEFAULT に設定されていない場合は、OUTBOUND CORRELATOR を NONE に設定する必要があります (SQLSTATE 42613)。

OUTBOUND CORRELATOR NONE

サービス・スーパークラスの場合、このサービス・クラスに関連付けられた外部ワークロード・マネージャー・サービス・クラスがないことを指定します。サービス・サブクラスの場合、外部ワークロード・マネージャー・サービス・クラスの間接がその親と同じであることを指定します。

OUTBOUND CORRELATOR *string-constant*

このサービス・クラスのスレッドを外部ワークロード・マネージャー・サービス・クラスに関連付けるための相関関係子として使用する *string-constant* を指定します。その外部ワークロード・マネージャーはアクティブでなければなりません (SQLSTATE 5U030)。その外部ワークロード・マネージャーは、*string-constant* の値を認識するように設定されている必要があります。

COLLECT ACTIVITY DATA

このサービス・クラスで実行する各アクティビティに関する情報を、アクティビティ完了時にアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。COLLECT ACTIVITY DATA 節は、サービス・サブクラスでのみ有効です。

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

アクティビティのコーディネーターのデータベース・パーティションでのみ、アクティビティ・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティが処理されるすべてのデータベース・パーティションでアクティビティ・データを収集することを指定します。ただし、アクティビティの詳細と値は、コーディネーターのデータベース・パーティションでのみ収集されます。

WITHOUT DETAILS

このサービス・クラスで実行される各アクティビティに関するデータを、アクティビティの実行完了時にアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境のデータを含むアクティビティについて、これらのデータをアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティーに入力データ値が含まれている場合、それをアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

このサービス・クラスで実行する各アクティビティーについて、アクティビティー・データを収集しないことを指定します。

COLLECT AGGREGATE ACTIVITY DATA

このサービス・クラスについて、集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。デフォルトは **COLLECT AGGREGATE ACTIVITY DATA BASE** です。 **COLLECT AGGREGATE ACTIVITY DATA** 節は、サービス・サブクラスでのみ有効です。

BASE

このサービス・クラスについて、基礎集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。基礎集約アクティビティー・データには以下のものが含まれます。

- アクティビティー・コストの最高水準点の見積もり
- 戻り行数の最高水準点
- **TEMPORARY** 表スペース使用量の最高水準点
- アクティビティー存続時間のヒストグラム
- アクティビティー・キュー時間のヒストグラム
- アクティビティー実行時間のヒストグラム

EXTENDED

このサービス・クラスについて、すべての集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。これには、すべての基礎集約アクティビティー・データに加えて、以下のものが含まれます。

- アクティビティー・データ操作言語 (DML) の見積コスト・ヒストグラム
- アクティビティー DML の到着間隔時間のヒストグラム

NONE

このサービス・クラスについて集約アクティビティー・データをキャプチャーしないことを指定します。

COLLECT AGGREGATE REQUEST DATA

このサービス・クラスについて、集約要求データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。デフォルトは **COLLECT AGGREGATE REQUEST DATA NONE** です。 **COLLECT AGGREGATE REQUEST DATA** 節は、サービス・サブクラスでのみ有効です。

ALTER SERVICE CLASS

BASE

このサービス・クラスについて、基礎集約要求データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

このサービス・クラスについて集約要求データをキャプチャーしないことを指定します。

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 アクティビティの所要時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、キューに入っていた時間と実行時間の両方が含まれます。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にものみ収集されます。この節は、サービス・サブクラスでのみ有効です。

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 アクティビティがキューに入っている時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にものみ収集されます。この節は、サービス・サブクラスでのみ有効です。

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 アクティビティの実行のための所要時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、アクティビティがキューに入っていた時間は含まれません。このヒストグラムにおいて、アクティビティの実行時間はコーディネーター・データベース・パーティションでのみ収集されます。アイドル時間はこの時間に含まれません。アイドル時間とは、要求が実行されてから同じアクティビティに属する別の要求が実行されるまでの間、何も作業が実行されていない時間のことです。アイドル時間の一例として、カーソルのオープンが終了してからカーソルからのフェッチが開始するまでの間の時間があります。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にものみ収集されます。この節は、サービス・サブクラスでのみ有効です。

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 要求の実行のための所要時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、アクティビティがキューに入っていた時間は含まれません。このヒストグラムにおいて要求実行時間は、要求が実行されるデータベース・パーティションごとに収集されます。この情報は、COLLECT AGGREGATE REQUEST DATA 節とその BASE オプションが指定されている場合にものみ収集されます。この節は、サービス・サブクラスでのみ有効です。

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*

このサービス・クラスで実行される DML アクティビティの見積コスト (timeron 単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節とその EXTENDED オプションが指定されている場合にのみ収集されます。この節は、サービス・サブクラスでのみ有効です。

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*

1 つの DML アクティビティの到着から次の DML アクティビティの到着までの間の時間の長さ (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節とその EXTENDED オプションが指定されている場合にのみ収集されます。この節は、サービス・サブクラスでのみ有効です。

ENABLE または DISABLE

接続とアクティビティをサービス・クラスにマップできるかどうかを指定します。

ENABLE

接続およびアクティビティをサービス・クラスにマップできます。

DISABLE

接続およびアクティビティをサービス・クラスにマップできません。

DISABLE (無効) に設定されているサービス・クラスに新しくマップされる接続やアクティビティは拒否されます (SQLSTATE 5U028)。サービス・スーパークラスが無効として設定されている場合は、サービス・サブクラスも無効になります。サービス・スーパークラスが再度有効になった場合、そのサービス・サブクラスはシステム・カタログで定義された状態に戻ります。デフォルト・サービス・クラスを無効にすることはできません (SQLSTATE 5U032)。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

ALTER SERVICE CLASS

注

- 全パーティションを通じて、同時に実行できる非コミットの WLM 排他 SQL ステートメントは 1 つのみです。非コミットの WLM 排他 SQL ステートメントが実行されている場合、後続の WLM 排他 SQL ステートメントは、現行の WLM 排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、COMMIT ステートメントが実行されるまで有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。
- ALTER SERVICE CLASS ステートメントがコミットされた場合、AGENT PRIORITY、PREFETCH PRIORITY、OUTBOUND CORRELATOR、および COLLECT への変更は、サービス・クラスの次の新しいアクティビティから有効になります。サービス・クラスに既存のアクティビティでは、引き続き古い設定を使用して作業が完了されます。

例

例 1: サービス・スーパークラス PETSALLES のエージェントのエージェント優先順位を、DEFAULT から可能な最高の値 (UNIX および Linux オペレーティング・システムで表示; Windows オペレーティング・システムでは 6 に置換) に変更します。

```
ALTER SERVICE CLASS PETSALLES AGENT PRIORITY -20
```

例 2: サービス・スーパークラス BARNSALLES を変更して、アウトバウンド相関関係子 'osLowPriority' を追加します。サービス・スーパークラスで実行されるスレッドとサービス・サブクラスには、関連付けられたアウトバウンド相関関係子 'osLowPriority' を持つようになります。

```
ALTER SERVICE CLASS BARNSALLES OUTBOUND CORRELATOR 'osLowPriority'
```

ALTER TABLE

ALTER TABLE ステートメントは、表の定義を変更します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 変更する表に対する ALTER 特権
- 変更する表に対する CONTROL 特権
- 表のスキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

外部キーの作成/ドロップの場合、このステートメントの許可 ID には、親表に対する以下の特権が少なくとも 1 つ含まれている必要があります。

- その表に対する REFERENCES 特権
- 指定の親キーのそれぞれの列に対する REFERENCES 特権
- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

表 T の主キーまたはユニーク制約をドロップするには、この親キー T に従属しているすべての表において、ステートメントの許可 ID に以下の特権が少なくとも 1 つ含まれている必要があります。

- その表に対する ALTER 特権
- 表に対する CONTROL 特権
- 表のスキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

(全選択を使用して) 表をマテリアライズ照会表に変更するには、このステートメントの許可 ID によって保持されている特権に、以下のうち少なくとも 1 つが含まれている必要があります。

- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

さらに、全選択で識別された個々の表またはビューに対する以下の特権 (グループ特権を除く) が少なくとも 1 つ含まれている必要があります。

- その表またはビューに対する SELECT 特権および ALTER 特権 (グループ特権を含む)
- 表またはビューに対する CONTROL 特権
- その表またはビューに対する SELECT 特権と、その表またはビューのスキーマに対する ALTERIN 特権 (グループ特権を含む)
- SYSADM または DBADM 権限

表を変更してマテリアライズ照会表でなくなるようにするには、ステートメントの許可 ID が持っている特権に、このマテリアライズ照会表を定義するのに使用する全選択で識別される各表またはビューに対して、少なくとも以下の 1 つが含まれている必要があります。

- その表またはビューに対する ALTER 特権
- 表またはビューに対する CONTROL 特権
- その表またはビューのスキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

タイプ DB2SECURITYLABEL の列を表に追加するには、ステートメントの許可 ID の保持する特権に、少なくとも、表に関連するセキュリティー・ポリシーからのセキュリティー・ラベルが含まれている必要があります。

表からセキュリティー・ポリシーを削除するには、ステートメントの許可 ID の保持する特権に、SECADM 権限が含まれている必要があります。

ALTER TABLE

データ・パーティションにアタッチするよう表を変更するには、ステートメントの許可 ID の保持する特権に、ソース表に関して少なくとも以下のうちの 1 つが含まれている必要があります。

- その表に対する SELECT 特権と、その表のスキーマに対する DROPIN 特権
- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

また、ターゲット表に関しては、以下のうち少なくとも 1 つが含まれている必要があります。

- その表に対する ALTER および INSERT 特権
- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

データ・パーティションをデタッチするよう表を変更するには、ステートメントの許可 ID の保持する特権に、デタッチされたパーティションのターゲット表に関して少なくとも以下のうちの 1 つが含まれている必要があります。

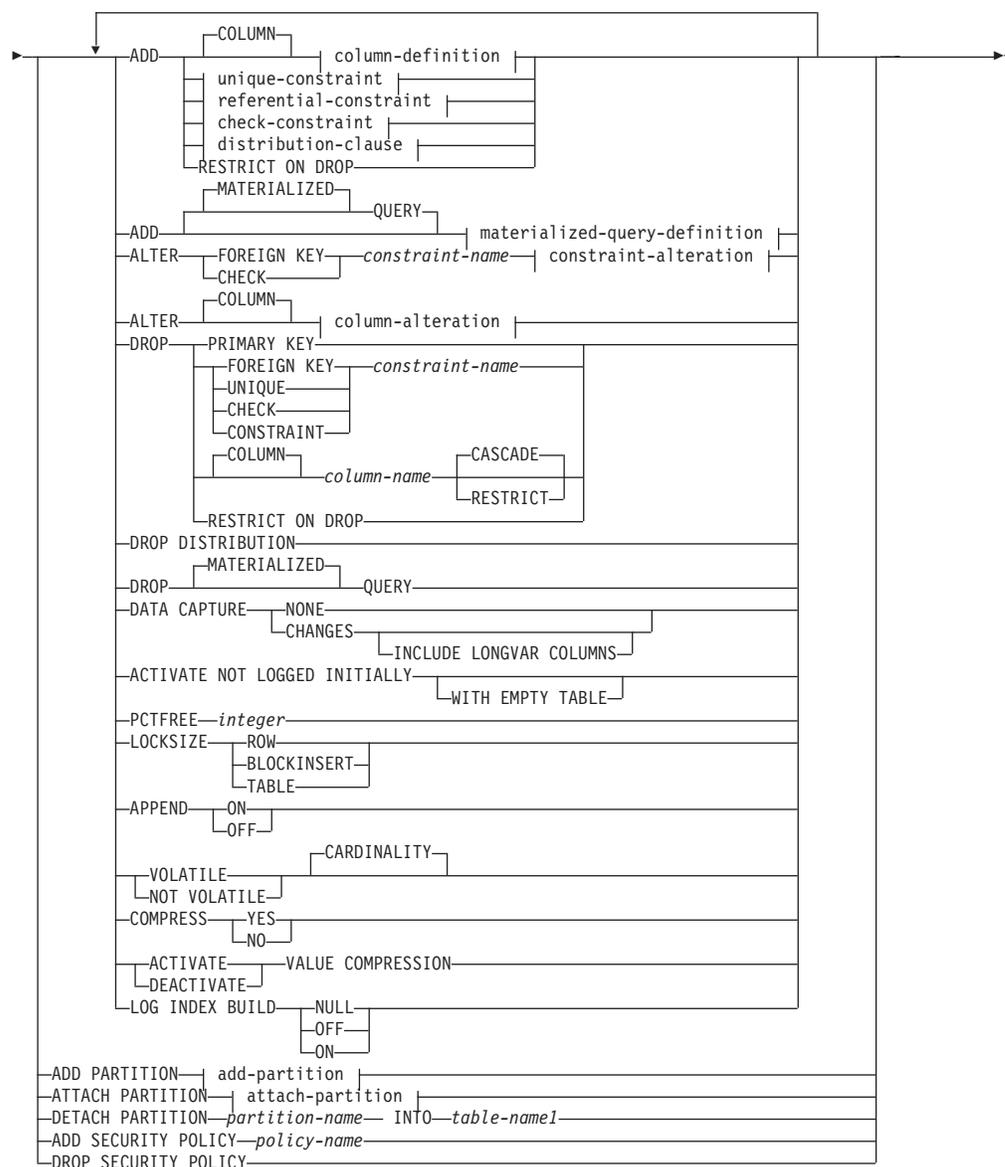
- データベースに対する CREATETAB 権限、および表の使用する表スペースに対する USE 特権に加えて、以下のいずれか。
 - データベースに対する IMPLICIT_SCHEMA 権限 (新規表の暗黙的または明示的スキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (新規表のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

また、ソース表に関しては、以下のうち少なくとも 1 つが含まれている必要があります。

- その表に対する SELECT、ALTER、および DELETE 特権
- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

構文

▶▶ALTER TABLE—*table-name*————→

**add-partition:**

```

┌──────────────────┐ boundary-spec ┌──────────────────┐ ┌──────────────────┐
└──partition-name──┘ ┌──IN──tablespace-name──┘ ┌──LONG IN──tablespace-name──┘

```

boundary-spec:

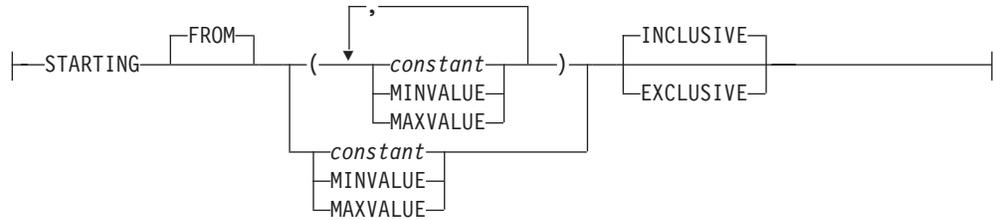
```

┌──┐ starting-clause ┌──┐ ending-clause ┌──────────────────┐
└──┘ ┌──ending-clause──┘ ┌──────────────────┐

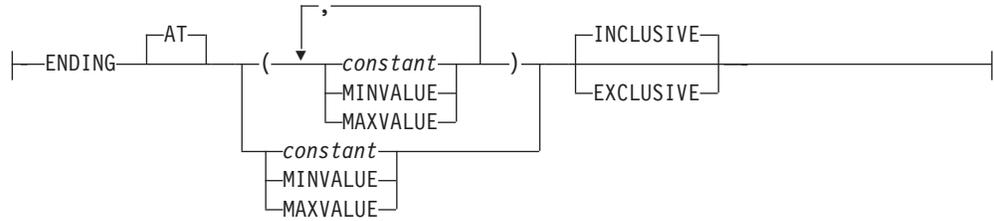
```

starting-clause:

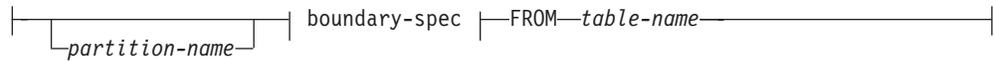
ALTER TABLE



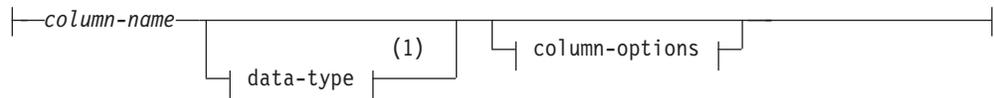
ending-clause:



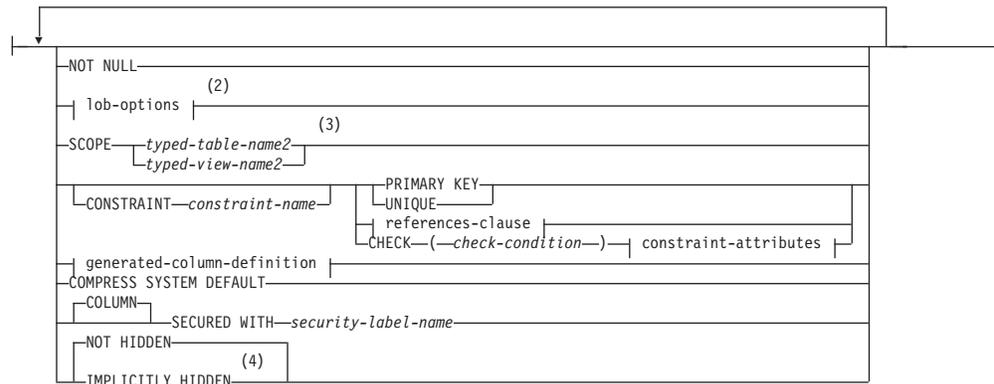
attach-partition:



column-definition:

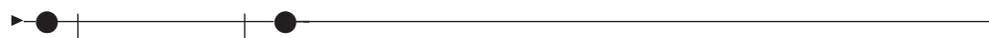
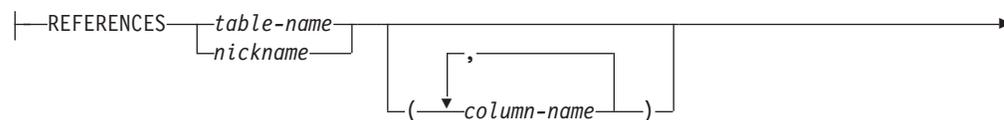
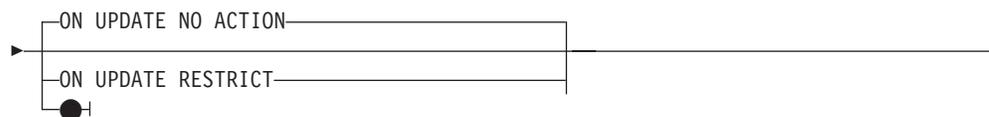
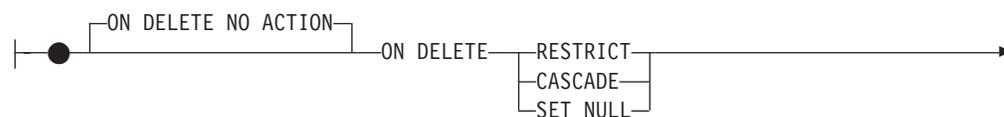
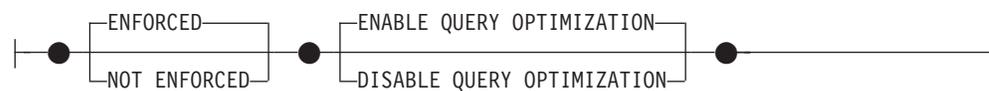
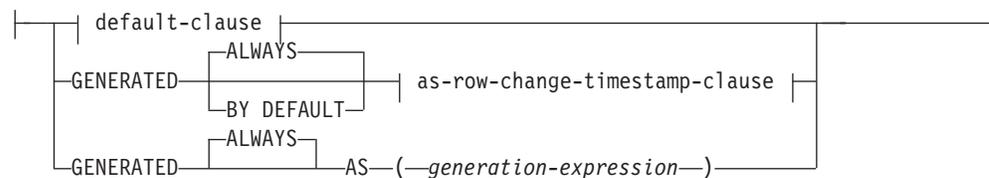


column-options:



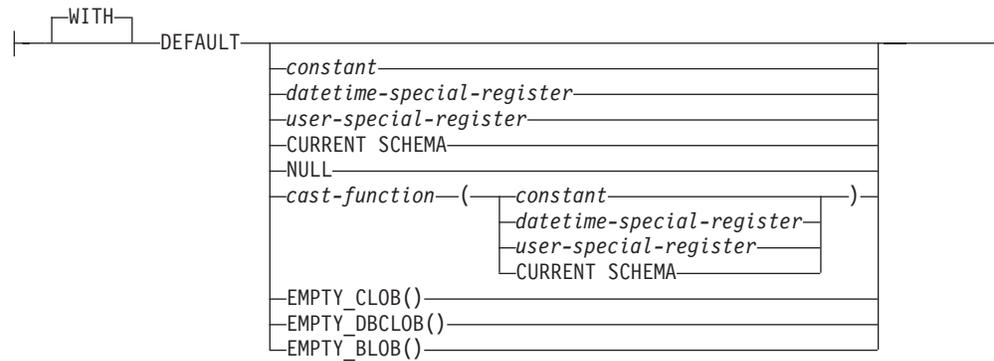
lob-options:



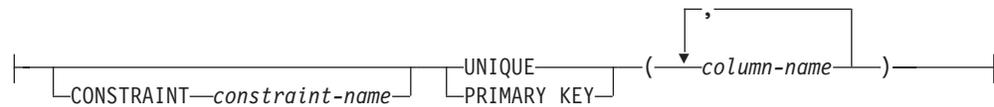
**references-clause:****rule-clause:****constraint-attributes:****generated-column-definition:**

ALTER TABLE

default-clause:



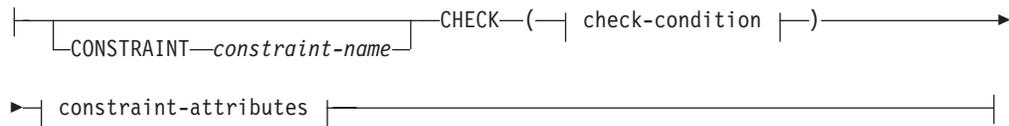
unique-constraint:



referential-constraint:



check-constraint:



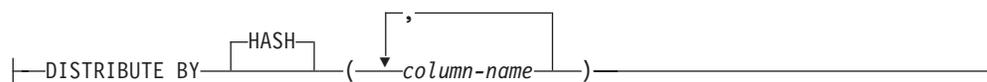
check-condition:



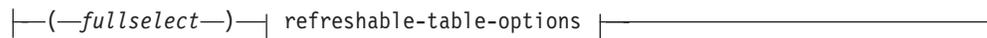
functional-dependency:



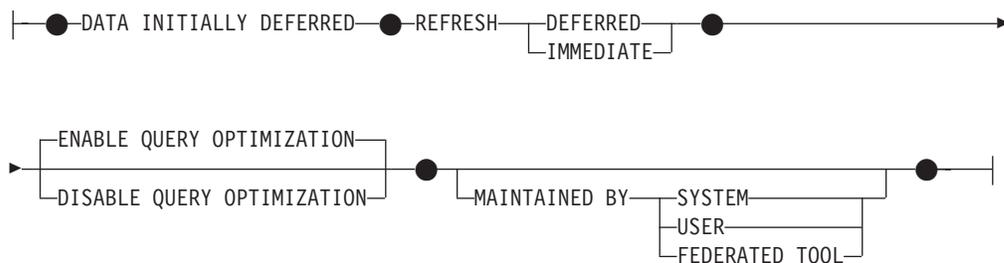
distribution-clause:



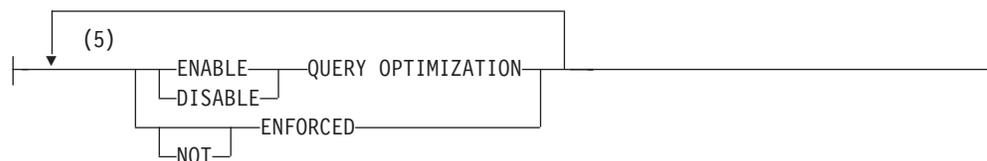
materialized-query-definition:



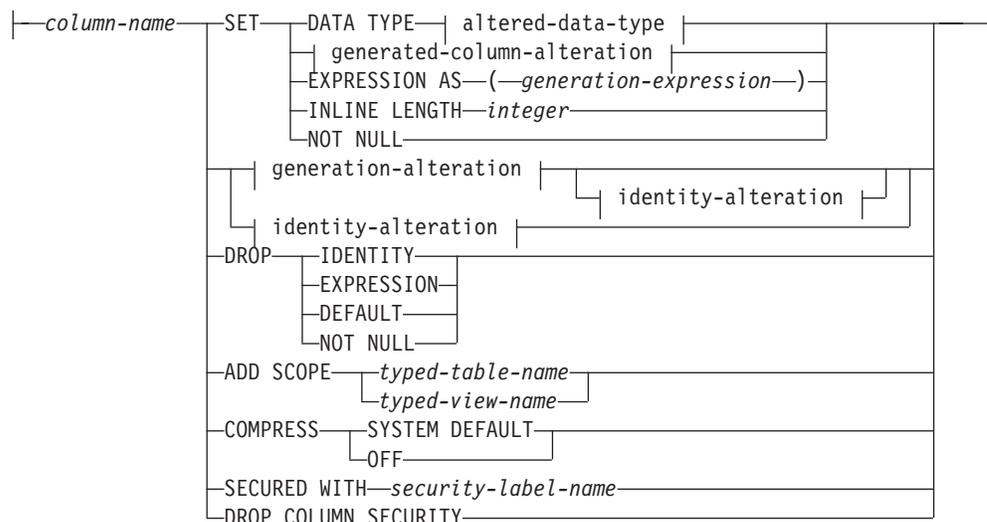
refreshable-table-options:



constraint-alteration:



column-alteration:

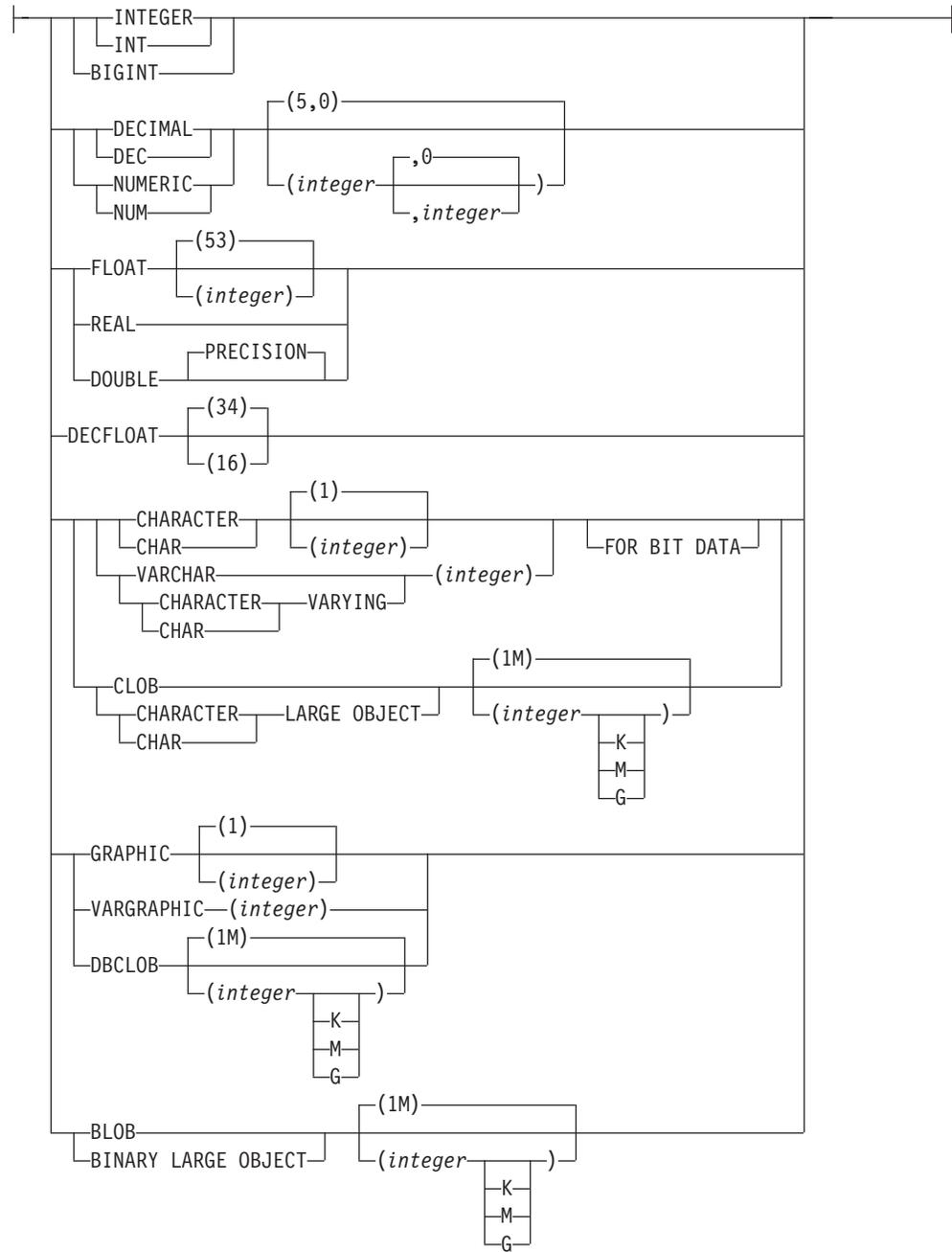


ALTER TABLE

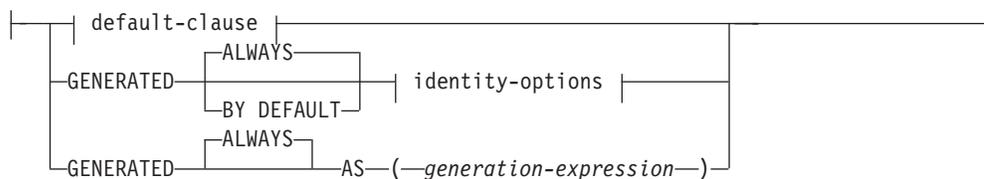
altered-data-type:



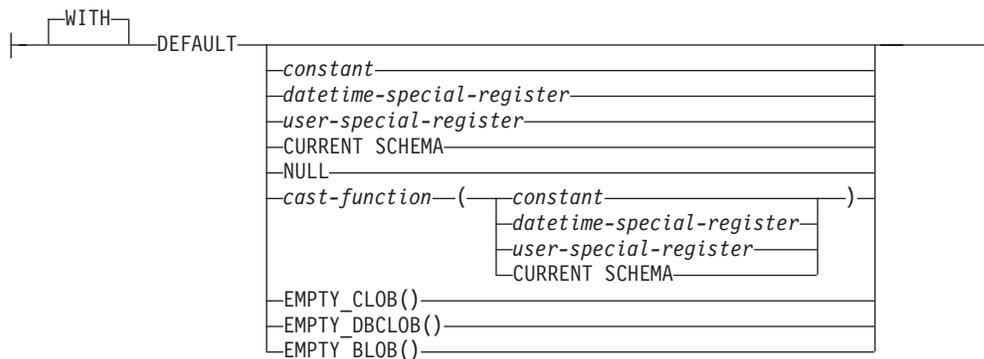
built-in-type:



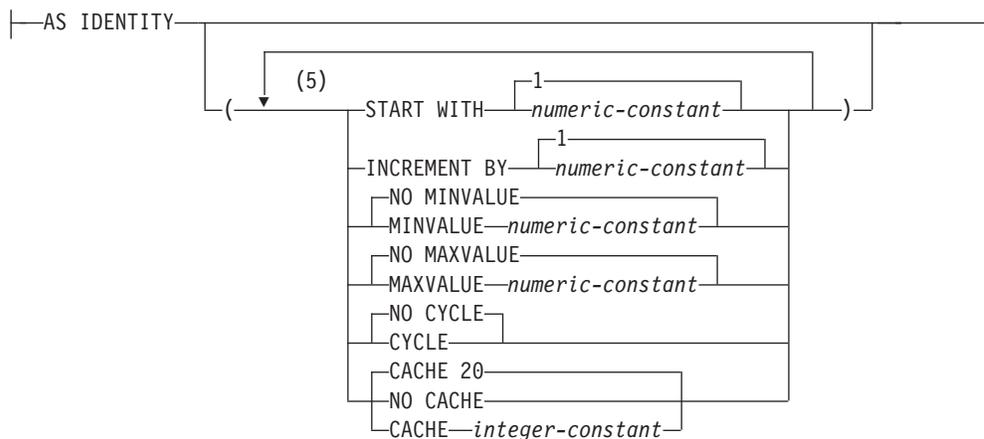
generated-column-alteration:



default-clause:



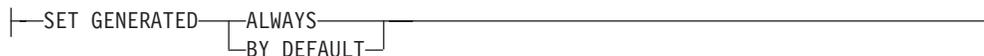
identity-options:



as-row-change-timestamp-clause:

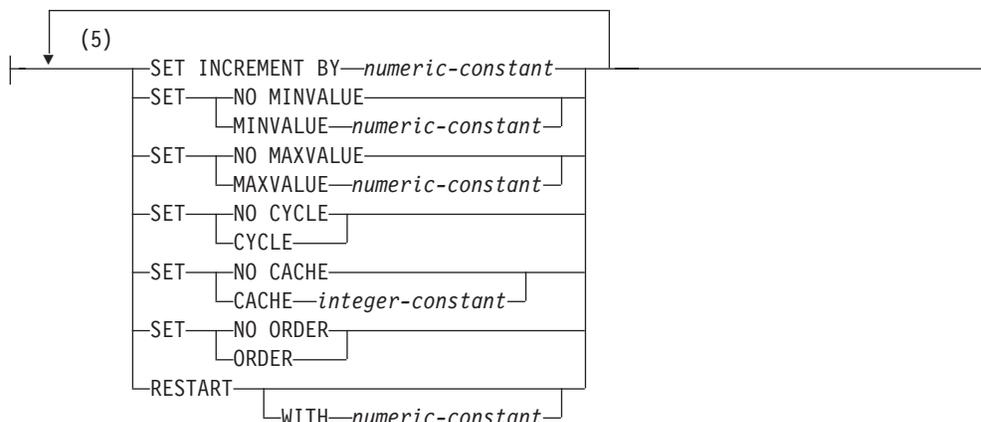


generation-alteration:



identity-alteration:

ALTER TABLE



注:

- 1 最初に選択された列オプションが *generated-column-definition* であれば、*data-type* は省略できます。これは、生成式によって計算されることとなります。
- 2 *lob-options* 節は、ラージ・オブジェクト・タイプ (CLOB、DBCLOB、および BLOB) と、ラージ・オブジェクト・タイプに基づく特殊タイプに対してのみ適用されます。
- 3 SCOPE 節は REF タイプに対してのみ適用されます。
- 4 IMPLICITLY HIDDEN を指定できるのは、ROW CHANGE TIMESTAMP も指定される場合のみです。
- 5 同じ節を複数回指定することはできません。
- 6 行変更タイム・スタンプ列のデータ・タイプはオプションです。

説明

table-name

table-name は、現行サーバーに存在する表を示していなければなりません。これはニックネームであってはならず (SQLSTATE 42809)、ビュー、カタログ表、または宣言済み一時表であつてもなりません (SQLSTATE 42995)。

table-name でマテリアライズ照会表を指定している場合、変更は、マテリアライズ照会表の追加またはドロップ、NOT LOGGED INITIALLY の活動化、RESTRICT ON DROP の追加またはドロップ、pctfree、locksize、append、または volatile の変更のみを行うことができます。

table-name が範囲クラスター表を示している場合、変更は、制約の追加・変更・ドロップ、NOT LOGGED INITIALLY の活動化、RESTRICT ON DROP の追加またはドロップ、locksize、data capture、または volatile の変更、および列のデフォルト値の設定のみ行うことができます。

ADD PARTITION *add-partition*

パーティション表に 1 つ以上のデータ・パーティションを追加します。指定された表がパーティション表でない場合、エラーが戻されます (SQLSTATE 428FT)。データ・パーティションの数は、32 767 を超えてはなりません。

partition-name

データ・パーティションの名前を指定します。この名前は、表の他のいずれのデータ・パーティションとも同じであってはなりません (SQLSTATE 42710)。この節を指定しない場合は、「PART」の後に文字形式の整数値が付いた名前になります (このようにして、その表での固有の名前になります)。

boundary-spec

新規のデータ・パーティションに対応する値の範囲を指定します。この範囲は、既存のデータ・パーティションの範囲と重なり合ってはなりません (SQLSTATE 56016)。 *starting-clause* および *ending-clause* についての説明は、『CREATE TABLE』を参照してください。

starting-clause が省略されると、新規のデータ・パーティションは、表の末尾に入るものと想定されます。 *ending-clause* が省略されると、新規のデータ・パーティションは、表の先頭に入るものと想定されます。パーティション・キーの最初の列が DESC であれば、上記の想定は逆になります。

IN *tablespace-name*

データ・パーティションが保管される表スペースを指定します。指定する表スペースは、パーティション表の他の表スペースと同じページ・サイズ、同じデータベース・パーティション・グループ、同じスペース管理方式でなければなりません (SQLSTATE 42838)。これは、同じ表の他のデータ・パーティションにすでに使用されている表スペース、あるいは、この表によって現在使用されていない表スペースとすることもできます。しかし、ステートメントの許可 ID には、その表スペースに対する USE 特権が必要です (SQLSTATE 42727)。この節を指定しない場合は、表の、最初の可視の、またはアタッチされたデータ・パーティションの表スペースが使用されます。

LONG IN *tablespace-name*

長い列データを含むデータ・パーティションが保管される表スペースを指定します。指定する表スペースは、パーティション表の他の表スペースおよびデータ・パーティションと同じページ・サイズ、同じデータベース・パーティション・グループ、同じスペース管理方式でなければなりません (SQLSTATE 42838)。ステートメントの許可 ID には、その表スペースに対する USE 特権が必要です。指定した表スペースのページ・サイズおよびエクステント・サイズは、パーティション表の他のデータ・パーティションのページ・サイズおよびエクステント・サイズと異なる可能性があります。

パーティション表で LONG IN 節の使用を制御する規則については、『パーティション表でのラージ・オブジェクトの動作』を参照してください。

ATTACH PARTITION *attach-partition*

他の表を新規のデータ・パーティションとしてアタッチします。アタッチされる表のデータ・オブジェクトは、アタッチ先の表の新規パーティションになります。データ移動は一切ありません。表は SET INTEGRITY ペンディング状態となり、参照整合性検査は、SET INTEGRITY ステートメントの実行まで据え置かれます。ALTER TABLE ATTACH 操作では、IN 節または LONG IN 節を使用することはできません。そのデータ・パーティションの LOB の配置は、ソース表の作成時に決定されます。パーティション表で LONG IN 節の使用を制御する規則については、『パーティション表でのラージ・オブジェクトの動作』を参照してください。

partition-name

データ・パーティションの名前を指定します。この名前は、表の他のいずれのデータ・パーティションとも同じであってはなりません (SQLSTATE 42710)。この節を指定しない場合は、「PART」の後に文字形式の整数値が付いた名前になります (このようにして、その表での固有の名前になります)。

boundary-spec

新規のデータ・パーティションに対応する値の範囲を指定します。この範囲は、既存のデータ・パーティションの範囲と重なり合ってはなりません (SQLSTATE 56016)。starting-clause および ending-clause についての説明は、『CREATE TABLE』を参照してください。

starting-clause が省略されると、新規のデータ・パーティションは、表の末尾に入るものと想定されます。ending-clause が省略されると、新規のデータ・パーティションは、表の先頭に入るものと想定されます。

FROM table-name1

表を、新規パーティションのデータのソースとして使用されるよう指定します。table-name1 の表定義は、複数のデータ・パーティションを持つことができず、以下のいずれかの形で、変更される表と一致してはなりません (SQLSTATE 428G3)。

- 列数が同じ。
- 表内の同じ位置 (順に並べた時の同じ位置) にある列のデータ・タイプが同じ。
- 表内の同じ位置 (順に並べた時の同じ位置) にある列の NULL 可能特性が同じ。
- またデータが配分されている場合は、同じ配分方法を使用して、同じデータベース・パーティション・グループ上で配分する。
- いずれかの表のデータが編成されている場合は、編成が一致する。

table-name1 のデータが正しくアタッチされた後は、DROP TABLE table-name1 と同等の操作が実行され、この表はデータを持たなくなり、データベースから削除されます。

DETACH PARTITION partition-name INTO table-name1

変更される表からデータ・パーティション partition-name をデタッチし、そのデータ・パーティションを使用して、table-name1 という名前の新規表を作成します。このデータ・パーティションは、論理的に、データ移動なしで新規表にアタッチされます。変更される表の最後に残ったデータ・パーティションを指定することはできません (SQLSTATE 428G2)。

ADD SECURITY POLICY policy-name

セキュリティー・ポリシーを表に追加します。セキュリティー・ポリシーは、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。表に既存のセキュリティー・ポリシーがあってはなりません (SQLSTATE 55065)。また、型付き表 (SQLSTATE 428DH)、マテリアライズ照会表 (MQT)、またはステージング表 (SQLSTATE 428FG) であってはなりません。

DROP SECURITY POLICY

表からセキュリティー・ポリシーと、すべての LBAC 保護を削除します。

table-name によって指定される表は、セキュリティー・ポリシーによって保護される必要があります (SQLSTATE 428GT)。表に、DB2SECURITYLABEL データ・タイプの列があれば、そのデータ・タイプは VARCHAR (128) FOR BIT DATA に変更されます。表に保護された列が 1 つ以上あれば、それらの列は無保護になります。

ADD column-definition

列を表に追加します。型付き表を使用することはできません (SQLSTATE 428DH)。表の既存の行に関して、新しい列の値はデフォルト値に設定されます。新しい列はその表の最後の列になります。つまり、当初 *n* 個の列があった場合、追加された列は列 *n+1* になります。

新しい列を追加する場合、すべての列のバイト・カウントの合計が、最大レコード・サイズを超えてはなりません。

column-name

表に追加する列の名前です。名前は非修飾でなければなりません。表にすでにある列名は使用できません (SQLSTATE 42711)。

data-type

『CREATE TABLE』の項に示されるデータ・タイプのいずれかです。

NOT NULL

列に NULL 値が入るのを防止します。 *default-clause* (DEFAULT 節) も指定する必要があります (SQLSTATE 42601)。

NOT HIDDEN または IMPLICITLY HIDDEN

列を隠し列と定義するかどうかを指定します。列を表の暗黙的参照に組み込むかどうか、SQL ステートメントで明示的に参照できるかどうかは隠し属性によって決まります。デフォルトは NOT HIDDEN です。

NOT HIDDEN

列を表の暗黙的参照に組み込むこと、および列を明示的に参照できることを指定します。

IMPLICITLY HIDDEN

名前でも明示的に参照されない限り列は SQL ステートメントから不可視であることを指定します。例えば、表に IMPLICITLY HIDDEN 節によって定義された列が組み込まれていると仮定した場合、暗黙的に隠された列は SELECT * の結果に組み込まれません。しかし、暗黙的に隠された列の名前を明示的に参照する SELECT の結果については、結果表にその列が組み込まれます。

IMPLICITLY HIDDEN は、ROW CHANGE TIMESTAMP 列にのみ指定する必要があります (SQLSTATE 42867)。ROW CHANGE

TIMESTAMP FOR *table-designator* 式は、IMPLICITLY HIDDEN ROW CHANGE TIMESTAMP 列に解決されます。したがって、ROW CHANGE TIMESTAMP 列は IMPLICITLY HIDDEN として表に追加することができ、この表から SELECT * を実行する既存のアプリケーションを、列を処理するために変更する必要はありません。新規アプリケーションは列名が分からなくても、式を使用することによって常に列にアクセスすることができます。

ALTER TABLE

lob-options

LOB データ・タイプのオプションを指定します。『CREATE TABLE』の *lob-options* を参照してください。

SCOPE

参照タイプ列の有効範囲を指定します。

typed-table-name2

型付き表の名前。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-table-name2* のタイプを表します (SQLSTATE 428DM)。値が *typed-table-name2* の既存行を実際に参照していることを確認するための、 *column-name* のデフォルト値の検査は行われません。

typed-view-name2

型付きビューの名前。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-view-name2* のタイプを表します (SQLSTATE 428DM)。値が *typed-view-name2* の既存行を実際に参照していることを確認するための、 *column-name* のデフォルト値の検査は行われません。

CONSTRAINT *constraint-name*

制約の名前を指定します。制約名 (*constraint-name*) は、同じ ALTER TABLE ステートメントにすでに指定されている制約、あるいは表に既存の他の制約の名前であってはなりません (SQLSTATE 42710)。

ユーザーが制約名を指定しない場合は、表に定義されている既存の制約の ID の中でユニークな 18 バイトの長さの ID がシステムによって生成されます。(ID は、"SQL" と、タイム・スタンプに基づいて生成される一連の 15 の数字から構成されます。)

PRIMARY KEY 制約またはユニーク制約とともに使用した場合、この *constraint-name* は、制約をサポートするために作成される索引の名前として使用されます。ユニーク制約に関連した索引名の詳細については、Notes®を参照してください。

PRIMARY KEY

これは、1 つの列からなる主キーを定義する簡単な方法です。つまり、PRIMARY KEY が列 C の定義で指定されている場合、その効果は、PRIMARY KEY(C) 節が独立した節として指定された場合と同じです。列に NULL 値を含めることはできないので、NOT NULL 属性も指定する必要があります (SQLSTATE 42831)。

後述の *unique-constraint* の説明の中の PRIMARY KEY を参照してください。

UNIQUE

これは、1 つの列からなるユニーク・キーを定義する簡単な方法です。すなわち、UNIQUE を列 C の定義に指定すると、UNIQUE(C) 節を独立した節として指定した場合と同じ結果になります。

後述の *unique-constraint* の説明の中の UNIQUE を参照してください。

references-clause

これは、1 つの列からなる外部キーを定義する簡単な方法です。つま

り、*references-clause* が列 C の定義に指定されている場合、その効果は、列として C しか指定されていない FOREIGN KEY 節の一部として *references-clause* が指定された場合と同じになります。

『CREATE TABLE』の *references-clause* を参照してください。

CHECK (*check-condition*)

これは、1 つの列に適用されるチェック制約を定義する簡単な方法です。『CREATE TABLE』の *check-condition* を参照してください。

generated-column-definition

列の生成の詳細については、『CREATE TABLE』を参照してください。

default-clause

列のデフォルト値を指定します。

WITH

オプション・キーワード。

DEFAULT

INSERT で値が提供されなかった場合、もしくは INSERT や UPDATE で DEFAULT が指定されている場合に、デフォルト値を提供します。DEFAULT キーワードの後に特定のデフォルト値の指定がない場合のデフォルト値は、列のデータ・タイプによって異なります。表 13 を参照してください。列が XML または構造化タイプとして定義されている場合、DEFAULT 節を指定することはできません。

列が特殊タイプを使用して定義される場合、列のデフォルト値は、特殊タイプにキャストされたソース・データ・タイプのデフォルト値になります。

表 13. デフォルト値 (値が指定されない場合)

データ・タイプ	デフォルト値
数値	0
固定長文字列	ブランク
可変長文字列	長さ 0 の文字列
固定長 GRAPHIC 文字列	2 バイトのブランク
可変長 GRAPHIC 文字列	長さ 0 の文字列
日付	既存の行の場合、0001 年 1 月 1 日に対応する日付。追加行の場合には、現在の日付。
時刻	既存の行の場合、0 時間 0 分 0 秒に対応する時刻。追加行の場合には、現在の時刻。
Timestamp	既存の行の場合、0001 年 1 月 1 日 0 時 0 分 0 秒 0 マイクロ秒に対応する日付。追加行の場合には、現在のタイム・スタンプ。
バイナリ・文字列 (blob)	長さ 0 の文字列

column-definition から DEFAULT を省略すると、その列のデフォルト値として NULL 値が使用されます。

ALTER TABLE

DEFAULT キーワードに指定できる値のタイプは、次のとおりです。

constant

列のデフォルト値として定数を指定します。指定する定数は、次の条件を満たしていなければなりません。

- 第 3 章に示されている割り当ての規則に従って、その列に割り当てることができる値でなければなりません。
- その列が浮動小数点データ・タイプとして定義されている場合を除き、浮動小数点の定数を指定してはなりません。
- 数値定数、または列のデータ・タイプが 10 進浮動小数点の場合は 10 進浮動小数点特殊値でなければなりません。浮動小数点定数はまず DOUBLE として解釈され、次に 10 進浮動小数点数に変換されます。DECFLOAT(16) 列の場合、10 進定数の精度は 16 以下でなければなりません。
- 定数が 10 進定数の場合、その列のデータ・タイプの位取りを超えるゼロ以外の数字を含めてはなりません (例えば、DECIMAL(5,2) の列のデフォルト値として 1.234 を指定することはできません)。
- 指定する定数が 254 バイトを超えてはなりません。この制約には、引用符文字や 16 進定数の X などの接頭部文字も含まれます。さらに、定数が *cast-function* の引数の場合には、完全修飾された関数名から取った文字や括弧も含めて、この制限を超えてはなりません。

datetime-special-register

INSERT、UPDATE、または LOAD の実行時における日時特殊レジスターの値 (CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP) を、その列のデフォルト値として指定します。その列のデータ・タイプは、指定した特殊レジスターに対応するデータ・タイプでなければなりません (例えば、CURRENT DATE を指定した場合、データ・タイプは DATE でなければなりません)。既存の行の場合は、ALTER TABLE ステートメントが処理される時点の現行日付、現行時刻、または現行タイム・スタンプが値として使用されます。

user-special-register

INSERT、UPDATE、または LOAD の実行時におけるユーザー特殊レジスターの値 (CURRENT USER、SESSION_USER、SYSTEM_USER) を、その列のデフォルトとして指定します。その列のデータ・タイプは、ユーザー特殊レジスターの長さ属性よりも短い文字ストリングであってはなりません。なお、SESSION_USER の代わりに USER を、CURRENT USER の代わりに CURRENT_USER を指定することもできます。既存の行の場合、値は ALTER TABLE ステートメントの CURRENT_USER、SESSION_USER、または SYSTEM_USER になります。

CURRENT SCHEMA

INSERT、UPDATE、または LOAD の実行時における

CURRENT SCHEMA 特殊レジスタの値を、その列のデフォルト値として指定します。CURRENT SCHEMA を指定した場合、その列のデータ・タイプは、CURRENT SCHEMA 特殊レジスタの長さ属性よりも短い文字ストリングであってはなりません。既存の行の場合は、ALTER TABLE ステートメントが処理される時点における CURRENT SCHEMA 特殊レジスタの値です。

NULL

その列のデフォルト値として NULL を指定します。NOT NULL の指定がある場合には、DEFAULT NULL を同じ列定義に指定してはなりません。

cast-function

この形式のデフォルト値は、特殊タイプ (distinct type)、BLOB、または日時 (DATE、TIME、または TIMESTAMP) データ・タイプとして定義された列に対してのみ使用することができます。特殊タイプの場合、BLOB や日時タイプに基づく例外があり、関数名が列の特殊タイプの名前に一致していなければなりません。スキーマ名で修飾されている場合には、その特殊タイプのスキーマ名と同じでなければなりません。修飾されていない場合には、関数の解決に用いるスキーマ名は特殊タイプのスキーマ名と同じでなければなりません。日時タイプに基づく特殊タイプで、デフォルト値が定数の場合、必ず関数を使用する必要があります。さらに、その関数名は、暗黙または明示のスキーマ名 SYSIBM を持つ特殊タイプのソース・タイプ名に一致していなければなりません。他の日時列の場合は、対応する日時関数も使用できます。BLOB に基づく BLOB または特殊タイプの場合も、関数を使用する必要があります。その関数名は、暗黙または明示のスキーマ名 SYSIBM を持つ BLOB でなければなりません。

constant

引数として定数を指定します。指定する定数は、特殊タイプのソース・タイプに関する定数の規則 (特殊タイプでない場合は、データ・タイプに関する定数の規則) に従っていなければなりません。cast-function が BLOB の場合には、定数としてストリング定数を指定する必要があります。

datetime-special-register

CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP を指定します。列の特殊タイプのソース・タイプは、指定した特殊レジスタに対応するデータ・タイプでなければなりません。

user-special-register

CURRENT USER、SESSION_USER、または SYSTEM_USER を指定します。列の特殊タイプのソース・タイプのデータ・タイプは、少なくとも 8 バイトの長さの

ALTER TABLE

ストリング・データ・タイプでなければなりません。
cast-function が BLOB の場合には、長さ属性が 8 バイト以上でなければなりません。

CURRENT SCHEMA

CURRENT SCHEMA 特殊レジスタの値を指定します。列の特殊タイプのソース・タイプのデータ・タイプは、CURRENT SCHEMA 特殊レジスタの長さ属性よりも短い文字ストリングであってはなりません。 *cast-function* が BLOB の場合には、長さ属性が 8 バイト以上でなければなりません。

EMPTY_CLOB(), EMPTY_DBCLOB(), または EMPTY_BLOB()

その列のデフォルト値として長さゼロのストリングを指定します。その列は、この関数の結果データ・タイプに対応するデータ・タイプを持っている必要があります。

指定した値が無効な場合、エラー (SQLSTATE 42894) が戻されます。

GENERATED

DB2 が列の値を生成することを指定します。

ALWAYS

行が表に挿入されるときや、*generation-expression* の結果値が変更されるたびに、DB2 が常に列の値を生成することを指定します。この式の結果は、表に保管されます。データ伝搬や、アンロードおよび再ロード操作を実行しているものでなければ、GENERATED ALWAYS が推奨されるオプションです。GENERATED ALWAYS は、生成列に必須指定のオプションです。

BY DEFAULT

行が表に挿入されたり、更新されるときに、明示的に値を指定しないかぎり、列に DEFAULT を指定して DB2 が列に値を生成することを指定します。データ伝搬を使用したり、アンロードおよび再ロードを実行したりするときは、BY DEFAULT が推奨されるオプションです。

AS (*generation-expression*)

列定義が式に基づくことを指定します。SET INTEGRITY ステートメントを OFF オプションとともに使用して、表を SET INTEGRITY ペンディング状態にする必要があります。ALTER TABLE ステートメントの後、IMMEDIATE CHECKED および FORCE GENERATED オプションを伴う SET INTEGRITY ステートメントを使用して、新しい式に対してこの列にあるすべての値を更新および検査しなければなりません。 *generation-expression* による列の指定の詳細については、『CREATE TABLE』を参照してください。

COMPRESS SYSTEM DEFAULT

システム・デフォルト値 (つまり、特定の値が指定されない場合にデータ・タイプとして使用されるデフォルト値) が最小限のスペースを使用して保管されるように指定します。VALUE COMPRESSION 節が指定されていない

場合には警告が出され (SQLSTATE 01648)、システム・デフォルト値が最小限のスペースを使用して保管されるようにはなりません。

システム・デフォルト値がこのような方法で保管されると、列に対する許可や更新操作の際に余分な検査が行われるために、若干パフォーマンスが低下します。

基本データ・タイプは、DATE、TIME、TIMESTAMP、XML、または構造化データ・タイプであってはなりません (SQLSTATE 42842)。基本データ・タイプが可変長ストリングの場合には、この節は無視されます。表が VALUE COMPRESSION に設定されている場合は、長さ 0 のストリング値は自動的に圧縮されます。

COLUMN SECURED WITH *security-label-name*

表に関連するセキュリティー・ポリシーに対応して存在するセキュリティー・ラベルを識別します。名前は非修飾でなければなりません (SQLSTATE 42601)。表にはセキュリティー・ポリシーが関連付けられている必要があります (SQLSTATE 55064)。

ADD *unique-constraint*

ユニーク制約または主キー制約を定義します。主キー制約またはユニーク制約を、副表に追加することはできません (SQLSTATE 429B3)。階層最上部のスーパー表の場合、制約はその表および関連する副表すべてに適用されます。

CONSTRAINT *constraint-name*

主キー制約、またはユニーク制約の名前を指定します。詳細については、『CREATE TABLE』で *constraint-name* を参照してください。

UNIQUE (*column-name...*)

指定した列で構成されるユニーク・キーを定義します。指定する列は NOT NULL として定義されていなければなりません。各 *column-name* (列名) は、表の列を指定するものでなければなりません。また、同じ列を複数回指定することはできません。名前は非修飾でなければなりません。指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。保管される列の長さについては、『CREATE TABLE』の『バイト・カウント』を参照してください。キー長の制限については、『SQL の制限』を参照してください。列の長さ属性がページ・サイズに対する索引キーの長さの上限を超えない場合でも、LOB、これらのタイプのうちのいずれかに基づく特殊タイプ、または構造化タイプは、ユニーク・キーの一部として使用できません (SQLSTATE 54008)。ユニーク・キーにある列セットは、主キーまたは他のユニーク・キーの列セットと同じにすることはできません (SQLSTATE 01543)。(LANGLEVEL が SQL92E または MIA の場合は、エラーが戻されます。SQLSTATE 42891) 指定した列セットに存在する値は、ユニークである必要があります (SQLSTATE 23515)。

既存の索引がユニーク・キー定義と一致しているかどうか判別するために、チェックが実行されます (索引内の INCLUDE 列はすべて無視されます)。列の順序や方向 (ASC/DESC) の指定に関係なく、同じ列セットを指定していると、索引定義は一致します。一致する索引定義が見つかり、その索引の記述は、システムによりその索引が必要であることを示すように変更され、索引がユニークでない場合はユニーク索引に変更されます (ユニーク性を確実にした後)。その表に一致する索引が複数ある場合、既存のユニーク

索引が選択されます (選択は任意に行われます)。一致する索引が見つからない場合は、CREATE TABLE で説明するように、その列に対してユニーク双方向索引が自動的に作成されます。ユニーク制約に関連した索引名の詳細については、注を参照してください。

PRIMARY KEY ...(column-name,)

指定された列で構成される主キーを定義します。各 *column-name* (列名) は、表の列を指定していなければなりません。また、同じ列を複数回指定することはできません。名前は非修飾でなければなりません。指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。保管される列の長さについては、『CREATE TABLE』の『バイト・カウント』を参照してください。キー長の制限については、『SQL の制限』を参照してください。表には主キーがあってはならず、指定する列は NOT NULL として定義されているものでなければなりません。列の長さ属性がページ・サイズの索引キーの長さの上限を超えない場合でも、LOB、これらのタイプのうちのいずれかに基づく特殊タイプ、または構造化タイプは、主キーの一部として使用できません (SQLSTATE 54008)。主キーの列セットは、ユニーク・キーの列セットと同じであってはなりません (SQLSTATE 01543)。 (LANGLEVEL が SQL92E または MIA の場合は、エラーが戻されます。SQLSTATE 42891) 指定した列セットに存在する値は、ユニークである必要があります (SQLSTATE 23515)。

既存の索引が主キー定義と一致しているかどうか判断するために、チェックが実行されます (索引内の INCLUDE 列はすべて無視されます)。列の順序や方向 (ASC/DESC) の指定に関係なく、同じ列セットを指定していると、索引定義は一致します。一致する索引定義が見つかったら、その索引の記述は、その索引が 1 次索引である (システムが必要としている) ことを示すように変更され、索引がユニークでない場合はユニーク索引に変更されます (ユニーク性を確実にした後)。その表に一致する索引が複数ある場合、既存のユニーク索引が選択されます (選択は任意に行われます)。一致する索引が見つからない場合は、CREATE TABLE で説明するように、その列に対してユニーク双方向索引が自動的に作成されます。ユニーク制約に関連した索引名の詳細については、注を参照してください。

1 つの表には、主キーを 1 つだけ定義することができます。

ADD referential-constraint

参照制約を定義します。『CREATE TABLE』の *referential-constraint* を参照してください。

ADD check-constraint

チェック制約または機能従属関係を定義します。『CREATE TABLE』の *check-constraint* を参照してください。

ADD distribution-clause

分散キーを定義します。表は、単一パーティションのデータベース・パーティション・グループにある表スペースに定義する必要があります (SQLSTATE 55037)、既存の分散キーがあってはなりません (SQLSTATE 42889)。分散キーが表にすでに存在している場合には、新しい分散キーを追加する前に既存のキーをドロップする必要があります。分散キーは、副表となっている表に追加することはでき

ず (SQLSTATE 428DH)、データ・タイプが XML になっている列を持つ表に追加することもできません (SQLSTATE 42997)。

DISTRIBUTE BY HASH (*column-name...*)

指定した列を使用して、分散キーを定義します。各 *column-name* (列名) は、表の列を指定していなければなりません。また、同じ列を複数回指定することはできません。名前は非修飾でなければなりません。列のデータ・タイプが BLOB、CLOB、DBCLOB、XML、これらのいずれかのタイプの特殊タイプ、または構造化タイプである場合、分散キーの一部として列を使用することはできません。

ADD RESTRICT ON DROP

表をドロップできないように、また、表を含む表スペースをドロップできないように指定します。

ADD MATERIALIZED QUERY

materialized-query-definition

照会の最適化で使用するために、正規表をマテリアライズ照会表に変更します。 *table-name* で指定する表には、以下の条件があります。

- マテリアライズ照会表として以前に定義されてはならない。
- 型付き表であってはならない。
- 何らかの定数、ユニーク索引、またはトリガーが定義されてはならない。
- キャッシング使用不可とマークされたニックネームを参照してはならない。
- 他のマテリアライズ照会表の定義で参照されてはならない。
- 照会の最適化に使用可能になっているビューの定義で参照されてはならない。

表名が基準に適合しない場合、エラーが戻されます (SQLSTATE 428EW)。

fullselect

表の基礎となる照会を定義します。既存の表の列は、 *fullselect* の結果列と以下のような関係になければなりません (SQLSTATE 428EW)。

- 列の数が同数でなければなりません。
- 全く同じデータ・タイプでなければなりません。
- 同じ順序を示す位置に同じ列名がなければなりません。

マテリアライズ照会表に *fullselect* を指定することについての詳細は、『CREATE TABLE』を参照してください。追加の制限事項としては、 *table-name* は全選択で直接的にも間接的にも参照できません。

refreshable-table-options

マテリアライズ照会表を変更するためのリフレッシュ可能オプションを指定します。

DATA INITIALLY DEFERRED

REFRESH TABLE または SET INTEGRITY ステートメントを使用して、表のデータを妥当性検査する必要があります。

REFRESH

表のデータを保守する方法を示します。

DEFERRED

REFRESH TABLE ステートメントを使っていつでも表のデータをリフレッシュできます。表のデータには、REFRESH TABLE ステートメント処理時のスナップショットである照会結果が反映されるにすぎません。この属性を定義したマテリアライズ照会表には、INSERT、UPDATE、または DELETE ステートメントを使用できません (SQLSTATE 42807)。

IMMEDIATE

DELETE、INSERT、または UPDATE の一部として基礎表に加えられた変更は、マテリアライズ照会表にカスケードされます。その場合、表の内容は、どのポイント・イン・タイム指定でも、指定した subselect (副選択) を処理する場合と同じ内容になります。この属性を定義したマテリアライズ照会表には、INSERT、UPDATE、または DELETE ステートメントを使用できません (SQLSTATE 42807)。

ENABLE QUERY OPTIMIZATION

マテリアライズ照会表を照会の最適化に使用できるようにします。

DISABLE QUERY OPTIMIZATION

マテリアライズ照会表を照会の最適化に使用しません。それでもその表を直接照会することはできます。

MAINTAINED BY

マテリアライズ照会表のデータが、システム、ユーザー、またはレプリケーション・ツールのいずれによって保守されるかを指定します。

SYSTEM

マテリアライズ照会表のデータがシステムによって保守されるように指定します。

USER

マテリアライズ照会表のデータがユーザーによって保守されるように指定します。ユーザーは、ユーザー保守済みマテリアライズ照会表に対して、更新、削除、また挿入操作を許可されます。システム保守のマテリアライズ照会表で使用される REFRESH TABLE ステートメントは、ユーザー保守のマテリアライズ照会表に対しては呼び出せません。REFRESH DEFERRED マテリアライズ照会表だけが、MAINTAINED BY USER として定義できます。

FEDERATED_TOOL

マテリアライズ照会表のデータがレプリケーション・ツールによって保守されるように指定します。システム保守のマテリアライズ照会表で使用される REFRESH TABLE ステートメントは、フェデレーテッド・ツール保守のマテリアライズ照会表に対しては呼び出せません。REFRESH DEFERRED のマテリアライズ照会表だけが、MAINTAINED BY FEDERATED_TOOL として定義されることが可能です。

ALTER FOREIGN KEY *constraint-name*

参照制約 *constraint-name* の制約属性を変更します。 *constraint-name* は既存の参照制約を指定する必要があります (SQLSTATE 42704)。

ALTER CHECK *constraint-name*

チェック制約または機能従属関係 *constraint-name* の制約属性を変更します。 *constraint-name* は、既存のチェック制約または機能従属関係を指定していなければなりません (SQLSTATE 42704)。

constraint-alteration

参照制約またはチェック制約に関連付けられた属性の変更のオプションです。

ENABLE QUERY OPTIMIZATION または DISABLE QUERY OPTIMIZATION

適切な状況下で、照会の最適化のために、制約または機能従属関係を使用できるかどうかを指定します。

ENABLE QUERY OPTIMIZATION

制約が真であると想定され、照会の最適化に使用できます。

DISABLE QUERY OPTIMIZATION

制約を照会の最適化に使用できません。

ENFORCED または NOT ENFORCED

挿入、更新、削除などの通常の操作中に、データベース・マネージャーによって制約が課せられるかどうかを指定します。

ENFORCED

制約を ENFORCED に変更します。関数的依存関係に ENFORCED を指定することはできません (SQLSTATE 42621)。

NOT ENFORCED

制約を NOT ENFORCED に変更します。この制約に適合することが分かっている表データだけに、制約を指定してください。データが実際には制約に準拠していない場合、照会結果が予測不能になる可能性があります。

ALTER *column-alteration*

列の定義を変更します。指定した属性だけが変更され、他の属性は変更されません。型付き表の列は変更できません (SQLSTATE 428DH)。

column-name

変更される列の名前を指定します。 *column-name* は、表の既存列を指定するものでなければなりません (SQLSTATE 42703)。名前は非修飾でなければなりません。同じ ALTER TABLE ステートメントで追加、変更、またはドロップされる列を指す名前は指定できません (SQLSTATE 42711)。

SET DATA TYPE *altered-data-type*

列の新規データ・タイプを指定します。新規データ・タイプは、列の既存データ・タイプと互換性のあるものでなくてはなりません (SQLSTATE 42837)。組み込みデータ・タイプの詳細については、『CREATE TABLE』を参照してください。98 ページの表 14 に、互換性のあるデータ・タイプを列挙します。「REORG の推奨の有無」という列は、表への完全アクセスが再度可能になる前に表の再編成が必要になる、データ・タイプの変更を示

ALTER TABLE

します (SQLSTATE 57016)。この場合、変更される列が、XML データ・タイプの列を含む表の一部となることはできません (SQLSTATE 42997)。

ID 列のデータ・タイプは変更できません (SQLSTATE 42997)。

表ではデータのキャプチャーを使用可能にすることはできません (SQLSTATE 42997)。

長さ、精度、またはスケールは、既存の値以上を指定できます (SQLSTATE 42837)。

表 14. 互換データ・タイプ

変更前のタイプ	変更後のタイプ	表パーティション・キー列に有効	MDC 編成ディメンション列に有効	分散キー列に有効	REORG の推奨
SMALLINT	INTEGER	はい	はい	はい	はい
SMALLINT	BIGINT	はい	はい	はい	はい
SMALLINT	DECIMAL(<i>p,m</i>); <i>p-m</i> >4	はい	はい	いいえ	はい
SMALLINT	REAL	はい	はい	いいえ	はい
SMALLINT	DOUBLE	はい	はい	いいえ	はい
SMALLINT	DECFLOAT(16) または DECFLOAT(34)	はい	はい	いいえ	はい
INTEGER	BIGINT	はい	はい	はい	はい
INTEGER	DECIMAL(<i>p,m</i>); <i>p-m</i> >9	はい	はい	いいえ	はい
INTEGER	DOUBLE	はい	はい	いいえ	はい
INTEGER	DECFLOAT(16) または DECFLOAT(34)	はい	はい	いいえ	はい
BIGINT	DECIMAL(<i>p,m</i>); <i>p-m</i> >19	はい	はい	いいえ	はい
BIGINT	DECFLOAT(34)	はい	はい	いいえ	はい
REAL	DOUBLE	はい	はい	はい	はい
REAL	DECFLOAT(16) または DECFLOAT(34)	はい	はい	いいえ	はい
DOUBLE	DECFLOAT(16) または DECFLOAT(34)	はい	はい	いいえ	はい
DECIMAL(<i>n,m</i>)	DECIMAL(<i>p,q</i>); <i>p</i> >= <i>n</i> ; <i>q</i> >= <i>m</i> ; (<i>p-q</i>)>= <i>(n-m)</i>	はい	はい	いいえ	はい
DECIMAL (<i>n, m</i>); <i>n</i> <= 15	DECFLOAT(16)	はい	はい	いいえ	はい
DECIMAL(<i>n,m</i>)	DECFLOAT(34)	はい	はい	いいえ	はい
DECFLOAT(16)	DECFLOAT(34)	はい	はい	はい	はい
CHARACTER(<i>n</i>)	CHARACTER(<i>n+x</i>)	いいえ	はい	はい	はい
CHARACTER(<i>n</i>)	VARCHAR(<i>n+x</i>)	いいえ	はい	はい	はい
VARCHAR(<i>n</i>)	CHARACTER(<i>n+x</i>)	いいえ	はい	はい	はい
VARCHAR(<i>n</i>)	VARCHAR(<i>n+x</i>)	いいえ	はい	はい	いいえ
GRAPHIC(<i>n</i>)	GRAPHIC(<i>n+x</i>)	いいえ	はい	はい	はい
GRAPHIC(<i>n</i>)	VARGRAPHIC(<i>n+x</i>)	いいえ	はい	はい	はい
VARGRAPHIC(<i>n</i>)	VARGRAPHIC(<i>n+x</i>)	いいえ	はい	はい	いいえ
VARGRAPHIC(<i>n</i>)	GRAPHIC(<i>n+x</i>)	いいえ	はい	はい	はい
BLOB (<i>n</i>)	BLOB(<i>n+x</i>)	n/a	n/a	n/a	はい

表 14. 互換データ・タイプ (続き)

変更前のタイプ	変更後のタイプ	表パーティション・キー列に有効	MDC 編成ディメンション列に有効	分散キー列に有効	REORG の推奨
CLOB (n)	CLOB(n+x)	n/a	n/a	n/a	はい
DBCLOB (n)	DBCLOB(n+x)	n/a	n/a	n/a	はい

列を変更する場合、すべての列のバイト・カウントの合計が、最大レコード・サイズを超えてはなりません (SQLSTATE 54010)。ユニーク制約または索引で列を使用する場合、新しい長さは、ユニーク制約または索引の列の保管長の合計が、ページ・サイズに対応する索引キーの長さの上限を超えないようにしなければなりません (SQLSTATE 54008)。保管される列の長さについては、『CREATE TABLE』の『バイト・カウント』を参照してください。キー長の制限については、『SQL の制限』を参照してください。

表 15. 列の変更のカスケード効果

操作	影響
ビューまたはチェック制約によって参照される列を変更する。	変更処理中にオブジェクトが再生成されます。ビューの場合、変更操作の後、オブジェクトのセマンティクスが変わり、オブジェクトについての関数やメソッドの解決方法が変わる可能性もあります。チェック制約の場合、変更操作の結果としてオブジェクトのセマンティクスが変わるようであれば、操作は失敗します。
従属パッケージ、トリガー、または SQL ルーチンを持つ表の列を変更する。	オブジェクトが無効とマークされ、次回の使用時に再度有効性を確かめられます。
分解が使用可能になった XSROBJECT によって参照される表内の列のタイプを変更する。	XSROBJECT が分解操作不能とマークされます。XSROBJECT の分解を再度使用可能にするには、マッピングの再調整が必要になる可能性があります。それに続いて、XSROBJECT に対して ALTER XSROBJECT ENABLE DECOMPOSITION ステートメントを実行します。
グローバル変数のデフォルトの式で参照される列の変更	グローバル変数のデフォルトの式は変更処理中に妥当性検査されます。デフォルトの式で使用されるユーザー定義関数を解決できない場合、操作は失敗します。

SET generated-column-alteration

列の値を生成するために使用される技法を指定します。これは、特定のデフォルト値、式、または ID 列としての列の定義という形を取ることができます。列の既存のデフォルトが、別の生成技法に由来する場合は、そのデフォルトをドロップする必要があります。それは、いずれかの DROP 節を使用して、同じ *column-alteration* の中で行うことができます。

default-clause

変更される列の新規デフォルト値を指定します。その列は ID 列としてすでに定義されているはならず、生成式が定義されているはなりません

(SQLSTATE 42837)。指定されるデフォルト値は、『割り当ておよび比較』で説明されている割り当ての規則に従って、その列に割り当てることができる値でなければなりません。デフォルト値を変更しても、既存の行については、この列に関連した値が変更されるわけではありません。

GENERATED ALWAYS または GENERATED BY DEFAULT

データベース・マネージャーがその列の値をいつ生成するかを指定します。GENERATED BY DEFAULT は、値が提供されないときか、列への割り当てに DEFAULT キーワードが使用されたときだけ、値が生成されることを指定します。GENERATED ALWAYS は、データベース・マネージャーが常にその列の値を生成することを指定します。

GENERATED BY DEFAULT を *generation-expression* と一緒に指定することはできません。

identity-options

その列が表の ID 列であることを指定します。その列は ID 列としてすでに定義されているはずで、生成式を持つことはできず、明示的デフォルトを持つこともできません (SQLSTATE 42837)。1 つの表には 1 つしか ID 列があってはなりません (SQLSTATE 428C1)。その列は NULL 可能でないものとして指定される必要があります (SQLSTATE 42997)、列に関連したデータ・タイプは、位取りがゼロの完全な数値データ・タイプでなければなりません (SQLSTATE 42815)。完全な数値データ・タイプは次のいずれかです: SMALLINT、INTEGER、BIGINT、DECIMAL、位取りがゼロの NUMERIC、またはこれらのいずれかのタイプに基づく特殊タイプ。IDENTITY オプションの詳細については、『CREATE TABLE』を参照してください。

AS (*generation-expression*)

列定義が式に基づくことを指定します。その列は生成式ですすでに定義されているはずで、ID 列であってはならず、明示的デフォルトを持つこともできません (SQLSTATE 42837)。

generation-expression は、生成される列を定義する際に適用されるのと同じ規則に適合する必要があります。 *generation-expression* の結果データ・タイプは、列のデータ・タイプに割り当て可能でなければなりません (SQLSTATE 42821)。その列は、分散キー列で、または ORGANIZE BY 節の中で、参照されているはずで、または (SQLSTATE 42997)。

SET EXPRESSION AS (*generation-expression*)

列の式を、指定された *generation-expression* に変更します。SET EXPRESSION AS では、SET INTEGRITY ステートメントを OFF オプションとともに使用して、表を SET INTEGRITY ペンディング状態にする必要があります。ALTER TABLE ステートメントの後、IMMEDIATE CHECKED および FORCE GENERATED オプションを伴う SET INTEGRITY ステートメントを使用して、新しい式に対してこの列にあるすべての値を更新および検査しなければなりません。列は、式に基づいて生成される列として定義されていなければなりません (SQLSTATE 42837)。また表の PARTITIONING KEY、DIMENSIONS、または KEY SEQUENCE 節に現れないようにする必要があります (SQLSTATE 42997)。

generation-expression は、生成される列を定義する際に適用されるのと同じ規則に適合する必要があります。 generation-expression の結果データ・タイプは、列のデータ・タイプに割り当て可能でなければなりません (SQLSTATE 42821)。

SET INLINE LENGTH *integer*

既存の構造化タイプ列または XML タイプ列のインライン長を変更します。インライン長は、基本表の行に格納する構造化タイプまたは XML タイプのインスタンスの最大バイト・サイズを指示します。基本表の行にインラインで格納できない構造化タイプまたは XML タイプのインスタンスは、LOB 値が処理されるのに似た方法で、別個に格納されます。

column-name のデータ・タイプは、構造化タイプまたは XML タイプでなければなりません (SQLSTATE 42842)。

構造化タイプ列のデフォルトのインライン長は、このタイプのインライン長になります (明示的に指定するか、または CREATE TYPE ステートメント内のデフォルトとして)。構造化タイプのインライン長が 292 未満の場合、列のインライン長には値 292 が使われます。

明示的なインライン長の値は増やすことのみ可能で (SQLSTATE -1)、 292 以上 32672 以下でなければなりません (SQLSTATE 54010)。

列を変更する場合、すべての列のバイト・カウントの合計が、最大レコード・サイズを超えてはなりません (SQLSTATE 54010)。

既に他の行とは別に保管されたデータは、このステートメントにより基本表の行にインラインで格納されるようになることはありません。構造化タイプ列のインライン長を変更した利点を生かすには、列のインライン長を変更した後に指定された表に対して REORG コマンドを呼び出します。既存の表内の XML タイプ列のインライン長を変更した利点を生かすには、UPDATE ステートメントによってすべての行を更新する必要があります。 REORG コマンドは、XML 文書の行保管に対して影響を及ぼしません。

SET NOT NULL

列に NULL 値を含むことができないよう指定します。表の既存の行内のこの列に関しては、どの値も NULL 値にすることはできません (SQLSTATE 23502)。この節は、列が参照制約の外部キーで SET NULL の DELETE 規則によって指定されていて、外部キー内に、他にまったく NULL 可能な列が存在しない場合には許可されません (SQLSTATE 42831)。列に関してこの属性を変更すると、その後の表アクセスが許可される前に表の再編成が必要になります (SQLSTATE 57016)。この操作は表データの妥当性検査を必要とするため、表が REORG ペンディング状態になっている時には実行できないという点に注意してください (SQLSTATE 57016)。変更される列は、XML データ・タイプの列を含む表の一部となることはできません (SQLSTATE 42997)。表ではデータのキャプチャーを使用可能にすることはできません (SQLSTATE 42997)。

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

列が表のタイム・スタンプ列であることを指定します。行が挿入されるたびにその列に値が生成され、列が更新されるたびにその行に値が生成されます。 ROW CHANGE TIMESTAMP 列に生成される値は、その行の挿入または更新の時刻に対応するタイム・スタンプです。1 つのステートメントに

よって複数の行が挿入または更新される場合、ROW CHANGE
TIMESTAMP 列の値は行ごとに異なる可能性があります。

ROW CHANGE TIMESTAMP 列は 1 つの表内に 1 つだけ含めることができます (SQLSTATE 428C1)。 *data-type* を指定する場合は、TIMESTAMP でなければなりません (SQLSTATE 42842)。ROW CHANGE TIMESTAMP 列は DEFAULT 節を持つことができません (SQLSTATE 42623)。ROW CHANGE TIMESTAMP 列には NOT NULL を指定する必要があります (SQLSTATE 42831)。

SET GENERATED ALWAYS または GENERATED BY DEFAULT

データベース・マネージャーがその列の値をいつ生成するかを指定します。GENERATED BY DEFAULT は、値が提供されないときか、列への割り当てに DEFAULT キーワードが使用されたときだけ、値が生成されることを指定します。GENERATED ALWAYS は、データベース・マネージャーが常にその列の値を生成することを指定します。その列は、ID 列に基づく生成列としてすでに定義されている、すなわち、AS IDENTITY 節によって定義されている必要があります (SQLSTATE 42837)。

identity-alteration

ID 列の識別属性を変更します。

SET INCREMENT BY *numeric-constant*

連続した ID 列値のインターバルを指定します。次に生成される ID 列の値は、最後に割り当てられた値に増分を適用することによって決まります。列はあらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。

この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815)、長精度整数定数の値を超えることはありません (SQLSTATE 42820)。

この値が負の場合、ALTER ステートメント以降は降順になります。この値が 0 の場合、または正の場合は、ALTER ステートメント以降は昇順になります。

SET NO MINVALUE または MINVALUE *numeric-constant*

降順 ID 列が値の生成を循環または停止する最小値、あるいは最大値に達した後、昇順 ID 列が循環する最小値を指定します。列は、指定した表の中に存在していなければならず (SQLSTATE 42703)、あらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。

NO MINVALUE

昇順シーケンスの場合、値は元の開始値です。降順シーケンスの場合、列のデータ・タイプの最小値になります。

MINVALUE *numeric-constant*

最小値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815) が、最大値以下でなければなりません (SQLSTATE 42815)。

SET NO MAXVALUE または **MAXVALUE** *numeric-constant*

昇順 ID 列が値の生成を循環または停止する最大値、あるいは最小値に達した後、降順 ID 列が循環する最大値を指定します。列は、指定した表の中に存在していなければならず (SQLSTATE 42703)、あらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。

NO MAXVALUE

昇順シーケンスの場合、値は列のデータ・タイプの最大値です。降順シーケンスの場合、値は最初の開始値です。

MAXVALUE *numeric-constant*

最大値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815) が、最小値よりも大きいかまたは等しくなければなりません (SQLSTATE 42815)。

SET NO CYCLE または **CYCLE**

その最大値または最小値が生成された後、この ID 列が値の生成を続行するかどうかを指定します。列は、指定した表の中に存在していなければならず (SQLSTATE 42703)、あらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。

NO CYCLE

最大値または最小値に達した後、ID 列について値が生成されないことを指定します。

CYCLE

最大値または最小値に達した後、この列について値の生成が続行されることを指定します。このオプションが使用されると、昇順 ID 列が最大値に達した後は、その最小値が生成されます。降順 ID 列が最小値に達した後は、その最大値が生成されます。ID 列の最大値および最小値は、循環に使用される範囲を決定します。

CYCLE が有効な場合、ID 列について重複する値が生成される可能性があります。固有値が必要であれば (実際には必要ありません)、ID 列を使用して 1 列のユニーク索引を定義することによって、ユニーク性を確実にしてください。このような ID 列にユニーク索引が存在し、ユニークではない値が生成されると、エラーが起こります (SQLSTATE 23505)。

SET NO CACHE または **CACHE** *integer-constant*

特定の事前割り振り値を、高速アクセスできるようメモリーに保存するかどうかを指定します。これはパフォーマンスおよびチューニング・オプションです。列はあらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。

NO CACHE

ID 列の値を事前割り振りしないことを指定します。データ共有環境では、IDENTITY 値は要求の順序で生成されなければならず、NO CACHE オプションを使用する必要があります。

このオプションが指定されると、ID 列の値はキャッシュに保管されません。この場合、新しい ID 値が要求されるたびに、ログに対して同期入出力が行われます。

CACHE *integer-constant*

事前割り振りされ、メモリーに保管される IDENTITY シーケンスの値の数を指定します。ID 列について値が生成される場合、値を事前割り振りしてキャッシュに保管しておくこと、ログへの同期入出力が少なくなります。

ID 列に新しい値が必要でも未使用の値がキャッシュにない場合、値の割り振りはログへの入出力を待機する必要があります。ただし、ID 列に新しい値が必要で、未使用の値がキャッシュにあれば、その ID 値の割り振りが、ログへの入出力なしで素早く行われます。

システム障害に起因するものであっても通常のものであっても、データベース非活動化が起これば、コミットされたステートメントで使用されていないキャッシュ済みシーケンス値はすべて失われます(使用されなくなります)。CACHE オプションに指定する値は、システム障害の際に失われても構わない ID 列の値の最大数です。

最小値は 2 です (SQLSTATE 42815)。

SET NO ORDER または **ORDER**

要求の順序で ID 列の値が生成されるかどうかを指定します。列は、指定した表の中に存在していなければならず (SQLSTATE 42703)、あらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。

NO ORDER

要求の順序で ID 列の値を生成する必要がないことを指定します。

ORDER

要求の順序で ID 列の値が生成されることを指定します。

RESTART または **RESTART WITH** *numeric-constant*

ID 列に関連付けられたシーケンスの状態をリセットします。WITH *numeric-constant* が指定されていないと、ID 列のシーケンスは、作成されたときに開始値として (暗黙的または明示的のいずれかで) 定義された値で再開されます。

列は、指定した表の中に存在していなければならず (SQLSTATE 42703)、あらかじめ IDENTITY 属性で定義されていなければなりません (SQLSTATE 42837)。RESTART は、START WITH の元の値を変更することはありません。

numeric-constant は数値定数で、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815)。*numeric-constant* が列の次の値として使用されます。

DROP IDENTITY

列の識別属性をドロップし、その列を単なる数値データ・タイプ列にします。その列が ID 列でない場合は、DROP IDENTITY は許可されません (SQLSTATE 42837)。

DROP EXPRESSION

列の生成された式属性をドロップし、その列を非生成の列にします。その列が、生成された式列でない場合は、DROP EXPRESSION は許可されません (SQLSTATE 42837)。

DROP DEFAULT

列の現行デフォルトをドロップします。指定される列には、デフォルト値がなければなりません (SQLSTATE 42837)。

DROP NOT NULL

列の NOT NULL 属性をドロップし、列が NULL 値を持つことを許可します。この節は、列が主キーで指定されている場合、また表のユニーク制約で指定されている場合には許可されません (SQLSTATE 42831)。列に関してこの属性を変更すると、その後の表アクセスが許可される前に表の再編成が必要になります (SQLSTATE 57016)。変更される列は、XML データ・タイプの列を含む表の一部となることはできません (SQLSTATE 42997)。表ではデータのキャプチャーを使用可能にすることはできません (SQLSTATE 42997)。

ADD SCOPE

有効範囲が未定義である既存の参照タイプ列に、有効範囲を追加します (SQLSTATE 428DK)。変更する表が型付き表である場合、列をスーパー表から継承することはできません (SQLSTATE 428DJ)。

typed-table-name

型付き表の名前。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-table-name* のタイプを表します (SQLSTATE 428DM)。値が *typed-table-name* の既存行を実際に参照していることを確認するための、 *column-name* の既存値の検査は行われません。

typed-view-name

型付きビューの名前。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-view-name* のタイプを表します (SQLSTATE 428DM)。値が *typed-view-name* の既存行を実際に参照していることを確認するための、 *column-name* の既存値の検査は行われません。

COMPRESS

この列のデフォルト値をさらに効率よく保管するかどうかを指定します。

SYSTEM DEFAULT

システム・デフォルト値 (つまり、特定の値が指定されない場合にデータ・タイプとして使用されるデフォルト値) が最小限のスペースを使用して保管されるように指定します。活動化された VALUE COMPRESSION 属性に表が設定されていない場合には、警告が出され (SQLSTATE 01648)、システム・デフォルト値が最小限のスペースを使用して保管されるようにはなりません。

システム・デフォルト値がこのような方法で保管されると、列上での挿入や更新操作の際に余分な検査が行われるために、若干パフォーマンスが低下します。

ALTER TABLE

列の既存のデータは変更されません。既存のデータを使用可能にして、最小限のスペースを使用してシステム・デフォルト値を保管することの利点を生かすため、オフラインでの表再編成を考慮してください。

OFF

システム・デフォルト値が列の正規値として保管されるように指定します。列の既存のデータは変更されません。既存のデータの変更のために、オフラインでの再編成をお勧めします。

基本データ・タイプは、DATE、TIME、または TIMESTAMP であってはなりません (SQLSTATE 42842)。基本データ・タイプが可変長ストリングの場合には、この節は無視されます。表が VALUE COMPRESSION に設定されている場合は、長さ 0 のストリング値は自動的に圧縮されます。

変更する表が型付き表である場合、列をスーパー表から継承することはできません (SQLSTATE 428DJ)。

SECURED WITH *security-label-name*

表に関連するセキュリティー・ポリシーに対応して存在するセキュリティー・ラベルを識別します。名前を修飾してはなりません (SQLSTATE 42601)。表にはセキュリティー・ポリシーが関連付けられている必要があります (SQLSTATE 55064)。

DROP COLUMN SECURITY

列を無保護列に変更します。

DROP PRIMARY KEY

主キーの定義、およびその主キーに従属するすべての参照制約をドロップします。表には主キーがなければなりません (SQLSTATE 42888)。

DROP FOREIGN KEY *constraint-name*

制約名が *constraint-name* の参照制約をドロップします。 *constraint-name* には参照制約を指定する必要があります (SQLSTATE 42704)。参照制約のドロップにより起こることについては、注を参照してください。

DROP UNIQUE *constraint-name*

ユニーク制約 *constraint-name* の定義、およびこのユニーク制約に従属するすべての参照制約をドロップします。 *constraint-name* は既存のユニーク制約を指定する必要があります (SQLSTATE 42704)。ユニーク制約のドロップにより起こることについては、注を参照してください。

DROP CHECK *constraint-name*

制約名が *constraint-name* のチェック制約をドロップします。 *constraint-name* は、表に定義されている既存のチェック制約を指定していなければなりません (SQLSTATE 42704)。

DROP CONSTRAINT *constraint-name*

制約名が *constraint-name* の制約をドロップします。 *constraint-name* は、表に定義されている既存のチェック制約、参照制約、主キー、またはユニーク制約のいずれかを指定していなければなりません (SQLSTATE 42704)。制約のドロップにより起こることについては、注を参照してください。

DROP COLUMN

指定の列を表から削除します。型付き表を使用することはできません (SQLSTATE 428DH)。表ではデータのキャプチャーを使用可能にすることはでき

きません (SQLSTATE 42997)。列をドロップする場合、更新、挿入、削除を行ったり、表に対して索引スキャンを実行したりする前に、表を再編成する必要があります (SQLSTATE 57016)。

column-name

ドロップする列を指定します。列名は非修飾でなければなりません。この名前は、指定の表の列を特定するものでなければなりません (SQLSTATE 42703)。この名前は、表の唯一の列を示すものであってはなりません (SQLSTATE 42814)。この名前は、表の分散キー、表のパーティション・キー、または編成ディメンションの一部となっている列を指定するものであってはなりません (SQLSTATE 42997)。この名前は、XML データ・タイプ列を含む表の一部となっている列を指定するものであってはなりません (SQLSTATE 42997)。

CASCADE

ドロップされる列に従属するあらゆるビュー、索引、トリガー、SQL 関数、制約、またはグローバル変数もドロップされること、または、その列を含む表に従属する、分解を使用可能にしたあらゆる XSROBJECT は、分解に関して作動不能にされることを指定します。トリガーは、UPDATE OF 列リスト、またはトリガー・アクションのどこかで参照されていれば、列に従属します。分解使用可能 XSROBJECT は、XML 要素または属性の表へのマッピングを含む場合、表に従属します。SQL 関数またはグローバル変数が他のデータベース・オブジェクトに従属する場合、CASCADE オプションによってその関数またはグローバル関数をドロップできない可能性があります。CASCADE はデフォルトです。

RESTRICT

いずれかのビュー、索引、トリガー、制約、またはグローバル変数が列に従属している場合、または、いずれかの分解使用可能 XSROBJECT がその列を含む表に従属している場合に、列をドロップできないことを指定します (SQLSTATE 42893)。トリガーは、UPDATE OF 列リスト、またはトリガー・アクションのどこかで参照されていれば、列に従属します。分解使用可能 XSROBJECT は、XML 要素または属性の表へのマッピングを含む場合、表に従属します。検出された最初の従属オブジェクトは、管理ログで識別されます。

表 16. 列のドロップのカスケード効果

操作	RESTRICT の効果	CASCADE の効果
ビューまたはトリガーによって参照される列をドロップする。	列のドロップは許可されません。	そのオブジェクト自身とそのオブジェクトに従属するすべてのオブジェクトがドロップされます。
索引のキーで参照される列をドロップする。	索引で参照されているすべての列が、同じ ALTER TABLE ステートメントでドロップされていれば、索引のドロップが許可されます。それ以外の場合は、列のドロップは許可されません。	索引がドロップされます。

ALTER TABLE

表 16. 列のドロップのカスケード効果 (続き)

操作	RESTRICT の効果	CASCADE の効果
ユニーク制約で参照されている列をドロップする。	ユニーク制約で参照されているすべての列が、同じ ALTER TABLE ステートメントでドロップされていれば、また、ユニーク制約が参照制約によって参照されていない場合は、列と制約がドロップされます。(制約を成立させるために使用されている索引もドロップされます) それ以外の場合は、列のドロップは許可されません。	ユニーク制約と、そのユニーク制約を参照するあらゆる参照制約がドロップされます。(これらの制約によって使用される索引もすべてドロップされます。)
参照制約で参照されている列をドロップする。	参照制約で参照されているすべての列が、同じ ALTER TABLE ステートメントでドロップされていれば、列と制約がドロップされます。それ以外の場合は、列のドロップは許可されません。	参照制約がドロップされます。
ドロップされないシステム生成列によって参照されている列をドロップする。	列のドロップは許可されません。	列のドロップは許可されません。
チェック制約で参照されている列をドロップする。	列のドロップは許可されません。	チェック制約がドロップされます。
分解使用可能 XSROBJECT で参照されている列をドロップする。	列のドロップは許可されません。	XSROBJECT が分解操作不能とマークされます。XSROBJECT の分解を再度使用可能にするには、マッピングの再調整が必要になる可能性があります。それに続いて、XSROBJECT に対して ALTER XSROBJECT ENABLE DECOMPOSITION ステートメントを実行します。
グローバル変数のデフォルトの式で参照される列のドロップ	列のドロップは許可されません。	グローバル変数はドロップされます。ただし、グローバル変数に依存する他のオブジェクト (カスケードを許可しないオブジェクト) があるためにグローバル変数のドロップが許可されなければ、ドロップされません。

DROP RESTRICT ON DROP

表、および表を含む表スペースのドロップに関する制約事項があれば削除します。

DROP DISTRIBUTION

表の配分定義をドロップします。表には配分定義がなければなりません (SQLSTATE 428FT)。表の表スペースは、単一パーティションのデータベース・パーティション・グループ上で定義される必要があります。

DROP MATERIALIZED QUERY

表がマテリアライズ照会表と見なされなくなるように、マテリアライズ照会表を変更します。 *table-name* で指定される表は、複製ではないマテリアライズ照会表として定義されていなければなりません (SQLSTATE 428EW)。 *table-name* の列の定義は変更されませんが、照会の最適化にこの表を使用することはできなくなり、 REFRESH TABLE ステートメントも使用できなくなります。

DATA CAPTURE

データの複製に関する追加情報をログに記録するか否かを指定します。

表が型付き表である場合、このオプションはサポートされません (ルート表の場合は SQLSTATE 428DH で、他の副表の場合は 428DR)。

データ・キャプチャーと行圧縮の間には互換性がありません (SQLSTATE 42997)。

NONE

追加情報をログに記録しないことを指定します。

CHANGES

この表に対する SQL 変更についての追加情報をログに書き込むことを指定します。このオプションは、表を複製する場合で、キャプチャー・プログラムを使用してログからこの表に対する変更内容をキャプチャーする場合に必須です。

カタログ・パーティション以外のデータベース・パーティションにデータが置かれるように表が定義されている場合 (複数パーティションのデータベース・パーティション・グループ、またはカタログ・パーティション以外のデータベース・パーティションを持つデータベース・パーティション・グループ)、このオプションはサポートされません (SQLSTATE 42997)。

表のスキーマ名 (暗黙または明示名) が 18 バイトより長い場合、このオプションはサポートされません (SQLSTATE 42997)。

INCLUDE LONGVAR COLUMNS

データ複製ユーティリティーが、LONG VARCHAR または LONG VARCHARIC 列に対する変更をキャプチャーするようにします。この節は、LONG VARCHAR または LONG VARCHARIC 列のない表に指定することもできます。これは、LONG VARCHAR または LONG VARCHARIC 列を含むよう、表を ALTER することができるためです。

ACTIVATE NOT LOGGED INITIALLY

現行の作業単位の表の NOT LOGGED INITIALLY 属性を活動化します。

このステートメントにより表を変更した後に、同一の作業単位の INSERT、DELETE、UPDATE、CREATE INDEX、DROP INDEX、または ALTER TABLE によって表に対して行われた変更は、ログ記録されません。NOT LOGGED INITIALLY 属性が活動状態にあるときに、ALTER ステートメントによってシ

ALTER TABLE

ステム・カタログに対して行われた変更は、ログ記録されます。同一の作業単位内でのシステム・カタログ情報に対する以降の変更は、ログ記録されます。

現行の作業単位が完了すると、NOT LOGGED INITIALLY 属性は非活動化され、それ以降の作業単位の表で行われるすべての操作はログ記録されます。

カタログ表へのデータの挿入中にロックを避けるためにこのフィーチャーを使用する場合、ALTER TABLE ステートメントにはこの節だけを指定してください。ALTER TABLE ステートメントでこの節以外のものを指定すると、カタログはロックされてしまいます。ALTER TABLE ステートメントでこの節のみが指定されている場合、SHARE ロックのみがシステム・カタログ表で獲得されます。これにより、このステートメントの実行時から、このステートメントの実行作業単位の終了時までの間で、並行する競合の可能性をかなり低減させることができます。

表が型付き表である場合、このオプションは、型付き表階層のルート表に対してのみサポートされます (SQLSTATE 428DR)。

NOT LOGGED INITIALLY 属性の詳細については、『CREATE TABLE』にあるこの属性に関する記述を参照してください。

注: 活動化された NOT LOGGED INITIALLY 属性を持つ表に対してログに記録されない活動が生じ、ステートメントに障害が起こる (ロールバックが発生する)、または ROLLBACK TO SAVEPOINT が実行される場合には、その作業単位全体がロールバックされます (SQL1476N)。さらに、NOT LOGGED INITIALLY 属性が活動化されている表は、ロールバックされた後にアクセス不能としてマークされ、ドロップしかできなくなります。したがって、NOT LOGGED INITIALLY 属性が活動化されている作業単位内のエラーは、最小限に抑えるべきです。

WITH EMPTY TABLE

現在表にあるすべてのデータを除去します。一度データが除去されると、RESTORE 機能を使用しなければ、そのデータの回復を行うことができません。この ALTER ステートメントを発行した作業単位をロールバックしても、表データは元の状態には回復できません。

この処置が必要な場合、対象の表に定義された DELETE トリガーは行われません。その表にある索引もすべて削除されます。

アタッチされたデータ・パーティションを持つパーティション表は空にできません (SQLSTATE 42928)。

PCTFREE *integer*

ロードまたは表再編成操作において、各ページに残すフリー・スペースのパーセンテージを指定します。各ページの最初の行は、制約なしに追加されます。ページに行が追加される際には、少なくとも *integer* で指定された分 (%) のフリー・スペースがページに残されます。PCTFREE 値は、load および table reorg ユーティリティでのみ考慮されます。*integer* の値は 0 から 99 です。システム・カタログ (SYSCAT.TABLES) の PCTFREE 値 -1 は、デフォルト値として解釈されます。表ページのデフォルトの PCTFREE 値は 0 です。表が型付き表である場合、このオプションがサポートされるのは、型付き表階層のルート表だけです (SQLSTATE 428DR)。

LOCKSIZE

表へのアクセス時に使用されるロックのサイズ (細分性) を指定します。表定義でこのオプションを使用しても、通常のロック・エスカレーションを妨げることはありません。表が型付き表である場合、このオプションは、型付き表階層のルート表に対してのみサポートされます (SQLSTATE 428DR)。

ROW

行ロックの使用を指定します。これは、表の作成時のデフォルトのロック・サイズです。

BLOCKINSERT

挿入操作時のブロック・ロックの使用を指定します。これは、挿入の前にブロックに関して適切な排他ロックが取得され、挿入される行に関しては行ロックは行われないということを意味します。このオプションは、独立したトランザクションが、表内の独立したセルに挿入を行う際に役立ちます。同じセルへの挿入を行うトランザクションはやはり同時に挿入を行う可能性があります、それぞれに個別のブロックに対して挿入を行います。より多くのブロックが必要になる場合、これがセルのサイズに影響することもあります。このオプションは、MDC 表に対してのみ有効です (SQLSTATE 628N)。

TABLE

表ロックの使用を指定します。これは、適切な共用ロックまたは排他ロックが表で獲得されており、意図ロック ("意図なし" は除く) が使用されないことを意味します。パーティション表に関しては、このロック方針は、表ロックおよびデータ・パーティション・ロックの両方に、アクセスされるあらゆるデータ・パーティションについて適用されます。この値を使用すると、獲得すべきロック数が限定されるため、照会のパフォーマンスが向上する可能性があります。しかし、すべてのロックが表全体に対して獲得されるので、並行性も限定されます。

APPEND

データを表データの終わりに追加するか、またはデータ・ページの使用可能なフリー・スペースを位置に追加するかを指定します。表が型付き表である場合、このオプションは、型付き表階層のルート表に対してのみサポートされます (SQLSTATE 428DR)。

ON

表データが追加され、各ページのフリー・スペース情報は保持されません。表にはクラスター索引があってはなりません (SQLSTATE 428CA)。

OFF

表データは使用可能なスペースに入れられます。これは、表の作成時のデフォルト値です。

APPEND OFF を設定した後に表の再編成が必要となります。これは、使用可能なフリー・スペース情報が不正確となるため、データ挿入時のパフォーマンスの低下につながるからです。

VOLATILE CARDINALITY または NOT VOLATILE CARDINALITY

表 *table-name* のカーディナリティーが、実行時に相当に変化し得るかどうか

ALTER TABLE

をオプティマイザーに知らせます。揮発性は、表そのものに対してではなく、表の行数に適用されます。 **CARDINALITY** はオプション・キーワードです。デフォルトは **NOT VOLATILE** です。

VOLATILE

表 *table-name* のカーディナリティーが、空から非常に大きなものに至るまで、実行時に変化し得ることを知らせます。表にアクセスするために、オプティマイザーは、その統計に関係なく、表のスキャンではなく索引のスキャンを使います。ただし、その場合、その索引は索引専用である (参照されるすべての列がその索引内にある) か、索引のスキャンで述部を適用できることが条件になります。リスト・プリフェッチ・アクセス方式は、この表へのアクセスには使用されません。表が型付き表である場合、このオプションは、型付き表階層のルート表に対してのみサポートされます (SQLSTATE 428DR)。

NOT VOLATILE

table-name のカーディナリティーが揮発性でないことを指定します。この表へのアクセス・プランは、既存の統計と、現行の最適化レベルに基づいて続けられます。

COMPRESS

表の行にデータ圧縮を適用するかどうかを指定します。

YES

データ行圧縮を使用可能にすることを指定します。表に対する挿入と更新の操作で、圧縮が行われるようになります。表のコンプレッション・ディクショナリーが存在していなければ、表に十分なデータが取り込まれた後で、コンプレッション・ディクショナリーが自動的に作成され、行が圧縮の対象になります。表に既存のコンプレッション・ディクショナリーがあれば、そのディクショナリーによる圧縮が再びアクティブになり、行が圧縮の対象になります。

NO

データ行圧縮を使用不可にすることを指定します。表に対する挿入と更新の操作で、圧縮が行われなくなります。圧縮フォーマットの表内の行は、更新時に非圧縮フォーマットに変換されるまで、そのまま圧縮フォーマットになります。表の非インプレース再編成を行うと、圧縮された行はすべて解凍されます。コンプレッション・ディクショナリーが存在する場合は、表の再初期設定または切り捨ての際に廃棄されます (例えば置換操作の際など)。

VALUE COMPRESSION

使用される行形式を判別します。それぞれのデータ・タイプは、使用される行形式に応じた、異なるバイト・カウントを持ちます。詳細については、『CREATE TABLE』の『バイト・カウント』を参照してください。更新操作により、既存の行を新しい行形式に変更します。既存の行の更新操作のパフォーマンスを向上させるには、オフラインでの表再編成をお勧めします。これはまた、表の消費スペースの削減にもつながります。『データ・タイプごとの列のバイト・カウント』という表の適切な列を使用して行サイズを計算すると (『CREATE TABLE』を参照のこと)、『各表スペースのページ・サイズの列数および行サイズの制限』という名前の表に示された行サイズの制限内に収まらなくなる場合に

は、エラーが戻されます (SQLSTATE 54010)。表が型付き表である場合、このオプションは、型付き表階層のルート表に対してのみサポートされます (SQLSTATE 428DR)。

ACTIVATE

NULL 値は 3 バイトを使用して保管されます。これは、すべてのデータ・タイプの列に対して VALUE COMPRESSION がアクティブでない場合と同じスペース、またはそれより少ないスペースです (CHAR(1) は例外)。列が NULL 可能として定義されているかいないかは、行サイズ計算には影響しません。データ・タイプが VARCHAR、VARGRAPHIC、LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB、または BLOB である列の、長さがゼロのデータ値は、2 バイトだけを使用して保管されます。これは、VALUE COMPRESSION がアクティブでない場合に必要とされるストレージを下回ります。COMPRESS SYSTEM DEFAULT オプションを使用して列を定義すると、列のシステム・デフォルト値も 3 バイトの合計ストレージを使用して保管できるようになります。これをサポートする行形式は、各データ・タイプのバイト・カウントを識別し、NULL、長さがゼロの値、またはシステム・デフォルト値の関係した更新を行う際に、データ・フラグメントの原因となる傾向があります。

DEACTIVATE

NULL 値は、今後の更新のために確保されたスペースと一緒に保管されます。このスペースは、可変長の列のためには確保されません。またそのようにすると、列のシステム・デフォルト値のストレージを効率的にサポートしません。列が COMPRESS SYSTEM DEFAULT 属性を既に有していると、警告が出されます (SQLSTATE 01648)。

LOG INDEX BUILD

この表で索引の作成、再作成、または再編成の操作を行う際に実行されるロギングのレベルを指定します。

NULL

logindexbuild データベース構成パラメーターの値を使用して、索引作成操作を完全にログに記録するかどうかを指定します。これは、表が作成される時のデフォルトです。

OFF

この表での索引作成操作が最小限ログに記録されることを指定します。この値は、**logindexbuild** データベース構成パラメーターの設定をオーバーライドします。

ON

この表での索引作成操作が完全にログに記録されることを指定します。この値は、**logindexbuild** データベース構成パラメーターの設定をオーバーライドします。

規則

- 表に対して定義されたユニーク・キー制約または主キー制約は、分散キー (存在する場合) のスーパーセットである必要があります (SQLSTATE 42997)。
- 主キーまたはユニーク・キーは、ディメンションのサブセットにはなりません (SQLSTATE 429BE)。

ALTER TABLE

- 1 つの列の参照は、1 つの ALTER TABLE ステートメント内の 1 つの ADD、ALTER または DROP COLUMN 節でのみ可能です (SQLSTATE 42711)。
- 表にマテリアライズ照会表があり、その表に従属している場合、列の長さまたはデータ・タイプを変更したり、列をドロップすることはできません (SQLSTATE 42997)。
- それぞれ 4000 および 2000 より大きい数値に変更された VARCHAR および VARGRAPHIC 列は、SYSFUN スキーマの関数での入力パラメーターとして使用しないでください (SQLSTATE 22001)。
- 表に従属するビューがあり、照会最適化に使用可能になっている場合、列の長さを変更することはできません (SQLSTATE 42997)。
- 以下のことを行う前に、SET INTEGRITY ステートメントを OFF オプションとともに使用して、表を SET INTEGRITY ペンディング状態に設定しなければなりません (SQLSTATE 55019)。
 - 生成式を使って列を追加する
 - 列の生成式を変更する
 - 生成式を持つよう、列を変更する
- XML データ・タイプの列は、タイプ 1 索引がある表には追加できません (SQLSTATE 42997)。索引は、CONVERT オプションを付けた REORG INDEXES コマンドを使用して、タイプ 2 索引に変換できます。
- 既存の列は DB2SECURITYLABEL タイプとなるよう変更することはできません (SQLSTATE 42837)。
- DB2SECURITYLABEL タイプの列を定義しようとしても、表にそれに関連するセキュリティ・ポリシーがなければ失敗します (SQLSTATE 55064)。
- DB2SECURITYLABEL タイプの列は、変更またはドロップできません (SQLSTATE 42817)。
- 表に保護のマークを付ける ALTER TABLE 操作は、その表に従属する MQT が存在する場合に失敗します (SQLSTATE 55067)。
- 保護されたパーティション表にパーティションをアタッチしようとしても、ソース表およびターゲット表が同じセキュリティ・ポリシーで保護されておらず、同じ行セキュリティ・ラベル列を持ち、同じ保護列のセットを持っている場合は失敗します (SQLSTATE 428GE)。
- 生成される列が表パーティション・キーで参照される場合、生成される列の式は変更できません (SQLSTATE 42837)。

注

- *REORG 推奨操作*は、ALTER TABLE ステートメントに起因する変更がデータの行フォーマットに影響する場合に行われています。この操作が行われた場合、その後の表に対する操作はほとんど、表再編成操作が正しく完了するまで制限されます。このタイプの ALTER TABLE ステートメントは、再編成が必要になるまでに最高で 3 つまで表に対して実行できます (SQLSTATE 57016)。REORG 推奨操作を構成する複数の変更は、単一の ALTER TABLE ステートメント (列あたり 1 つ) の一部とすることができます。これは、単一の REORG 推奨操作とみなされます。例えば、単一の ALTER TABLE ステートメントの中で 2 つの列をドロップする場合、2 つの REORG 推奨操作とはみなされません。ただし、2

つの別の ALTER TABLE ステートメントで 2 つの列をドロップする場合は、REORG 推奨操作を含む 2 つのステートメントとみなされます。

- REORG 推奨操作が正しく行われた後は、以下の表操作が許可されます。
 - 行データ妥当性検査を必要としない ALTER TABLE。ただし、以下の操作は許可されません (SQLSTATE 57007)。
 - ADD CHECK CONSTRAINT
 - ADD REFERENTIAL CONSTRAINT
 - ADD UNIQUE CONSTRAINT
 - ALTER COLUMN SET NOT NULL
 - DROP TABLE
 - RENAME TABLE
 - REORG TABLE
 - TRUNCATE TABLE
 - 表データの表スキャン・アクセス
- 表をマテリアライズ照会表に変更すると、この表は SET INTEGRITY ペンディング状態になります。表が REFRESH IMMEDIATE として定義されている場合、この表は SET INTEGRITY ペンディング状態を解除しない限り、全選択で参照されている表で INSERT、DELETE、または UPDATE コマンドを呼び出すことはできません。IMMEDIATE CHECKED オプションを指定して REFRESH TABLE または SET INTEGRITY を使用することで、表の SET INTEGRITY ペンディング状態を解除し、全選択に基づいて表内のデータを完全にリフレッシュできます。表にあるデータが完全に全選択の結果を反映する場合、SET INTEGRITY の IMMEDIATE UNCHECKED オプションを使用して、表の SET INTEGRITY ペンディング状態を解除できます。
- 表を変更して REFRESH IMMEDIATE マテリアライズ照会表にすると、全選択により参照される表に対して INSERT、DELETE、または UPDATE を使用するパッケージはどれも無効になります。
- 表をマテリアライズ照会表から正規表に変更すると、表に関連するパッケージはどれも無効になります。
- 表を MAINTAINED BY FEDERATED_TOOL マテリアライズ照会表から正規表に変更しても、レプリケーション・ツールのサブスクリプション・セットアップには変更は生じません。MAINTAINED BY SYSTEM マテリアライズ照会表に対する以降の変更は、レプリケーション・ツールの失敗をもたらすので、MAINTAINED BY FEDERATED_TOOL マテリアライズ照会表の変更の際には、サブスクリプション設定を変更する必要があります。
- 据え置かれたマテリアライズ照会表がステージング表に関連付けられる場合、マテリアライズ照会表が正規表に変更されると、ステージング表はドロップされます。
- ADD COLUMN 節は、他のいずれの節よりも先に処理されます。他の節は、指定された順序で処理されます。
- 表変更操作によって追加される列は、表の既存のビューに自動的に追加されることはありません。

ALTER TABLE

- データ・パーティションのパーティション表への追加またはアタッチ、あるいはパーティション表からのデータ・パーティションのデタッチを行うと、表に従属するあらゆるパッケージは無効になります。
- 表についてのパーティションをドロップするには、表をドロップした上で再作成する必要があります。
- 表についての編成をドロップするには、表をドロップした上で再作成する必要があります。
- ユニーク・キー制約または主キー制約に関して索引が自動的に作成される場合、データベース・マネージャーは、指定された制約名を表のスキーマ名と一致するスキーマ名を伴う索引名として使用することを試みます。この名前が既存の索引名と一致する場合、または制約の名前が指定されなかった場合、索引は **SYSIBM** スキーマに作成され、"SQL" とタイム・スタンプに基づいて生成される一連の 15 個の数字からなるシステム生成の名前が付けられます。
- アタッチされたデータ・パーティションを伴うパーティション表上に索引が作成される場合、索引には、アタッチされたデータ・パーティションのデータは含まれません。すべてのアタッチされたデータ・パーティションに対応する全索引を維持するには、**SET INTEGRITY** ステートメントを使用します。
- 表 T での **DELETE** 操作に関係する可能性のある表は、「T に連結削除されている」と呼ばれます。つまり、ある表が T の従属表であるか、または T からの削除のカスケード先の表の従属表である場合、この表は T に対して連結削除されることとなります。
- パッケージに表 T での挿入 (更新/削除) 使用があるとは、パッケージ内のステートメントによって直接的に、あるいは、そのいずれかのステートメントの代わりにパッケージによって実行される制約やトリガーによって間接的に、レコードが T に挿入 (更新または削除) されることです。同様に、パッケージに更新使用があるとは、パッケージ内のステートメントによって直接的に、あるいは、そのいずれかのステートメントの代わりにパッケージによって実行される制約やトリガーによって間接的に、列が変更されることです。
- フェデレーテッド・システムでは、透過性 DDL を使用して作成されたりリモート基本表は変更できます。ただし、可能な変更に関して、透過 DDL には以下のようないくつかの制限があります。
 - リモート基本表は、新規の列の追加か、または主キーの指定によってのみ、変更できます。
 - 透過性 DDL によってサポートされる節には、具体的には以下のものがあります。
 - **ADD COLUMN** *column-definition*
 - *column-options* 節の中の **NOT NULL** と **PRIMARY KEY**
 - **ADD unique-constraint** (**PRIMARY KEY** のみ)
 - リモート基本表内の既存の列には、コメントを指定できません。
 - リモート基本表内の既存の主キーは、変更またはドロップできません。
 - リモート基本表の変更を行うと、そのリモート基本表に関連したニックネームに従属するパッケージはいずれも無効になります。

- リモート・データ・ソースは、ALTER TABLE ステートメントを通じて要求された変更をサポートする必要があります。データ・ソースの、サポートしていない要求への応答方法によって、エラーが返されるか、または要求が無視される可能性があります。
- 透過性 DDL を使用して作成されたのではないリモート基本表を変更しようとすると、エラーが返されます。
- 主キー、ユニーク・キー、または外部キーに対する変更は、明示的、暗黙的を問わず、パッケージ、索引、およびその他の外部キーに以下の影響を与える可能性があります。
 - 主キーまたはユニーク・キーが追加された場合、
 - パッケージ、外部キー、または既存のユニーク・キーに影響はありません。(主キーまたはユニーク・キーが、前のバージョンで作成された既存のユニーク索引を使用しており、ユニーク性の据え置きをサポートするように変換されていない場合、索引は変換され、関連した表の更新を行うパッケージは無効になります。)
 - 主キーまたはユニーク・キーがドロップされた場合、
 - 制約に関してその索引が自動的に作成されていた場合には、その索引はドロップされます。索引に従属しているパッケージはすべて無効になります。
 - 索引が制約に関してユニークであるように変換されており、現在システムが索引に関してユニークであることを必要としていない場合、索引は非ユニークに戻されます。索引に従属しているパッケージはすべて無効になります。
 - 索引が制約のために使用された既存のユニーク索引だった場合、索引はシステムが必要としていないことを示すよう設定されます。パッケージに影響はありません。
 - すべての従属外部キーはドロップされます。次の項目に示すように、各従属外部キーごとに、さらにアクションが取られます。
 - 外部キーが追加される、ドロップされる、または NOT ENFORCED から ENFORCED に (または ENFORCED から NOT ENFORCED に) 変更される場合:
 - オブジェクト表に対して挿入を行うパッケージは、すべて無効になります。
 - 外部キー内の少なくとも 1 つの列に対して更新使用の指定があるパッケージは、すべて無効になります。
 - 親表の削除を行うパッケージはすべて無効になります。
 - 親キーの少なくとも 1 つの列に対して更新使用の指定があるパッケージは、すべて無効になります。
 - 外部キーまたは機能従属関係が、ENABLE QUERY OPTIMIZATION から DISABLE QUERY OPTIMIZATION に変更される場合:
 - 最適化のための制約と従属関係にあるパッケージすべては、無効です。
- 表に列を追加すると、変更された表に対して挿入を行うパッケージはすべて無効になります。追加された列が、表内の最初のユーザー定義構造化タイプ列である場合、変更された表で DELETE を行おうとするパッケージは無効になります。
- SET INTEGRITY ペンディング状態ではない既存の表に対してチェック制約または参照制約を追加するか、または SET INTEGRITY ペンディング状態にない既存の表の既存のチェック制約か参照制約を NOT ENFORCED から ENFORCED に

変更すると、その表の既存の行は、制約に関して直ちに評価されます。検証に失敗すると、エラーが戻されます (SQLSTATE 23512)。表が SET INTEGRITY ペンディング状態の場合は、チェック制約または参照制約を追加しても、または制約を NOT ENFORCED から ENFORCED に変更しても、制約が直ちに適用されるわけではありません。制約の適用を開始するには、IMMEDIATE CHECKED オプションを付けて SET INTEGRITY ステートメントを発行します。

- チェック制約の追加、変更、またはドロップを行うと、対象の表に対する挿入、制約に関係している少なくとも 1 つの列に対する更新、またはパフォーマンスを向上させるための制約の選択使用のいずれかを含むすべてのパッケージが無効になります。
- 分散キーを追加すると、分散キーの少なくとも 1 つの列に対して更新使用の指定があるパッケージは、すべて無効になります。
- デフォルトで主キーの最初の列を使用して定義された分散キーは、主キーのドロップや異なる主キーの追加によっては影響を受けません。
- 列をドロップするか、そのデータ・タイプを変更すると、変更される表からすべての RUNSTATS 情報が除去されます。表に対する RUNSTATS は、その表へのアクセスが再び可能になった後に行ってください。表の統計プロファイルは、表に明示的にドロップされた列が含まれていない場合には、保持されます。
- 列を変更する (長さを増やす、データ・タイプを変更する、または NULL 可能属性を変更するために) か、列をドロップすると、その表を (直接的に、あるいは参照制約やトリガーによって間接的に) 参照するすべてのパッケージが無効になります。
- 列を変更する (長さを増やす、データ・タイプを変更する、または NULL 可能属性を変更するといったことを行う) と、表に従属するビュー (型付きビューを除く) が再生成されます。そうしたビューの再生成時に問題が生じると、エラーが戻されます (SQLSTATE 56098)。表に従属する型付きビューは、作動不能としてマークされます。
- 列を変更し、長さを増やす、データ・タイプを変えるとといったことを行うと、従属するトリガーや SQL 関数はすべて無効とマークされます。これらは次回の使用時に暗黙的に再コンパイルされます。そうしたオブジェクトの再生成時に問題が生じると、エラーが戻されます (SQLSTATE 56098)。
- 列を変更する (長さを増やす、データ・タイプを変更する、または NULL 可能属性を変更するといったことを行う) と、トリガーや SQL 関数を伴うステートメントの準備またはバインドの際に、トリガーや SQL 関数の処理中にエラーが発生する可能性があります (SQLSTATE 54010)。このことは、遷移変数および遷移表列の長さの合計に基づく行の長さが長すぎる場合に生じる可能性があります。このようなトリガーまたは SQL 関数がドロップされると、それ以降にそれを再作成しようとしてもエラーが戻されます (SQLSTATE 54040)。
- 構造化タイプ列または XML タイプ列を変更してインライン長を長くすると、参照制約またはトリガーによって直接または間接的に表を参照するパッケージはすべて無効になります。
- 構造化タイプ列または XML タイプ列を変更してインライン長を長くすると、表に従属するビューは再生成されます。
- 表の LOCKSIZE を変更すると、変更された表に従属するすべてのパッケージは無効になります。

- VOLATILE または NOT VOLATILE CARDINALITY を変更すると、変更された表に従属するすべてのパッケージは無効になります。
- **レプリケーション:** 列の長さを増やす時や、列のデータ・タイプを変更する際には十分に注意してください。アプリケーション表と関連付けられた変更データ表は、すでに DB2 行サイズの限界近くに設定されている可能性があります。変更データ表をアプリケーション表よりも前に変更するか、これら 2 つの表を同じ作業単位内で変更するようにして、両方の表で変更が完了できるようにしてください。コピーについても考慮すべき点があります。これも、行サイズの限界近くに設定されていたり、既存の列の長さを長くする機能のないプラットフォームに存在している可能性があります。

属性を変更したログ・レコードをキャプチャー・プログラムが処理する前に、変更データ表を変更していなければ、キャプチャー・プログラムが失敗する場合があります。コピーを保持しているサブスクリプションを実行する前に、変更対象の列が含まれるコピーを変更していなければ、そのサブスクリプションは失敗する可能性があります。

- パーティションを保護表からデタッチする際、DB2 によって自動的に作成されるターゲット表は、ソース表とまったく同じ方法で保護されます。
- 表を、行レベルの細分度で保護されるよう変更すると、そうした表に従属するキャッシュ動的 SQL セクションはすべて無効にされます。同様に、そうした表に従属するパッケージも無効にされます。
- 表 T の列を、保護列になるよう変更すると、表 T に従属するキャッシュ動的 SQL セクションはすべて無効にされます。同様に、表 T に従属するパッケージも無効にされます。
- 表 T の列を、非保護列になるよう変更すると、表 T に従属するキャッシュ動的 SQL セクションはすべて無効にされます。同様に、表 T に従属するパッケージも無効にされます。
- 表内の既存の行に関し、セキュリティー・ラベル列の値は、行セキュリティー・ラベル列を追加する ALTER ステートメントの実行時に、デフォルトである、セッション許可 ID の書き込みアクセス権限に対応するセキュリティー・ラベルになります。
- **暗黙的に隠される列に関する考慮事項:** 暗黙的に隠されると定義される列は、ALTER TABLE ステートメントで明示的に参照することができます。例えば、暗黙的に隠される列は、参照制約、チェック制約、またはマテリアライズ照会表の定義の一部として変更または指定することができます。

- **互換性**

- 以前のバージョンの DB2 との互換性:
 - 以下についての ADD キーワードはオプションです。
 - 名前のない PRIMARY KEY 制約
 - 名前のない参照制約
 - FOREIGN KEY 句の後に名前を指定した参照制約
 - CONSTRAINT キーワードは、参照節を定義する *column-definition* から省略できます。
 - *constraint-name* (制約名) を FOREIGN KEY に続けて (CONSTRAINT キーワードなし) 指定することができます。

ALTER TABLE

- SET MATERIALIZED QUERY AS の代わりに SET SUMMARY AS を指定できます。
- DROP MATERIALIZED QUERY の代わりに SET MATERIALIZED QUERY AS DEFINITION ONLY を指定できます。
- ADD MATERIALIZED QUERY (全選択) の代わりに SET MATERIALIZED QUERY AS (全選択) を指定できます。
- ADD DISTRIBUTE BY HASH の代わりに ADD PARTITIONING KEY を指定することができます。この場合、オプションの USING HASHING 節も指定できます。
- DROP DISTRIBUTION の代わりに DROP PARTITIONING KEY を指定できます。
- データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっています (特に移植可能なアプリケーションではお勧めしていません)。
- 以前のバージョンの DB2 との互換性と整合性:
 - *identity-alteration* 節では、コンマを使って複数のオプションを分離することができます。
- DB2 for z/OS との互換性:
 - PARTITION の代わりに PART を指定できます。
 - ENDING AT の代わりに VALUES を指定できます。
- 以下の構文もサポートされています。
 - NOMINVALUE、NOMAXVALUE、NOCYCLE、NOCACHE、および NOORDER

例

例 1: 1 文字の長さの RATING という名前の新しい列を、DEPARTMENT 表に追加します。

```
ALTER TABLE DEPARTMENT
ADD RATING CHAR(1)
```

例 2: SITE_NOTES という名前の新しい列を PROJECT 表に追加します。SITE_NOTES は、最大 1000 バイトの長さの可変長列として作成します。この列の値には関連する文字セットがなく、変換されません。

```
ALTER TABLE PROJECT
ADD SITE_NOTES VARCHAR(1000) FOR BIT DATA
```

例 3: 以下の列が定義された EQUIPMENT という表が存在するものと想定します。

Column Name	Data Type
EQUIP_NO	INT
EQUIP_DESC	VARCHAR(50)
LOCATION	VARCHAR(50)
EQUIP_OWNER	CHAR(3)

EQUIPMENT 表に、所有者 (EQUIP_OWNER) は DEPARTMENT 表に存在する部門番号 (DEPTNO) でなければならない、という参照制約を追加します。DEPTNO は、DEPARTMENT 表の主キーです。DEPARTMENT 表からある部門を削除する

場合は、その部門の所有するすべての備品の所有者 (EQUIP_OWNER) の値を割り当て解除する必要があります (つまり NULL 値に設定する必要があります)。制約の名前は、DEPTQUIP です。

```
ALTER TABLE EQUIPMENT
  ADD CONSTRAINT DEPTQUIP
  FOREIGN KEY (EQUIP_OWNER)
  REFERENCES DEPARTMENT
  ON DELETE SET NULL
```

さらに、備品レコードに関係した数量を記録できるようにするため、追加の列が必要になります。特に指定されない限り、EQUIP_QTY 列には値 1 を入れます。NULL 値にしてはなりません。

```
ALTER TABLE EQUIPMENT
  ADD COLUMN EQUIP_QTY
  SMALLINT NOT NULL DEFAULT 1
```

例 4: 表 EMPLOYEE を変更します。各従業員の給与と歩合の合計が \$30,000 を超えていなければならない、という定義済みの REVENUE という名前のチェック制約を追加します。

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT REVENUE
  CHECK (SALARY + COMM > 30000)
```

例 5: 表 EMPLOYEE を変更します。前に定義した制約 REVENUE をドロップします。

```
ALTER TABLE EMPLOYEE
  DROP CONSTRAINT REVENUE
```

例 6: SQL の変更内容をデフォルトのフォーマットでログに記録するように表を変更します。

```
ALTER TABLE SALARY1
  DATA CAPTURE NONE
```

例 7: SQL の変更内容を拡張フォーマットでログに記録するように表を変更します。

```
ALTER TABLE SALARY2
  DATA CAPTURE CHANGES
```

例 8: EMPLOYEE 表を変更して、デフォルト値を指定して 4 つの新しい列を追加します。

```
ALTER TABLE EMPLOYEE
  ADD COLUMN HEIGHT MEASURE DEFAULT MEASURE(1)
  ADD COLUMN BIRTHDAY BIRTHDATE DEFAULT DATE('01-01-1850')
  ADD COLUMN FLAGS BLOB(1M) DEFAULT BLOB(X'01')
  ADD COLUMN PHOTO PICTURE DEFAULT BLOB(X'00')
```

デフォルト値の指定時に、これらのデフォルト値はさまざまな関数名を使用します。MEASURE は INTEGER に基づく特殊タイプなため、MEASURE 関数が使用されます。HEIGHT 列のデフォルト値は、MEASURE のソース・タイプは、BLOB または日時データ・タイプでないため、関数を使用しなくても指定しておくことができました。BIRTHDATE は DATE に基づく特殊タイプなので、DATE 関数を使用しています (この場合、BIRTHDATE は使用できません)。FLAGS 列と PHOTO 列では、PHOTO が特殊タイプであるにもかかわらず、BLOB 関数を使用

ALTER TABLE

してデフォルト値が指定されています。 BIRTHDAY、FLAGS、および PHOTO 列のデフォルト値を指定するためには、関数を使用する必要があります。タイプが、 BLOB や日時データ・タイプのソースに基づく BLOB や特殊タイプだからです。

例 9: 以下の列のある CUSTOMERS という表が定義されます。

Column Name	Data Type
BRANCH_NO	SMALLINT
CUSTOMER_NO	DECIMAL(7)
CUSTOMER_NAME	VARCHAR(50)

この表では、主キーは BRANCH_NO 列と CUSTOMER_NO 列からなります。表を分散するには、表に対して分散キーを作成する必要があります。表は単一のデータベース・パーティションからなるデータベース・パーティション・グループの表スペースに定義する必要があります。主キーは、分散キー列のスーパーセットである必要があります、主キーの少なくとも 1 つの列が分散キーとして使用されている必要があります。以下のようにして、BRANCH_NO を分散キーとして定義します。

```
ALTER TABLE CUSTOMERS
ADD DISTIBUTE BY HASH (BRANCH_NO)
```

例 10: リモート表 EMPLOYEE が、フェデレーテッド・システムに透過 DDL を使用して作成されました。リモート表 EMPLOYEE を変更して、列 PHONE_NO および WORK_DEPT を追加します。また、主キーを既存の列 EMP_NO および新規列 WORK_DEPT に追加します。

```
ALTER TABLE EMPLOYEE
ADD COLUMN PHONE_NO CHAR(4) NOT NULL
ADD COLUMN WORK_DEPT CHAR(3)
ADD PRIMARY KEY (EMP_NO, WORK_DEPT)
```

例 11: DEPARTMENT 表を変更して機能従属関係 FD1 を追加し、次いで DEPARTMENT 表から機能従属関係をドロップします。

```
ALTER TABLE DEPARTMENT
ADD CONSTRAINT FD1
CHECK ( DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED
```

```
ALTER TABLE DEPARTMENT
DROP CHECK FD1
```

例 12: EMPLOYEE 表の WORKDEPT 列のデフォルト値を 123 に変更します。

```
ALTER TABLE EMPLOYEE
ALTER COLUMN WORKDEPT
SET DEFAULT '123'
```

例 13: セキュリティー・ポリシー DATA_ACCESS を EMPLOYEE 表に関連付けます。

```
ALTER TABLE EMPLOYEE
ADD SECURITY POLICY DATA_ACCESS
```

例 14: 表 EMPLOYEE を変更して、SALARY 列を保護します。

```
ALTER TABLE EMPLOYEE
ALTER COLUMN SALARY
SECURED WITH EMPLOYEESECLABEL
```

例 15: 以下の列を伴って定義された SALARY_DATA という表があるとします。

Column Name	Data Type
-----	-----
EMP_NAME	VARCHAR(50) NOT NULL
EMP_ID	SMALLINT NOT NULL
EMP_POSITION	VARCHAR(100) NOT NULL
SALARY	DECIMAL(5,2)
PROMOTION_DATE	DATE NOT NULL

この表を変更して、給料を DECIMAL(6,2) 列に保管できるようにし、PROMOTION_DATE を NULL 値への設定が可能なオプションのフィールドにし、EMP_POSITION 列を除去します。

```
ALTER TABLE SALARY_DATA
ALTER COLUMN SALARY SET DATA TYPE DECIMAL(6,2)
ALTER COLUMN PROMOTION_DATE DROP NOT NULL
DROP COLUMN EMP_POSITION
```

ALTER TABLESPACE

ALTER TABLESPACE ステートメントは、以下の方法で既存の表スペースを変更する場合に使用されます。

- DMS 表スペース (つまり MANAGED BY DATABASE オプションによって作成される表スペース) にコンテナを追加する、または DMS 表スペースからコンテナをドロップする。
- DMS 表スペースのコンテナのサイズを変更する。
- コンテナのないデータベース・パーティション上の SMS 表スペースにコンテナを追加する。
- 表スペースの PREFETCHSIZE 設定値を変更する。
- 表スペースの表に対して使用する BUFFERPOOL を変更する。
- 表スペースの OVERHEAD 設定値を変更する。
- 表スペースの TRANSFERRATE 設定値を変更する。
- 表スペースに対するファイル・システムのキャッシング・ポリシーを変更する。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

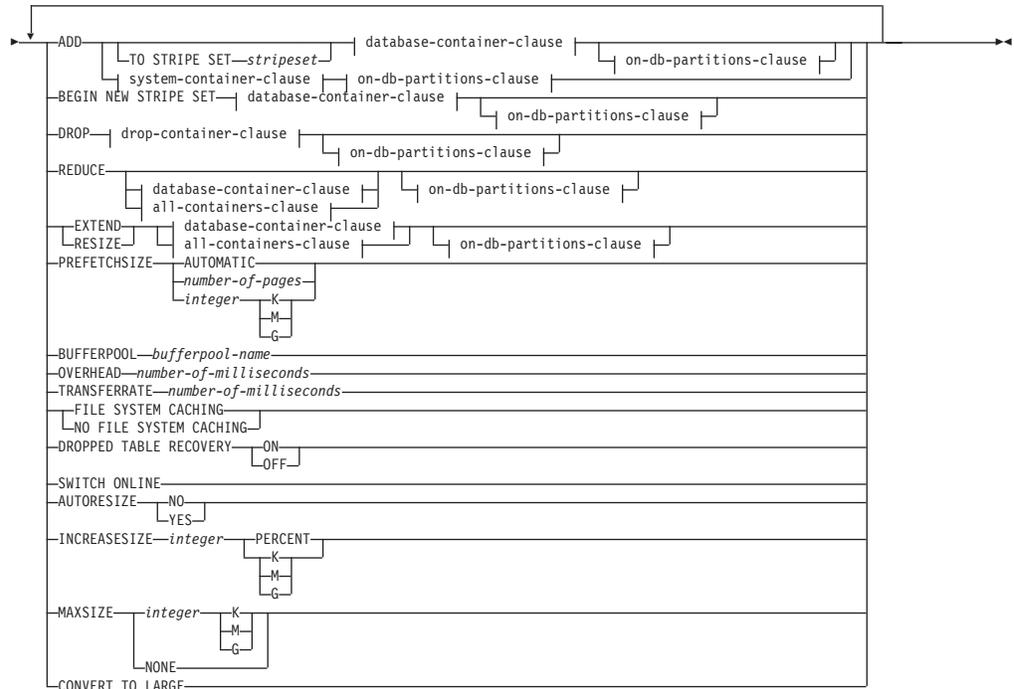
許可

このステートメントの許可 ID が持つ特権には、SYSCtrl または SYSADM 権限が含まれている必要があります。

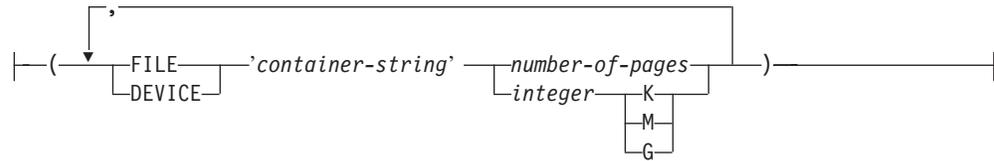
構文

```
→ALTER TABLESPACE—tablespace-name→
```

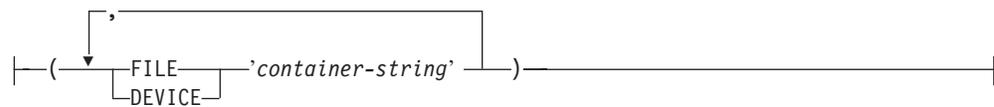
ALTER TABLESPACE



database-container-clause:



drop-container-clause:

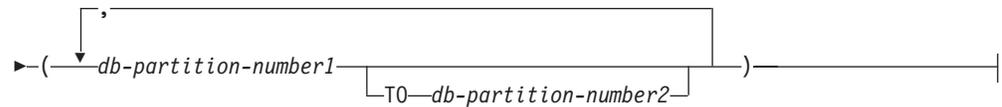
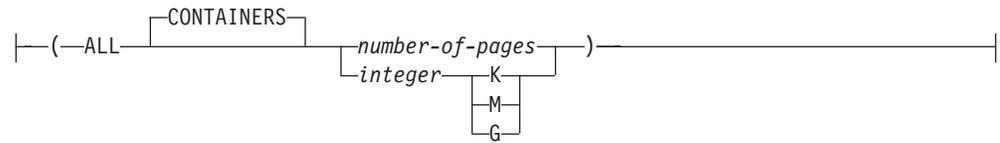


system-container-clause:



on-db-partitions-clause:



**all-containers-clause:****説明***tablespace-name*

表スペースの名前を指定します。これは、1 部構成の名前です。これは、長形式 SQL ID です (通常 ID または区切り ID のいずれか)。

ADD

表スペースに 1 つまたは複数の新しいコンテナを追加するように指定します。

TO STRIPE SET *stripeset*

1 つまたは複数の新しいコンテナを表スペースに追加し、指定されたストライプ・セットに配置するように指定します。

BEGIN NEW STRIPE SET

表スペースに新しいストライプ・セットを作成し、その新しいストライプ・セットに 1 つまたは複数のコンテナを追加するように指定します。ADD オプションを使用して後に追加されるコンテナは、TO STRIPE SET が指定されない場合にはこの新しいストライプ・セットに追加されます。

DROP

1 つまたは複数のコンテナを表スペースからドロップするように指定します。

REDUCE

非自動ストレージ表スペースでは、既存のコンテナのサイズを減らすように指示します。指定するサイズは、既存のコンテナから減らすサイズです。

all-containers-clause が指定されると、表スペースにあるすべてのコンテナがこのサイズだけ縮小されます。サイズを減らすことによって表スペースのサイズが現行の最高水準点より小さくなる場合、コンテナを縮小する前に最高水準点を下げを試みます。非自動ストレージ表スペースでは、REDUCE 節の後に *database-container-clause* または *all-containers-clause* を指定する必要があります。

自動ストレージ表スペースでは、可能な場合は現行の最高水準点を下げること、そして表スペースのサイズを新しい最高水準点まで減らすことを指定します。自動ストレージ表スペースの場合は、REDUCE 節の後に *database-container-clause* または *all-containers-clause* を指定してはなりません。

EXTEND

既存のコンテナのサイズを増やすように指示します。指定するサイズは、既存

ALTER TABLESPACE

のコンテナに追加されるサイズです。 *all-containers-clause* が指定されると、表スペースにあるすべてのコンテナがこのサイズで拡張されます。

RESIZE

既存のコンテナのサイズを変更するよう指定します。指定されるサイズが、コンテナの新しいサイズになります。 *all-containers-clause* が指定されると、表スペースにあるすべてのコンテナがこのサイズに変更されます。この操作が複数のコンテナに影響を与える場合には、そうしたコンテナすべてのサイズは増やすか減らすかのどちらかにする必要があります。一部を増やし、その他を減らすことはできません (SQLSTATE 429BC)。

database-container-clause

DMS 表スペースに 1 つ以上のコンテナを追加します。表スペースは、すでにアプリケーション・サーバーに存在する DMS 表スペースを指定するものでなければなりません。

drop-container-clause

1 つまたは複数のコンテナを DMS 表スペースからドロップします。表スペースは、すでにアプリケーション・サーバーに存在する DMS 表スペースを指定するものでなければなりません。

system-container-clause

指定のデータベース・パーティションにある SMS 表スペースに、1 つ以上のコンテナを追加します。表スペースは、すでにアプリケーション・サーバーに存在する SMS 表スペースを指定するものでなければなりません。表スペースに対して指定するデータベース・パーティションにコンテナがあってはなりません (SQLSTATE 42921)。

on-db-partitions-clause

1 つ以上のデータベース・パーティションを、それに対応するコンテナ操作に対して指定します。

all-containers-clause

DMS 表スペースにあるコンテナすべてを拡張、縮小またはサイズ変更します。表スペースは、すでにアプリケーション・サーバーに存在する DMS 表スペースを指定するものでなければなりません。

PREFETCHSIZE

照会によって参照される前に、照会に必要なデータを読み取るよう指定し、照会が入出力の実行を待たずに済むようにします。

AUTOMATIC

表スペースのプリフェッチ・サイズが自動的に更新されるように指定します。プリフェッチ・サイズは、次の公式を使って DB2 により管理されます。

$$\begin{aligned} \text{プリフェッチ・サイズ} = & \\ & (\text{コンテナの数}) * \\ & (\text{コンテナごとの物理ディスクの数}) * \\ & (\text{エクステント・サイズ}) \end{aligned}$$

コンテナごとの物理ディスクの数のデフォルトは、DB2_PARALLEL_IO レジストリー変数によって指定されているのでない限り、1 です。

表スペース内のコンテナー数が増えるたびに (1 つ以上のコンテナーを追加またはドロップする ALTER TABLESPACE ステートメントの正常実行に続いて)、DB2 はプリフェッチ・サイズを自動的に更新します。プリフェッチ・サイズはデータベースの開始時に更新されます。

PREFETCHSIZE 節に数値を指定することにより、プリフェッチ・サイズの自動更新をオフにすることができます。

number-of-pages

データのプリフェッチの実行中に、表スペースから読み取られる PAGESIZE ページの数を指定します。このプリフェッチ・サイズ値は、後に K (K バイトの場合)、M (M バイトの場合)、または G (G バイトの場合) を付けた整数値としても指定できます。このように指定した場合、バイト数をページ・サイズで割った値を超えない最大の整数が、プリフェッチ・サイズのページ値の数を判別するために使用されます。

BUFFERPOOL *bufferpool-name*

この表スペースの表に対して使用するバッファ・プールを指定します。バッファ・プールは、現在データベースに存在する必要があります (SQLSTATE 42704)。バッファ・プールに対して、この表スペースのデータベース・パーティション・グループを定義する必要があります (SQLSTATE 42735)。

OVERHEAD *number-of-milliseconds*

入出力制御装置のオーバーヘッドとディスク・シーク待ち時間をミリ秒単位で指定する数値リテラルです (整数、10 進数、または浮動小数点数)。この数値がすべてのコンテナーで同一でない場合、それは表スペースに属するすべてのコンテナーの平均でなければなりません。この値は、照会の最適化の過程で入出力コストを判別するのに使用されます。

TRANSFERRATE *number-of-milliseconds*

1 ページ (4K または 8K) をメモリーに読み込むための時間をミリ秒単位で指定する数値リテラルです (整数、10 進数、または浮動小数点数)。この数値がすべてのコンテナーで同一でない場合、それは表スペースに属するすべてのコンテナーの平均でなければなりません。この値は、照会の最適化の過程で入出力コストを判別するのに使用されます。

FILE SYSTEM CACHING または **NO FILE SYSTEM CACHING**

入出力操作をファイル・システム・レベルでキャッシュに入れるかどうかを指定します。新しいキャッシング・ポリシーが有効になる前に、データベースへの接続を終了する必要があります。長いデータまたは LOB データへの入出力アクセスは、SMS コンテナーと DMS コンテナーの両方でバッファに入れられることに注意してください。

FILE SYSTEM CACHING

ターゲット表スペース内のすべての入出力操作が、ファイル・システム・レベルでキャッシュに入れられます。

NO FILE SYSTEM CACHING

すべての入出力操作がファイル・システム・レベルのキャッシュを迂回します。

DROPPED TABLE RECOVERY

tablespace-name からドロップされた表を、ROLLFORWARD DATABASE コマ

ALTER TABLESPACE

ンドの RECOVER DROPPED TABLE ON オプションを使用して回復させることができるかどうかを指定します。パーティション表に関しては、ドロップされた表のリカバリーは、たとえ 1 つ以上の表スペースで非パーティション表についてオフにされたとしても、常にオンになります。

ON

ドロップされた表が回復可能であることを指定します。

OFF

ドロップされた表が回復不能であることを指定します。

SWITCH ONLINE

OFFLINE 状態の表スペースが、コンテナがアクセス可能であれば、オンラインになることを指定します。コンテナがアクセス可能でなければ、エラーが戻されます (SQLSTATE 57048)。

AUTORESIZE

データベース管理スペース (DMS) 表スペースまたは自動ストレージ表スペースの自動サイズ変更機能を使用可能にするかどうかを指定します。自動サイズ変更可能表スペースは、いっぱいになると、自動的にサイズが大きくなります。

NO

DMS 表スペースまたは自動ストレージ表スペースの自動サイズ変更機能が、使用不可であることを指定します。自動サイズ変更機能が使用不可の場合、先に INCREASESIZE または MAXSIZE に指定した値は保持されません。

YES

DMS 表スペースまたは自動ストレージ表スペースの自動サイズ変更機能が、使用可能であることを指定します。

INCREASESIZE *integer* PERCENT または INCREASESIZE *integer* K | M | G

自動サイズ変更可能な表スペースがいっぱいになって、スペースの要求がなされたとき、自動的に増加させる量を、データベース・パーティションごとに指定します。整数値の後に以下のものを指定しなければなりません。

- PERCENT。スペースの要求がなされた時点の表スペース・サイズのパーセンテージとして量を指定します。PERCENT を指定する場合、整数値は 0 と 100 の間でなければなりません (SQLSTATE 42615)。
- K (K バイト)、M (M バイト)、または G (G バイト)。バイト単位で量を指定します。

使用される実際の値は、指定されたものよりも若干小さかったり大きかったりすることにご注意ください。これは、データベース・マネージャーが表スペース内の各コンテナの増大の整合性を維持しようとするためです。

MAXSIZE *integer* K | M | G または MAXSIZE NONE

自動サイズ変更可能な表スペースを自動的に大きくできる、最大サイズを指定します。

integer

DMS 表スペースまたは自動ストレージ表スペースを自動的に大きくできる、サイズ上のハード限界を、データベース・パーティションごとに指定します。整数値の後に K (K バイトの場合)、M (M バイトの場合)、または G (G バイトの場合) を指定する必要があります。使用される実際の値は、

指定されたものよりも若干小さい場合があることにご注意ください。これは、データベース・マネージャーが表スペース内の各コンテナの増大の整合性を維持しようとするためです。

NONE

表スペースをファイル・システムの容量まで、または表スペースの最大サイズ（『SQL の制限値』で解説）まで増大できるようにすることを指定します。

CONVERT TO LARGE

既存の REGULAR DMS 表スペースを LARGE DMS 表スペースになるよう変更します。表スペースとその内容は、変換の際、ロックされます。このオプションは、REGULAR DMS 表スペースにのみ使用できます。SMS 表スペース、TEMPORARY 表スペース、またはシステム・カタログ表スペースが指定されると、エラーが戻されます (SQLSTATE 560CF)。他の表スペースにデータ・パーティションを持つパーティション表のデータ・パーティションを含む表スペースは変換できません (SQLSTATE 560CF)。変換は、コミット後に元に戻すことはできません。表スペース内の表が DATA CAPTURE CHANGES によって定義されている場合は、ターゲット表と表スペースの記憶容量制限を考慮するようにします。

規則

- BEGIN NEW STRIPE SET 節は、ADD、DROP、EXTEND、REDUCE、および RESIZE 節が別のデータベース・パーティションへ指示されない限り、同じステートメントでそれらを指定して使用することはできません (SQLSTATE 429BC)。
- TO STRIPE SET 節を使用して指定されたストライプ・セットは、変更される表スペースの有効な範囲内になければなりません (SQLSTATE 42615)。
- 表スペースにスペースを追加、または表スペースからスペースを削除する場合、以下の規則に従います。
 - EXTEND および RESIZE は、各コンテナのサイズを拡張する、同じステートメントで使用できます (SQLSTATE 429BC)。
 - REDUCE および RESIZE は、各コンテナのサイズを軽減する、同じステートメントで使用できます (SQLSTATE 429BC)。
 - EXTEND および REDUCE は、別のデータベース・パーティションに向けられない限りは、同じステートメントでは使用できません (SQLSTATE 429BC)。
 - ADD は、REDUCE または DROP が別のデータベース・パーティションに向けられない限りは、それらと共に同じステートメントでは使用できません (SQLSTATE 429BC)。
 - DROP は、EXTEND または ADD が別のデータベース・パーティションに向けられない限りは、それらと共に同じステートメントでは使用できません (SQLSTATE 429BC)。
- AUTORESIZE、INCREASESIZE、または MAXSIZE 節は、システム管理スペース (SMS) 表スペース、自動ストレージを使用して作成された TEMPORARY 表スペース、またはロー・デバイス・コンテナを使用するよう定義された DMS 表スペースには指定できません (SQLSTATE 42601)。

ALTER TABLESPACE

- INCREASESIZE または MAXSIZE 節は、表スペースが自動サイズ変更不可である場合には指定できません (SQLSTATE 42601)。
- 表スペースの新規最大サイズを指定する際、その値は各データベース・パーティション上の現行サイズよりも大きくなければなりません (SQLSTATE 560B0)。
- コンテナ操作 (ADD、EXTEND、RESIZE、DROP、または BEGIN NEW STRIPE SET) は、自動ストレージ表スペースに対しては実行できません。なぜならそのような表スペースのスペース管理は、データベース・マネージャーが制御しているからです (SQLSTATE 42858)。
- ロー・デバイス・コンテナを、自動サイズ変更可能 DMS 表スペースに追加することはできません (SQLSTATE 42601)。
- CONVERT TO LARGE 節は、他の節と同じステートメントに指定することができません (SQLSTATE 429BC)。

注

- 各コンテナ定義には、53 バイトに加えて、コンテナ名を保管するのに必要なバイト数が必要です。表スペースのすべてのコンテナ名を結合した長さは、20480 バイトを超えることはできません (SQLSTATE 54034)。
- デフォルトのコンテナ操作は、ALTER TABLESPACE ステートメントで指定されるコンテナ操作ですが、この操作は特定のデータベース・パーティションに明示的に向けられません。こうしたコンテナ操作は、ステートメントにリストされていない任意のデータベース・パーティションに向けられます。デフォルトのコンテナ操作がどのデータベース・パーティションにも向けられない場合には、コンテナ操作ではすべてのデータベース・パーティションに明示的に言及するので、警告が出されます (SQLSTATE 1758W)。
- スペースが表スペースに追加または、表スペースから削除され、トランザクションがコミットされると、表スペースのコンテンツはコンテナ間でバランスの再調整がなされるかもしれません。バランス再調整中も、表スペースへのアクセスは制限されません。
- 表スペースが OFFLINE 状態で、コンテナがアクセス可能である場合、すべてのアプリケーションを切断してから、もう一度データベースへ接続すれば、表スペースは OFFLINE 状態から脱することができます。別の方法として、SWITCH ONLINE オプションを使用すると、残りのデータベースは稼働状態で使用中のまま、表スペースは OFFLINE から脱する (稼働状態になる) ことができます。
- 表スペースに複数のコンテナを追加する場合は、バランスの再調整のコストが一度だけで済むように、これらのコンテナを同じステートメントで追加することをお勧めします。単一トランザクションで別々の ALTER TABLESPACE ステートメントを使用して、同じ表スペースにコンテナを追加するとエラーになります (SQLSTATE 55041)。
- 存在しないコンテナについて拡張、縮小、またはサイズ変更、またはドロップをしようとする、エラーが発生します (SQLSTATE 428B2)。
- コンテナを拡張、軽減またはサイズ変更する場合、このコンテナ・タイプは、コンテナが作成されたときに使用されたタイプと適合しなければなりません (SQLSTATE 428B2)。
- 1 つのトランザクションで、同じ表スペースに対して別個の ALTER TABLESPACE ステートメントを使用して、複数のコンテナ・サイズを変更しようとする、エラーが発生します (SQLSTATE 55041)。

- パーティション・データベースで、複数のデータベース・パーティションが同じ物理ノードに存在する場合、このようなデータベース・パーティションに同じデバイスまたは特定のパスを指定することはできません (SQLSTATE 42730)。この環境の場合、それぞれのデータベース・パーティションごとにユニークな *container-string* を指定するか、または相対パス名を使用してください。
- 表スペース定義はトランザクションであり、表スペース定義に対する変更はコミット時にカタログ表に反映されますが、新しい定義のバッファ・プールは、データベースの次回始動時まで使用することはできません。ALTER TABLESPACE ステートメントが出されたときに使用中のバッファ・プールは、それまで引き続き使用されます。
- REDUCE、RESIZE、または DROP オプションは必要に応じて DMS 表スペースの未使用エクステンツの解放を試み、REDUCE オプションは自動ストレージ表スペースの未使用エクステンツの解放を試みます。未使用エクステンツを除去すると、スペースの使用量を正確に表す値まで表スペースの最高水準点を下げることができ、その後、表スペース・サイズをさらに減らすことができます。
- **LARGE DMS 表スペースへの変換に関して:**
 - 変換後は、COMMIT ステートメントを実行してから、表スペースの記憶容量を増やすことを推奨します。
 - 表スペースで自動サイズ変更が使用可能に設定されている場合、MAXSIZE 表スペース属性が NONE に設定済みでなければ増やす必要があります。
 - 表スペースで自動サイズ変更が使用不可に設定されている場合、以下の一方または両方を行ってください。
 - AUTORESIZE YES オプションを使用して ALTER TABLESPACE ステートメントを実行し、自動サイズ変更を使用可能にする。
 - ストライプ・セットを追加し、既存のコンテナのサイズを拡張して、記憶容量を増やす。
 - 変換された表スペースにある表の索引は、再編成または再作成しない限り、LARGE レコード ID (RID) をサポートできません。
 - 索引の再編成には、REORG INDEXES ALL コマンド (CLEANUP ONLY 節なし) が使用できます。パーティション表については、ALLOW NO ACCESS オプションを指定します。
 - あるいは、表を (INPLACE ではなく) 再編成することもできます。それにより、すべての索引が再作成され、表がページあたり 255 を超える行をサポートできるようになります。
 - 再作成されたタイプ 1 索引はすべて自動的にタイプ 2 索引に変換されます。
 - どの表がまだ LARGE RID をサポートしていないかを判別するには、ADMIN_GET_TAB_INFO 表関数を使用します。
- **互換性**
 - バージョン 8 より前のバージョンのキーワードとの互換性:
 - DBPARTITIONNUM を NODE に置換できます。
 - DBPARTITIONNUMS を NODES に置換できます。

ALTER TABLESPACE

例

例 1: PAYROLL 表スペースにデバイスを追加します。

```
ALTER TABLESPACE PAYROLL
  ADD (DEVICE '/dev/rhdisk9' 10000)
```

例 2: ACCOUNTING 表スペースのプリフェッチ・サイズと入出力オーバーヘッドを変更します。

```
ALTER TABLESPACE ACCOUNTING
  PREFETCHSIZE 64
  OVERHEAD 19.3
```

例 3: 表スペース TS1 を作成した後、コンテナをサイズ変更して、すべてのコンテナのサイズが 2000 ページになるようにします。(このサイズ変更を実行する 3 つの異なる ALTER TABLESPACE ステートメントを示します。)

```
CREATE TABLESPACE TS1
  MANAGED BY DATABASE
  USING (FILE '/conts/cont0' 1000,
        DEVICE '/dev/rcont1' 500,
        FILE 'cont2' 700)
ALTER TABLESPACE TS1
  RESIZE (FILE '/conts/cont0' 2000,
        DEVICE '/dev/rcont1' 2000,
        FILE 'cont2' 2000)
```

または

```
ALTER TABLESPACE TS1
  RESIZE (ALL 2000)
```

または

```
ALTER TABLESPACE TS1
  EXTEND (FILE '/conts/cont0' 1000,
        DEVICE '/dev/rcont1' 1500,
        FILE 'cont2' 1300)
```

例 4: DATA_TS 表スペースにあるすべてのコンテナを 1000 ページだけ拡張します。

```
ALTER TABLESPACE DATA_TS
  EXTEND (ALL 1000)
```

例 5: INDEX_TS 表スペースにあるすべてのコンテナのサイズを 100 メガバイト (MB) に変更します。

```
ALTER TABLESPACE INDEX_TS
  RESIZE (ALL 100 M)
```

例 6: 3 つの新規コンテナを追加します。1 番目のコンテナを拡張し、2 番目をサイズ変更します。

```
ALTER TABLESPACE TS0
  ADD (FILE 'cont2' 2000, FILE 'cont3' 2000)
  ADD (FILE 'cont4' 2000)
  EXTEND (FILE 'cont0' 100)
  RESIZE (FILE 'cont1' 3000)
```

例 7: 表スペース TSO がデータベース・パーティション 0、1、および 2 に存在します。データベース・パーティション 0 に新規のコンテナを追加し、データベー

ス・パーティション 1 のすべてのコンテナを拡張し、明示的に指定されたデータベース・パーティション (つまり、データベース・パーティション 0 および 1) 以外のすべてのデータベース・パーティションのコンテナのサイズを変更します。

```
ALTER TABLESPACE TS0
  ADD (FILE 'A' 200) ON DBPARTITIONNUM (0)
  EXTEND (ALL 200) ON DBPARTITIONNUM (1)
  RESIZE (FILE 'B' 500)
```

この例では RESIZE 節がデフォルトのコンテナ節で、データベース・パーティション 2 で実行されます。他の操作は明示的にデータベース・パーティション 0 および 1 に向けられるからです。しかしデータベース・パーティションが 2 つしかない場合、ステートメントは正常に実行されますが、デフォルトのコンテナが指定されたものの使用されていないことを示す警告が出されます (SQL1758W)。

例 8: 表スペース DMS_TS1 の自動サイズ変更オプションを使用可能にし、その最大サイズを 256 メガバイトに設定します。

```
ALTER TABLESPACE DMS_TS1
  AUTORESIZE YES MAXSIZE 256 M
```

例 9: 表スペース AUTOSTORE1 の自動サイズ変更オプションを使用可能にし、その増加率を 5% に変更します。

```
ALTER TABLESPACE AUTOSTORE1
  AUTORESIZE YES INCREASESIZE 5 PERCENT
```

例 10: MY_TS という自動サイズ変更可能表スペースの増加率を 512 K バイトに変更し、その最大サイズを最大限に設定します。

```
ALTER TABLESPACE MY_TS
  INCREASESIZE 512 K MAXSIZE NONE
```

ALTER THRESHOLD

ALTER THRESHOLD ステートメントは、しきい値の定義を変更します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

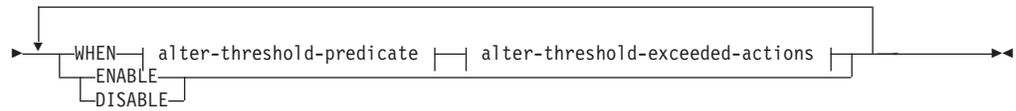
許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

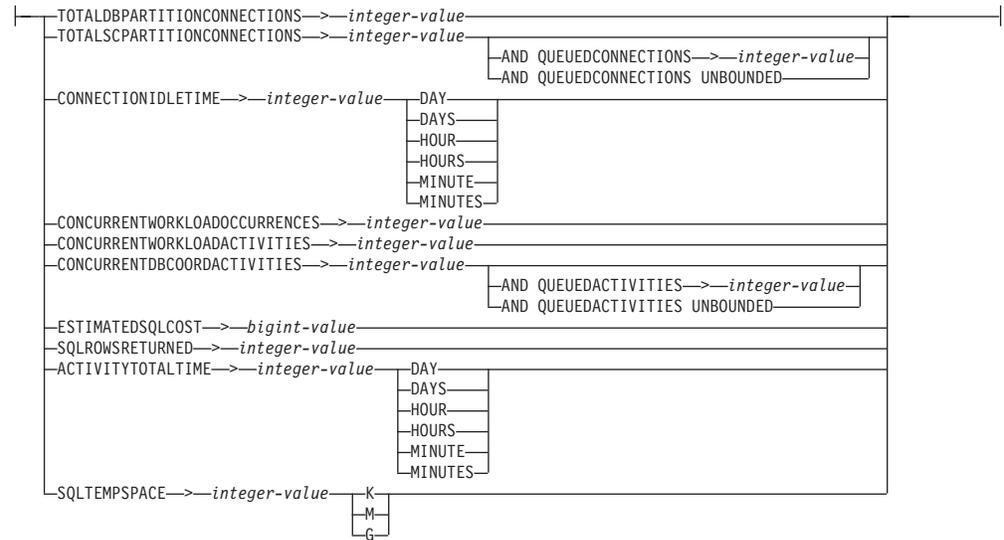
構文

```
▶▶—ALTER THRESHOLD—threshold-name—————▶▶
```

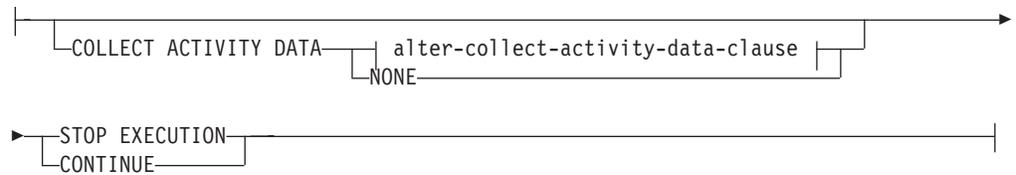
ALTER THRESHOLD



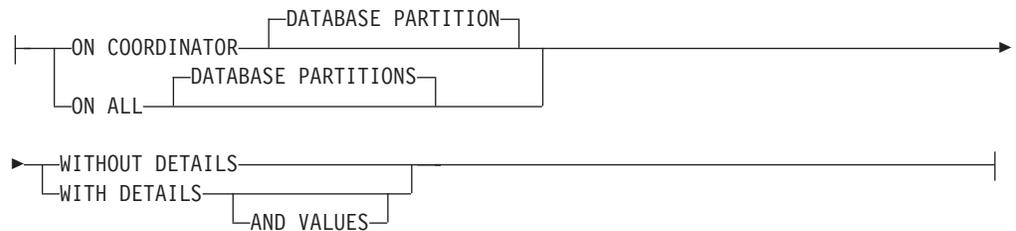
altered-threshold-predicate:



alter-threshold-exceeded-actions:



alter-collect-activity-data-clause:



説明

threshold-name

変更するしきい値を識別します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。名前は、現行のサーバー上の既存のしきい値を一意的に識別するものでなければなりません (SQLSTATE 42704)。

WHEN *alter-threshold-predicate*

しきい値条件の既存の上限の値を新しい上限の値に置き換えます。しきい値の条件を別のものに変更することはできません。

TOTALDBPARTITIONCONNECTIONS > *integer-value*

この条件は、データベース・パーティション上で並行して実行できるコーディネーター接続の数の上限を定義します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、新しいコーディネーター接続を接続させないことを意味します。現在実行中の接続やキューに入っている接続は続行されます。

TOTALSCPARTITIONCONNECTIONS > *integer-value*

この条件は、特定のサービス・スーパークラスのデータベース・パーティション上で並行して実行できるコーディネーター接続の数の上限を定義します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、新しい接続をサービス・クラスに結合しないことを意味します。現在実行中の接続やキューに入っている接続は続行されます。

AND QUEUEDCONNECTIONS > *integer-value* または AND QUEUEDCONNECTIONS UNBOUNDED

コーディネーター接続が最大数を越えた場合のキュー・サイズを指定します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、コーディネーター接続をキューに入れないことを意味します。UNBOUNDED を指定した場合は、指定されたコーディネーター接続の最大数を越えた接続がすべてキューに入れられ、*threshold-exceeded-actions* は実行されません。

CONNECTIONIDLETIME > *integer-value* (DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)

この条件は、データベース・マネージャーが接続をアイドル状態のままにしておく時間の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。有効な期間キーワードを使用して、*integer-value* に適切な時間の単位を指定してください。この条件は、コーディネーターのデータベース・パーティションで論理的に強制されます。

STOP EXECUTION アクションを CONNECTIONIDLETIME しきい値とともに指定した場合、しきい値を超過すると、アプリケーションの接続はドロップされます。それ以降にアプリケーションがデータ・サーバーへのアクセスを試行しても、SQLSTATE 5U026 を受け取ることはありません。

このしきい値の最小の細分は 5 分であるため、指定された値は、最も近いゼロ以外の 5 分の倍数に切り上げられます。切り上げの結果がオーバーフローまたはアンダーフローになる場合は、整数の範囲の中で最も近い値が選択されます。

このしきい値の最大値は、2,147,400 秒です。2,147,400 秒よりも大きい秒に相当する値が指定された場合は、この秒数に設定されます。

DB2CHECKCLIENTINTERVAL レジストリー変数を変更してチェックの頻度を下げると、このしきい値の細分が影響を受けることがあります。

CONCURRENTWORKLOADOCCURRENCES > *integer-value*

この条件は、各データベース・パーティションでの並行するワークロード・オカレンスの数の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。

CONCURRENTWORKLOADACTIVITIES > *integer-value*

この条件は、各データベース・パーティションのワークロードで並行して実行されるコーディネーター・アクティビティーとネストされたアクティビティーの数の上限を定義します。この値には、ゼロ以外の正整数を指定できません (SQLSTATE 42820)。

ネストされるアクティビティーは、それぞれ以下の条件を満たしている必要があります。

- 認識されているコーディネーター・アクティビティーである必要があります。認識されているアクティビティーのタイプに含まれないネストされたコーディネーター・アクティビティーは、カウントされません。同じように、リモート・ノード要求などのネストされたサブエージェント・アクティビティーもカウントされません。
- SQL ステートメントを発行するユーザー作成のプロシージャなどの、ユーザー・ロジックから直接呼び出される必要があります。

したがって、DB2 ユーティリティーや SYSIBM、SYSFUN、または SYSPROC スキーマのルーチンの呼び出しによって自動的に開始された、ネストされたコーディネーター・アクティビティーは、このしきい値で指定された上限にカウントされません。

制約の設定やマテリアライズ照会表のリフレッシュによって生成されるアクティビティーなどの内部 SQL アクティビティーも、データベース・マネージャーによって開始されるもので、ユーザー・ロジックによって直接呼び出されるものではないため、このしきい値にはカウントされません。

CONCURRENTDBCOORDACTIVITIES > *integer-value*

この条件は、指定されたドメイン内のすべてのデータベース・パーティションで並行して実行できる、認識されているデータベース・コーディネーター・アクティビティーの数の上限を定義します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、新しいデータベース・コーディネーター・アクティビティーを実行しないことを意味します。現在実行中またはキューに入っているデータベース・コーディネーター・アクティビティーは続行されます。

重要: CONCURRENTDBCOORDACTIVITIES しきい値が原因で、外部からの操作を必要とするキュー・ベースの競合が発生することがあります。詳しくは、『CONCURRENTDBCOORDACTIVITIES しきい値』を参照してください。

AND QUEUEDACTIVITIES > *integer-value* または **AND QUEUEDACTIVITIES UNBOUNDED**

データベース・コーディネーター・アクティビティーが最大数を越えた場合のキュー・サイズを指定します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、データベース・コーディネーター・アクティビティーをキューに入れなかったことを意味します。UNBOUNDED を指定した場合、指定されたデータベース・コーデ

イネーター・アクティビティの最大数を越えたデータベース・コーディネーター・アクティビティがすべてキューに入れられ、*threshold-exceeded-actions* は実行されません。

ESTIMATEDSQLCOST > *bigint-value*

この条件は、アクティビティのオブティマイザー割り当てコスト (timeron 単位) の上限を定義します。この値には、ゼロ以外の任意の 64 ビット正整数を指定できます (SQLSTATE 42820)。この条件は、コーディネーターのデータベース・パーティションで強制されます。この条件では、以下のアクティビティが追跡されます。

- データ操作言語 (DML) タイプのコーディネーター・アクティビティ。
- ユーザー・ロジックから呼び出されるネストされた DML アクティビティ。したがって、データベース・マネージャー (ユーティリティ、プロシージャー、内部 SQL など) によって開始される DML アクティビティは、この条件では追跡されません (ただし、コストが親の見積もりに含まれている場合は、これらのアクティビティは間接的に追跡されます)。

SQLROWSRETURNED > *integer-value*

この条件は、アプリケーション・サーバーからクライアント・アプリケーションに戻される行の数の上限を定義します。この値には、ゼロ以外の任意の整数を指定できます (SQLSTATE 42820)。この条件は、コーディネーターのデータベース・パーティションで強制されます。この条件では、以下のアクティビティが追跡されます。

- DML タイプのコーディネーター・アクティビティ。
- ユーザー・ロジックから派生するネストされた DML アクティビティ。ユーティリティ、プロシージャー、または内部 SQL によってデータベース・マネージャーから開始されたアクティビティは、この条件による影響を受けません。

プロシージャー内から戻される結果セットは、個々のアクティビティとして別個に扱われます。プロシージャーそのものから戻される行の集約はありません。

ACTIVITYTOTALTIME > *integer-value* (DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)

この条件は、アクティビティがキューに入れている時間を含んだ、データベース・マネージャーがアクティビティの実行のために許可する時間の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。有効な期間キーワードを使用して、*integer-value* に適切な時間の単位を指定してください。この条件は、コーディネーターのデータベース・パーティションで論理的に強制されます。

このしきい値の最小の細分は 5 分であるため、指定された値は、最も近いゼロ以外の 5 分の倍数に切り上げられます。切り上げの結果がオーバーフローまたはアンダーフローになる場合は、整数の範囲の中で最も近い値が選択されます。

このしきい値の最大値は、2,147,400 秒です。2,147,400 秒よりも大きい秒に相当する値が指定された場合は、この秒数に設定されます。

ALTER THRESHOLD

DB2CHECKCLIENTINTERVAL レジストリー変数を変更してチェックの頻度を下げると、このしきい値の細分が影響を受けることがあります。

SQLTEMPSPACE > integer-value (K | M | G)

この条件は、すべてのデータベース・パーティションでの SYSTEM TEMPORARY 表スペースのサイズの上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。

integer-value K (大文字または小文字のどちらでも可) を指定した場合、最大サイズは *integer-value* の 1024 倍です。 *integer-value M* を指定した場合は、最大サイズは *integer-value* の 1 048 576 倍です。 *integer-value G* を指定した場合は、最大サイズは *integer-value* の 1 073 741 824 倍です。

この条件では、以下のアクティビティーが追跡されます。

- DML タイプのコーディネーター・アクティビティーと、それに対応するサブエージェント作業 (サブセクション実行)。
- ユーザー・ロジックから派生するネストされた DML アクティビティーと、それに対応するサブエージェント作業 (サブセクション実行)。ユーティリティ、プロシージャ、または内部 SQL によってデータベース・マネージャーから開始されたアクティビティーは、この条件による影響を受けません。

alter-threshold-exceeded-actions

条件を超過したときに実行するアクションを指定します。条件を超過するたびに、しきい値違反イベント・モニター (アクティブになっている場合) にイベントが記録されます。

COLLECT ACTIVITY DATA

しきい値を超過した各アクティビティーに関するデータを、アクティビティーが完了したときに、アクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

アクティビティーのコーディネーターのデータベース・パーティションでのみアクティビティー・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティーが処理されるすべてのデータベース・パーティションでアクティビティー・データを収集することを指定します。予測しきい値の場合、しきい値を超過した場合の CONTINUE アクションも指定した場合にのみ、アクティビティー情報がすべてのパーティションで収集されます。反応しきい値の場合、ON ALL DATABASE PARTITIONS 節を指定しても効果はなく、アクティビティー情報は常にコーディネーター・パーティションでのみ収集されます。予測しきい値および反応しきい値の両方について、アクティビティーの詳細または値は、コーディネーター・パーティションでのみ収集されます。

WITHOUT DETAILS

しきい値を超過する各アクティビティーに関するデータを、アクテ

アクティビティが完了したときに、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境の情報はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティについて、これらの情報をアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティに入力データ値が含まれている場合、それをアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

しきい値を超過する各アクティビティについて、アクティビティ・データを収集しないことを指定します。

STOP EXECUTION

アクティビティの実行を停止し、エラーを戻します (SQLSTATE 5U026)。

CONTINUE

アクティビティの実行を停止しません。条件にキューも含まれている場合にこのオプションを指定すると、キューイングがキューのサイズを超えて拡張します。

ENABLE または DISABLE

データベース・マネージャーでしきい値を使用可能にするかどうかを指定します。

ENABLE

データベース・アクティビティの実行を制限するために、データベース・マネージャーでしきい値を使用します。現在実行中のデータベース・アクティビティは、このしきい値の制限に関係なく、引き続き実行されます。

DISABLE

データベース・アクティビティの実行を制限するためにデータベース・マネージャーでしきい値を使用しません。新しいデータベース・アクティビティはこのしきい値による制限を受けません。

TOTALSCPARTITIONCONNECTIONS や

CONCURRENTDBCOORDACTIVITIES などのキューで使用されるしきい値は、無効にしてからドロップする必要があります。

注

- しきい値に新しく指定された値は、操作がコミットされた後に実行が開始された DB2 アクティビティに対してのみ影響を及ぼします。

例

しきい値 MAXBIGQUERIESCONCURRENCY のアクティビティの最大数を 2 から 3 に変更します。

ALTER THRESHOLD

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY  
WHEN CONCURRENTDBCOORDACTIVITIES > 3  
STOP EXECUTION
```

これはキューを伴うしきい値であるため、このしきい値は、次のように無効にしてからでなければドロップできません。

```
ALTER THRESHOLD MAXBIGQUERIESCONCURRENCY DISABLE
```

ALTER TRUSTED CONTEXT

ALTER TRUSTED CONTEXT ステートメントは、現行サーバーでトラステッド・コンテキストの定義を変更します。

呼び出し

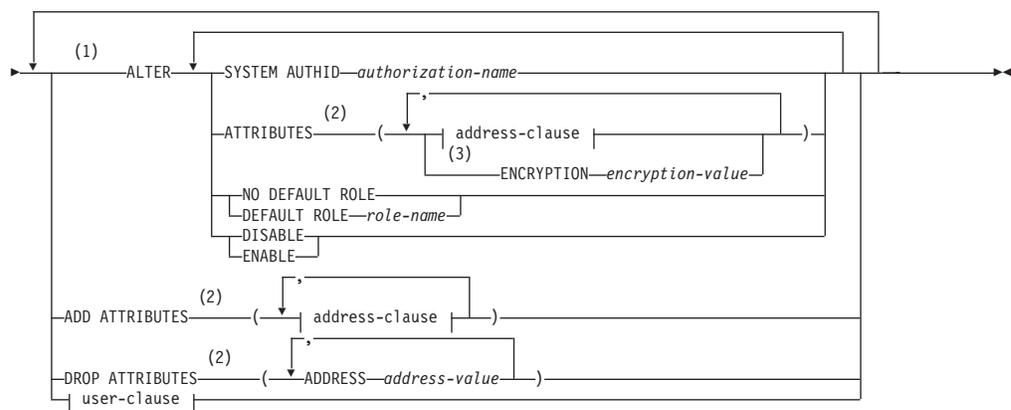
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

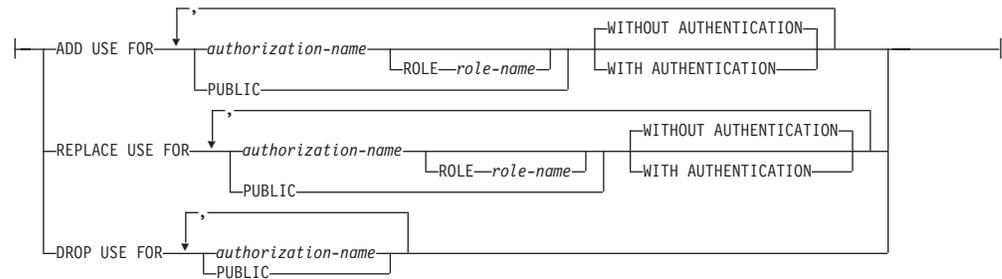
構文

```
▶▶ALTER TRUSTED CONTEXT—context-name▶▶
```



address-clause:

```
|ADDRESS—address-value | WITH ENCRYPTION—encryption-value |
```

user-clause:**注:**

- 1 ATTRIBUTES、DEFAULT ROLE、ENABLE、および WITH USE 節はそれぞれ 1 度しか指定できません (SQLSTATE 42614)。
- 2 属性名およびそれに対応する値はそれぞれ固有でなければなりません (SQLSTATE 4274D)。
- 3 ENCRYPTION を複数回指定することはできませんが (SQLSTATE 42614)、WITH ENCRYPTION は指定されている ADDRESS ごとに指定できます。

説明*context-name*

変更するトラステッド・コンテキストを指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *context-name* は現行のサーバーに存在するトラステッド・コンテキストを識別するものでなければなりません (SQLSTATE 42704)。

ALTER

トラステッド・コンテキストのオプションと属性を変更します。

SYSTEM AUTHID *authorization-name*

コンテキストがシステム許可 ID *authorization-name* によって確立される接続であることを指定します。これを既存のトラステッド・コンテキストと関連付けてはなりません (SQLSTATE 428GL)。これにステートメントの許可 ID を指定することはできません (SQLSTATE 42502)。

ATTRIBUTES (...)

変更する 1 つ以上の接続トラスト属性のリスト (これにトラステッド・コンテキストが定義される) を指定します。指定された属性の既存の値は新規値で置き換えられます。属性が現在トラステッド・コンテキスト定義の一部となっていない場合にはエラーが戻されます (SQLSTATE 4274C)。指定されない属性はその以前の値を保持します。

ADDRESS *address-value*

クライアントがデータベース・サーバーと通信するために使用する実際の通信アドレスを指定します。サポートされるプロトコルは TCP/IP のみです。指定されたトラステッド・コンテキストの以前の ADDRESS 値は除去されます。ADDRESS 属性は複数回指定できますが、*address-value* の対はそれぞれ属性のセットで固有でなければなりません (SQLSTATE 4274D)。

トラステッド接続を確立するときにトラステッド・コンテキストの ADDRESS 属性に対して複数の値が定義されている場合、候補となる接続によって使用されるアドレスがトラステッド・コンテキストの ADDRESS 属性の定義値のいずれかと一致していると、その接続はこの属性と一致しているとみなされます。

address-value

ADDRESS トラスト属性と関連付けられる値を含むストリング定数を指定します。 *address-value* は、IPv4 アドレス、IPv6 アドレス、またはセキュア・ドメイン名でなければなりません。

- IPv4 アドレスの先頭にスペースが含まれてはなりません。このアドレスは小数点付き 10 進数アドレスとして表されます。例えば IPv4 アドレスは 9.112.46.111 のようになります。値 'localhost' またはそれに相当する表現 '127.0.0.1' は、一致という結果になりません。代わりにホストの実 IPv4 アドレスを指定する必要があります。
- IPv6 アドレスの先頭にスペースが含まれてはなりません。このアドレスはコロン区切りの 16 進アドレスとして表されます。例えば IPv6 アドレスは 2001:0DB8:0000:0000:0008:0800:200C:417A のようになります。IPv4 がマップされた IPv6 アドレス (例えば ::ffff:192.0.2.128) は、一致という結果になりません。同じように、'localhost' またはその IPv6 短表現 '::1' も一致という結果になりません。
- ドメイン・ネームはドメイン・ネーム・サーバーで IP アドレスに変換されます。結果として生成される IPv4 または IPv6 アドレスはこのサーバーで決定されます。例えばドメイン・ネームは corona.torolab.ibm.com のようになります。ドメイン・ネームが IP アドレスに変換されたとき、この変換の結果が 1 つ以上の IP アドレスのセットになる場合があります。その場合、接続開始時の IP アドレスがドメイン名変換後の IP アドレスのいずれかと一致すると、着信接続はトラステッド・コンテキスト・オブジェクトの ADDRESS 属性と一致しているとみなされます。トラステッド・コンテキスト・オブジェクトを作成するとき、特に動的ホスト構成プロトコル (DHCP) 環境では、静的 IP アドレスの代わりにドメイン・ネーム値を ADDRESS 属性に提供することをお勧めします。DHCP ではデバイスがネットワークと接続するたびに IP アドレスが変わります。そのため、トラステッド・コンテキスト・オブジェクトの ADDRESS 属性に静的 IP アドレスを提供すると、デバイスによっては意図せずにトラステッド接続を取得してしまう場合があります。トラステッド・コンテキスト・オブジェクトの ADDRESS 属性にドメイン・ネームを指定すると、DHCP 環境におけるこの問題を回避できます。

WITH ENCRYPTION *encryption-value*

この特定の *address-value* に関するデータ・ストリームまたはネットワーク暗号化の最小暗号化レベルを指定します。この *encryption-value* は、この特定の *address-value* に関するグローバル ENCRYPTION 属性の設定をオーバーライドします。

encryption-value

この特定の *address-value* に関する ENCRYPTION トラスト属性と関連付けられる値を含むストリング定数を指定しません。 *encryption-value* は以下のいずれかでなければなりません (SQLSTATE 42615)。

- NONE。特定レベルの暗号化は不要です。
- LOW。最小の低レベルの暗号化が必要です。着信接続がこの特定アドレスの暗号化設定と一致する場合、データベース・マネージャーの認証タイプは DATA_ENCRYPT でなければなりません。
- HIGH。着信接続がこの特定アドレスの暗号化設定と一致する場合、DB2 クライアントと DB2 サーバーの間のデータ通信に関して Secure Sockets Layer (SSL) 暗号化を使用する必要があります。

ENCRYPTION *encryption-value*

データ・ストリームまたはネットワーク暗号化の最小暗号化レベルを指定します。デフォルトは NONE です。

encryption-value

この特定の *address-value* に関する ENCRYPTION トラスト属性と関連付けられる値を含むストリング定数を指定します。

encryption-value は以下のいずれかでなければなりません (SQLSTATE 42615)。

- NONE。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、特定レベルの暗号化は不要です。
- LOW。最小の低レベルの暗号化が必要です。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、データベース・マネージャーの認証タイプは DATA_ENCRYPT でなければなりません。
- HIGH。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、DB2 クライアントと DB2 サーバーの間のデータ通信に関して Secure Sockets Layer (SSL) 暗号化を使用する必要があります。

ENCRYPTION トラスト属性について詳しくは、『CREATE TRUSTED CONTEXT』を参照してください。

NO DEFAULT ROLE または **DEFAULT ROLE** *role-name*

デフォルトのロールをこのトラステッド・コンテキストに基づくトラステッド接続と関連付けるかどうかを指定します。このコンテキストのトラステッド接続がアクティブの場合、変更は次にユーザーの切り替えを要求するときまたは新規接続を要求するとき有効になります。

NO DEFAULT ROLE

トラステッド・コンテキストがデフォルトのロールを持たないことを指定します。

DEFAULT ROLE *role-name*

role-name がトラステッド・コンテキストのデフォルトのロールであること

ALTER TRUSTED CONTEXT

を指定します。 *role-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。トラステッド・コンテキストの定義の一部としてユーザー固有のロールがユーザーに定義されていない場合、このトラステッド・コンテキストに基づいて、トラステッド接続の中でこのロールがそのユーザーに使用されます。

ENABLE または DISABLE

トラステッド・コンテキストが使用可能かまたは使用不可かを指定します。

ENABLE

トラステッド・コンテキストが使用可能であることを指定します。

DISABLE

トラステッド・コンテキストが使用不可であることを指定します。トラステッド接続を確立するとき、使用不可のトラステッド・コンテキストは考慮されません。

ADD ATTRIBUTES

トラステッド・コンテキストが定義される 1 つ以上の追加トラスト属性のリストを指定します。

ADDRESS *address-value*

クライアントがデータベース・サーバーと通信するために使用する実際の通信アドレスを指定します。サポートされるプロトコルは TCP/IP のみです。ADDRESS 属性は複数回指定できますが、*address-value* の対はそれぞれ属性のセットで固有でなければなりません (SQLSTATE 4274D)。

トラステッド接続を確立するときにトラステッド・コンテキストの ADDRESS 属性に対して複数の値が定義されている場合、候補となる接続によって使用されるアドレスがトラステッド・コンテキストの ADDRESS 属性の定義値のいずれかと一致していると、その接続はこの属性と一致しているとみなされます。

address-value

ADDRESS トラスト属性と関連付けられる値を含むストリング定数を指定します。 *address-value* は、IPv4 アドレス、IPv6 アドレス、またはセキュア・ドメイン名でなければなりません。

- IPv4 アドレスの先頭にスペースが含まれてはなりません。このアドレスは小数点付き 10 進数アドレスとして表されます。例えば IPv4 アドレスは 9.112.46.111 のようになります。値 'localhost' またはそれに相当する表現 '127.0.0.1' は、一致という結果になりません。代わりにホストの実 IPv4 アドレスを指定する必要があります。
- IPv6 アドレスの先頭にスペースが含まれてはなりません。このアドレスはコロン区切りの 16 進アドレスとして表されます。例えば IPv6 アドレスは 2001:0DB8:0000:0000:0008:0800:200C:417A のようになります。IPv4 がマップされた IPv6 アドレス (例えば ::ffff:192.0.2.128) は、一致という結果になりません。同じように、'localhost' またはその IPv6 短表現 '::1' も一致という結果になりません。
- ドメイン・ネームはドメイン・ネーム・サーバーで IP アドレスに変換されます。結果として生成される IPv4 または IPv6 アドレスはこ

のサーバーで決定されます。例えばドメイン・ネームは `corona.torolab.ibm.com` のようになります。

WITH ENCRYPTION *encryption-value*

この特定の *address-value* に関するデータ・ストリームまたはネットワーク暗号化の最小暗号化レベルを指定します。この *encryption-value* は、この特定の *address-value* に関するグローバル ENCRYPTION 属性の設定をオーバーライドします。

encryption-value

この特定の *address-value* に関する ENCRYPTION トラスト属性と関連付けられる値を含むストリング定数を指定します。

encryption-value は以下のいずれかでなければなりません (SQLSTATE 42615)。

- NONE。特定レベルの暗号化は不要です。
- LOW。最小の低レベルの暗号化が必要です。着信接続がこの特定アドレスの暗号化設定と一致する場合、データベース・マネージャーの認証タイプは DATA_ENCRYPT でなければなりません。
- HIGH。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、DB2 クライアントと DB2 サーバーの間のデータ通信に関して Secure Sockets Layer (SSL) 暗号化を使用する必要があります。

DROP ATTRIBUTES

1 つ以上の属性をトラステッド・コンテキストの定義からドロップすることを指定します。属性と属性値の対が現在トラステッド・コンテキスト定義の一部となっていない場合にはエラーが戻されます (SQLSTATE 4274C)。

ADDRESS *address-value*

指定された通信アドレスをトラステッド・コンテキストの定義から除去することを指定します。 *address-value* は、既存の ADDRESS トラスト属性の値を含むストリング定数を指定します。

ADD USE FOR

このトラステッド・コンテキストに基づくトラステッド接続を使用できる追加ユーザーを指定します。トラステッド・コンテキストの定義で PUBLIC とユーザーのリストからのアクセスが許可されている場合、ユーザーの指定が PUBLIC の指定をオーバーライドします。

authorization-name

指定された *authorization-name* でトラステッド接続を使用できることを指定します。 *authorization-name* はトラステッド・コンテキストを使用することがすでに定義されている許可 ID を識別するものであってはならず、ADD USE FOR 節内に複数回指定することはできません (SQLSTATE 428GM)。これにステートメントの許可 ID を指定することもできません (SQLSTATE 42502)。

ROLE *role-name*

使用されるユーザーのロールが *role-name* であることを指定します。 *role-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。ユーザーに対して明示的に指定され

ALTER TRUSTED CONTEXT

たロールは、トラステッド・コンテキストと関連付けられているすべてのデフォルトのロールをオーバーライドします。

PUBLIC

すべてのユーザーがこのトラステッド・コンテキストに基づくトラステッド接続を使用できることを指定します。トラステッド・コンテキストを使用するために **PUBLIC** が既に定義されてはならず、**PUBLIC** を **ADD USE FOR** 節内で複数回指定されていてもなりません (SQLSTATE 428GM)。

WITHOUT AUTHENTICATION または **WITH AUTHENTICATION**

このトラステッド・コンテキストに基づくトラステッド接続で現行のユーザーを切り替えるときに認証を必要とするかどうかを指定します。

WITHOUT AUTHENTICATION

このトラステッド接続に基づくトラステッド接続で現行のユーザーをこのユーザーに切り替えるときに認証を必要としないことを指定します。

WITH AUTHENTICATION

このトラステッド接続に基づくトラステッド接続で現行のユーザーをこのユーザーに切り替えるときに認証を必要とすることを指定します。

REPLACE USE FOR

特定のユーザーまたは **PUBLIC** がトラステッド・コンテキストを使用する方法が変更されることを指定します。

authorization-name

トラステッド接続の使用が変更されるユーザーの *authorization-name* を指定します。トラステッド・コンテキストで *authorization-name* による使用許可が既に定義されていなければなりません (SQLSTATE 428GN)。また、*authorization-name* を **REPLACE USE FOR** 節内で複数回指定されてはなりません (SQLSTATE 428GM)。これにステートメントの許可 ID を指定することもできません (SQLSTATE 42502)。

ROLE *role-name*

role-name がユーザーのロールであることを指定します。*role-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。ユーザーに対して明示的に指定されたロールは、トラステッド・コンテキストと関連付けられているすべてのデフォルトのロールをオーバーライドします。

PUBLIC

PUBLIC によって使用されるトラステッド接続の属性が変更されることを指定します。トラステッド・コンテキストで **PUBLIC** による使用許可が既に定義されていなければなりません (SQLSTATE 428GN)。また、**PUBLIC** を **REPLACE USE FOR** 節内で複数回指定されてはなりません (SQLSTATE 428GM)。

WITHOUT AUTHENTICATION または **WITH AUTHENTICATION**

このトラステッド・コンテキストに基づくトラステッド接続で現行のユーザーを切り替えるときに認証を必要とするかどうかを指定します。

WITHOUT AUTHENTICATION

このトラステッド接続に基づくトラステッド接続で現行のユーザーをこのユーザーに切り替えるときに認証を必要としないことを指定します。

WITH AUTHENTICATION

このトラステッド接続に基づくトラステッド接続で現行のユーザーをこのユーザーに切り替えるときに認証を必要とすることを指定します。

DROP USE FOR

トラステッド・コンテキストを使用できなくなるユーザーを指定します。トラステッド・コンテキストの定義から除去されるのは、現在トラステッド・コンテキストの使用を許可されているユーザーです。1人以上のユーザー (全ユーザーではない) をトラステッド・コンテキストの定義から除去できる場合、指定されたユーザーが除去され、警告が戻されます (SQLSTATE 01682)。指定されたユーザーをトラステッド・コンテキストの定義から除去できない場合にはエラーが戻されます (SQLSTATE 428GN)。

authorization-name

指定された許可 ID がこのトラステッド・コンテキストを使用できないようにします。

PUBLIC

すべてのユーザー (ただしシステム許可 ID と明示的に使用可能にされた個々の許可 ID は除く) がこのトラステッド・コンテキストを使用できないようにします。

規則

- トラステッド・コンテキスト排他 SQL ステートメントの後は、COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。トラステッド・コンテキスト排他 SQL ステートメントは次のとおりです。
 - CREATE TRUSTED CONTEXT、ALTER TRUSTED CONTEXT、または DROP (TRUSTED CONTEXT)
- グローバル・トランザクション内でトラステッド・コンテキスト排他 SQL ステートメントを発行することはできません。例えば、フェデレーテッド・トランザクションにおける 2 フェーズ・コミットの一部として開始されるグローバル・トランザクションまたは XA トランザクションなどの場合です (SQLSTATE 51041)。

注

- トラステッド・コンテキスト定義の一部として IP アドレスを提供するとき、そのアドレスの形式はネットワークで有効なものでなければなりません。例えば、ネットワークが IPv4 であるのに IPv6 形式のアドレスを提供しても一致には至りません。混合環境では、IPv4 と IPv6 両方のアドレス表現を指定するとよいでしょう。さらに望ましいのは、セキュア・ドメイン・ネーム (例えば corona.torolab.ibm.com) を指定することです。その場合、アドレス・フォーマットの詳細が非表示になるからです。
- データベース・パーティション全体を通じて、同時に実行できる非コミットのトラステッド・コンテキスト排他 SQL ステートメントは 1 つのみです。非コミットのトラステッド・コンテキスト排他 SQL ステートメントが実行されている場合、後続のトラステッド・コンテキスト排他 SQL ステートメントは、現行のトラステッド・コンテキスト排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。

ALTER TRUSTED CONTEXT

- システム・カタログに変更が書き込まれますが、それらがコミットされるまでは (たとえステートメントを発行する接続であったとしても) 有効になりません。
- 操作順序:** ALTER TRUSTED CONTEXT ステートメント内の操作順序は次のとおりです。
 - DROP
 - ALTER
 - ADD ATTRIBUTES
 - ADD USE FOR
 - REPLACE USE FOR
- 既存のトラステッド接続における変更の有効化:** 変更されるトラステッド・コンテキストについてのトラステッド接続が存在する場合、次にユーザーの切り替えを要求するときまたは接続を終了するときまで接続はトラステッドの状態を保ち、それまでは ALTER TRUSTED CONTEXT ステートメント発行前の定義が有効となります。このコンテキストのトラステッド接続がアクティブの間にトラステッド・コンテキストが使用不可になる場合、次にユーザーの切り替えを要求するときまたは接続を終了するときまで接続はトラステッドの状態を保ちます。ALTER TRUSTED CONTEXT ステートメントによってトラスト属性が変更される場合、トラステッド・コンテキストを使用する ALTER TRUSTED CONTEXT ステートメント発行時に存在するトラステッド接続は継続を許可されます。
- ロールの特権:** ユーザーまたはトラステッド・コンテキストにロールが関連付けられていない場合、ユーザーに関連付けられている特権のみ適用可能です。これはトラステッド・コンテキストにない場合と同じです。

例

例 1: APPSERVER というトラステッド・コンテキストが存在し、それが使用可能になっているとします。ALTER TRUSTED CONTEXT ステートメントを発行し、Bill がトラステッド・コンテキスト APPSERVER を使用できるようにし、さらにトラステッド・コンテキストが使用不可の状態となるようにします。

```
ALTER TRUSTED CONTEXT APPSERVER
DISABLE
ADD USE FOR BILL
```

例 2: SECUREROLE というトラステッド・コンテキストが存在するとします。ALTER TRUSTED CONTEXT ステートメントを発行して、既存のユーザー Joe が認証付きでトラステッド・コンテキストを使用するように変更し、それ以外の全員が認証なしでトラステッド・コンテキストを使用するように追加します。

```
ALTER TRUSTED CONTEXT SECUREROLE
REPLACE USE FOR JOE WITH AUTHENTICATION
ADD USE FOR PUBLIC WITHOUT AUTHENTICATION
```

例 3: ADDRESS 属性値 '9.13.55.100' および '9.12.30.112'、ENCRYPTION 属性値 'NONE' を持つ SECUREROLEENCRYPT というトラステッド・コンテキストが存在するとします。ALTER ステートメントを発行して、ADDRESS 属性値を変更し、暗号化属性を 'LOW' に変更します。

```
ALTER TRUSTED CONTEXT SECUREROLEENCRYPT
ALTER ATTRIBUTES (ADDRESS '9.12.155.200',
ENCRYPTION 'LOW')
```

ALTER TYPE (構造化)

ALTER TYPE ステートメントは、ユーザー定義の構造化タイプの属性またはメソッド指定を追加またはドロップします。既存のメソッドのプロパティも変更が可能です。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

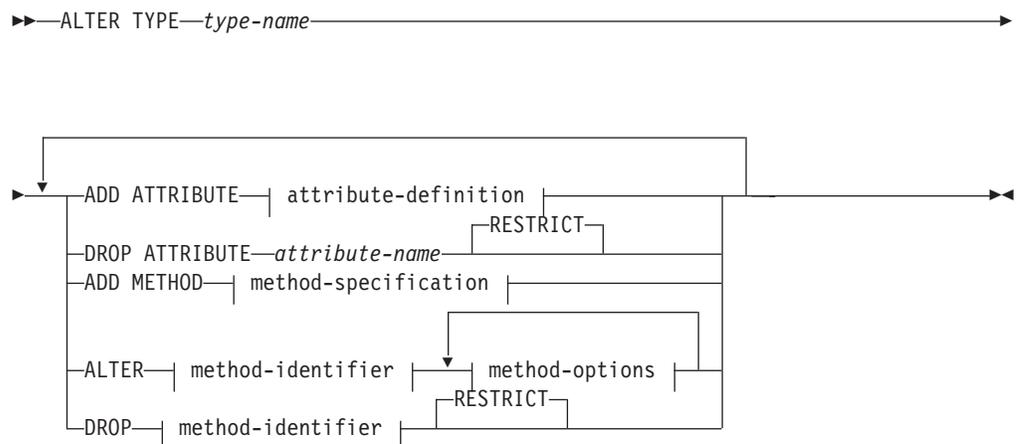
- タイプのスキーマに対する ALTERIN 特権
- SYSCAT.DATATYPES カタログ・ビューの OWNER 列に記録されているそのタイプの所有者
- SYSADM または DBADM 権限

fenced でないようにメソッドを変更するには、ステートメントの許可 ID の特権に以下の特権の少なくとも 1 つが含まれている必要があります。

- データベースに対する CREATE_NOT_FENCED_ROUTINE 権限
- SYSADM または DBADM 権限

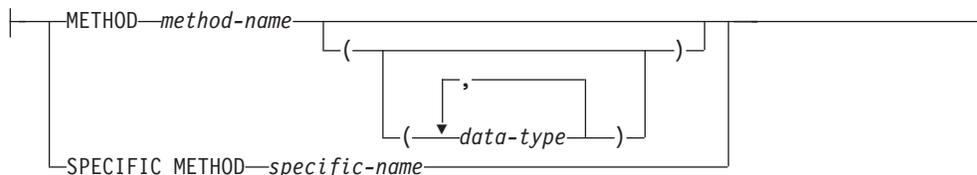
fenced であるようにメソッドを変更するには、さらに別の権限や特権は必要ありません。

構文



method-identifier:

ALTER TYPE (構造化)



method-options:



説明

type-name

変更する構造化タイプを識別します。指定するタイプは、カタログに定義されている既存のタイプであり (SQLSTATE 42704)、かつ構造化タイプでなければなりません (SQLSTATE 428DP)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

ADD ATTRIBUTE

既存の構造化タイプの最後の属性の後に、属性を追加します。

attribute-definition

構造化タイプの属性を定義します。

attribute-name

属性の名前を指定します。この名前は、この構造化タイプの他のどの属性 (継承された属性も含む) とも同じであってはならず、この構造化タイプのどのサブタイプとも同じであってはなりません (SQLSTATE 42711)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*attribute-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH、および比較演算子です。

data-type 1

属性のデータ・タイプを指定します。これは、『CREATE TABLE』でリストされているデータ・タイプの 1 つで、XML 以外のものです (SQLSTATE 42601)。このデータ・タイプには既存のデータ・タイプを指定する必要があります (SQLSTATE 42704)。*data-type* がスキーマ名なしで指定される場合、SQL パス上でスキーマを検索することにより、タイプは解決されます。『CREATE TABLE』に種々のデータ・タイプの説明が記載されています。属性データ・タイプが参照タイプであ

る場合、参照するターゲット・タイプはこのステートメントに既に存在する構造化タイプでなければなりません (SQLSTATE 42704)。

実行時にタイプのインスタンスが直接または間接的に同じタイプやそのサブタイプのインスタンスを含むタイプ定義を避けるために、タイプの定義において、属性タイプのいずれかが直接または間接的にそれ自身を使用するように定義してはならないという制限があります (SQLSTATE 428EP)。

lob-options

LOB タイプと関連したオプション (あるいは LOB に基づく特殊タイプ) を指定します。 *lob-options* の詳細については、『CREATE TABLE』を参照してください。

DROP ATTRIBUTE

既存の構造化タイプの属性をドロップします。

attribute-name

属性の名前。属性は、そのタイプの属性として存在していなければなりません (SQLSTATE 42703)。

RESTRICT

type-name が既存の表、ビュー、列、列のタイプ内でネストされた属性、または索引拡張のタイプとして使用される場合に、どの属性もドロップできないという規則を課します。

ADD METHOD *method-specification*

メソッド指定を、*type-name* で識別されるタイプに追加します。別個の CREATE METHOD ステートメントを使用してメソッドに本体を与えるまでは、このメソッドを使用することはできません。 *method-specification* についての詳細は、『CREATE TYPE (構造化)』を参照してください。

ALTER *method-identifier*

変更されるメソッドのインスタンスを一意的に指定します。指定されたメソッドには、既存のメソッド本体があるかもしれませんし、ないかもしれません。LANGUAGE SQL として宣言されたメソッドは変更できません (SQLSTATE 42917)。

method-identifier

METHOD *method-name*

特定のメソッドを指定します。 *type-name* というサブジェクト・タイプの *method-name* という名前のメソッド・インスタンスが 1 つだけ存在している場合にのみ有効です。このように指定されたメソッドには、任意の数のパラメーターを定義できます。タイプに、指定された名前のメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。タイプに、そのメソッドのインスタンスが複数存在する場合も、エラーが戻されます (SQLSTATE 42725)。

METHOD *method-name* (*data-type*,...)

メソッドを一意に指定するメソッド・シグニチャーを指定します。メソッド解決のアルゴリズムは使用されません。

method-name

type-name タイプのメソッドの名前を指定します。

ALTER TYPE (構造化)

(*data-type*,...)

値は、CREATE TYPE ステートメント上で (対応する位置に) 指定されたデータ・タイプと一致していなければなりません。データ・タイプの数、およびデータ・タイプを論理的に連結した値が、特定のメソッド・インスタンスを識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索してタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE TYPE ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つメソッドのタイプがない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC METHOD *specific-name*

メソッドの作成時に指定された名前か、デフォルト値として与えられた名前を使用して、特定のメソッドを識別します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。*specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定のメソッド・インスタンスを指定していなければなりません。そうでない場合、エラー (SQLSTATE 42704) になります。

method-options

メソッドに対して変更されるオプションを指定します。

FENCED または NOT FENCED

メソッドをデータベース・マネージャーのオペレーティング環境のプロセスまたはアドレス・スペースで実行しても安全か (NOT FENCED)、そうでないか (FENCED) を指定します。多くのメソッドは、FENCED または NOT FENCED のどちらかで実行するように選択することができます。

メソッドが FENCED として登録されると、データベース・マネージャーは、その内部リソース (データ・バッファーなど) を fenced して、そのメ

ソッドからアクセスされないようにします。一般に、FENCED として実行されるメソッドは、NOT FENCED として実行されるものと同じようには実行されません。

注意:

適切にコード化、検討、および検査されていないメソッドに NOT FENCED を使用すると、DB2 の保全性に危険を招く場合があります。DB2 では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられています。NOT FENCED メソッドが使用される場合には、完全な整合性を確保できません。

NOT THREADSAFE を宣言したメソッドは、NOT FENCED には変更できません (SQLSTATE 42613)。

メソッドが定義済みの AS LOCATOR の任意のパラメーターを有していて、NO SQL オプションが指定されていた場合には、このメソッドは FENCED には変更できません (SQLSTATE 42613)。

このオプションは LANGUAGE OLE メソッドを変更できません (SQLSTATE 42849)。

THREADSAFE または NOT THREADSAFE

メソッドを他のルーチンと同じプロセスで実行しても安全か (THREADSAFE)、そうでないか (NOT THREADSAFE) を指定します。

メソッドが OLE 以外の LANGUAGE で定義される場合:

- メソッドが THREADSAFE に定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスでメソッドを呼び出すことができます。一般に、スレッド・セーフにするには、メソッドはどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。FENCED および NOT FENCED メソッドの両方が THREADSAFE になることが可能です。メソッドが LANGUAGE OLE とともに定義される場合には、THREADSAFE は指定されません (SQLSTATE 42613)。
- メソッドが NOT THREADSAFE として定義される場合には、データベース・マネージャーは他のルーチンと同じプロセスにメソッドを決して呼び出しません。fenced されたメソッドだけが、NOT THREADSAFE になり得ます (SQLSTATE 42613)。

DROP *method-identifier*

一意的にドロップするメソッドのインスタンスを指定します。指定されたメソッドには、既存のメソッド本体があってはなりません (SQLSTATE 428ER)。

DROP METHOD ステートメントを使用してメソッド本体をドロップしてから、ALTER TYPE DROP METHOD を使用してください。CREATE TYPE ステートメントで暗黙的に生成されたメソッド (mutators および observers など) は、ドロップできません (SQLSTATE 42917)。

RESTRICT

指定されたメソッドが、既存のメソッド本体を所持できないように制限を受けることを指示します。DROP METHOD ステートメントを使用してメソッド本体をドロップしてから、ALTER TYPE DROP METHOD を使用してください。

規則

- 以下の場合には、タイプ *type-name* で属性を追加またはドロップすることは許可されていません (SQLSTATE 55043)。
 - あるタイプまたはそのタイプのサブタイプの 1 つが既存の表のタイプである場合。
 - タイプが直接または間接的に *type-name* を使用する表の列が存在する場合。直接使用 および間接使用 という用語は、『構造化タイプ』で定義されています。
 - 索引拡張で、このタイプまたはサブタイプのいずれかが使用される場合。
- 属性の追加によるタイプの変更で、このタイプまたはサブタイプの属性の合計が 4082 を超えてはなりません (SQLSTATE 54050)。
- ADD ATTRIBUTE オプション:
 - ADD ATTRIBUTE は、新しい属性に *observer* および *mutator* メソッドを生成します。これらのメソッドは、構造化タイプが作成される際に生成されるタイプに類似しています (『CREATE TYPE (構造化)』を参照)。これらのメソッドが任意の既存のメソッドまたは関数と競合したり、これらをオーバーライドしたりする場合には、ALTER TYPE ステートメントは失敗します (SQLSTATE 42745)。
 - ユーザーがタイプ (またはこの任意のサブタイプ) の *INLINE LENGTH* を明示的に 292 よりも小さい値に指定した場合で、追加したこの属性が原因で、指定されたインライン長が、変更されたタイプのコンストラクター関数の結果のサイズよりも小さくなる場合 (32 バイト + 属性ごとに 10 バイト)、エラーになります (SQLSTATE 42611)。
- DROP ATTRIBUTE オプション:
 - 既存のスーパータイプから継承される属性は、ドロップできません (SQLSTATE 428DJ)。
 - DROP ATTRIBUTE は、ドロップされた属性の *mutator* および *observer* メソッドをドロップし、これらのドロップされたメソッドの従属性を検査します。
- DROP METHOD オプション:
 - 他のメソッドによってオーバーライドされた元のメソッドは、ドロップできません (SQLSTATE -2)。

注

- SYSIBM、SYSFUN、または SYSPROC スキーマのメソッドは変更できません (SQLSTATE 42832)。
- 属性を追加またはドロップしてタイプを変更すると、そのタイプまたはそのタイプのサブタイプをパラメーターまたは結果として使用する関数またはメソッドに依存するすべてのパッケージは無効になります。
- 構造化タイプから属性を追加またはドロップする場合:
 - タイプが作成されたときにシステムによりタイプの *INLINE LENGTH* が計算された場合、*INLINE LENGTH* 値は自動的に、変更されたタイプについて修正され、そのサブタイプもすべて変更に対応するように修正されます。すべての構造化タイプについても、*INLINE LENGTH* 値は自動的に (再帰的に) 変更されます。この場合、*INLINE LENGTH* はシステムにより計算され、変更された *INLINE LENGTH* を持つタイプの属性がタイプに含まれています。

- 属性の追加またはドロップにより影響を受けるタイプの `INLINE LENGTH` がユーザーにより明示的に指定されたものである場合、この特定のタイプの `INLINE LENGTH` は変更されません。明示的に指定されたインライン長については、十分に注意してください。後でタイプに属性が追加されることがある場合、列定義でこのタイプまたはサブタイプの 1 つを使用するために、インスタンス化されたオブジェクトの長さの増加の可能性に対応できるように、インライン長を十分に大きくしておかなければなりません。
 - 新しい属性がアプリケーション・プログラムから見えるようにするには、データ・タイプの新しい構造に適合するように、既存のトランスフォーム機能を修正しなければなりません。
- パーティション・データベース環境では、外部ユーザー定義関数またはメソッドでの SQL の使用はサポートされていません (SQLSTATE 42997)。

• 特権

`EXECUTE` 特権は、メソッド本体が `CREATE METHOD` ステートメントを使用して定義されるまでは、`ALTER TYPE` ステートメントで明示的に指定されたすべてのメソッドには与えられません。ユーザー定義タイプの所有者には、`ALTER TYPE` ステートメントを使用して、メソッド指定をドロップする特権があります。

例

例 1: `ALTER TYPE` ステートメントを使用して、相互参照するタイプおよび表の循環を許可します。EMPLOYEE および DEPARTMENT という名前の表を手動で参照しているとします。

次のシーケンスで、タイプおよび表の作成ができます。

```
CREATE TYPE DEPT ...
CREATE TYPE EMP ... (タイプ REF(DEPT) の DEPTREF という属性を含む)
ALTER TYPE DEPT ADD ATTRIBUTE MANAGER REF(EMP)
CREATE TABLE DEPARTMENT OF DEPT ...
CREATE TABLE EMPLOYEE OF EMP (DEPTREF WITH OPTIONS SCOPE DEPARTMENT)
ALTER TABLE DEPARTMENT ALTER COLUMN MANAGER ADD SCOPE EMPLOYEE
```

次のシーケンスで、タイプおよび表のドロップができます。

```
DROP TABLE EMPLOYEE (DEPARTMENT の MANAGER 列が有効範囲解除となる)
DROP TABLE DEPARTMENT
ALTER TYPE DEPT DROP ATTRIBUTE MANAGER
DROP TYPE EMP
DROP TYPE DEPT
```

例 2: `ALTER TYPE` ステートメントを使用して、サブタイプを参照する属性を持つタイプを作成します。

```
CREATE TYPE EMP ...
CREATE TYPE MGR UNDER EMP ...
ALTER TYPE EMP ADD ATTRIBUTE MANAGER REF(MGR)
```

例 3: `ALTER TYPE` ステートメントを使用して、属性を追加します。以下のステートメントは、EMP タイプに `SPECIAL` 属性を追加します。元の `CREATE TYPE` ステートメントでインライン長が指定されなかったため、DB2 は、13 (新しい属性の 10 + 属性長 + 非 LOB 属性の 2 バイト) を追加してインライン長を計算しなおします。

ALTER TYPE (構造化)

```
ALTER TYPE EMP ...  
  ADD ATTRIBUTE SPECIAL CHAR(1)
```

例 4: ALTER TYPE ステートメントを使用して、タイプに関連するメソッドを追加します。以下のステートメントは、BONUS というメソッドを追加します。

```
ALTER TYPE EMP ...  
  ADD METHOD BONUS (RATE DOUBLE)  
  RETURNS INTEGER  
  LANGUAGE SQL  
  CONTAINS SQL  
  NO EXTERNAL ACTION  
  DETERMINISTIC
```

CREATE METHOD ステートメントを発行してメソッド本体を作成するまでは、BONUS メソッドは使用できないことに注意してください。タイプ EMP に SALARY という属性が含まれているとすると、メソッド本体の定義例は以下のようになります。

```
CREATE METHOD BONUS(RATE DOUBLE) FOR EMP  
  RETURN CAST(SELF.SALARY * RATE AS INTEGER)
```

ALTER USER MAPPING

ALTER USER MAPPING ステートメントは、指定したフェデレーテッド・サーバーの許可 ID について、データ・ソースで使用する許可 ID またはパスワードを変更するときに使います。

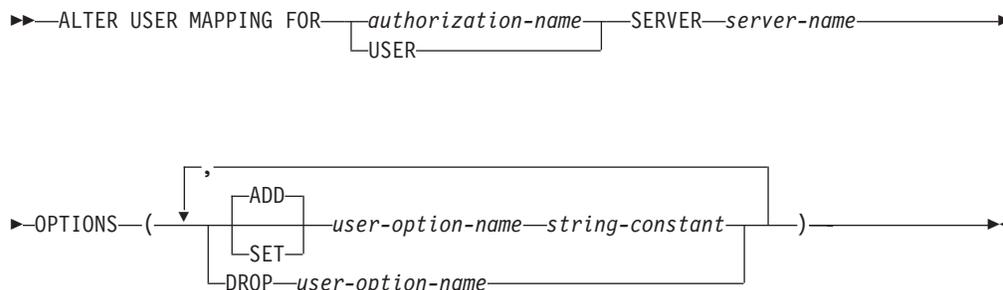
呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID がデータ・ソースへマップされる許可名とは異なる場合、そのステートメントの許可 ID が持つ特権には SYSADM または DBADM 権限が含まれている必要があります。許可 ID と許可名が一致する場合には、必要とされる特権または権限はありません。

構文



説明

authorization-name

ユーザーまたはアプリケーションがフェデレーテッド・データベースへ接続するときの、許可名を指定します。

USER

特殊レジスター **USER** の値。 **USER** を指定すると、ALTER USER MAPPING ステートメントの許可 ID は、 **REMOTE_AUTHID** ユーザー・オプションで指定したデータ・ソースの許可 ID にマップされます。

SERVER *server-name*

ローカル許可 ID へマップするリモート許可 ID を使ってアクセスできるデータ・ソースを指定します。このローカル許可 ID は、 *authorization-name* で示されるか、または **USER** によって参照されるものです。

OPTIONS

変更するマッピングに関して、使用可能にする、リセットする、またはドロップするユーザー・オプションを指定します。

ADD

ユーザー・オプションを使用可能にします。

SET

ユーザー・オプションの設定を変更します。

user-option-name

使用可能にする、あるいはリセットするユーザー・オプションを指定します。

string-constant

user-option-name の設定を、文字ストリング定数として指定します。

DROP *user-option-name*

ユーザー・オプションをドロップします。

注

- ユーザー・オプションは、同じ ALTER USER ステートメントに複数回指定することはできません (SQLSTATE 42853)。ユーザー・オプションを使用可能にする、リセットする、あるいはドロップする場合、使用中の他のユーザー・オプションには影響はありません。
- 所定の作業単位 (UOW) 内の ALTER USER MAPPING ステートメントは、UOW に以下のいずれかがすでに含まれている場合には処理できません (SQLSTATE 55007)。
 - マッピングに含めるソース・データの表またはビューのニックネームを参照する SELECT ステートメント。
 - マッピングに含めるソース・データの表またはビューのニックネーム上のオープン・カーソル。
 - マッピングに含めるソース・データの表またはビューのニックネームに対して発行された、INSERT、DELETE、または UPDATE ステートメント。

ALTER USER MAPPING

例

例 1: Jim はローカル・データベースを使い、ORACLE1 という Oracle データ・ソースに接続します。そして許可 ID KLEEWEIN を使ってローカル・データベースにアクセスします。KLEEWEIN は、CORONA (ORACLE1 へアクセスするときの許可 ID) へマップします。Jim は新しい ID である JIMK を使用して ORACLE1 へのアクセスを開始します。ここで、KLEEWEIN は JIMK へマップすることが必要になります。

```
ALTER USER MAPPING FOR KLEEWEIN
  SERVER ORACLE1
  OPTIONS ( SET REMOTE_AUTHID 'JIMK' )
```

例 2: Mary はフェデレーテッド・データベースを使用して、DORADO という DB2 for z/OS データ・ソースへ接続します。そしてある許可 ID を使って DB2 にアクセスし、別の許可 ID で DORADO にアクセスします。これら 2 つの ID のマッピングは作成してあります。どちらの ID でも同じパスワードを使っていますが、ここで、DORADO の ID 用に固有のパスワード ZNYQ を使うことにしました。その結果、使用しているフェデレーテッド・データベースのパスワードを ZNYQ へマップしなければならなくなります。

```
ALTER USER MAPPING FOR MARY
  SERVER DORADO
  OPTIONS ( ADD REMOTE_PASSWORD 'ZNYQ' )
```

ALTER VIEW

ALTER VIEW ステートメントは以下の方法で既存のビューを変更します。

- 参照タイプ列を変更して有効範囲を追加する。
- 照会の最適化でのビューの使用を可能化あるいは使用不可化する。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

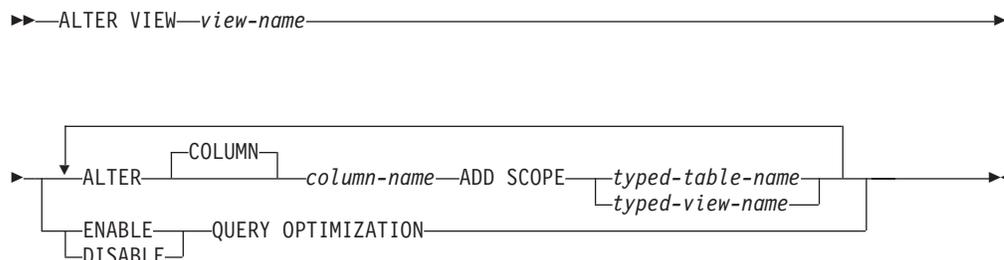
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- ビューのスキーマに対する ALTERIN 特権
- 変更するビューの所有者
- 変更するビューに対する CONTROL 特権
- SYSADM または DBADM 権限

照会の最適化へのビューの使用を可能化または不能化するには、ステートメントの許可 ID の保持する特権に、各表、またはビュー全選択の FROM 節で参照されるビューの基礎表に対し、少なくとも以下のいずれか 1 つが含まれている必要があります。

- その表に対する ALTER 特権
- 表のスキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

構文



説明

view-name

変更するビューを指定します。ビューはカタログに記述されている必要があります。

ALTER COLUMN *column-name*

変更される列の名前を指定します。 *column-name* は、ビューの既存の列を指定するものでなければなりません (SQLSTATE 42703)。名前は非修飾でなければなりません。

ADD SCOPE

有効範囲が未定義である既存の参照タイプ列に、有効範囲を追加します (SQLSTATE 428DK)。列をスーパービューから継承することはできません (SQLSTATE 428DJ)。

typed-table-name

型付き表の名前を指定します。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-table-name* のタイプを表します (SQLSTATE 428DM)。値が *typed-table-name* の既存行を実際に参照していることを確認するための、 *column-name* の既存値の検査は行われません。

typed-view-name

型付きビューの名前を指定します。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-view-name* のタイプを表します (SQLSTATE 428DM)。値が *typed-view-name* の既存行を実際に参照していることを確認するための、 *column-name* の既存値の検査は行われません。

ENABLE QUERY OPTIMIZATION または **DISABLE QUERY OPTIMIZATION**

ビューおよびあらゆる関連の統計を照会の最適化の改善に使用するかどうかを指定します。 **DISABLE QUERY OPTIMIZATION** は、ビューの作成時のデフォルト値です。

ENABLE QUERY OPTIMIZATION

ビューに、そのビューに関わる照会、またはそのビューの全選択に似た副照会を含む照会の最適化の向上に使用できる統計が含まれることを指定します。

ALTER VIEW

DISABLE QUERY OPTIMIZATION

ビューおよびあらゆる関連の統計を照会の最適化の改善に使用しないことを指定します。

規則

- 以下の場合には、照会の最適化へのビューの使用可能化はできなくなります。
 - ビューが直接、間接に、マテリアライズ照会表 (MQT) を参照している。
MQT または統計ビューが統計ビューを参照する可能性があることに注意してください。
 - 型付きビューになっている。

注

- 照会の最適化への使用を考える場合、ビューに関しては以下の点に注意してください。
 - 集約操作や特殊操作を含むことはできない。
 - UNION、EXCEPT、INTERSECT といった操作を含むことはできない。
 - スカラー集約 (OLAP) 関数を含むことはできない。
- 照会の最適化への使用が不可になるようビューが変更された場合は、そのビューを照会最適化に使用していたキャッシュ照会プランが無効になります。照会の最適化への使用が可能になるようビューが変更された場合は、直接、あるいは他のビューを通じて間接的に、新たに使用可能にされたビュー参照と同じ表を参照しているキャッシュ照会プランが無効になります。これらのキャッシュ照会プランの無効化により、ビューの変更された照会最適化プロパティを考慮に入れた再有効化が暗黙的に行われます。

ビューの照会最適化プロパティは、組み込み静的 SQL ステートメントには影響しません。

ALTER WORK ACTION SET

ALTER WORK ACTION SET ステートメントは、作業アクション・セットの作業アクションを追加、変更、またはドロップすることによって、その作業アクション・セットに変更を加えます。

呼び出し

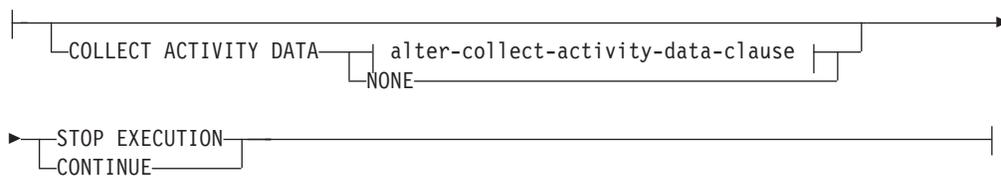
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

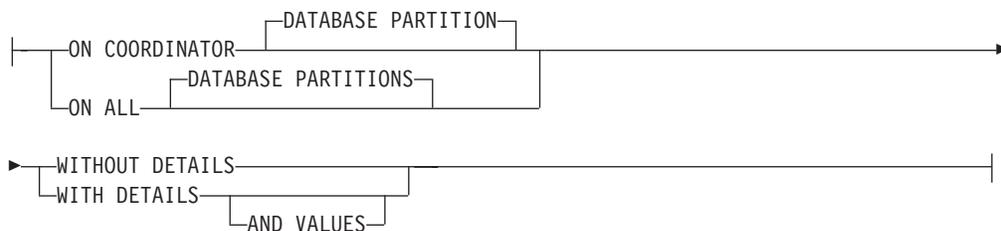
このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

ALTER WORK ACTION SET

threshold-exceeded-actions:



alter-collect-activity-data-clause:

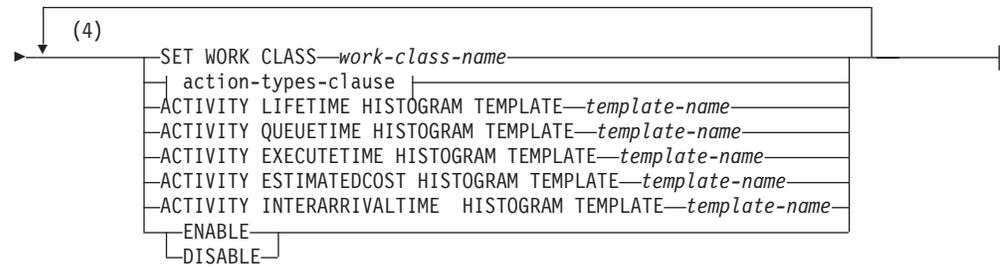


histogram-template-clause:



work-action-alteration:



**注:**

- 1 ADD、ALTER、および DROP 節は、指定されている順に処理されます。
- 2 ENABLE または DISABLE 節は、1 つのステートメント内で 1 度だけ指定することができます。
- 3 1 つの作業クラスに一度に適用できる同一しきい値タイプの作業アクションは、1 つだけです。しきい値作業アクションを変更するときに、しきい値述部を変更することはできません。
- 4 同じ節を複数回指定することはできません。

説明*work-action-set-name*

変更する作業アクション・セットを識別します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *work-action-set-name* には、現行のサーバー上の既存の作業アクション・セットを指定する必要があります (SQLSTATE 42704)。

ADD

作業アクション・セットに作業アクションを追加します。

WORK ACTION *work-action-name*

作業アクションの名前を指定します。 *work-action-name* は、現行のサーバーのこの作業アクション・セットの下に既存の作業アクションを識別するものであってはなりません (SQLSTATE 42710)。 *work-action-name* の先頭に 'SYS' を使用することはできません (SQLSTATE 42939)。

ON WORK CLASS *work-class-name*

この作業アクションが適用されるデータベース・アクティビティを識別する作業クラスを指定します。 *work-class-name* は、現行のサーバーの *work-class-set-name* に存在するものでなければなりません (SQLSTATE 42704)。

MAP ACTIVITY

アクティビティのマッピングの作業アクションを指定します。このアクションは、この作業アクション・セットが定義されているオブジェクトがサービス・スーパークラスである場合にのみ指定できます (SQLSTATE 5U034)。

WITH NESTED または WITHOUT NESTED

このアクティビティの下にネストされているアクティビティをサービス・サブクラスにマップするかどうかを指定します。デフォルトは WITH NESTED です。

WITH NESTED

作業クラスに分類されるすべてのデータベース・アクティビティーのうちネ스팅・レベルがゼロのもの、およびこのアクティビティーの下でネストされているすべてのデータベース・アクティビティーは、サービス・サブクラスにマップされます。つまり、ネ스팅・レベルがゼロより大きいアクティビティーは、ネ스팅・レベルがゼロのアクティビティーと同じサービス・クラスの下で実行されます。

WITHOUT NESTED

作業クラスの下に分類されているデータベース・アクティビティーのうち、ネ스팅・レベルがゼロのものだけがサービス・サブクラスにマップされます。このアクティビティーの下にネストされているデータベース・アクティビティーは、そのアクティビティー・タイプに従って処理されます。

TO *service-subclass-name*

アクティビティーのマップ先となるサービス・サブクラスを指定します。*service-subclass-name* は、現行サーバーの *service-superclass-name* に既に存在するものでなければなりません (SQLSTATE 42704)。*service-subclass-name* にデフォルトのサービス・サブクラス SYSDEFAULTSUBCLASS を指定することはできません (SQLSTATE 5U018)。

WHEN

この作業アクションが定義されている作業クラスに関連するデータベース・アクティビティーに適用するしきい値を指定します。しきい値は、この作業アクション・セットが定義されているデータベース・マネージャー・オブジェクトがデータベースである場合にのみ指定できます (SQLSTATE 5U034)。これらのしきい値は、データベース・マネージャーによって開始された内部のデータベース・アクティビティー、または管理 SQL ルーチンによって生成されたデータベース・アクティビティーには適用されません。

threshold-types-clause

有効なしきい値タイプについての説明は、『CREATE THRESHOLD』ステートメントの項を参照してください。

threshold-exceeded-actions

しきい値を超過した場合の有効なアクションについて説明は、『CREATE THRESHOLD』ステートメントを参照してください。

PREVENT EXECUTION

この作業アクションが定義されている作業クラスに関連するデータベース・アクティビティーの実行を許可しないことを指定します (SQLSTATE 5U033)。

COUNT ACTIVITY

作業クラスに関連するすべてのデータベース・アクティビティーを実行し、1 つのアクティビティーが実行されるたびに作業クラスのカウンターを増加させることを指定します。

COLLECT ACTIVITY DATA

この作業アクションが定義されている作業クラスに関連する各アクティビテ

ィーについてのデータを、アクティビティ完了時に、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

アクティビティのコーディネーターのデータベース・パーティションでのみアクティビティ・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティが処理されるすべてのデータベース・パーティションでアクティビティ・データを収集することを指定します。予測しきい値の場合、しきい値を超過した場合の CONTINUE アクションも指定した場合にのみ、アクティビティ情報がすべてのパーティションで収集されます。反応しきい値の場合、ON ALL DATABASE PARTITIONS 節を指定しても効果はなく、アクティビティ情報は常にコーディネーター・パーティションでのみ収集されます。予測しきい値および反応しきい値の両方について、アクティビティの詳細または値は、コーディネーター・パーティションでのみ収集されます。

WITHOUT DETAILS

この作業アクションが定義されている作業クラスに関連する各アクティビティについてのデータを、アクティビティ完了時に、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境の情報はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティについて、これらの情報をアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティに入力データ値が含まれている場合、それをアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

この作業アクションが定義されている作業クラスに関連付けられている各アクティビティについてはアクティビティ・データを収集しないことを指定します。

COLLECT AGGREGATE ACTIVITY DATA

この作業アクションが定義されている作業クラスに関連するアクティビティについて、集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。デフォルトは COLLECT

AGGREGATE ACTIVITY DATA BASE です。この節は、データベースに適用される作業アクション・セットで定義されている作業アクションには指定できません。

BASE

この作業アクションが定義されている作業クラスに関連するアクティビティについて、基礎集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。基礎集約アクティビティ・データには以下のものが含まれます。

- アクティビティ・コストの最高水準点の見積もり
- 戻り行数の最高水準点
- TEMPORARY 表スペース使用量の最高水準点
- アクティビティ存続時間のヒストグラム
- アクティビティ・キュー時間のヒストグラム
- アクティビティ実行時間のヒストグラム

EXTENDED

この作業アクションが定義されている作業クラスに関連するアクティビティについて、すべての集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。これには、すべての基礎集約アクティビティ・データに加えて、以下のものが含まれます。

- アクティビティ・データ操作言語 (DML) の見積コスト・ヒストグラム
- アクティビティ DML の到着間隔時間のヒストグラム

ENABLE または DISABLE

データベース・アクティビティをサブミットする際にこの作業アクションを考慮するかどうかを指定します。デフォルトは ENABLE です。

ENABLE

この作業アクションが有効であり、データベース・アクティビティのサブミット時に考慮することを指定します。

DISABLE

作業アクションが無効であり、データベース・アクティビティのサブミット時に考慮の対象にならないことを指定します。

histogram-template-clause

この作業アクションの割り当て先の作業クラスに関連するアクティビティの集約アクティビティ・データを収集する際に使用するヒストグラム・テンプレートを指定します。作業クラスの集約アクティビティ・データが収集されるのは、作業アクション・タイプが COLLECT AGGREGATE ACTIVITY DATA となっている場合だけです。

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で実行された (この作業アクションの割り当て先作業クラスに関連する) DB2 アクティビティの所要時間 (マイクロ秒) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、キューに入っていた

時間と実行時間の両方が含まれます。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY QUEUE TIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で (この作業アクションが割り当てられている作業クラスに関連する) DB2 アクティビティがキューに入れられていた時間の長さ (マイクロ秒単位) に関する、統計データの収集に使用されるヒストグラムを記述する、テンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY EXECUTE TIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で (この作業アクションが割り当てられている作業クラスに関連する) DB2 アクティビティが実行されている時間の長さ (マイクロ秒単位) に関する、統計データの収集に使用されるヒストグラムを記述する、テンプレートを指定します。この時間には、アクティビティがキューに入っていた時間は含まれません。このヒストグラムでは、アクティビティが実行される各データベース・パーティションごとにアクティビティの実行時間が収集されます。アクティビティのコーディネーターのデータベース・パーティションの場合、これはエンドツーエンドの実行時間です (つまり、存続時間からキューに入っていた時間を差し引いた時間)。コーディネーター・データベース・パーティション以外の場合、これはこれらのパーティションがアクティビティの代わりに費やした時間です。特定のアクティビティの実行中、DB2 は、リモート・データベース・パーティションに対して作業を複数回提示することがあります。リモート・パーティションはそのたびにアクティビティのオカレンスの実行時間を収集します。そのため、実行時間のヒストグラムの数は、データベース・パーティションで実行された固有アクティビティの実数の数とは異なる場合があります。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY ESTIMATED COST HISTOGRAM TEMPLATE *template-name*

この作業アクションの割り当て先作業クラスに関連する DML アクティビティの見積コスト (timeron 単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節とその EXTENDED オプションが指定されている場合にのみ収集されます。

ACTIVITY INTERARRIVAL TIME HISTOGRAM TEMPLATE

template-name

この作業アクションの割り当て先作業クラスに関連するすべてのアクティビティについて、1 つの DML アクティビティの到着から次の

ALTER WORK ACTION SET

DML アクティビティーの到着までの間の時間の長さ (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE ACTIVITY DATA` 節とその `EXTENDED` オプションが指定されている場合にのみ収集されます。

ALTER

作業アクションの定義を変更します。この作業アクションが適用される作業クラス、およびその作業クラスに属するデータベース・アクティビティーに適用されるアクションを変更できます。

WORK ACTION *work-action-name*

作業アクションを識別します。 *work-action-name* には、現行のサーバーに存在する作業アクションを、この作業アクション・セットの下に指定する必要があります (SQLSTATE 42704)。

SET WORK CLASS *work-class-name*

この作業アクションが適用されるデータベース・アクティビティーを識別する作業クラスを指定します。 *work-class-name* は、現行のサーバーの *work-class-set-name* に存在するものでなければなりません (SQLSTATE 42704)。

MAP ACTIVITY

アクティビティーのマッピングの作業アクションを指定します。このアクションは、この作業アクション・セットが定義されているオブジェクトがサービス・スーパークラスである場合にのみ指定できます (SQLSTATE 5U034)。

WITH NESTED または WITHOUT NESTED

このアクティビティーの下にネストされているアクティビティーをサービス・サブクラスにマップするかどうかを指定します。デフォルトは `WITH NESTED` です。

WITH NESTED

作業クラスに分類されているネスティング・レベルがゼロのすべてのデータベース・アクティビティー、およびこのアクティビティーの下にネストされているすべてのデータベース・アクティビティーが、サービス・サブクラスにマップされます。

WITHOUT NESTED

作業クラスの下に分類されているデータベース・アクティビティーのうち、ネスティング・レベルがゼロのものだけがサービス・サブクラスにマップされます。このアクティビティーの下にネストされているデータベース・アクティビティーは、そのアクティビティー・タイプに従って処理されます。

TO *service-subclass-name*

アクティビティーのマッピング先となるサービス・サブクラスを指定します。 *service-subclass-name* は、現行サーバーの *service-superclass-name* に既に存在するものでなければなりません (SQLSTATE 42704)。

service-subclass-name にデフォルトのサービス・サブクラス SYSDEFAULTSUBCLASS を指定することはできません (SQLSTATE 5U018)。

WHEN

この作業アクションが定義されている作業クラスに関連するデータベース・アクティビティーに適用するしきい値を指定します。しきい値は、この作業アクション・セットが定義されているデータベース・マネージャー・オブジェクトがデータベースである場合のみ指定できます (SQLSTATE 5U034)。これらのしきい値は、データベース・マネージャーによって開始された内部のデータベース・アクティビティー、または管理 SQL ルーチンによって生成されたデータベース・アクティビティーには適用されません。

threshold-types-clause

有効なしきい値タイプについての説明は、『CREATE THRESHOLD』ステートメントの項を参照してください。

threshold-exceeded-actions

しきい値を超過した場合の有効なアクションについて説明は、『CREATE THRESHOLD』ステートメントを参照してください。

PREVENT EXECUTION

この作業アクションが定義されている作業クラスに関連するデータベース・アクティビティーの実行を許可しないことを指定します (SQLSTATE 5U033)。

COUNT ACTIVITY

作業クラスに関連するすべてのデータベース・アクティビティーを実行し、1 つのアクティビティーが実行されるたびに作業クラスのカウンターを増加させることを指定します。

COLLECT ACTIVITY DATA

この作業アクションが定義されている作業クラスに関連する各アクティビティーについてのデータを、アクティビティー完了時に、アクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

*alter-collect-activity-data-clause***ON COORDINATOR DATABASE PARTITION**

アクティビティーのコーディネーターのデータベース・パーティションでのみアクティビティー・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティーが処理されるすべてのデータベース・パーティションでアクティビティー・データを収集することを指定します。予測しきい値の場合、しきい値を超過した場合の CONTINUE アクションも指定した場合にのみ、アクティビティー情報がすべてのパーティションで収集されます。反応しきい値の場合、ON ALL DATABASE PARTITIONS 節を指定しても効果はなく、アクティビティー情報は常にコーディネーター・パーティションでのみ収集さ

ALTER WORK ACTION SET

れます。予測しきい値および反応しきい値の両方について、アクティビティの詳細または値は、コーディネーター・パーティションでのみ収集されます。

WITHOUT DETAILS

この作業アクションが定義されている作業クラスに関連する各アクティビティについてのデータを、アクティビティ完了時に、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境の情報はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティについて、これらの情報をアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティに入力データ値が含まれている場合、それをアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

この作業アクションが定義されている作業クラスに関連付けられている各アクティビティについてはアクティビティ・データを収集しないことを指定します。

COLLECT AGGREGATE ACTIVITY DATA

この作業アクションが定義されている作業クラスに関連するアクティビティについて、集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。デフォルトは **COLLECT AGGREGATE ACTIVITY DATA BASE** です。この節は、データベースに適用される作業アクション・セットで定義されている作業アクションには指定できません。

BASE

この作業アクションが定義されている作業クラスに関連するアクティビティについて、基礎集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。基礎集約アクティビティ・データには以下のものが含まれます。

- アクティビティ・コストの最高水準点の見積もり
- 戻り行数の最高水準点
- **TEMPORARY** 表スペース使用量の最高水準点
- アクティビティ存続時間のヒストグラム
- アクティビティ・キュー時間のヒストグラム
- アクティビティ実行時間のヒストグラム

EXTENDED

この作業アクションが定義されている作業クラスに関連するアクティビ

ティーについて、すべての集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。これには、すべての基礎集約アクティビティー・データに加えて、以下のものが含まれます。

- アクティビティー DML の見積コストのヒストグラム
- アクティビティー DML の到着間隔時間のヒストグラム

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で実行された (この作業アクションの割り当て先作業クラスに関連する) DB2 アクティビティーの所要時間 (マイクロ秒) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、キューに入っていた時間と実行時間の両方が含まれます。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で (この作業アクションが割り当てられている作業クラスに関連する) DB2 アクティビティーがキューに入れられていた時間の長さ (マイクロ秒単位) に関する、統計データの収集に使用されるヒストグラムを記述する、テンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で (この作業アクションが割り当てられている作業クラスに関連する) DB2 アクティビティーが実行されている時間の長さ (マイクロ秒単位) に関する、統計データの収集に使用されるヒストグラムを記述する、テンプレートを指定します。この時間には、アクティビティーがキューに入っていた時間は含まれません。このヒストグラムでは、アクティビティーが実行される各データベース・パーティションごとにアクティビティーの実行時間が収集されます。アクティビティーのコーディネーターのデータベース・パーティションの場合、これはエンドツーエンドの実行時間です (つまり、存続時間からキューに入っていた時間を差し引いた時間)。コーディネーター・データベース・パーティション以外の場合、これはこれらのパーティションがアクティビティーの代わりに費やした時間です。特定のアクティビティーの実行中、DB2 は、リモート・データベース・パーティションに対して作業を複数回提示することがあります。リモート・パーティションはそのたびにアクティビティーのオカレンスの実行時間を収集します。そのため、実行時間のヒストグラムの数は、データベース・パーティションで実行された固有アクティビティーの実際の数とは異なる場合があります。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*

この作業アクションの割り当て先作業クラスに関連するデータ操作言語 (DML) アクティビティーの見積コスト (timeron 単位) に関する統計データ

ALTER WORK ACTION SET

を収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE ACTIVITY DATA` 節とその `EXTENDED` オプションが指定されている場合にのみ収集されます。

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*

この作業アクションの割り当て先作業クラスに関連するすべてのアクティビティについて、1 つの `DML` アクティビティの到着から次の `DML` アクティビティの到着までの間の時間の長さ (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE ACTIVITY DATA` 節とその `EXTENDED` オプションが指定されている場合にのみ収集されます。

ENABLE または **DISABLE**

データベース・アクティビティをサブミットする際にこの作業アクションを考慮するかどうかを指定します。

ENABLE

この作業アクションが有効であり、データベース・アクティビティのサブミット時に考慮することを指定します。

DISABLE

作業アクションが無効であり、データベース・アクティビティのサブミット時に考慮の対象にならないことを指定します。

DROP *work-action-name*

作業アクション・セットから作業アクションをドロップします。

work-action-name には、現行のサーバーに存在する作業アクションを、この作業アクション・セットの下に指定する必要があります (`SQLSTATE 42704`)。

ENABLE または **DISABLE**

データベース・アクティビティをサブミットする際にこの作業アクション・セットを考慮するかどうかを指定します。

ENABLE

この作業アクション・セットが有効であり、データベース・アクティビティのサブミット時に考慮することを指定します。

DISABLE

この作業アクション・セットが無効であり、データベース・アクティビティのサブミット時に考慮の対象とならないことを指定します。

規則

- ワークロード管理 (WLM) 排他 `SQL` ステートメントの後は `COMMIT` または `ROLLBACK` ステートメントでなければなりません (`SQLSTATE 5U021`)。WLM 排他 `SQL` ステートメントは次のとおりです。
 - `CREATE HISTOGRAM TEMPLATE`、`ALTER HISTOGRAM TEMPLATE`、または `DROP (HISTOGRAM TEMPLATE)`
 - `CREATE SERVICE CLASS`、`ALTER SERVICE CLASS`、または `DROP (SERVICE CLASS)`
 - `CREATE THRESHOLD`、`ALTER THRESHOLD`、または `DROP (THRESHOLD)`

- CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
- CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
- CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
- GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

例 1: DATABASE_ACTIONS 作業アクション・セットに変更を加え、作業クラス LARGE_SELECTS を使用して 2 つの作業アクションを追加します。作業アクション ONE_CONCURRENT_SELECT では、一度に並行して実行できるアクティビティの数を制御するために 1 をしきい値として適用し、キューに入れることのできるアクティビティの最大数を 3 にします。作業アクション BIG_ROWS_RETURNED では、そのクラスに含まれるデータベース・アクティビティから戻せる行の数を 1 000 000 に制限します。

```
ALTER WORK ACTION SET DATABASE_ACTIONS
ADD WORK ACTION ONE_CONCURRENT_SELECT ON WORK CLASS LARGE_SELECTS
  WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 3 STOP EXECUTION
ADD WORK ACTION BIG_ROWS_RETURNED ON WORK CLASS LARGE_SELECTS
  WHEN SQLROWSRETURNED > 1000000 STOP EXECUTION
```

例 2: ADMIN_APPS_ACTIONS 作業アクション・セットに変更を加えて、MAP_SELECTS 作業アクションを変更し、作業クラス SELECT_CLASS の下のスーパー・サービス・クラス ADMIN_APPS で実行されるすべてのアクティビティをサービス・サブクラス ALL_SELECTS にマップします。また、作業クラス UPDATE_CLASS で実行されるすべてのアクティビティをサービス・サブクラス ALL_SELECTS にマップする、MAP_UPDATES という名前の新規作業アクションを追加します。

```
ALTER WORK ACTION SET ADMIN_APPS_ACTIONS
ALTER WORK ACTION MAP_SELECTS MAP ACTIVITY TO ALL_SELECTS
ADD WORK ACTION MAP_UPDATES ON WORK CLASS UPDATE_CLASS
  MAP ACTIVITY TO ALL_SELECTS
```

ALTER WORK CLASS SET

ALTER WORK CLASS SET ステートメントは、作業クラス・セットに作業クラスを追加、変更、またはドロップします。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効

ALTER WORK CLASS SET

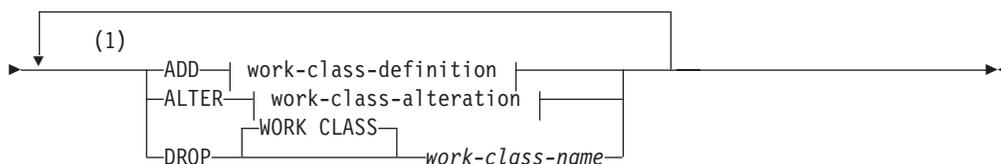
力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

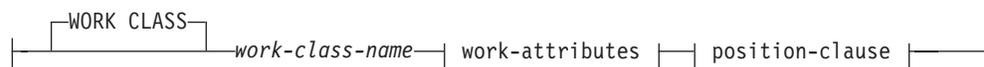
このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文

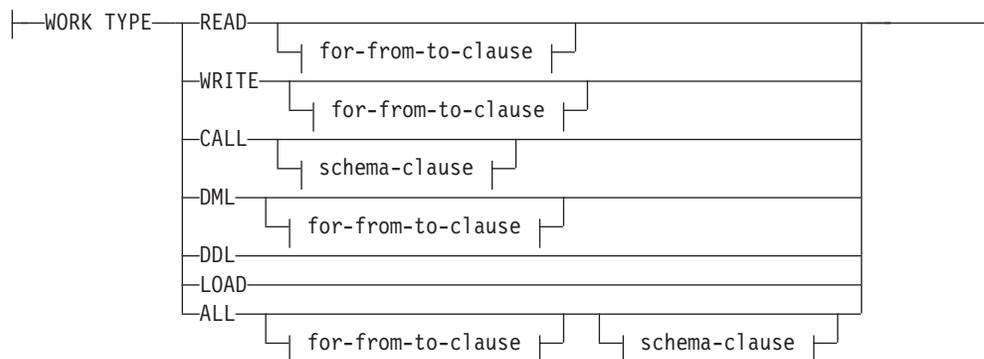
▶▶ ALTER WORK CLASS SET *work-class-set-name* ▶▶



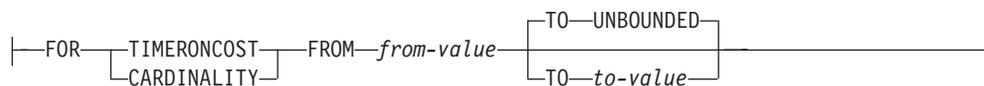
work-class-definition:



work-attributes:

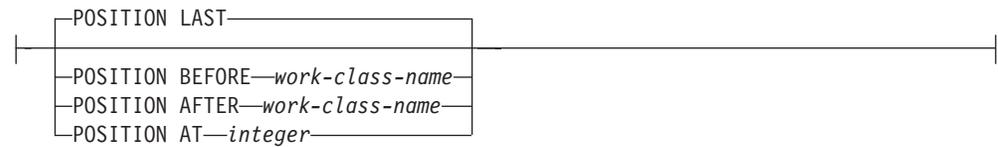
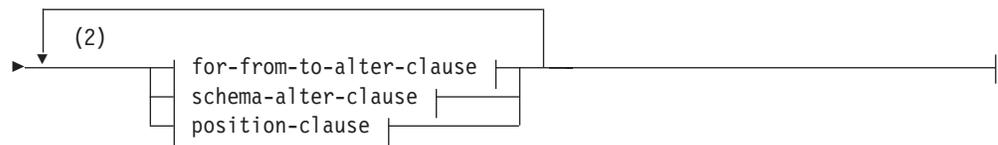
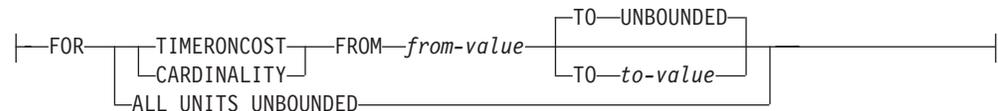
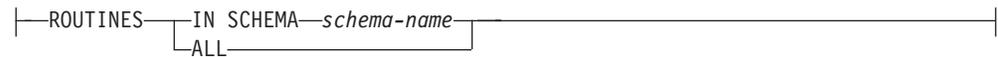


for-from-to-clause:



schema-clause:



position-clause:**work-class-alteration:****for-from-to-alter-clause:****schema-alter-clause:****注:**

- 1 ADD、ALTER、および DROP 節は、指定されている順に処理されます。
- 2 同じ節を複数回指定することはできません。

説明*work-class-set-name*

変更する作業クラス・セットを識別します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *work-class-set-name* には、現行のサーバー上に既存の作業クラス・セット名を指定する必要があります (SQLSTATE 42704)。

ADD

作業クラス・セットに作業クラスを追加します。詳細は、『CREATE WORK CLASS SET』の項を参照してください。

ALTER

データベース・アクティビティの属性と、作業クラス・セット内での特定の作業クラスの位置を変更します。

ALTER WORK CLASS SET

WORK CLASS *work-class-name*

変更する作業クラスを識別します。*work-class-name* には、現行のサーバー上の作業クラス・セットに既存の作業クラスを指定する必要があります (SQLSTATE 42704)。

DROP

作業クラス・セットから作業クラスをドロップします。

WORK CLASS *work-class-name*

ドロップする作業クラスを識別します。*work-class-name* には、現行のサーバー上の作業クラス・セットに既存の作業クラスを指定する必要があります (SQLSTATE 42704)。この作業クラス・セットに関連付けられている作業アクション・セットのいずれかに従属する作業アクションがある作業クラスはドロップできません (SQLSTATE 42893)。

for-to-from-alter-clause

FOR

FROM *from-value* TO *to-value* 節で指定されている情報のタイプを示します。FOR 節は、以下の作業タイプでのみ使用されます。

- READ
- WRITE
- DML
- ALL

TIMERONCOST

作業の見積コスト (timeron 単位)。この値は、作業が FROM *from-value* TO *to-value* 節で指定された範囲に入るかどうかを判断するのに使用されます。

CARDINALITY

作業の見積カーディナリティー。この値は、作業が FROM *from-value* TO *to-value* 節で指定された範囲に入るかどうかを判断するのに使用されます。

FROM *from-value* **TO UNBOUNDED** または **FROM** *from-value* **TO** *to-value*

timeron 値 (見積コストの場合) またはカーディナリティーの範囲を指定します。データベース・アクティビティーがこの作業クラスに属するためには、この範囲に収まっていなければなりません。この範囲には、*from-value* および *to-value* が含まれています。この範囲は以下の作業タイプでのみ使用されます。

- READ
- WRITE
- DML
- ALL

FROM *from-value* **TO UNBOUNDED**

from-value は、ゼロまたは正の DOUBLE 値でなければなりません (SQLSTATE 5U019)。範囲に上限はありません。

FROM *from-value* **TO** *to-value*

from-value はゼロまたは正の DOUBLE 値でなければならず、
to-value は正の DOUBLE 値でなければなりません。 *from-value* は
to-value 以下でなければなりません (SQLSTATE 5U019)。

ALL UNITS UNBOUNDED

FROM *from-value* TO *to-value* 節に範囲を指定せず、指定した作業タイプに含まれるすべての作業を組み込むことを示します。

*schema-alter-clause***ROUTINES**

この節は、作業タイプが CALL または ALL で、データベース・アクティビティが CALL ステートメントの場合にのみ使用されます。

IN SCHEMA *schema-name*

CALL ステートメントが呼び出すプロシージャのスキーマ名を指定します。

ALL

すべてのスキーマを組み込むことを指定します。

*position-clause***POSITION**

この作業クラスを作業クラス・セット内のどの位置に配置するかを指定します。この位置によって、作業クラスが評価される順序が決まります。実行時に作業クラスの割り当てを実行する際、データベース・マネージャは、まずデータベースまたはサービス・スーパークラスのどちらかのオブジェクトに関連付けられている作業クラス・セットを判別します。次に、その作業クラス・セット内で最初に一致する作業クラスが選択されます。このキーワードが指定されていない場合、作業クラスは最後尾に配置されます。

LAST

作業クラスを、作業クラス・セット内で作業クラスの番号付きリストの最後尾に配置することを指定します。

BEFORE *work-class-name*

作業クラスを、リストの作業クラス *work-class-name* の前に配置することを指定します。 *work-class-name* は、現行のサーバー上に存在する作業クラス・セットの作業クラスを識別するものでなければなりません (SQLSTATE 42704)。

AFTER *work-class-name*

作業クラスを、リストの作業クラス *work-class-name* の後に配置することを指定します。 *work-class-name* は、現行のサーバー上に存在する作業クラス・セットの作業クラスを識別するものでなければなりません (SQLSTATE 42704)。

AT *position*

作業クラス・セット内で作業クラスを配置する位置を、作業クラスの番号付きリストの中での絶対位置として指定します。この値には、任意の正整数値を指定できます (SQLSTATE 42615)。 *position*

ALTER WORK CLASS SET

が既存の作業クラスの数に 1 を足した値より大きい場合、その作業クラスは作業クラス・セットの最後尾に配置されます。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

例 1: 作業クラス・セット LARGE_QUERIES に変更を加えて、既存の 2 つの作業クラスの範囲がそれぞれ 100 000 から始まって上限なしとなるように設定します。見積コスト (timeron 単位) が 10 000 以上のすべての SELECT ステートメントに 3 つ目の作業クラスを追加し、この作業クラスが既存の 2 つの作業クラスよりも高い優先順位を持つように配置します。

```
ALTER WORK CLASS SET LARGE_QUERIES
ALTER WORK CLASS LARGE_ESTIMATED_COST
FOR TIMERONCOST FROM 100000 TO UNBOUNDED
ALTER WORK CLASS LARGE_CARDINALITY
FOR CARDINALITY FROM 100000 TO UNBOUNDED
ADD WORK CLASS LARGE_SELECTS WORK TYPE READ
FOR TIMERONCOST FROM 10000 TO UNBOUNDED POSITION AT 1
```

例 2: DML_STATEMENTS という名前の作業クラス・セットに変更を加えて、DELETE、INSERT、MERGE、または UPDATE ステートメントを含むすべての DML SELECT ステートメントを表す作業クラスを追加します。

```
ALTER WORK CLASS SET DML_STATEMENTS
ADD WORK CLASS UPDATE_CLASS WORK TYPE WRITE
```

ALTER WORKLOAD

ALTER WORKLOAD ステートメントは、ワークロードを変更します。

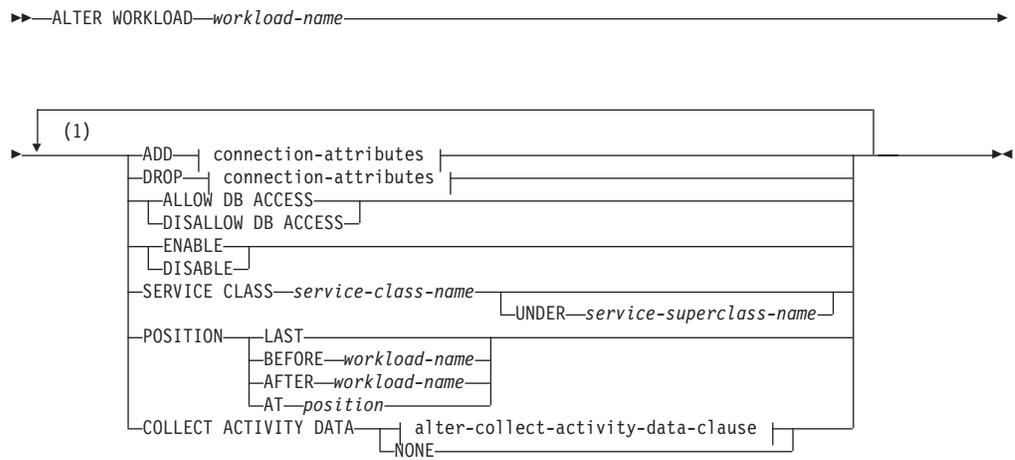
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

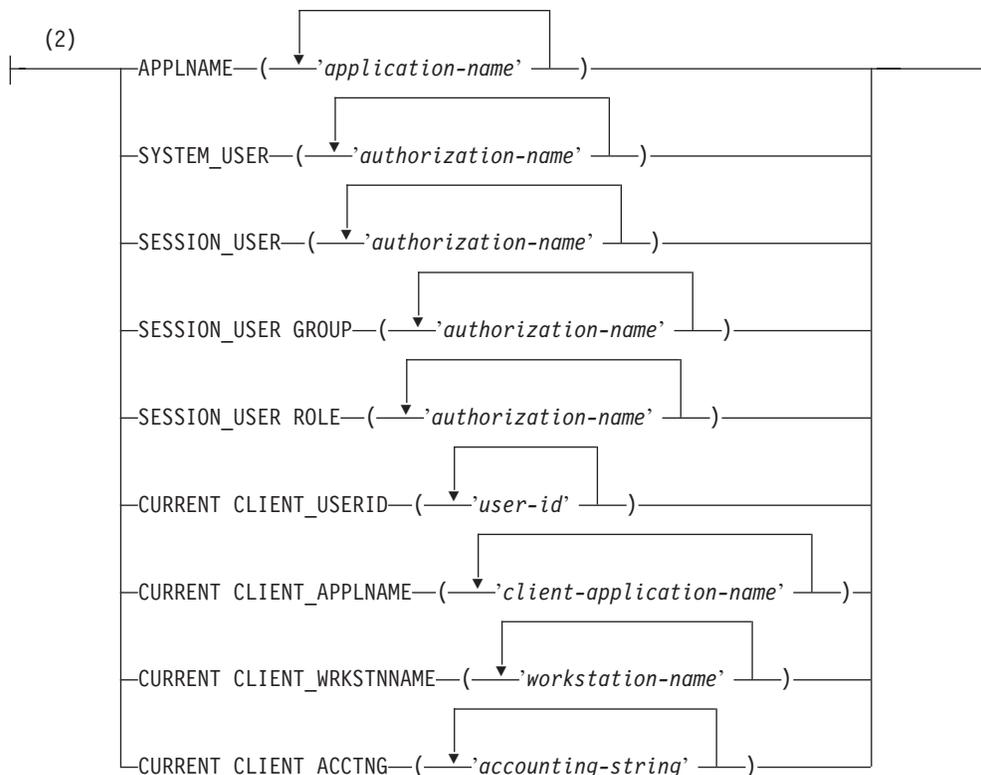
このステートメントの許可 ID が持つ特権には、 SYSADM または DBADM 権限が含まれている必要があります。

構文

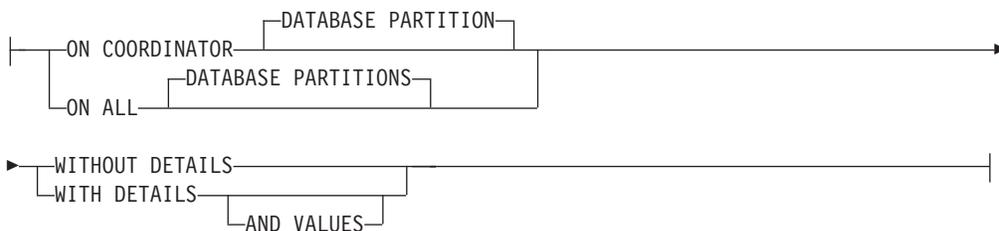


connection-attributes:

ALTER WORKLOAD



alter-collect-activity-data-clause:



注:

- 1 同じ節を複数回指定することはできません。
- 2 各接続属性節は、一度しか指定できません。

説明

workload-name

変更するワークロードを識別します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。

ADD *connection-attributes*

ワークロードの定義に 1 つ以上の接続属性の値を追加します。指定する各接続属性の値は、ワークロードに既に定義されている値であってはなりません

(SQLSTATE 5U039)。 *workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、ADD オプションを指定できません (SQLSTATE 42832)。

DROP connection-attributes

ワークロードの定義から 1 つ以上の接続属性の値をドロップします。指定する各接続属性の値は、ワークロードに定義されている値でなければなりません (SQLSTATE 5U040)。 *workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、DROP オプションを指定できません (SQLSTATE 42832)。少なくとも 1 つの接続属性の値が定義されている必要があります。最後の接続属性の値はドロップできません (SQLSTATE 5U022)。

connection-attributes

ワークロードの接続属性の値を指定します。

APPLNAME ('application-name', ...)

APPLNAME 接続属性に 1 つ以上のアプリケーションを指定します。1 つのアプリケーション名をリストの中で複数回指定することはできません (SQLSTATE 42713)。 *application-name* は、大文字と小文字を区別して扱います。この値は、システム・モニター出力や LIST APPLICATIONS コマンドからの出力の「アプリケーション名」フィールドに表示される値と等しくなります。

SYSTEM_USER ('authorization-name', ...)

SYSTEM_USER 接続属性に 1 つ以上の許可 ID を指定します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

SESSION_USER ('authorization-name', ...)

SESSION_USER 接続属性に 1 つ以上の許可 ID を指定します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

SESSION_USER GROUP ('authorization-name', ...)

SESSION_USER GROUP 接続属性に 1 つ以上の許可 ID を指定します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

SESSION_USER ROLE ('authorization-name', ...)

SESSION_USER ROLE 接続属性に 1 つ以上の許可 ID を指定します。このコンテキストでの SESSION 許可 ID のロールは、どのように取得されたロールであるかに関係なく、SESSION 許可 ID に使用可能なすべてのロールを参照します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

CURRENT_CLIENT_USERID ('user-id', ...)

CURRENT_CLIENT_USERID 接続属性に 1 つ以上のクライアント・ユーザー ID を指定します。1 つのクライアント・ユーザー ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

CURRENT_CLIENT_APPLNAME ('client-application-name', ...)

CURRENT_CLIENT_APPLNAME 接続属性に 1 つ以上のアプリケーションを指定します。1 つのアプリケーション名をリストの中で複数回指定することはできません (SQLSTATE 42713)。 *client-application-name* は、大文字

ALTER WORKLOAD

と小文字を区別して扱います。この値は、システム・モニター出力の「TP
モニター・クライアント・アプリケーション名」フィールドに表示される値
と等しくなります。

CURRENT CLIENT_WRKSTNNAME (*'workstation-name'*, ...)

CURRENT CLIENT_WRKSTNNAME 接続属性に 1 つ以上のクライアント
・ワークステーション名を指定します。1 つのクライアント・ワークステ
ーション名をリストの中で複数回指定することはできません (SQLSTATE
42713)。

CURRENT CLIENT_ACCTNG (*'accounting-string'*, ...)

CURRENT CLIENT_ACCTNG 接続属性に 1 つ以上のクライアント会計情
報ストリングを指定します。1 つのクライアント会計情報ストリングをリス
トの中で複数回指定することはできません (SQLSTATE 42713)。

ALLOW DB ACCESS または **DISALLOW DB ACCESS**

このワークロードに関連付けられているワークロード・オカレンスにデータベ
ースへのアクセスを許可するかどうかを指定します。

ALLOW DB ACCESS

このワークロードに関連付けられているワークロード・オカレンスにデー
タベースへのアクセスを許可することを指定します。

DISALLOW DB ACCESS

このワークロードに関連付けられているワークロード・オカレンスにデー
タベースへのアクセスを許可しないことを指定します。このワークロードに関
連付けられている次の作業単位は拒否されます (SQLSTATE 5U020)。既に
実行中のワークロード・オカレンスは完了まで実行できます。

workload-name が 'SYSDEFAULTADMWORKLOAD' の場合は、このオプシ
ョンを指定できません (SQLSTATE 42832)。

ENABLE または **DISABLE**

ワークロードを選択する際にこのワークロードを考慮するかどうかを指定し
ます。

ENABLE

このワークロードを有効にし、ワークロードを選択する際にこのワークロ
ードを考慮することを指定します。

DISABLE

このワークロードを無効にし、ワークロードを選択する際にこのワークロ
ードを考慮しないことを指定します。 *workload-name* が
SYSDEFAULTUSERWORKLOAD または **SYSDEFAULTADMWORKLOAD**
の場合は、このオプションを指定できません (SQLSTATE 42832)。

SERVICE CLASS*service-class-name*

このワークロードに関連付けられている要求をサービス・クラス
service-class-name で実行することを指定します。 *service-class-name* には、現行
のサーバー上の既存のサービス・クラスを指定する必要があります (SQLSTATE
42704)。 *service-class-name* を 'SYSDEFAULTSUBCLASS'、
'SYSDEFAULTSYSTEMCLASS'、または
'SYSDEFAULTMAINTENANCECLASS' にすることはできません (SQLSTATE
5U032)。 *workload-name* が 'SYSDEFAULTADMWORKLOAD' の場合は、このオ
プションを指定できません (SQLSTATE 42832)。

UNDER *service-superclass-name*

この節は、サービス・サブクラスを指定するときに使用されます。
service-superclass-name は、*service-class-name* のサービス・スーパークラスを識別します。*service-superclass-name* には、現行のサーバー上の既存のサービス・スーパークラスを指定する必要があります (SQLSTATE 42704)。*service-superclass-name* を 'SYSDEFAULTSYSTEMCLASS' または 'SYSDEFAULTMAINTENANCECLASS' にすることはできません (SQLSTATE 5U032)。

POSITION

ワークロードの番号付きリストの中でこのワークロードをどの位置に配置するかを指定します。実行時には、このリストの順番で、必須の接続属性と一致する最初のワークロードが検索されます。*workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、このオプションを指定できません (SQLSTATE 42832)。

LAST

ワークロードが、デフォルトのワークロード SYSDEFAULTUSERWORKLOAD および SYSDEFAULTADMWORKLOAD の前の、リストで最後のワークロードであることを指定します。

BEFORE *relative-workload-name*

リストの中でワークロードをワークロード *relative-workload-name* の前に配置することを指定します。*relative-workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。*relative-workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、BEFORE オプションを指定できません (SQLSTATE 42832)。

AFTER *relative-workload-name*

リストの中でワークロードをワークロード *relative-workload-name* の後に配置することを指定します。*relative-workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。*relative-workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、AFTER オプションを指定できません (SQLSTATE 42832)。

AT *position*

リストの中でワークロードの絶対位置を指定します。この値には、任意の正整数値を指定できます (SQLSTATE 42615)。*position* が既存のワークロードの数に 1 を足した値より大きい場合、ワークロードはリストの最後、SYSDEFAULTUSERWORKLOAD および SYSDEFAULTADMWORKLOAD のすぐ前に置かれます。

COLLECT ACTIVITY DATA

このワークロードに関連付けられている各アクティビティーに関するデータを、アクティビティーが完了したときに、アクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

alter-collect-activity-data-clause

ON COORDINATOR DATABASE PARTITION

アクティビティのコーディネーターのデータベース・パーティションでのみアクティビティ・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティが処理されるすべてのデータベース・パーティションでアクティビティ・データを収集することを指定します。ただし、アクティビティの詳細や値は、アクティビティのコーディネーターのデータベース・パーティションでのみ収集されます。

WITHOUT DETAILS

このワークロードに関連付けられている各アクティビティに関するデータを、アクティビティの実行が完了したときに、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティについて、これらの情報をアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティに入力データ値が含まれている場合、それをアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

このワークロードに関連付けられている各アクティビティについてはアクティビティ・データを収集しないことを指定します。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。 WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。新しくサブミットされるワークロード・オカレンスの場合は、変更は ALTER WORKLOAD ステートメントがコミットされた後に有効になります。アクティブなワークロード・オカレンスの場合は、変更は次の作業単位の開始時に有効になります。
- DISABLE オプションが指定される場合、ワークロードはステートメントのコミット後に無効になります。次回ワークロードの選択が行われる際には、このワークロードは考慮されません。ALTER WORKLOAD ステートメントのコミット時に、このワークロードに関連付けられているアクティブなワークロード・オカレンスが存在する場合、そのワークロード・オカレンスは、現行の作業単位が終了するまで引き続き実行されます。次の作業単位が開始されるときにワークロードの再評価が行われ、接続は別のワークロードに関連付けられます。

例

例 1: ワークロード PAYROLL の現在の位置づけでは、実行時に DB2 がワークロードを選択する際、ワークロード INVENTORY が最初に考慮されるようになっています。評価の順序を変更して、PAYROLL が最初に考慮されるようにします。

```
ALTER WORKLOAD PAYROLL
  POSITION BEFORE INVENTORY
```

例 2: DB2 がカタログ内の他のワークロードよりも先にワークロード BENCHMARK を評価するように、評価の順序を変更します。

```
ALTER WORKLOAD BENCHMARK
  POSITION AT 1
```

例 3: APPLNAME を appl1、appl2、および appl3 に設定し、SYSTEM_USER を BOB および MARY に設定して、ワークロード REPORTS が作成されました。このワークロードを変更して、アプリケーション名のリストに新しいアプリケーション appl4 を追加し、REPORTS にマップされなくなった appl2 を除去します。

```
ALTER WORKLOAD REPORTS
  ADD APPLNAME ('appl4')
  DROP APPLNAME ('appl2')
```

ALTER WRAPPER

ALTER WRAPPER ステートメントは、ラッパーのプロパティの更新に使用されます。

呼び出し

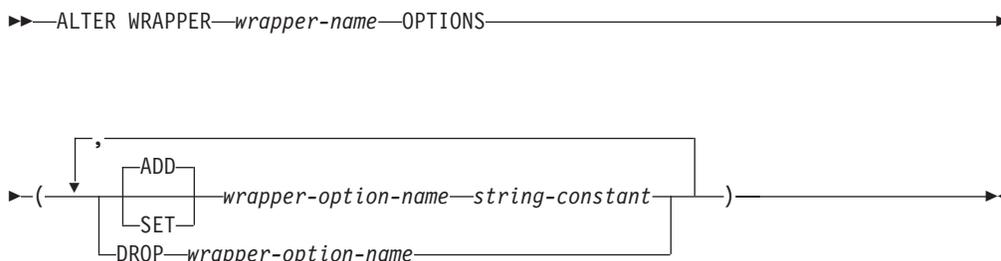
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

ALTER WRAPPER

許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文



説明

wrapper-name

ラッパーの名前を指定します。

OPTIONS

使用可能にする、リセットする、またはドロップするラッパー・オプションを指定します。

ADD

サーバー・オプションを使用可能にします。

SET

ラッパー・オプションの設定を変更します。

wrapper-option-name

使用可能にする、またはリセットするラッパー・オプションを指定します。現在、唯一サポートされているラッパー・オプション名は DB2_FENCED です。

string-constant

wrapper-option-name の設定を、文字ストリング定数として指定します。有効な値は 'Y' または 'N' です。リレーショナル・ラッパーのデフォルト値は 'N' で、非リレーショナル・ラッパーのデフォルト値は 'Y' です。

DROP *wrapper-option-name*

ラッパー・オプションをドロップします。

注

- ALTER WRAPPER ステートメントの実行には、ラッパー固有のオプションの有効性検査は含まれていません。
- 所定の作業単位 (UOW) 内の ALTER WRAPPER ステートメントは、UOW に以下のいずれかがすでに含まれている場合には処理できません (SQLSTATE 55007)。
 - ラッパーに属するニックネームを参照する SELECT ステートメント。
 - ラッパーに属するニックネーム上のオープン・カーソル。

- ラッパーに属するニッケネームに対して発行される INSERT、DELETE、または UPDATE ステートメント。

例

例 1: DB2_FENCED オプションをラッパー NET8 に設定します。

```
ALTER WRAPPER NET8 OPTIONS (SET DB2_FENCED 'Y')
```

ALTER XSROBJECT

このステートメントは、特定の XML スキーマの分解サポートを有効または無効にするのに使用されます。分解が有効になっているアノテーション付き XML スキーマを使用することによって、XML 文書をリレーショナル表に分解することができます。

呼び出し

ALTER XSROBJECT ステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

以下のいずれかの権限が必要です。

- SYSADM または DBADM
- SQL スキーマに対する ALTERIN
- 変更される XSR オブジェクトの所有権

構文

```
▶▶ ALTER XSROBJECT xsobject-name {
    ENABLE DECOMPOSITION
  | DISABLE DECOMPOSITION
}
```

説明

xsobject-name

変更する XSR オブジェクトを示します。 *xsobject-name* (暗黙的または明示的スキーマ修飾子を含む) は、現行のサーバーに存在する XSR オブジェクトを固有に識別しなければなりません。この ID を持つ XSR オブジェクトが存在しない場合、エラーが戻されます (SQLSTATE 42704)。

ENABLE DECOMPOSITION または DISABLE DECOMPOSITION

XSR オブジェクトを分解に使用することを有効または無効にします。識別された XSR オブジェクトは、XML スキーマでなければなりません (SQLSTATE 42809)。分解を有効にするためには、XML スキーマに分解規則を含むアノテーションが付けられている必要があります (SQLSTATE 225DE)、この分解規則によって参照されるオブジェクトが現行サーバーに存在しなければなりません (SQLSTATE 42704)。

ALTER XSROBJECT

注:

- XSR オブジェクトの分解サポートを無効にすると、関連するすべてのカタログ項目が除去されます。
- XSR オブジェクトが従属するオブジェクト (表など) がドロップまたは変更されて XSR オブジェクトと非互換になった場合、XSR オブジェクトの分解サポートは無効になります。

ASSOCIATE LOCATORS

ASSOCIATE LOCATORS ステートメントは、ストアード・プロシージャから戻される結果セットごとの結果セット・ロケータ値を入手します。

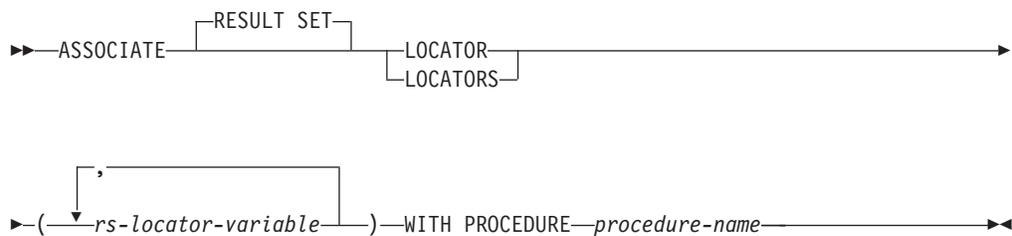
呼び出し

このステートメントは、SQL プロシージャに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

許可

必要ありません。

構文



説明

rs-locator-variable

コンパウンド SQL (プロシージャ) ステートメントで宣言されている結果セット・ロケータ変数を指定します。

WITH PROCEDURE

ここで指定したプロシージャ名で、結果セット・ロケータを戻すストアード・プロシージャを識別します。

procedure-name

プロシージャ名は修飾または非修飾の名前です。

完全修飾のプロシージャ名は、2 つの部分からなる名前です。最初の部分は、ストアード・プロシージャのスキーマ名を含んでいる ID です。最後の部分は、ストアード・プロシージャの名前を含んでいる ID です。それぞれの部分の間はピリオドで区切らなければなりません。それらの部分の一部またはすべてを区切り ID にできます。

プロシージャー名が非修飾の場合、暗黙的なスキーマ名が修飾子としてプロシージャー名に追加されることはないので、プロシージャー名に含まれる名前は 1 つだけです。ASSOCIATE LOCATOR ステートメントを正常に実行するのに必要なことは、ステートメント内の非修飾のプロシージャー名を、最近実行した CALL ステートメント (ただし、非修飾のプロシージャー名を使用するように指定したもの) でのプロシージャー名と同じにすることだけです。その CALL ステートメントでの非修飾の名前の暗黙的なスキーマ名は、マッチングにおいて考慮されません。プロシージャー名を指定する際に従わなければならない規則について、以下で説明します。

ASSOCIATE LOCATORS ステートメントを実行する場合、プロシージャーの名前または指定で、CALL ステートメントを使用して要求側がすでに呼び出しているストアード・プロシージャーを識別しなければなりません。ASSOCIATE LOCATORS ステートメントのプロシージャー名は、CALL ステートメントで指定したのと同じ方法で指定しなければなりません。たとえば CALL ステートメントで 2 つの部分からなる名前を指定した場合、ASSOCIATE LOCATORS ステートメントでも 2 つの部分からなる名前を使用しなければなりません。

注

- ASSOCIATE LOCATORS ステートメントでリストされている結果セット・ロケータ変数の数が、ストアード・プロシージャーで戻されるロケータの数より少ない場合、ステートメント内のすべての変数に値が割り当てられ、警告が出されます。
- ASSOCIATE LOCATORS ステートメントでリストされている結果セット・ロケータ変数の数が、ストアード・プロシージャーで戻されるロケータの数より多い場合、超過した変数に値 0 が割り当てられます。
- あるストアード・プロシージャーが同じ呼び出し側から複数回呼び出される場合、アクセス可能なのは、最新の結果セットだけです。
- 結果セット・ロケータ値は、PREPARE ステートメントによって事前に準備された CALL ステートメントを実行する EXECUTE ステートメントを使用して呼び出されるプロシージャーで使用可能です。しかし、結果セット・ロケータ値は、EXECUTE IMMEDIATE ステートメントを使用して呼び出されるプロシージャーでは使用できません。

例

以下の例のステートメントは、SQL プロシージャーに組み込まれることを想定しています。

例 1: 結果セット・ロケータ変数 LOC1 および LOC2 を使用して、ストアード・プロシージャー P1 から戻される 2 つの結果セットの結果セット・ロケータ値を入手します。ストアード・プロシージャーが 1 つの部分だけからなる名前呼び出されることを想定しています。

```
CALL P1;
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)
WITH PROCEDURE P1;
```

ASSOCIATE LOCATORS

例 2: 例 1 のシナリオを繰り返しますが、スキーマ MYSCHEMA で確実にストアド・プロシージャ P1 が使用されるように、2 つの部分からなる名前を使用し、ストアド・プロシージャの明示的なスキーマ名を指定します。

```
CALL MYSCHEMA.P1;  
ASSOCIATE RESULT SET LOCATORS (LOC1, LOC2)  
WITH PROCEDURE MYSCHEMA.P1;
```

AUDIT

AUDIT ステートメントは、現行サーバーの特定のデータベースまたはデータベース・オブジェクトで使用する監査ポリシーを決定します。オブジェクトが使用されている場合は常に、そのポリシーに従って監査されます。

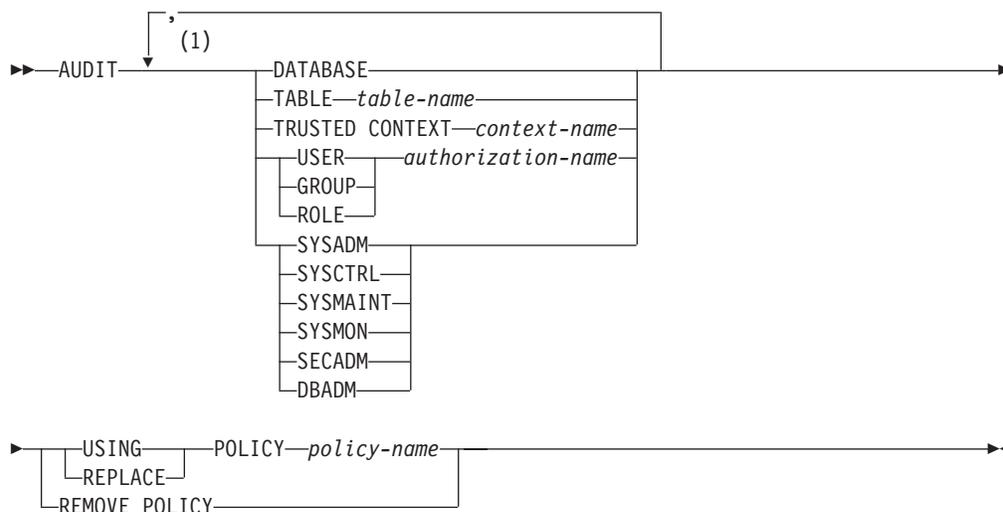
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文



注:

- 1 各節 (適切な場合は同じオブジェクト名を持つもの) は 1 回以上指定できます (SQLSTATE 42713)。

説明

DATABASE

監査ポリシーを現行サーバーのデータベースと関連付けること、またはそこから除去することを指定します。データベース内で発生する監査可能イベントはすべて、関連付けられている監査ポリシーに従って監査されます。

TABLE *table-name*

監査ポリシーを *table-name* と関連付けること、またはそこから除去することを指定します。 *table-name* は、現行のサーバーに存在する表、マテリアライズ照会表 (MQT)、またはニックネームを識別するものでなければなりません (SQLSTATE 42704)。これに、ビュー、カタログ表、宣言済み一時表 (SQLSTATE 42995)、または型付き表 (SQLSTATE 42997) を指定することはできません。表にアクセスされたときに生成されるのは EXECUTE カテゴリ監査イベントだけです (データを伴う場合と伴わない場合がある)。これは、ポリシーが他のカテゴリを監査することを指示している場合でも同じです。

TRUSTED CONTEXT *context-name*

監査ポリシーを *context-name* と関連付けること、またはそこから除去することを指定します。 *context-name* は現行のサーバーに存在するトラステッド・コンテキストを識別するものでなければなりません (SQLSTATE 42704)。トラステッド・コンテキスト *context-name* によって定義されるトラステッド接続内で発生する監査可能イベントはすべて、関連付けられている監査ポリシーに従って監査されます。

USER *authorization-name*

許可 ID *authorization-name* を持つユーザーと監査ポリシーを関連付けること、またはそこから除去することを指定します。 *authorization-name* によって開始される監査可能イベントはすべて、関連付けられている監査ポリシーに従って監査されます。

GROUP *authorization-name*

許可 ID *authorization-name* を持つグループと監査ポリシーを関連付けること、またはそこから除去することを指定します。 *authorization-name* のメンバーであるユーザーによって開始される監査可能イベントはすべて、関連付けられている監査ポリシーに従って監査されます。グループのユーザー・メンバーシップを判別できない場合、ポリシーはそのユーザーに適用されません。

ROLE *authorization-name*

許可 ID *authorization-name* を持つロールと監査ポリシーを関連付けること、またはそこから除去することを指定します。 *authorization-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。 *authorization-name* のメンバーであるユーザーによって開始される監査可能イベントはすべて、関連付けられている監査ポリシーに従って監査されます。他のロールまたはグループを介した間接的なロール・メンバーシップも有効です。

SYSADM, SYSMANT, SYSCTRL, SYSMON, DBADM, または SECADM

指定された権限と監査ポリシーを関連付けること、またはそこから除去することを指定します。(たとえイベントでその権限を必要としない場合でも) 指定された権限の付与されたユーザーによって開始される監査可能イベントはすべて、関連付けられている監査ポリシーに従って監査されます。

USING、REMOVE、または REPLACE

指定されたオブジェクトで監査ポリシーを使用、除去、または置換するかどうかを指定します。

USING

指定されたオブジェクトで監査ポリシーを使用することを指定します。このオブジェクトについて定義されている監査ポリシーが既に存在してはなりません (SQLSTATE 5U041)。監査ポリシーが既に存在する場合はそれを除去するかまたは置換する必要があります。

REMOVE

指定されたオブジェクトから監査ポリシーを除去することを指定します。監査ポリシーに従ったオブジェクト使用の監査は行われなくなります。監査ポリシーからオブジェクトから除去されると、カタログから関連付けが削除されます。

REPLACE

指定されたオブジェクトの既存の監査ポリシーを置換することを指定します。これは REMOVE オプションと USING オプションの両方を 1 つのステップにまとめたもので、これを使用した場合、指定されたオブジェクトに監査ポリシーが未適用となる期間がないことが保証されます。指定されたオブジェクトでポリシーが使用されていなかった場合、REPLACE は USING と同じということになります。

POLICY *policy-name*

監査設定を決定するために使用する監査ポリシーを指定します。 *policy-name* は、現行サーバーの既存の監査ポリシーを識別するものでなければなりません (SQLSTATE 42704)。

規則

- AUDIT 排他 SQL ステートメントの後は、COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。AUDIT 排他 SQL ステートメントは次のとおりです。
 - AUDIT
 - CREATE AUDIT POLICY、ALTER AUDIT POLICY、または DROP (AUDIT POLICY)
 - DROP (監査ポリシーと関連付けられる場合は ROLE または TRUSTED CONTEXT)
- AUDIT 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。
- オブジェクトと関連付けることのできるポリシーは 1 つだけです (SQLSTATE 5U042)。

注

- カタログに変更が書き込まれますが、COMMIT ステートメントが実行されるまでは有効になりません。
- 変更は、監査ポリシーが適用されるオブジェクトを参照する次の作業単位が適用されるまでは有効になりません。例えば、監査ポリシーがデータベースで使用さ

れている場合、現行の作業単位は COMMIT または ROLLBACK ステートメントが完了するまではポリシーに従って監査を開始しません。

- 監査ポリシーと関連付けられている表にビューがアクセスする場合、そのビューは基礎表のポリシーに従って監査されます。
- 監査ポリシーが表に適用される場合、その監査ポリシーはその表に基づくマテリアライズ照会表 (MQT) には適用されません。監査ポリシーを表と関連付ける場合はそのポリシーをその表に基づく MQT にも関連付けることをお勧めします。SQL ステートメントが基本表を参照していても、コンパイラーは MQT を自動的に使用する可能性があります。ただし、基本表で使用されている監査ポリシーは引き続き有効です。
- トラストッド・コンテキスト内でユーザーの切り替え操作が実行される際、監査ポリシーはすべて新規ユーザーに応じて再評価され、古いユーザーのポリシーは現行セッションでは使用されません。これは特に、ユーザー、ユーザーのグループまたはロールのメンバーシップ、およびユーザーの権限と直接関連付けられている監査ポリシーに当てはまります。例えば、切り替え前のユーザーが監査対象のロールのメンバーだったために現行セッションが監査されたとします。切り替え先のユーザーがそのロールのメンバーではない場合、そのポリシーはセッションに適用されなくなります。
- SET SESSION USER ステートメントが実行される際、元のユーザー (およびそのユーザーのグループとロールのメンバーシップと権限) と関連付けられている監査ポリシーは、SET SESSION USER ステートメントで指定されたユーザーと関連付けられているポリシーに結合されます。元のユーザーと関連付けられている監査ポリシーは引き続き有効となり、SET SESSION USER ステートメントで指定されたユーザーのポリシーも同じように有効となります。1 つのセッション内で複数の SET SESSION USER ステートメントが発行される場合、元のユーザーおよび現行ユーザーと関連付けられている監査ポリシーだけが考慮されます。
- 監査ポリシーが関連付けられているオブジェクトがドロップされると、監査ポリシーとの関連付けはカタログから除去され、存在しなくなります。その後いつかそのオブジェクトが再作成される場合、そのオブジェクトがドロップされた時点で関連付けられていたポリシーによる監査は、そのオブジェクトに対して実行されません。

例

例 1: 監査ポリシー DBAUDPRF を使用して、現行のサーバーのデータベースの監査設定を決定します。

```
AUDIT DATABASE USING POLICY DBAUDPRF
```

例 2: EMPLOYEE 表から監査ポリシーを除去します。

```
AUDIT TABLE EMPLOYEE REMOVE POLICY
```

例 3: 監査ポリシー POWERUSERS を使用して、権限 SYSADM、DBADM、SECADM、およびグループ DBAS の監査設定を決定します。

```
AUDIT SYSADM, DBADM, SECADM, GROUP DBAS USING POLICY POWERUSERS
```

例 4: ロールの監査ポリシー TELLER を新規ポリシー TELLERPRF で置き換えます。

```
AUDIT ROLE TELLER REPLACE POLICY TELLERPRF
```

BEGIN DECLARE SECTION

BEGIN DECLARE SECTION ステートメントは、ホスト変数宣言セクションの始まりを示します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、実行可能ステートメントではありません。また、REXX に指定することはできません。

許可

必要ありません。

構文

▶—BEGIN DECLARE SECTION—▶

説明

BEGIN DECLARE SECTION ステートメントは、ホスト言語の規則に従って変数宣言が許される個所であれば、アプリケーション・プログラムのどのような個所にもコーディングできます。これは、ホスト変数宣言セクションの始まりを示すのに使用されます。ホスト変数セクションは、END DECLARE SECTION ステートメントで終了します。

規則

- BEGIN DECLARE SECTION と END DECLARE SECTION ステートメントは、対にして使用する必要があり、ネストすることはできません。
- 宣言セクションに SQL ステートメントを含めることはできません。
- REXX 以外のすべてのホスト言語において、SQL ステートメントで参照される変数は、宣言セクションで宣言する必要があります。また、そのセクションは、変数に対する最初の参照より前になければなりません。一般に、REXX では、LOB ロケータとファイル参照変数を除いて、ホスト変数は宣言されません。それらは BEGIN DECLARE SECTION では宣言されません。
- 宣言セクションの外部で宣言される変数の名前を、宣言セクションで宣言されている変数と同じ名前にすることはできません。
- LOB データ・タイプのデータ・タイプと長さの前には、SQL TYPE IS キーワードを付ける必要があります。

例

例 1: C プログラムで、ホスト変数 hv_smint (smallint)、 hv_vchar24 (varchar(24))、 hv_double (double)、 hv_blob_50k (blob(51200))、 hv_struct (構造化タイプ "struct_type" は blob(10240)) を定義します。

```
EXEC SQL BEGIN DECLARE SECTION;  
       short hv_smint;  
       struct {
```

```

short hv_vchar24_len;
char hv_vchar24_value[24];
}                                hv_vchar24;
double hv_double;
SQL TYPE IS BLOB(50K) hv_blob_50k;
SQL TYPE IS struct_type AS BLOB(10k) hv_struct;
EXEC SQL END DECLARE SECTION;

```

例 2: COBOL プログラムで、ホスト変数 HV-SMINT (smallint)、HV-VCHAR24 (varchar(24))、HV-DEC72 (dec(7,2))、および HV-BLOB-50k (blob(51200)) を定義します。

```

WORKING-STORAGE SECTION.
EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 HV-SMINT PIC S9(4) COMP-4.
01 HV-VCHAR24.
49 HV-VCHAR24-LENGTH PIC S9(4) COMP-4.
49 HV-VCHAR24-VALUE PIC X(24).
01 HV-DEC72 PIC S9(5)V9(2) COMP-3.
01 HV-BLOB-50K USAGE SQL TYPE IS BLOB(50K).
EXEC SQL END DECLARE SECTION END-EXEC.

```

例 3: FORTRAN プログラムで、ホスト変数 HVSMINT (smallint)、HVVCHAR24 (char(24))、HVDDOUBLE (double)、および HVBLOB50k (blob(51200)) を定義します。

```

EXEC SQL BEGIN DECLARE SECTION
INTEGER*2 HVSMINT
CHARACTER*24 HVVCHAR24
REAL*8 HVDDOUBLE
SQL TYPE IS BLOB(50K) HVBLOB50K
EXEC SQL END DECLARE SECTION

```

注: FORTRAN では、予期される値が 254 バイトを超える場合には、CLOB ホスト変数を使用する必要があります。

例 4: REXX プログラムで、ホスト変数 HVSMINT (smallint)、HVBLOB50K (blob(51200))、および HVCLOBLOC (CLOB ロケーター) を定義します。

```

DECLARE :HVCLOBLOC LANGUAGE TYPE CLOB LOCATOR
call sqlexec 'FETCH c1 INTO :HVSMINT, :HVBLOB50K'

```

変数 HVSMINT と HVBLOB50K は、FETCH ステートメントで使用することによって、暗黙に定義されています。

CALL

CALL ステートメントは、プロシージャまたは外部プロシージャを呼び出します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

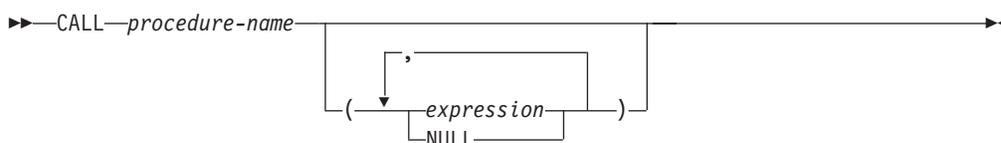
許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- プロシージャでの EXECUTE 特権
- SYSADM または DBADM 権限

実行の許可を与えられていないステートメントの許可 ID を持つ一致するプロシージャが存在する場合、エラーが出されます (SQLSTATE 42501)。

構文



説明

procedure-name

呼び出されるプロシージャを指定します。プロシージャはカタログに記述されている必要があります。呼び出す特定のプロシージャは、プロシージャ解決を使用して選択します。(詳細については、このステートメントの『注』セクションを参照してください。)

expression または NULL

指定される *expression* または NULL のどちらかは、CALL の引数です。CALL ステートメントの *n* 番目の引数は、プロシージャの CREATE PROCEDURE ステートメントで定義されている *n* 番目のパラメーターに対応します。

CALL の各引数は、以下のようなプロシージャ定義における対応するパラメーターと互換性がなければなりません。

- IN パラメーター
 - 引数は、パラメーターに割り当て可能でなければなりません。
 - ストリング引数の割り当てには、ストレージ割り当て規則を使用します。
- OUT パラメーター
 - 引数は、単一の変数またはパラメーター・マーカでなければなりません (SQLSTATE 42886)。
 - 引数は、パラメーターに割り当て可能でなければなりません。
 - ストリング引数の割り当てには、検索割り当て規則を使用します。
- INOUT パラメーター
 - 引数は、単一の変数またはパラメーター・マーカでなければなりません (SQLSTATE 42886)。
 - 引数は、パラメーターに割り当て可能でなければなりません。
 - ストリング引数の割り当てには、呼び出しについてはストレージ割り当て規則、および戻りに関しては検索割り当て規則を使用します。

注• **パラメーター割り当て:**

CALL ステートメントの実行時には、CALL ステートメントの引数のそれぞれの値が、プロシージャの対応するパラメーターに割り当てられます (ストレージ割り当てを使用して)。ホスト言語の呼び出し規則に応じて、制御がプロシージャに渡されます。プロシージャの実行が完了すると、プロシージャの各パラメーターの値が、OUT または INOUT として定義された CALL ステートメントの対応する引数に割り当てられます (ストレージ割り当てを使用して)。プロシージャによってエラーが戻される場合、OUT 引数は未定義で、INOUT 引数は未変更です。割り当て規則についての詳細は、『割り当てと比較』を参照してください。

CALL ステートメントが SQL プロシージャの中にあり、他の SQL プロシージャを呼び出していれば、XML パラメーターの割り当ては参照によって行われます。XML 引数が参照渡しであれば、入力ノード・ツリーがある場合は、XML 引数から、文書の順序、オリジナル・ノード ID、およびすべての親プロパティなどを含むすべてのプロパティを保持したまま、直接使用されます。

• **プロシージャ・シグニチャー:**

プロシージャは、そのスキーマ、プロシージャ名、およびパラメーター数によって識別されます。これはプロシージャ・シグニチャーと呼ばれ、データベース内でユニークである必要があります。プロシージャごとにパラメーターの数が違っていれば、1 つのスキーマに同じ名前プロシージャが複数存在してもかまいません。

• **SQL パス:**

プロシージャは、修飾名 (スキーマおよびプロシージャ名) を参照して呼び出すことができます。修飾名の後に、括弧で閉じられた引数のオプションのリストが続きます。また、スキーマ名を指定せずにプロシージャを呼び出すことも可能であり、その場合は、同じ数のパラメーターを持つ異なるスキーマのプロシージャが選択可能になります。このような場合、プロシージャ解決に役立つ SQL パスが使用されます。SQL パスとは、同じ名前、同じパラメーター数を持つプロシージャを識別するために探索されるスキーマのリストです。静的 CALL ステートメントに対する SQL パスは、FUNCPATH BIND オプションを使って指定されます。動的 CALL ステートメントの場合、SQL パスは CURRENT PATH 特殊レジスターの値です。

• **プロシージャ解決:**

特定のプロシージャの呼び出しに対して、データベース・マネージャーは、同じ名前をもつ呼び出し可能なプロシージャから呼び出すプロシージャを判別する必要があります。プロシージャ解決は、以下の手順で行われます。

1. カタログ (SYSCAT.ROUTINES) から、以下のすべての条件が真となるすべてのプロシージャを探します。
 - スキーマ名が指定された呼び出し (修飾子付き参照) の場合、スキーマ名とプロシージャ名が呼び出し名に一致する。

- スキーマ名が指定されていない呼び出し (修飾子なし参照) の場合、プロシージャ名が呼び出し名に一致し、SQL パス中のスキーマの 1 つに一致するスキーマ名がある。
 - 定義済みパラメーターの数が呼び出しと一致している。
 - 呼び出し側が、プロシージャで EXECUTE 特権を持っている。
2. スキーマがその SQL パスで最初に出現するプロシージャが選択されます。

ステップ 1 の後で候補となるプロシージャが残らなかった場合は、エラー (SQLSTATE 42884) になります。

• **SQL プロシージャからの DB2_RETURN_STATUS の検索:**

SQL プロシージャが RETURN ステートメントを状況値とともに正常に発行すると、この値が SQLCA の最初の SQLERRD フィールドに戻されます。SQL プロシージャで CALL ステートメントが発行される場合、GET DIAGNOSTICS ステートメントを使用して DB2_RETURN_STATUS 値を検索します。SQLSTATE がエラーを示す場合は、値は -1 になります。エラーが出ないで、RETURN ステートメントがプロシージャで指定されなかった場合には、値は 0 になります。

• **プロシージャから戻される結果セット:**

呼び出し側プログラムが CLI、JDBC、または SQLJ を使用して作成されている場合、または呼び出し側が SQL プロシージャの場合には、結果セットを呼び出し側に直接戻すことができます。プロシージャは、結果セットにカーソルを宣言して、その結果セットでカーソルをオープンし、プロシージャ終了時にカーソルをオープンしたままにすることによって、結果セットを戻すよう指定します。

プロシージャの終了時に、

- オープンされたままのカーソルすべてについて、結果セットは呼び出し側に戻されるか、(WITH RETURN TO CLIENT カーソルの場合) クライアントに直接戻されます。
- 未読の行だけが戻されます。たとえば、カーソルの結果セットに 500 行が含まれていて、そのうち 150 行がプロシージャの終了時にプロシージャによって読み取られた場合、第 151 行から第 500 行までが呼び出し側またはアプリケーションに戻されます (該当する場合)。

プロシージャが CLI または JDBC から呼び出され、複数のカーソルがオープンされたままの場合、結果セットはカーソルがオープンされた順序でのみ処理が可能です。

• **パフォーマンスの向上:**

すべての引数の値は、アプリケーションからプロシージャへ渡されます。この操作のパフォーマンスを向上させるには、OUT パラメーターに対応し、数バイト以上の長さを持つホスト変数を、CALL ステートメントを実行する前に NULL に設定しなければなりません。

• **CALL ステートメントのネスト:**

プロシージャは、ルーチンやアプリケーション・プログラムから呼び出すことができます。プロシージャをルーチンから呼び出す場合、その呼び出しはネストされるものと見なされます。

プロシージャが照会結果セットを戻す場合には、その結果セットは以下のように戻されます。

- RETURN TO CALLER 結果セットは、直前のネスト・レベルにあるプログラムでのみ可視です。
- RETURN TO CLIENT 結果セットは、そのプロシージャがネストされた一連のプロシージャから呼び出された場合に限り可視になります。呼び出しチェーンのどこかで関数やメソッドが実行されると、結果セットは不可視になります。結果セットが可視の場合、最初のプロシージャが呼び出されたクライアント・アプリケーションでのみ可視になります。

以下の例について考慮します。

```
Client program:
EXEC SQL CALL PROCA;

PROCA:
EXEC SQL CALL PROCB;

PROCB:
EXEC SQL DECLARE B1 CURSOR WITH RETURN TO CLIENT ...;
EXEC SQL DECLARE B2 CURSOR WITH RETURN TO CALLER ...;
EXEC SQL DECLARE B3 CURSOR FOR SELECT UDFA FROM T1;

UDFA:
EXEC SQL CALL PROCC;

PROCC:
EXEC SQL DECLARE C1 CURSOR WITH RETURN TO CLIENT ...;
EXEC SQL DECLARE C2 CURSOR WITH RETURN TO CALLER ...;
```

プロシージャ PROCB から:

- カーソル B1 はクライアント・アプリケーションでは可視ですが、プロシージャ PROCA では不可視です。
- カーソル B2 は PROCA では可視ですが、クライアントには不可視です。

プロシージャ PROCC から:

- カーソル C1 は UDFA でもクライアント・アプリケーションでも不可視です。(UDFA はクライアントと PROCC との間の呼び出しチェーンで現れ、結果セットはクライアントに戻されないからです。)
 - カーソル C2 は UDFA では可視ですが、上位のプロシージャでは不可視です。
- **トリガー、動的コンパウンド・ステートメント、関数、またはメソッド内のネストされたプロシージャ:**

トリガー、動的コンパウンド・ステートメント、関数、またはメソッドの中で、プロシージャが呼び出される時は、次のとおりです。

- プロシージャは COMMIT または ROLLBACK ステートメントを発行してはなりません。
- プロシージャから戻される結果セットにはアクセスできません。

CALL

- プロシージャが READS SQL DATA または MODIFIES SQL DATA として定義されている場合、プロシージャを呼び出したステートメントによって変更されている表には、プロシージャ内のどのステートメントからもアクセスできません (SQLSTATE 57053)。また、プロシージャが MODIFIES SQL DATA として定義されている場合は、プロシージャを呼び出したステートメントによって読み取りまたは変更されている表には、プロシージャ内のどのステートメントからもアクセスできません (SQLSTATE 57053)。

関数またはメソッドの中でプロシージャが呼び出されるときは、次のとおりです。

- プロシージャは、呼び出された関数またはメソッドと同じ表アクセスに関する制約事項を持ちます。
- 関数またはメソッドが呼び出される前に定義されたセーブポイントは、プロシージャでは不可視です。またプロシージャ内で定義されたセーブポイントは関数またはメソッド以外では不可視になります。
- プロシージャから戻される RETURN TO CLIENT 結果セットに、クライアントからアクセスすることはできません。

• 互換性:

- CALL_RESOLUTION DEFERRED オプションを使用してアプリケーションを事前にコンパイルすると、アプリケーションに組み込める旧書式の CALL ステートメントがあります。このオプションは SQL プロシージャおよびフェデレーテッド・プロシージャには使用できません。

例

例 1:

Java™ プロシージャが、以下のステートメントを使用してデータベースに定義されています。

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                                OUT COST DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
    EXTERNAL NAME 'parts!onhand'
    LANGUAGE JAVA
    PARAMETER STYLE DB2GENERAL;
```

Java アプリケーションは、以下のコードを使用してこのプロシージャを呼び出します。

```
...
CallableStatement stpCall;

String sql = "CALL PARTS_ON_HAND (?, ?, ?)";

stpCall = con.prepareStatement(sql); /*con is the connection */

stpCall.setInt(1, hvPartnum);
stpCall.setBigDecimal(2, hvCost);
stpCall.setInt(3, hvQuantity);

stpCall.registerOutParameter(2, Types.DECIMAL, 2);
stpCall.registerOutParameter(3, Types.INTEGER);

stpCall.execute();
```

```

hvCost = stpCall.getBigDecimal(2);
hvQuantity = stpCall.getInt(3);
...

```

このアプリケーションのコード部分は、クラス `parts` の Java メソッド `onhand` を呼び出します。これは、CALL ステートメントで指定されたプロシージャ名がデータベースで検出され、外部名 `parts!onhand` を持っているためです。

例 2:

4 つの異なるスキーマに 6 個の FOO プロシージャがあり、以下のように登録されているとします (必須キーワードの一部は省略されています)。

```

CREATE PROCEDURE AUGUSTUS.FOO (INT) SPECIFIC FOO_1 ...
CREATE PROCEDURE AUGUSTUS.FOO (DOUBLE, DECIMAL(15, 3)) SPECIFIC FOO_2 ...
CREATE PROCEDURE JULIUS.FOO (INT) SPECIFIC FOO_3 ...
CREATE PROCEDURE JULIUS.FOO (INT, INT, INT) SPECIFIC FOO_4 ...
CREATE PROCEDURE CAESAR.FOO (INT, INT) SPECIFIC FOO_5 ...
CREATE PROCEDURE NERO.FOO (INT,INT) SPECIFIC FOO_6 ...

```

以下のようにプロシージャが参照されるとします (I1 および I2 は INTEGER 値です)。

```
CALL FOO(I1, I2)
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"JULIUS", "AUGUSTUS", "CAESAR"
```

アルゴリズムに従っていきます。

スキーマ "NERO" が SQL パスに含まれていないため、特定の名前 FOO_6 のあるプロシージャは候補から除かれます。パラメーターの数が違うため、FOO_1、FOO_3、および FOO_4 は候補から除かれます。残った候補は順番に考慮され、SQL パスにより判別します。引数およびパラメーターのタイプは無視されることに注意してください。FOO_5 のパラメーターは CALL の引数と正確に一致しますが、SQL パスで "CAESAR" の前に "AUGUSTUS" が現れるため FOO_2 が選ばれます。

CASE

CASE ステートメントは、複数の条件に基づいて実行パスを選択します。CASE ステートメントを CASE 式と混同しないでください。CASE 式を使用すると、1 つ以上の条件の評価に基づいて 1 つの式を選択できます。

呼び出し

このステートメントは、SQL プロシージャに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

許可

CASE ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID によって保持される特権には、CASE ステートメントに組み込

明であると評価された場合、次の *search-condition* が評価されます。真であると評価される *search-condition* がなく、なおかつ ELSE 節が使用されている場合、ELSE 節内のステートメントが処理されます。

SQL-procedure-statement

呼び出すステートメントを指定します。『コンパウンド SQL (プロシージャー)』で、*SQL-procedure-statement* を参照してください。

END CASE

case-statement を終了します。

注

- WHEN で指定した条件がすべて真ではなく、なおかつ ELSE 節が指定されていない場合、実行時にエラーが出され、CASE ステートメントの実行が終了します (SQLSTATE 20000)。
- 使用する CASE ステートメントでは、考えられるあらゆる実行条件を全て網羅するようにしてください。

例

SQL 変数 *v_workdept* の値によっては、表 DEPARTMENT 内の更新列 DEPTNAME を適当な名前更新しなければなりません。

以下の例では、*simple-case-statement-when-clause* の構文を使用して、これを行う方法を示しています。

```

CASE v_workdept
  WHEN 'A00'
    THEN UPDATE department
    SET deptname = 'DATA ACCESS 1';
  WHEN 'B01'
    THEN UPDATE department
    SET deptname = 'DATA ACCESS 2';
  ELSE UPDATE department
    SET deptname = 'DATA ACCESS 3';
END CASE

```

以下の例では、*searched-case-statement-when-clause* の構文を使用して、これを行う方法を示しています。

```

CASE
  WHEN v_workdept = 'A00'
    THEN UPDATE department
    SET deptname = 'DATA ACCESS 1';
  WHEN v_workdept = 'B01'
    THEN UPDATE department
    SET deptname = 'DATA ACCESS 2';
  ELSE UPDATE department
    SET deptname = 'DATA ACCESS 3';
END CASE

```

CLOSE

CLOSE ステートメントは、カーソルをクローズします。カーソルのオープン時に結果表が作成された場合、その表は破棄されます。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。これは、動的に作成できない実行可能ステートメントです。

許可

必要ありません。カーソルの使用に必要な許可については、『DECLARE CURSOR』を参照してください。

構文

```

▶▶ CLOSE cursor-name [WITH RELEASE]

```

説明

cursor-name

クローズするカーソルを識別します。 DECLARE CURSOR ステートメントの項で説明されているように、 *cursor-name* は、宣言されたカーソルを指定しなければなりません。 CLOSE ステートメントを実行する場合、カーソルはオープン状態でなければなりません。

WITH RELEASE

カーソルのために保留されていた、すべてのロックを解放しようとします。すべてのロックを解放する必要はないことに注意してください。これらのロックは他の操作または活動のために保留することができます。

注

- 作業単位の終了時には、アプリケーション・プロセスに属し、 WITH HOLD オプションを指定せずに宣言されたすべてのカーソルは暗黙にクローズされます。
- WITH RELEASE 節は、関数またはメソッドで定義されたカーソルのクローズには効力を持っていません。またこの節は、関数またはメソッドから呼び出されるプロシージャで定義されたカーソルのクローズに対しても効力を持ちません。
- WITH RELEASE 節は、分離レベル CS または UR で機能しているカーソルに対しては影響を与えません。また、分離レベル RS または RR で機能しているカーソルに対して WITH RELEASE を指定した場合には、それらの分離レベルの保証の一部が終了させられます。特に、カーソルを再オープンする場合には、RS カーソルが '反復不可読み取り' 状態になったり、RR カーソルが '反復不可読み取り' か '幻像読み取り' 状態のどちらかになる可能性があります。

もともと RR か RS だったカーソルが、WITH RELEASE 節を使用してクローズされたのちに、再オープンされると、新しいロックを獲得できます。

- クローズされずに呼び出し側プログラムに戻ったプロシージャ内のカーソルには、特殊な規則が適用されます。
- カーソルがオープンしている間は (つまり、まだクローズしていない場合)、そのカーソルをステートメントが呼び出した結果 (たとえば、NEXT VALUE 式が組み込まれたカーソルをシーケンスに使用した FETCH または UPDATE) 生じたシーケンス値への変更が、そのカーソルが示したシーケンスを PREVIOUS VALUE

に更新することはありません。このように影響を受けるシーケンスの PREVIOUS VALUE 値は、CLOSE ステートメントを使ってカーソルを明示的にクローズした際に更新されます。パーティション・データベース環境では、コミットやロールバックによってカーソルを暗黙的にクローズした場合、PREVIOUS VALUE はシーケンスについて生成された最新の値に更新されない場合があります。

例

カーソルを使用して、C プログラム変数 dnum、dname、および mnum の中に、一度に 1 行ずつ取り出します。最後にカーソルをクローズします。再びカーソルをオープンすると、再びその位置は取り出される行の始めになります。

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT DEPTNO, DEPTNAME, MGRNO
  FROM TDEPT
  WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

while (SQLCODE==0) {
  EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
  .
  .
}
EXEC SQL CLOSE C1;
```

COMMENT

COMMENT ステートメントは、種々のオブジェクトのカatalog記述にコメントを追加するか、または置き換えます。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

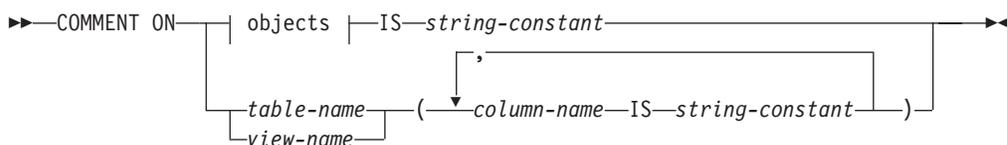
- オブジェクトのカatalog・ビューの OWNER 列に記録されているオブジェクト (列または制約の場合は基礎表) の所有者
- スキーマに対する ALTERIN 特権 (複数部分の名前を使用できるオブジェクトにのみ適用される)
- オブジェクトに対する CONTROL 特権 (索引、パッケージ、表、またはビューの各オブジェクトにのみ適用される)
- オブジェクトに対する ALTER 特権 (表オブジェクトにのみ適用される)
- WITH ADMIN OPTION (ロールにのみ適用される)

COMMENT

- SECADM 権限 (監査ポリシー、ロール、セキュリティー・ラベル、セキュリティー・ラベル・コンポーネント、セキュリティー・ポリシー、またはトラステッド・コンテキストの各オブジェクトにのみ適用される)
- SYSADM または DBADM 権限 (監査ポリシー、ロール、セキュリティー・ラベル、セキュリティー・ラベル・コンポーネント、セキュリティー・ポリシー、またはトラステッド・コンテキストの各オブジェクトには適用されない)

表スペースまたはデータベース・パーティション・グループの場合、許可 ID は SYSCTRL 権限または SYSADM 権限を持っている必要がある点に注意してください。

構文



オブジェクト:

ALIAS	<i>alias-name</i>
AUDIT POLICY	<i>policy-name</i>
COLUMN	<i>table-name.column-name</i> <i>view-name.column-name</i>
CONSTRAINT	<i>table-name.constraint-name</i>
FUNCTION	<i>function-name</i> (, <i>data-type</i>)
SPECIFIC FUNCTION	<i>specific-name</i>
FUNCTION MAPPING	<i>function-mapping-name</i>
HISTOGRAM TEMPLATE	<i>template-name</i> (1)
INDEX	<i>index-name</i>
NICKNAME	<i>nickname</i>
DATABASE PARTITION GROUP	<i>db-partition-group-name</i>
PACKAGE	<i>schema-name.</i> <i>package-id</i> VERSION <i>version-id</i>
PROCEDURE	<i>procedure-name</i> (, <i>data-type</i>)
ROLE	<i>role-name</i>
SPECIFIC PROCEDURE	<i>specific-name</i>
SCHEMA	<i>schema-name</i>
SECURITY LABEL	<i>sec-label-name</i>
SECURITY LABEL COMPONENT	<i>label-comp-name</i>
SECURITY POLICY	<i>label-pol-name</i>
SERVER	<i>server-name</i>
SERVER OPTION	<i>server-option-name</i> FOR remote-server
SERVICE CLASS	<i>service-class-name</i> UNDER <i>service-superclass-name</i>
TABLE	<i>table-name</i> <i>view-name</i>
TABLESPACE	<i>tablespace-name</i>
THRESHOLD	<i>threshold-name</i>
TRIGGER	<i>trigger-name</i>
TRUSTED CONTEXT	<i>context-name</i>
TYPE	<i>type-name</i>
TYPE MAPPING	<i>type-mapping-name</i>
VARIABLE	<i>variable-name</i>
WORK ACTION SET	<i>work-action-set-name</i>
WORK CLASS SET	<i>work-class-set-name</i>
WORKLOAD	<i>workload-name</i>
WRAPPER	<i>wrapper-name</i>
XSROBJECT	<i>xsrobject-name</i>

remote-server:

SERVER	<i>server-name</i>
SERVER TYPE	<i>server-type</i> VERSION <i>server-version</i> WRAPPER <i>wrapper-name</i>

server-version:

<i>version</i>	<i>release</i> <i>mod</i>
<i>version-string-constant</i>	

注:

- 1 *Index-name* には、索引、あるいは SPECIFICATION ONLY 指定の索引のどちらかの名前を指定できます。

説明

ALIAS *alias-name*

alias-name (別名) に対するコメントの追加または置き換えを行うことを指定します。*alias-name* には、現行のサーバー上の既存の別名を指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.TABLES カタログ・ビューの別名を記述する行の REMARKS 列の値を置き換えます。

AUDIT POLICY *policy-name*

監査ポリシーに対するコメントの追加または置き換えを行うことを指定します。*policy-name* は、現行のサーバーに存在する監査ポリシーを識別するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.AUDITPOLICIES カタログ・ビューのうち、監査ポリシーを記述する行の REMARKS 列の値を置き換えます。

COLUMN *table-name.column-name* または *view-name.column-name*

列に対するコメントを追加または置き換えることを指定します。

table-name.column-name または *view-name.column-name* の組み合わせは、現行のサーバー上の既存の列と表の組み合わせを指定していなければなりません (SQLSTATE 42704)。しかし、グローバル一時表を指定してはなりません (SQLSTATE 42995)。コメントは、SYSCAT.COLUMNS カタログ・ビューのその列を記述する行の REMARKS 列の値を置き換えます。

CONSTRAINT *table-name.constraint-name*

制約に対するコメントの追加または置き換えを指定します。

table-name.constraint-name の組み合わせは、制約とそれが制約する表を指定していなければなりません。これらは、現行のサーバーに存在していなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.TABCONST カタログ・ビューのその制約を記述する行の REMARKS 列の値を置き換えます。

FUNCTION

関数に対するコメントの追加または置き換えを指定します。指定する関数インスタンスは、現行のサーバーに存在するユーザー定義関数、または関数テンプレートでなければなりません。

関数のインスタンスを指定する方法としては、次のようにいくつかの方法があります。

FUNCTION *function-name*

特定の関数を指定します。*function-name* (関数名) の関数がちょうど 1 つだけ存在している場合にのみ有効です。このように指定する関数には、任意の数のパラメーターが定義されていても構いません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこの名前関数が存在しない場合は、エラー (SQLSTATE 42704)

になります。指定したスキーマまたは暗黙のスキーマに、この関数の特定インスタンスが複数存在する場合は、エラー (SQLSTATE 42725) になります。

FUNCTION *function-name* (*data-type*,...)

コメントを付ける関数名を固有に識別する関数シグニチャーを指定します。関数選択のアルゴリズムは使用されません。

function-name

コメントを付ける関数名を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(*data-type*, ...)

これは、CREATE FUNCTION ステートメント上で (対応する位置に) 指定されたデータ・タイプに一致していなければなりません。データ・タイプ (*data-type*) の数、およびそれらのデータ・タイプを論理的に連結したものが、コメントを追加または置換する特定の関数を識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。代わりに、空の括弧をコーディングすることによって、データ・タイプの一致を調べる際にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE FUNCTION ステートメントにおける指定に完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(n) のタイプは、n に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

(FOR BIT DATA 属性は、一致検索のためのシグニチャーの一部とは見なされません。したがって、例えばシグニチャーの中に CHAR FOR BIT DATA が指定されている場合、それは CHAR とだけ定義されている関数と一致し、シグニチャーに CHAR とだけ指定されているものは、CHAR FOR BIT DATA と指定されている関数と一致することになります。)

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つ関数がない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC FUNCTION *specific-name*

関数のコメントを追加または置換することを指定します (関数を指定する他の方法については、FUNCTION の部分を参照)。関数の作成時に指定された特定の関数名、またはデフォルト値として使用された特定の関数名を使用して、コメントを付ける特定のユーザー定義関数を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定関数のインスタンスを指定していなければなりません。そうでない場合、エラー (SQLSTATE 42704) になります。

SYSIBM、SYSFUN、または SYSPROC スキーマ (SQLSTATE 42832) の関数についてのコメントを付けることはできません。

コメントは、SYSCAT.ROUTINES カタログ・ビューのうち、その関数を記述する行の REMARKS 列の値を置き換えます。

FUNCTION MAPPING *function-mapping-name*

関数マッピングに対するコメントの追加または置き換えを指定します。 *function-mapping-name* には、現行のサーバー上の既存の関数マッピングを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.FUNCMAPPINGS カタログ・ビューのうち、その関数マッピングを記述する行の REMARKS 列の値を置き換えます。

HISTOGRAM TEMPLATE *template-name*

ヒストグラム・テンプレートに対するコメントを追加または置換することを指定します。 *template-name* は、現行のサーバーに存在するヒストグラム・テンプレートを識別するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.HISTOGRAMTEMPLATES カタログ・ビューのうち、そのヒストグラム・テンプレートを記述する行の REMARKS 列の値を置き換えます。

INDEX *index-name*

索引または SPECIFICATION ONLY 指定の索引に対するコメントを追加または置換することを指定します。 *index-name* には、現行のサーバー上の既存の特定の索引、または SPECIFICATION ONLY 指定の索引のいずれかを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.INDEXES カタログ・ビューのうち、その索引または SPECIFICATION ONLY 指定の索引を記述する行の REMARKS 列の値を置き換えます。

NICKNAME *nickname*

ニックネームに対するコメントの追加または置き換えを指定します。 *nickname* には、現行のサーバー上の既存のニックネームを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.TABLES カタログ・ビューのニックネームを記述する行の REMARKS 列の値を置き換えます。

DATABASE PARTITION GROUP *db-partition-group-name*

データベース・パーティション・グループに対するコメントの追加または置き換えを指定します。 *db-partition-group-name* には、現行のサーバー上の既存の特定のデータベース・パーティション・グループを指定する必要があります

(SQLSTATE 42704)。コメントは、SYSCAT.DBPARTITIONGROUPS カタログ・ビューのうち、そのデータベース・パーティション・グループを記述する行の REMARKS 列の値を置き換えます。

PACKAGE *schema-name.package-id*

パッケージに対するコメントを追加または置換することを指定します。スキーマ名が指定されていない場合、パッケージ ID は暗黙的にデフォルト・スキーマで修飾されます。スキーマ名およびパッケージ ID は、明示的または暗黙的に指定されたバージョン ID とともに、現在のサーバーに存在するパッケージを指定していなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.PACKAGES カタログ・ビューのうち、そのパッケージを記述する行の REMARKS 列の値を置き換えます。

VERSION *version-id*

コメントを付けるパッケージ・バージョンを指定します。値が指定されない場合には、空ストリングがバージョンのデフォルトになります。パッケージ名は同じですがバージョンが異なる複数のパッケージが存在する場合には、COMMENT ステートメントの単一の呼び出しで、1 つのパッケージ・バージョンにのみ、コメントを付けることができます。次のような場合は、バージョン ID を二重引用符で区切ってください。

- バージョン ID が VERSION(AUTO) プリコンパイラー・オプションによって生成された場合
- バージョン ID が数字で始まる場合
- バージョン ID が小文字であったり、大/小文字混合である場合

ステートメントをオペレーティング・システムのコマンド・プロンプトから呼び出す場合は、各二重引用符の区切り文字の前に円記号を置いて、オペレーティング・システムによって区切り文字が外されないようにします。

PROCEDURE

プロシージャに対するコメントを追加または置換することを指定します。指定するプロシージャ・インスタンスは、現行のサーバーに存在するプロシージャでなければなりません。

プロシージャ・インスタンスを指定する方法としては、次のようにいくつかの方法があります。

PROCEDURE *procedure-name*

特定のプロシージャを指定します。スキーマに *procedure-name* のプロシージャが 1 つだけ存在している場合にのみ有効です。この方法で指定するプロシージャには、パラメーターがいくつ定義されていても構いません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマに該当する名前プロシージャが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定したスキーマまたは暗黙のスキーマにこのプロシージャの特定のインスタンスが複数存在する場合は、エラーが戻されます (SQLSTATE 42725)。

PROCEDURE *procedure-name* (*data-type*,...)

これは、コメントを付けるプロシージャを固有に識別するプロシージャ・シグニチャーを指定するのに使用されます。

procedure-name

コメントを付けるプロシージャのプロシージャ名を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されません。

(data-type, ...)

データ・タイプを指定します。ここで指定されるデータ・タイプは、CREATE PROCEDURE ステートメントの対応する位置に指定されたデータ・タイプと一致していなければなりません。フェデレーテッド・プロシージャの場合、データ・タイプは、ローカル・カタログの情報と一致しなくてはなりません。データ・タイプの数、およびそれらのデータ・タイプを論理的に連結したものが、コメントを追加または置換する特定のプロシージャを識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。代わりに、空の括弧をコーディングすることによって、データ・タイプの一致を調べる際にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE PROCEDURE ステートメントにおける指定、またはフェデレーテッド・プロシージャの場合はローカル・カタログの情報に完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つプロシージャがない場合は、エラー (SQLSTATE 42883) になります。

SPECIFIC PROCEDURE *specific-name*

プロシージャのコメントを追加または置換することを指定します (プロシージャを指定する他の方法については、PROCEDURE を参照)。プロシージャの作成時に指定されたか、またはデフォルト値として付けられた特定のプロシージャ名を使用して、コメントを付ける特定のプロシージャを指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊

レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。*specific-name* に指定される名前は、指定したスキーマまたは暗黙のスキーマに含まれる特定プロシージャのインスタンスを識別するものでなければなりません。それ以外の名前が指定された場合は、エラーが戻されます (SQLSTATE 42704)。

SYSIBM、SYSFUN、または SYSPROC スキーマ (SQLSTATE 42832) のプロシージャについてのコメントを付けることはできません。

コメントは、SYSCAT.ROUTINES カタログ・ビューのうち、そのプロシージャを記述する行の REMARKS 列の値を置き換えます。

ROLE *role-name*

ロールに対するコメントの追加または置き換えを指定します。*role-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.ROLES カタログ・ビューのうち、そのロールを記述する行の REMARKS 列の値を置き換えます。

SCHEMA *schema-name*

スキーマに対するコメントを追加または置換することを指定します。*schema-name* には、現行のサーバー上の既存のスキーマを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.SCHEMATA カタログ・ビューのうち、そのスキーマを記述する行の REMARKS 列の値を置き換えます。

SECURITY LABEL *sec-label-name*

sec-label-name という名前のセキュリティー・ラベルに対するコメントを追加または置換することを指定します。名前は、セキュリティー・ポリシーで修飾され、現行のサーバーの既存のセキュリティー・ラベルを示すものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.SECURITYLABELS カタログ・ビューのうち、そのセキュリティー・ラベルを記述する行の REMARKS 列の値を置き換えます。

SECURITY LABEL COMPONENT *label-comp-name*

label-comp-name という名前のセキュリティー・ラベル・コンポーネントに対するコメントを追加または置換することを指定します。*label-comp-name* には、現行のサーバー上の既存のセキュリティー・ラベル・コンポーネントを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.SECURITYLABELCOMPONENTS カタログ・ビューのうち、そのセキュリティー・ラベル・コンポーネントを記述する行の REMARKS 列の値を置き換えます。

SECURITY POLICY *label-pol-name*

label-pol-name という名前のセキュリティー・ポリシーに対するコメントを追加または置換することを指定します。*label-pol-name* には、現行のサーバー上の既存のセキュリティー・ポリシーを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.SECURITYPOLICIES カタログ・ビューのうち、そのセキュリティー・ポリシーを記述する行の REMARKS 列の値を置き換えます。

SERVER *server-name*

データ・ソースに対するコメントの追加または置き換えを指定します。

server-name には、現行のサーバー上の既存のデータ・ソースを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.SERVERS カタログ・ビューのうち、そのデータ・ソースを記述する行の REMARKS 列の値を置き換えます。

SERVER OPTION *server-option-name* **FOR** *remote-server*

サーバー・オプションに対するコメントの追加または置き換えを指定します。

server-option-name

サーバー・オプションを指定します。このオプションは、現行のサーバーに存在するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.SERVEROPTIONS カタログ・ビューのうち、そのサーバー・オプションを記述する行の REMARKS 列の値を置き換えます。

remote-server

server-option が適用されるデータ・ソースを示します。

SERVER *server-name*

server-option が適用されるデータ・ソースを指定します。 *server-name* には、現行のサーバー上の既存のデータ・ソースを指定する必要があります。

TYPE *server-type*

server-option が適用されるデータ・ソースのタイプを指定します。例えば、DB2 for z/OS または Oracle など。 *server-type* の指定は、大文字でも小文字でもかまいません。カタログには大文字で格納されます。

VERSION

server-name で指定したデータ・ソースのバージョンを指定します。

version

バージョン番号を指定します。 *version* は整数でなければなりません。

release

version で示されたバージョンのリリース番号を指定します。 *release* は整数でなければなりません。

mod

release で示されたリリースのモディフィケーション番号を指定します。 *mod* は整数でなければなりません。

version-string-constant

バージョンの正式名称を指定します。 *version-string-constant* は単一値 (例えば、'8i') にすることができます。あるいは、*version*、*release*、そして該当する場合は *mod* を連結した値にすることができます (例えば、'8.0.3')。

WRAPPER *wrapper-name*

server-name で示されるデータ・ソースにアクセスするとき使用するラッパーを指定します。

SERVICE CLASS *service-class-name*

サービス・クラスに対するコメントの追加または置き換えを指定します。 *service-class-name* には、現行のサーバー上の既存のサービス・クラスを指定する必要があります (SQLSTATE 42704)。サービス・サブクラスのコメントを追

加または置換するためには、UNDER 節を使用して *service-superclass-name* が指定されている必要があります。コメントは、SYSCAT.SERVICECLASSES カタログ・ビューのうち、そのサービス・クラスを記述する行の REMARKS 列の値を置き換えます。

UNDER *service-superclass-name*

サービス・サブクラスのコメントを追加または置換するときの、サービス・サブクラスのサービス・スーパークラスを指定します。

service-superclass-name には、現行のサーバー上の既存のサービス・スーパークラスを指定する必要があります (SQLSTATE 42704)。

TABLE *table-name* または *view-name*

表またはビューに対するコメントを追加または置換することを指定します。

table-name または *view-name* は、現行のサーバーの既存の表またはビュー (別名またはニックネームではない) を指定していなければならず (SQLSTATE 42704)、宣言済み一時表を指定してはなりません (SQLSTATE 42995)。コメントは、SYSCAT.TABLES カタログ・ビューのうち、その表またはビューを記述する行の REMARKS 列の値を置き換えます。

TABLESPACE *tablespace-name*

表スペースに対するコメントを追加または置換することを指定します。

tablespace-name には、現行のサーバー上の既存の特定の表スペースを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.TABLESPACES カタログ・ビューのうち、その表スペースを記述する行の REMARKS 列の値を置き換えます。

THRESHOLD *threshold-name*

しきい値に対するコメントの追加または置き換えを指定します。 *threshold-name* は現行のサーバーに存在するしきい値を識別するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.THRESHOLDS カタログ・ビューのうち、そのしきい値を記述する行の REMARKS 列の値を置き換えます。

TRIGGER *trigger-name*

トリガーに対するコメントを追加または置換することを指定します。

trigger-name には、現行のサーバー上の既存の特定のトリガーを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.TRIGGERS カタログ・ビューのうち、そのトリガーを記述する行の REMARKS 列の値を置き換えます。

TRUSTED CONTEXT *context-name*

トラステッド・コンテキストに対するコメントの追加または置き換えを指定します。 *context-name* は現行のサーバーに存在するトラステッド・コンテキストを識別するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.CONTEXTS カタログ・ビューのうち、そのトラステッド・コンテキストを記述する行の REMARKS 列の値を置き換えます。

TYPE *type-name*

ユーザー定義タイプのコメントを追加または置換することを指示します。

type-name には、現行のサーバー上の既存のユーザー定義タイプを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.DATATYPES カタログ・ビューの REMARKS 列の値を、ユーザー定義タイプを記述する行に置き換えます。

動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

TYPE MAPPING *type-mapping-name*

ユーザー定義のデータ・タイプのマッピングに対するコメントを追加または置換することを指定します。 *type-mapping-name* には、現行のサーバー上の既存のデータ・タイプ・マッピングを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.TYPEMAPPINGS カタログ・ビューのうち、そのマッピングを記述する行の REMARKS 列の値を置き換えます。

VARIABLE *variable-name*

グローバル変数に対するコメントの追加または置き換えを指定します。 *variable-name* は、現在のサーバーに存在するグローバル変数を識別するものでなければなりません (SQLSTATE 42704)。コメントは、SYSCAT.VARIABLES カタログ・ビューの変数を記述する行の REMARKS 列の値を置き換えます。

WORK ACTION SET *work-action-set-name*

作業アクション・セットに対するコメントの追加または置き換えを指定します。 *work-action-set-name* には、現行のサーバー上の既存の作業アクション・セットを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.WORKACTIONSETS カタログ・ビューのうち、その作業アクション・セットを記述する行の REMARKS 列の値を置き換えます。

WORK CLASS SET *work-class-set-name*

作業クラス・セットに対するコメントの追加または置き換えを指定します。 *work-class-set-name* には、現行のサーバー上に既存の作業クラス・セット名を指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.WORKCLASSSETS カタログ・ビューのうち、その作業クラス・セットを記述する行の REMARKS 列の値を置き換えます。

WORKLOAD *workload-name*

ワークロードに対するコメントを追加または置換することを指定します。 *workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.WORKLOADS カタログ・ビューのうち、そのワークロードを記述する行の REMARKS 列の値を置き換えます。

WRAPPER *wrapper-name*

ラッパーに対するコメントの追加または置き換えを指定します。 *wrapper-name* には、現行のサーバー上の既存のラッパーを指定する必要があります (SQLSTATE 42704)。コメントは、SYSCAT.WRAPPERS カタログ・ビューのうち、そのラッパーを記述する行の REMARKS 列の値を置き換えます。

XROBJECT *xrobject-name*

XSR オブジェクトに対するコメントの追加または置き換えを行うことを指定します。 *xrobject-name* には、現行のサーバー上の既存の XSR オブジェクトを指定する必要があります (SQLSTATE 42704)。コメントは SYSCAT.XROBJECTS カタログ・ビューのうち、その XSR オブジェクトを記述する行の REMARKS 列の値を置き換えます。

IS string-constant

追加または置換するコメントを指定します。 *string-constant* (ストリング定数) には、最大 254 バイトの任意の文字ストリング定数を指定できます。(復帰文字 (CR) と改行文字 (LF) はそれぞれ 1 バイトとカウントされます。)

table-name|view-name ({ column-name IS string-constant } ...)

この形式の COMMENT ステートメントを使用すると、表またはビューの複数の列に対するコメントを指定することができます。列名は修飾できず、各名前は指定する表またはビューの列を指定するものでなければなりません。また、その表またはビューは現行のサーバーに存在していなければなりません。 *table-name* を宣言済み一時表にすることはできません (SQLSTATE 42995)。

作動不能ビューの列にコメントを作成することはできません (SQLSTATE 51024)。

注• **互換性**

- 以前のバージョンの DB2 との互換性:
 - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。
 - TYPE *type-name* の代わりに DISTINCT TYPE *type-name* を指定できます。
 - TYPE *type-name* の代わりに DATA TYPE *type-name* を指定できます。

例

例 1: EMPLOYEE 表についてのコメントを追加します。

```
COMMENT ON TABLE EMPLOYEE
  IS 'Reflects first quarter reorganization'
```

例 2: EMP_VIEW1 ビューについてのコメントを追加します。

```
COMMENT ON TABLE EMP_VIEW1
  IS 'View of the EMPLOYEE table without salary information'
```

例 3: EMPLOYEE 表の EDLEVEL 列についてのコメントを追加します。

```
COMMENT ON COLUMN EMPLOYEE.EDLEVEL
  IS 'highest grade level passed in school'
```

例 4: EMPLOYEE 表の異なる 2 つの列についてのコメントを追加します。

```
COMMENT ON EMPLOYEE
  (WORKDEPT IS 'see DEPARTMENT table for names',
  EDLEVEL IS 'highest grade level passed in school' )
```

例 5: Pellow は、自身の PELLOW スキーマに作成した CENTRE 関数についてのコメントを付けます。シグニチャーを使用して、コメントを付ける特定の関数を指定します。

```
COMMENT ON FUNCTION CENTRE (INT,FLOAT)
  IS 'Frank''s CENTRE fctn, uses Chebychev method'
```

例 6: McBride は、PELLOW スキーマに作成した別の CENTRE 関数にコメントを付けます。特定の名前を使用して、コメントを付ける関数インスタンスを指定します。

COMMENT

```
COMMENT ON SPECIFIC FUNCTION PELLOW.FOCUS92 IS
  'Louise's most triumphant CENTRE function, uses the
  Brownian fuzzy-focus technique'
```

例 7: CHEM スキーマの関数 ATOMIC_WEIGHT にコメントを付けます。このスキーマではこの名前関数は 1 つしかないことが分かっています。

```
COMMENT ON FUNCTION CHEM.ATOMIC_WEIGHT
  IS 'takes atomic nbr, gives atomic weight'
```

例 8: Eigler は、自身の EIGLER スキーマに作成した SEARCH プロシージャーについてのコメントを付けます。シグニチャーを使用して、コメントを付ける特定のプロシージャーを指定します。

```
COMMENT ON PROCEDURE SEARCH (CHAR,INT)
  IS 'Frank's mass search and replace algorithm'
```

例 9: Macdonald は、EIGLER スキーマに作成した別の SEARCH 関数にコメントを付けます。特定の名前を使用して、コメントを付けるプロシージャー・インスタンスを指定します。

```
COMMENT ON SPECIFIC PROCEDURE EIGLER.DESTROY IS
  'Patrick's mass search and destroy algorithm'
```

例 10: BIOLOGY スキーマのプロシージャー OSMOSIS にコメントを付けます。このスキーマではこの名前プロシージャーは 1 つしかないことが分かっています。

```
COMMENT ON PROCEDURE BIOLOGY.OSMOSIS
  IS 'Calculations modelling osmosis'
```

例 11: INDEXSPEC という SPECIFICATION ONLY 指定の索引にコメントを付けます。

```
COMMENT ON INDEX INDEXSPEC
  IS 'An index specification that indicates to the optimizer
  that the table referenced by nickname NICK1 has an index.'
```

例 12: デフォルト名が NET8 のラッパーにコメントを付けます。

```
COMMENT ON WRAPPER NET8
  IS 'The wrapper for data sources associated with
  Oracle's Net8 client software.'
```

例 13: XML スキーマ HR.EMPLOYEE にコメントを作成します。

```
COMMENT ON XSROBJECT HR.EMPLOYEE
  IS 'This is the base XML Schema for employee data.'
```

例 14: トラストド・コンテキスト APPSERVER にコメントを作成します。

```
COMMENT ON TRUSTED CONTEXT APPSERVER
  IS 'WebSphere Server'
```

COMMIT

COMMIT ステートメントは、作業単位を終了し、その作業単位で行われたデータベースへの変更をコミットします。

COMMIT

その開始以降に作業単位によって獲得されたロックは、WITH HOLD を宣言されたオープン・カーソルに必要なロック以外のすべてのロックが解放されます。WITH HOLD を定義されていないすべてのオープン・カーソルはクローズされます。WITH HOLD を定義されたオープン・カーソルはオープンされたままになり、そのカーソルは、結果表の次の論理行の前に置かれます。(位置指定の UPDATE ステートメントまたは DELETE ステートメントが出される前に、FETCH を実行する必要があります。) LOB ロケータはすべて解放されます。WITH HOLD 特性を持つカーソルによって取り出された LOB 値に関連したロケータも、すべて解放されます。

トランザクション内に設定されたセーブポイントはすべて解放されます。

以下のステートメントの動作は、他のデータ定義言語 (DDL) ステートメントやデータ制御言語 (DCL) ステートメントの動作と異なります。これらのステートメントによって行われた変更は、ステートメントがコミットされるまでは有効になりません。これは、ステートメントを発行する現行接続でも同じです。これらのステートメントはアプリケーションによって一度に 1 つのみ発行され、1 つの作業単位内で 1 つのみ許可されます。どのステートメントの場合も、発行後に COMMIT ステートメントまたは ROLLBACK ステートメントを発行する必要があります。その後、これらのステートメントの中の別のものを発行することができます。

- CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
- CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
- CREATE WORK ACTION、ALTER WORK ACTION、または DROP (WORK ACTION)
- CREATE WORK CLASS、ALTER WORK CLASS、または DROP (WORK CLASS)
- CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
- GRANT (ワークロード特権) または REVOKE (ワークロード特権)

注

- 各アプリケーション・プロセスを終了する前に、その作業単位を明示的に終了することを強くお勧めします。アプリケーション・プログラムが COMMIT ステートメントまたは ROLLBACK ステートメントを実行せずに正常に終了すると、データベース・マネージャーはアプリケーションの環境に応じてコミットまたはロールバックを試みます。
- キャッシュされた動的 SQL ステートメントに対する COMMIT の影響については、『EXECUTE』を参照してください。
- 宣言済み一時表に対する COMMIT の影響の可能性については、『DECLARE GLOBAL TEMPORARY TABLE』を参照してください。

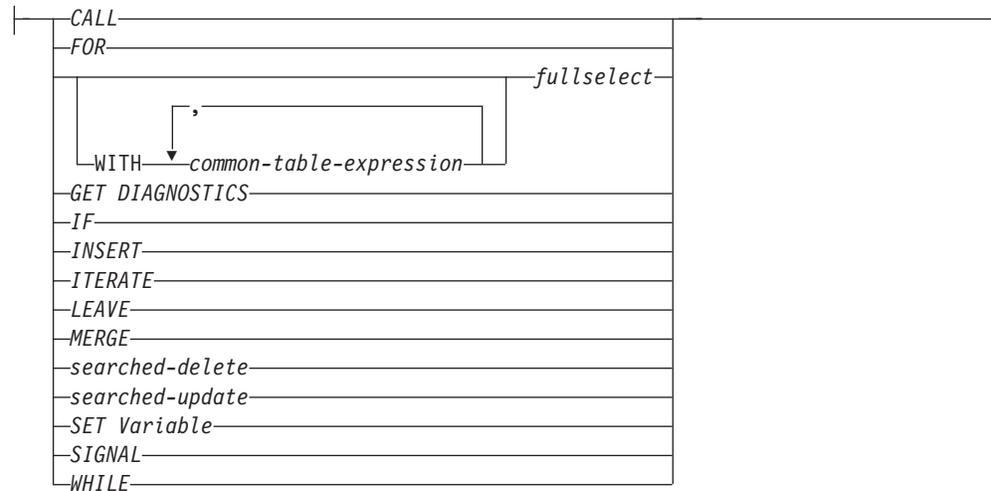
例

最後のコミット・ポイント以降にデータベースに対して行われた変更をコミットします。

```
COMMIT WORK
```


コンパウンド SQL (動的)

SQL-routine-statement:



注:

- 1 ステートメントが関数、メソッド、またはトリガー定義にある場合のみ、ラベルを指定できます。

説明

label

コード・ブロックのラベルを定義します。開始ラベルが指定されている場合、そのラベルを使用して動的コンパウンド・ステートメントで宣言されている SQL 変数を修飾したり、ラベルを LEAVE ステートメントに指定したりすることができます。終了ラベルを指定する場合、そのラベルは開始ラベルと同じでなければなりません。

ATOMIC

ATOMIC は、コンパウンド・ステートメントでエラーが起こった場合、そのコンパウンド・ステートメント内の SQL ステートメントがすべてロールバックされ、以降の SQL ステートメントは処理されないことを指示します。

SQL-routine-statement

動的コンパウンド・ステートメント内で使用する SQL ステートメントを指定します。SQL 関数または SQL メソッド内の動的コンパウンド・ステートメントでは、RETURN ステートメントも使用できます。コンパウンド SQL 内のニックネームに対する検索更新、検索削除、挿入、またはマージ操作はサポートされません。

SQL-variable-declaration

動的コンパウンド・ステートメントに対してローカルである変数を宣言します。

SQL-variable-name

ローカル変数の名前を定義します。DB2 は SQL 変数をすべて大文字に変換します。この名前を以下のものと同じにすることはできません。

- コンパウンド・ステートメント内の別の SQL 変数
- パラメーター名

SQL 変数および列参照と同じ名前の ID が SQL ステートメントに含まれている場合、DB2 はその ID を列と解釈します。

data-type

変数のデータ・タイプを指定します。XML タイプはサポートされず (SQLSTATE 429BB)、配列タイプはサポートされません (SQLSTATE 42815)。

DEFAULT *default-values* または **NULL**

SQL 変数のデフォルトを定義します。動的コンパウンド・ステートメントが呼び出されると、この変数は初期化されます。デフォルト値が指定されていない場合、変数は NULL に初期化されます。

condition-declaration

条件名および対応する SQLSTATE 値を宣言します。

condition-name

条件の名前を指定します。条件名は、それが宣言されるコンパウンド・ステートメント内で固有でなければなりません。ただし、そのようなコンパウンド・ステートメント内でネストされたコンパウンド・ステートメント内での宣言は例外です (SQLSTATE 42734)。条件名は、それが宣言されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です (SQLSTATE 42737)。

FOR SQLSTATE *string-constant*

条件に関連する SQLSTATE を指定します。*string-constant* は引用符で囲んだ 5 文字で指定しなければなりません、'00000' にすることはできません。

注

- 動的コンパウンド・ステートメントは、DB2 によって単一ステートメントとしてコンパイルされます。このステートメントは、小さな制御フロー・ロジックを含む短いスクリプトに有効ですが、大きな意味を持つデータ・フローには有効ではありません。複雑な制御フローのネストまたは条件処理が必要な大きな構成の場合、SQL プロシージャの使用をお勧めします。SQL プロシージャの使用法の詳細については、『CREATE PROCEDURE』を参照してください。
- コンパウンド・ステートメント内で呼び出されるプロシージャは、COMMIT または ROLLBACK ステートメントを発行できません (SQLSTATE 42985)。
- 表アクセスの制限:**

プロシージャが READS SQL DATA または MODIFIES SQL DATA として定義されている場合は、プロシージャ内のステートメントは、このプロシージャを呼び出したコンパウンド・ステートメントによって変更される表にアクセスすることはできません (SQLSTATE 57053)。プロシージャが MODIFIES SQL DATA として定義されている場合は、プロシージャ内のステートメントは、このプロシージャを呼び出したコンパウンド・ステートメントによって読み取られるまたは変更される表を変更できません (SQLSTATE 57053)。

例

例 1:

コンパウンド SQL (動的)

この例では、データ・クレンジングを行うために、データウェアハウジング・シナリオでインライン SQL PL を使用する方法を示します。

この例には、3 つの表があります。"target" 表には、クレンジングされたデータが入ります。"except" 表にはクレンジングできない行 (例外) が保管され、"source" 表にはクレンジングするロー・データが入ります。

データを分類して変更するために、"discretize" という単純な SQL 関数を使用されます。これは、不良データの場合はすべて NULL を戻します。次いで、動的コンパウンド・ステートメントがデータをクレンジングします。このステートメントは、FOR ループでソース表の中のすべての行を処理し、"discretize" 関数の結果に従って、現在行を "target" 表と "except" 表のどちらに挿入するのかを決定します。この技法を使用して、より複雑なメカニズム (複数ステージのクレンジング) にすることができます。

SQL プロシージャ、その他の任意のプロシージャ、またはホスト言語のアプリケーションで、同じコードを作成できます。ただし、動的コンパウンド・ステートメントには独特の利点があります。つまり、FOR ループでカーソルが開くことはなく、単一行挿入も実際の単一行挿入ではありません。実際には、共用選択からの複数表挿入という効果的な論理になっています。

これは、動的コンパウンドを単一ステートメントとしてコンパイルすることによって達成されます。ビューを使用する照会に統合され、照会コンテキスト内で全体としてコンパイルおよび最適化される本体を持つビューと同様に、DB2 オプティマイザは、制御フローとデータ・フローの両方をコンパイルおよび最適化します。したがって、全体の論理は、DB2 の実行時間内で実行されます。プロシージャの場合とは異なり、DB2 のコア・エンジンの外部に移動されるデータはありません。

最初のステップでは必要な表を作成します。

```
CREATE TABLE target
(pk INTEGER NOT NULL
PRIMARY KEY, c1 INTEGER)
```

クレンジングされたデータを入れるための TARGET という表が作成されます。

```
CREATE TABLE except
(pk INTEGER NOT NULL
PRIMARY KEY, c1 INTEGER)
```

例外を入れるための EXCEPT という表が作成されます。

```
CREATE TABLE source
(pk INTEGER NOT NULL
PRIMARY KEY, c1 INTEGER)
```

クレンジングするデータを保持する SOURCE という表が作成されます。

次に、[0..1000] の範囲外にあるすべての値を取り除き、それらの値を 10 のステップに整列させることによってデータをクレンジングするための、"discretize" 関数を作成します。

```
CREATE FUNCTION discretize(raw INTEGER) RETURNS INTEGER
RETURN CASE
  WHEN raw < 0 THEN CAST(NULL AS INTEGER)
  WHEN raw > 1000 THEN NULL
  ELSE ((raw / 10) * 10) + 5
END
```

次いで、次のようにして値を挿入します。

```
INSERT INTO source (pk, c1)
VALUES (1, -5),
       (2, NULL),
       (3, 1200),
       (4, 23),
       (5, 10),
       (6, 876)
```

次のようにして関数を呼び出します。

```
BEGIN ATOMIC
FOR row AS
  SELECT pk, c1, discretize(c1) AS d FROM source
DO
  IF row.d is NULL THEN
    INSERT INTO except VALUES(row.pk, row.c1);
  ELSE
    INSERT INTO target VALUES(row.pk, row.d);
  END IF;
END FOR;
END
```

次のようにして結果をテストできます。

```
SELECT * FROM except ORDER BY 1
PK      C1
-----
      1      -5
      2       -
      3     1200
3 record(s) selected.
```

```
SELECT * FROM target ORDER BY 1
PK      C1
-----
      4      25
      5      15
      6     875
3 record(s) selected.
```

最後のステップとして、次のようにしてクリーンアップを行います。

```
DROP FUNCTION discretize
DROP TABLE source
DROP TABLE target
DROP TABLE except
```

コンパウンド SQL (組み込み)

1 つ以上の異なる SQL ステートメント (サブステートメント) を結合して、1 つの実行可能なブロックにします。

コンパウンド SQL (組み込み)

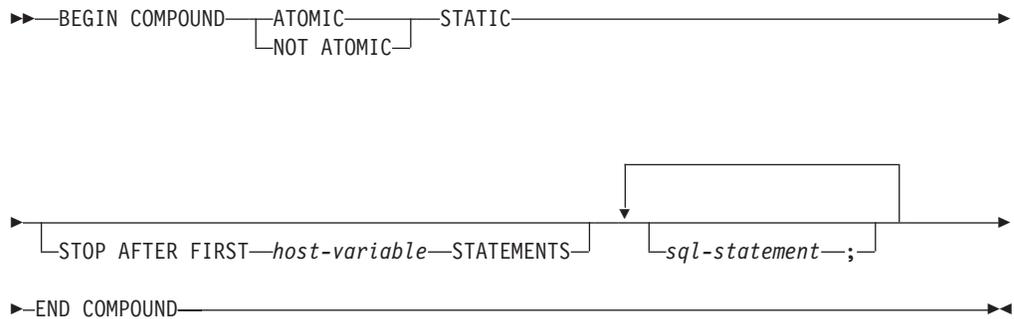
呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。コンパウンド SQL ステートメント構成全体は、動的に準備できない実行可能ステートメントです。このステートメントは REXX ではサポートされません。

許可

組み込みコンパウンド・ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID によって保持される特権には、コンパウンド・ステートメントに組み込まれている SQL ステートメントを呼び出すために必要なすべての特権が含まれていなければなりません。

構文



説明

ATOMIC

コンパウンド SQL ステートメント内のいずれかのサブステートメントが失敗した場合に、正常なサブステートメントによる変更も含め、サブステートメントによってデータベースに対して行われた変更すべてを取り消すことを指定します。

NOT ATOMIC

サブステートメントのエラーには関係なく、他のサブステートメントによってデータベースに対して行われた変更をコンパウンド SQL ステートメントが取り消さないことを指定します。

STATIC

すべてのサブステートメントに対する入力変数が、その当初の値を保持することを指定します。例えば、

```
SELECT ... INTO :abc ...
```

上記のステートメントの後に、次のステートメントが続いていると想定します。

```
UPDATE T1 SET C1 = 5 WHERE C2 = :abc
```

この UPDATE ステートメントは、SELECT INTO の後に続く値ではなく、コンパウンド SQL ステートメントの実行開始時の :abc の値が使用されます。

1 つの同じ変数が複数のサブステートメントによって設定される場合、コンパウンド SQL ステートメントの後のその変数の値は、最後のサブステートメントで設定された値になります。

注: 非静的動作はサポートされません。つまり、サブステートメントは、順次ではない方法で実行されているものとして見る必要があり、サブステートメントに相互関係があってはなりません。

STOP AFTER FIRST

特定の数のサブステートメントだけを実行することを指定します。

host-variable

実行されるサブステートメントの数を指定する短精度整数。

STATEMENTS

STOP AFTER FIRST *host-variable* 節を完結します。

sql-statement

組み込み静的コンパウンド SQL ステートメントには、以下のものを除くすべての実行可能ステートメントを含めることができます。

CALL	FETCH
CLOSE	OPEN
CONNECT	PREPARE
Compound SQL	RELEASE (Connection)
DESCRIBE	ROLLBACK
DISCONNECT	SET CONNECTION
EXECUTE IMMEDIATE	SET variable

注: コンパウンド SQL では、INSERT、UPDATE、および DELETE でのニックネームの使用がサポートされません。

COMMIT ステートメントを含める場合、それは最後のサブステートメントでなければなりません。COMMIT がこの位置にある場合には、STOP AFTER FIRST *host-variable* STATEMENTS 節ですべてのサブステートメントが実行されるわけではないことが指定されている場合でも、その COMMIT は発行されます。例えば、100 個のサブステートメントのあるコンパウンド SQL ブロックで、COMMIT が最後のサブステートメントであるとし、STOP AFTER FIRST STATEMENTS 節で 50 個のサブステートメントしか実行できないことが指定されている場合、COMMIT は 51 番目のサブステートメントになります。

CONNECT TYPE 2 を使用している場合や、XA 分散トランザクション処理環境で実行している場合に、COMMIT を組み込むと、エラーが戻されます (SQLSTATE 25000)。

規則

- DB2® Connect™ では、コンパウンド SQL ブロックの LOB 列を選択する SELECT ステートメントは、サポートされていません。
- コンパウンド SQL ステートメント内にホスト言語コードを使用することはできません。すなわち、コンパウンド SQL ステートメントを構成するサブステートメント相互間にホスト言語コードを使用することはできません。
- NOT ATOMIC コンパウンド SQL ステートメントのみが、DB2 Connect により受け入れられます。

コンパウンド SQL (組み込み)

- コンパウンド SQL ステートメントはネストできません。
- 準備済み COMMIT ステートメントは、ATOMIC コンパウンド SQL ステートメントでは許可されていません。

注

コンパウンド SQL ステートメント全体に対して、1 つの SQLCA が戻されます。その SQLCA の情報の多くは、最後のサブステートメントの処理時にアプリケーション・サーバーによって設定された値を反映しています。例えば、次のようになります。

- 通常、SQLCODE および SQLSTATE は、最後のサブステートメントに関するものです (例外については次の点で説明します)。
- 「no data found (データが見つからない)」の警告 (SQLSTATE '02000') が戻されると、その警告には他の警告よりも高い優先順位が与えられ、WHENEVER NOT FOUND 例外を立ち上げることができるようになります。(これは、最終的にアプリケーションに戻される SQLCA 内の SQLCODE、SQLERRML、SQLERRMC、および SQLERRP の各フィールドが、「no data found」の警告を引き起こしたサブステートメントによるものであることを意味しています。コンパウンド SQL ステートメントに複数の「no data found」の警告が生じた場合、戻されるフィールドは最後のサブステートメントに関するフィールドです。)
- SQLWARN 標識は、すべてのサブステートメントに対する標識の累計になります。

NOT ATOMIC コンパウンド SQL の実行の過程で 1 つまたは複数のエラーが発生し、その中に重大なエラーが含まれていない場合には、SQLERRMC には、それらのエラーのうち最大 7 つに関する情報が入れられます。SQLERRMC の最初のトークンは、発生したエラーの合計数を示します。残りのトークンには、コンパウンド SQL ステートメント内でエラーが生じたサブステートメントの順序位置と SQLSTATE が含まれます。これは、次の形式の文字ストリングです。

nnnXssscccc

X で始まるサブストリングは最大 6 回まで繰り返され、各ストリング・エレメントは、次のように定義されます。

nnn エラーが生じたステートメントの合計数。(数が 999 を超えると、カウントは 0 から再び始まります。) このフィールドは左寄せされ、空白で埋められます。

X トークン区切り記号 X'FF'。

sss エラーが生じたステートメントの順序位置。(数が 999 を超えると、カウントは 0 から再び始まります。) 例えば、最初のステートメントが失敗した場合、このフィールドには数字の 1 が左寄せで入れられます ('1')。

cccc エラーの SQLSTATE。

2 番目の SQLERRD フィールドには、エラーが生じたステートメント (負の SQLCODE が戻される) の数が入れられます。

SQLCA の 3 番目の SQLERRD フィールドは、すべてのサブステートメントによって影響を受けた行の数の累計です。

SQLCA の 4 番目の SQLERRD は、正常に実行されたサブステートメントの数を示します。例えば、コンパウンド SQL ステートメント内の 3 番目のサブステートメントが失敗した場合、4 番目の SQLERRD フィールドは 2 に設定され、2 つのサブステートメントが正常に処理された後でエラーが発生したことを示します。

SQLCA の 5 番目の SQLERRD フィールドは、制約活動を引き起こしたすべてのサブステートメントに対して参照整合性制約を課すことによって更新または削除された行の数の累計です。

例

例 1: C プログラムで、ACCOUNTS 表と TELLERS 表の両方を更新するコンパウンド SQL ステートメントを発行します。ステートメントのいずれかにエラーがある場合は、すべてのステートメントの結果を取り消します (ATOMIC)。エラーがない場合は、現行の作業単位をコミットします。

```
EXEC SQL BEGIN COMPOUND ATOMIC STATIC
  UPDATE ACCOUNTS SET ABALANCE = ABALANCE + :delta
  WHERE AID = :aid;
  UPDATE TELLERS SET TBALANCE = TBALANCE + :delta
  WHERE TID = :tid;
  INSERT INTO TELLERS (TID, BID, TBALANCE) VALUES (:i, :branch_id, 0);
  COMMIT;
END COMPOUND;
```

例 2: C プログラムで、データベースに 10 行のデータを挿入します。ホスト変数 :nbr の値は 10 であり、また S1 は準備済みの INSERT ステートメントであると想定します。さらに、エラーに関係なくすべての挿入を試行するものとします (NOT ATOMIC)。

```
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC STOP AFTER FIRST :nbr STATEMENTS
  EXECUTE S1 USING DESCRIPTOR :sqlda0;
  EXECUTE S1 USING DESCRIPTOR :sqlda1;
  EXECUTE S1 USING DESCRIPTOR :sqlda2;
  EXECUTE S1 USING DESCRIPTOR :sqlda3;
  EXECUTE S1 USING DESCRIPTOR :sqlda4;
  EXECUTE S1 USING DESCRIPTOR :sqlda5;
  EXECUTE S1 USING DESCRIPTOR :sqlda6;
  EXECUTE S1 USING DESCRIPTOR :sqlda7;
  EXECUTE S1 USING DESCRIPTOR :sqlda8;
  EXECUTE S1 USING DESCRIPTOR :sqlda9;
END COMPOUND;
```

コンパウンド SQL (プロシージャ)

コンパウンド・ステートメントは、別のステートメントを SQL プロシージャにグループ化します。コンパウンド・ステートメントで宣言できるのは、SQL 変数、カーソル、および条件処理ルーチンです。

呼び出し

このステートメントは、SQL プロシージャに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

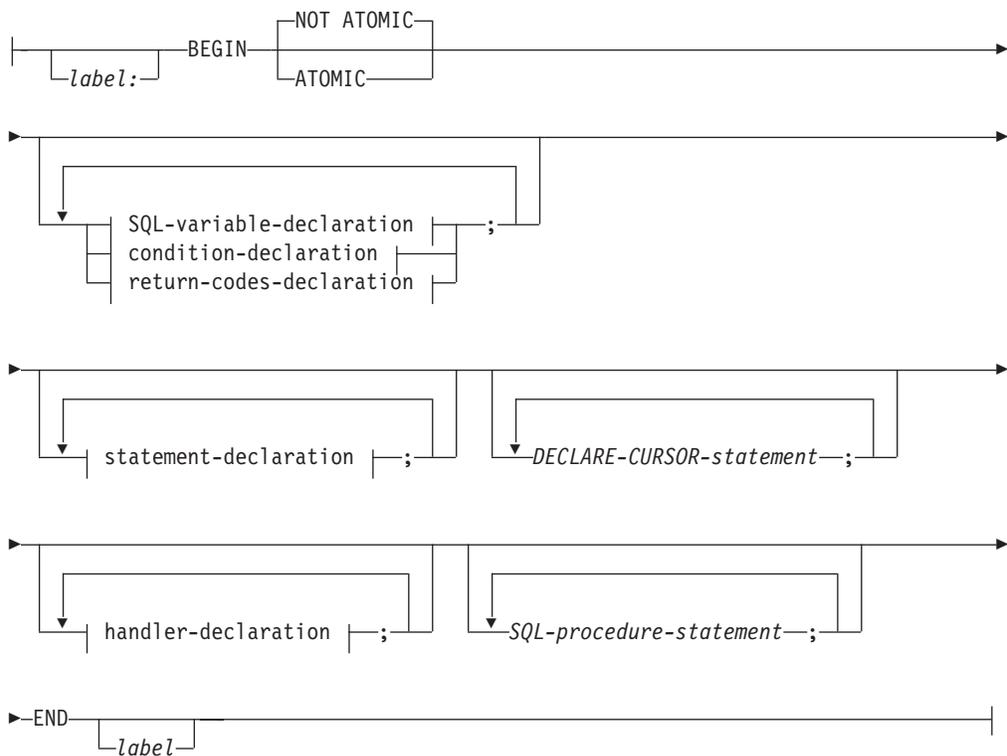
コンパウンド SQL (プロシージャー)

許可

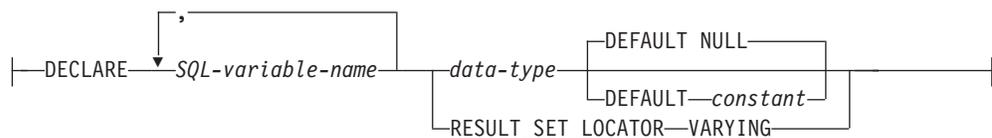
プロシージャー・コンパウンド・ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID によって保持される特権には、コンパウンド・ステートメントに組み込まれている SQL ステートメントを呼び出すために必要なすべての特権が含まれていなければなりません。カーソルの使用に必要な許可については、『DECLARE CURSOR』を参照してください。

構文

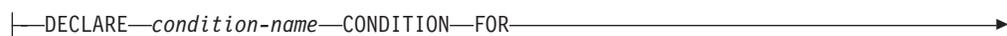
procedure-compound-statement:

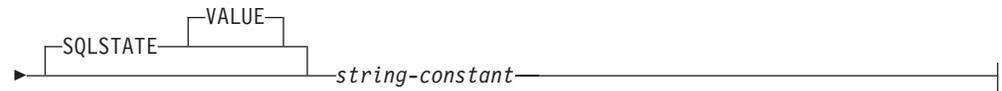


SQL-variable-declaration:



condition-declaration:

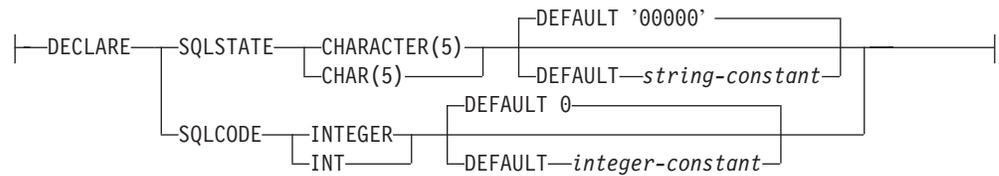




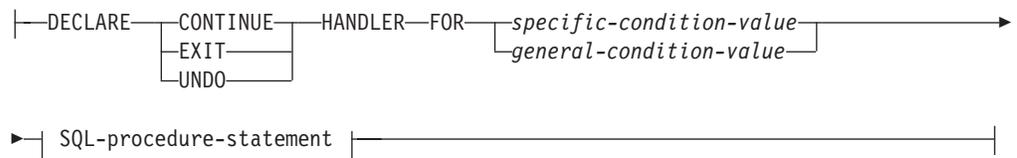
statement-declaration:



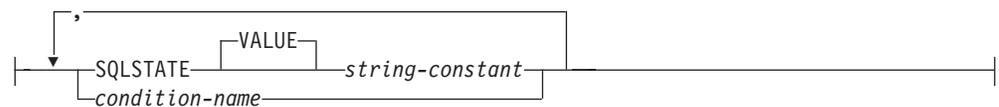
return-codes-declaration:



handler-declaration:



specific-condition-value:



general-condition-value:



SQL-procedure-statement:



説明

label

コード・ブロックのラベルを定義します。開始ラベルを指定した場合、そのラベ

コンパウンド SQL (プロシージャ)

ルを使用して、コンパウンド・ステートメントで宣言する SQL 変数を修飾することができます。また、開始ラベルは LEAVE ステートメントで指定することもできます。終了ラベルを指定する場合、そのラベルは開始ラベルと同じでなければなりません。

ATOMIC または NOT ATOMIC

ATOMIC は、コンパウンド・ステートメントで未処理の例外条件が発生したときに、そのコンパウンド・ステートメント内のすべての SQL ステートメントをロールバックします。NOT ATOMIC は、コンパウンド・ステートメントで未処理の例外条件が発生しても、そのコンパウンド・ステートメントをロールバックしません。

SQL-variable-declaration

コンパウンド・ステートメントに対してローカルな変数を宣言します。

SQL-variable-name

ローカル変数の名前を定義します。DB2 は SQL 変数をすべて大文字に変換します。この名前は、同じコンパウンド・ステートメント内にある別の SQL 変数と同じにすることはできず、パラメーター名と同じにすることもできません。SQL 変数名は、列名と同じにすることはできません。SQL 変数および列参照と同じ名前の ID が SQL ステートメントに含まれている場合、DB2 はその ID を列と解釈します。変数が宣言されているコンパウンド・ステートメントにラベルを付けると、変数の使用法をラベルで修飾できます。たとえば、ラベル C を付けたコンパウンド・ステートメント中で変数 V を宣言すると、その変数を C.V として参照できます。

data-type

変数のデータ・タイプを指定します。REFERENCE およびユーザー定義構造化タイプはサポートされません (SQLSTATE 429BB)。

DEFAULT *constant* または NULL

SQL 変数のデフォルトを定義します。SQL プロシージャが呼び出された時点で、この変数は初期化されます。デフォルト値が指定されていない場合、変数は NULL に初期化されます。

RESULT_SET_LOCATOR VARYING

結果セット・ロケータ変数のデータ・タイプを指定します。

condition-declaration

条件名および対応する SQLSTATE 値を宣言します。

condition-name

条件の名前を指定します。条件名は、それが宣言されるコンパウンド・ステートメント内で固有でなければなりません。ただし、そのようなコンパウンド・ステートメント内でネストされたコンパウンド・ステートメント内での宣言は例外です (SQLSTATE 42734)。条件名は、それが宣言されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です (SQLSTATE 42737)。

FOR SQLSTATE *string-constant*

条件に関連付ける SQLSTATE を指定します。 *string-constant* は単一引用符で囲まれている 5 つの文字として指定しなければならず、 '00000' にすることはできません。

statement-declaration

コンパウンド・ステートメントの 1 つ以上のローカルの名前を宣言します。ステートメント名は、同じコンパウンド・ステートメント内の別のステートメント名と同じにすることはできません。

return-codes-declaration

SQLSTATE および SQLCODE という特殊変数を宣言します。これらの変数は、SQL ステートメントの処理後に戻される値に自動的に設定されます。SQLSTATE および SQLCODE 変数は両方とも、SQL プロシージャー本体の最外部のコンパウンド・ステートメントでしか宣言できません。これらの変数は、SQL プロシージャーごとに一度しか宣言できません。

declare-cursor-statement

プロシージャー本体にカーソルを宣言します。各カーソルは、それが宣言されたコンパウンド・ステートメント内で固有の名前を持たなければなりません。ただし、そのようなコンパウンド・ステートメント内でネストされたコンパウンド・ステートメント内での宣言は例外です (SQLSTATE 42734)。カーソルは、それが宣言されたコンパウンド・ステートメント内でのみ参照が可能です。コンパウンド・ステートメント内でネストされたコンパウンド・ステートメントも同様です (SQLSTATE 34000)。

カーソルをオープンする場合は OPEN ステートメントを、カーソルを使用して行を読み取る場合は FETCH ステートメントを使用します。SQL プロシージャーからクライアント・アプリケーションに結果セットを戻す場合、WITH RETURN 節を使用してカーソルを宣言しなければなりません。以下の例では、クライアント・アプリケーションに結果セットを 1 つ戻します。

```
CREATE PROCEDURE RESULT_SET()
LANGUAGE SQL
RESULT SETS 1
BEGIN
  DECLARE C1 CURSOR WITH RETURN FOR
  SELECT id, name, dept, job
  FROM staff;
  OPEN C1;
END
```

注: 結果セットを処理する場合、DB2 コール・レベル・インターフェース (DB2 コール・レベル・インターフェース)、ODBC (Java Database Connectivity)、JDBC (Java Database Connectivity)、Java Embedded SQL (SQLJ) のいずれかのアプリケーション・プログラミング・インターフェースを使用して、クライアント・アプリケーションを作成しなければなりません。

カーソルの宣言について詳しくは、『DECLARE CURSOR』を参照してください。

handler-declaration

ハンドラー (つまり、コンパウンド・ステートメントで例外または完了条件が発生した場合に実行する *SQL-procedure-statement*) を指定します。

SQL-procedure-statement は、ハンドラーが制御を受け取る際に実行するステートメントです。

コンパウンド SQL (プロシージャー)

ハンドラーがアクティブになる対象は、そのハンドラーが宣言されているコンパウンド・ステートメント (ネストされたコンパウンド・ステートメントを含む) 中の、*handler-declarations* の集合の後の *SQL-procedure-statements* の集合です。

条件処理ルーチンには以下の 3 つのタイプがあります。

CONTINUE

ハンドラーが正常に呼び出された後に、例外が発生したステートメントの後の SQL ステートメントに制御が戻されます。例外が発生したエラーが FOR、IF、CASE、WHILE、または REPEAT ステートメント (ただし、それらのいずれかのステートメントの中の SQL-procedure-statement は除く) の場合、制御は、END FOR、END IF、END CASE、END WHILE、または END REPEAT の後のステートメントに戻されます。

EXIT

ハンドラーが正常に呼び出された後に、ハンドラーを宣言したコンパウンド・ステートメントの最後に制御が戻されます。

UNDO

ハンドラーが呼び出される前に、コンパウンド・ステートメントで行われたあらゆる SQL の変更がロールバックされます。ハンドラーが正常に呼び出された後に、ハンドラーを宣言したコンパウンド・ステートメントの最後に制御が戻されます。UNDO を指定する場合は、ハンドラーを宣言しているコンパウンド・ステートメントを ATOMIC にしなければなりません。

以下のようなハンドラーを活動化する条件を *handler-declaration* 中に定義します。

specific-condition-value

ハンドラーが、特定条件処理ルーチンであることを指定します。

SQLSTATE *string*

ハンドラーが呼び出される SQLSTATE を指定します。SQLSTATE 値の先頭 2 文字は "00" にしないでください。

condition-name

ハンドラーが呼び出される条件名を指定します。条件名は、条件宣言であらかじめ定義していなければなりません。

general-condition-value

ハンドラーが、一般条件処理ルーチンであることを指定します。

SQLEXCEPTION

例外条件が発生した場合に呼び出されるハンドラーを指定します。例外条件は、最初の 2 文字が "00"、"01"、または "02" ではない SQLSTATE 値で表されます。

SQLWARNING

警告条件が発生した場合に呼び出されるハンドラーを指定します。警告条件は、最初の 2 文字が "01" の SQLSTATE 値で表されます。

NOT FOUND

NOT FOUND 条件が発生した場合に呼び出されるハンドラーを指定します。NOT FOUND 条件は、最初の 2 文字が "02" の SQLSTATE 値で表されます。

SQL-procedure-statement

SQL プロシージャー・ステートメントを指定します。

label

SQL プロシージャー・ステートメントのラベルを指定します。ラベルは、リスト内でネストされたコンパウンド・ステートメントを含め、SQL プロシージャー・ステートメントのリスト内でユニークでなければなりません。ネストされていないコンパウンド・ステートメントは、同じラベルを使用できることに注意してください。SQL プロシージャー・ステートメントのリストは、おそらく SQL 制御ステートメントの中にあります。

SQL-statement

SQL プロシージャーの本体には、以下を除き、実行可能なすべての SQL ステートメントを入れることができます。

- ALTER
- CONNECT
- 索引、表、またはビュー以外のオブジェクトの CREATE
- DESCRIBE
- DISCONNECT
- 索引、表、またはビュー以外のオブジェクトの DROP
- FLUSH EVENT MONITOR
- REFRESH TABLE
- RELEASE (接続のみ)
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- SET CONNECTION
- SET INTEGRITY
- SET PASSTHRU
- SET SERVER OPTION

以下のステートメントは、SQL プロシージャーの有効範囲内でのみサポートされます。

- ALLOCATE CURSOR
- ASSOCIATE LOCATORS
- CASE
- GOTO
- LOOP
- コンパウンド SQL (プロシージャー)
- REPEAT

- RESIGNAL

規則

- ATOMIC コンパウンド・ステートメントはネストできません。
 - ハンドラーの宣言には、以下の規則が適用されます。
 - ハンドラーの宣言では、同一の *condition-name* または *SQLSTATE* 値を複数回含めることはできません。また、*SQLSTATE* 値と、その同じ *SQLSTATE* 値を表す *condition-name* を含めることもできません。
 - コンパウンド・ステートメント中で複数の条件処理ルーチンが宣言されている場合、以下の規則が適用されます。
 - 2 つのハンドラー宣言に、同一の一般条件カテゴリ (SQLEXCEPTION、SQLWARNING、NOT FOUND) を指定することはできません。
 - 2 つのハンドラー宣言に、同一の値を表す *SQLSTATE* 値または *condition-name* として、同一の特定条件カテゴリを指定することはできません。
 - 例外または完了条件が発生した場合、その条件に最も適したハンドラーが有効になります。以下の考慮事項に基づいて、最も適したハンドラーが判別されます。
 - ハンドラー宣言 *H* の有効範囲は、*H* のあるコンパウンド・ステートメント中に含まれるハンドラー宣言の後の *SQL-procedure-statement* のリストです。したがって、*H* の有効範囲には、条件処理ルーチン *H* の本体中に含まれるステートメントは入りません。つまり、条件処理ルーチンは条件処理ルーチン自体の本体中で生じる条件を処理できないこととなります。同様に、同一のコンパウンド・ステートメント中で *H1* と *H2* という 2 つのハンドラーが宣言されている場合、*H1* は *H2* の本体中で生じる条件を処理できず、*H2* は *H1* の本体中で生じる条件を処理できません。
 - 内側の有効範囲で宣言された *specific-condition-value* または *general-condition-value* *C* のハンドラーは、外側の有効範囲で宣言された *C* の別のハンドラーより優先します。
 - 条件 *C* に関する特定ハンドラーと、同じく *C* を処理する一般ハンドラーが、同一の有効範囲内で宣言されている場合は、特定ハンドラーの方が一般ハンドラーより優先します。
- 適したハンドラーのない例外条件が生じた場合は、失敗したステートメントに含まれる SQL プロシージャーは、未処理の例外条件で終了します。該当するハンドラーのない完了条件が生じた場合は、引き続き次の SQL ステートメントが実行されます。
- コミットまたはロールバック操作が生じた後に、SQL プロシージャー内でデータ・タイプ XML の変数またはパラメーターを参照することは、まずこれらの変数に新しい値を割り当ててからでなければサポートされません (SQLSTATE 560CE)。

注

- **XML 割り当て:** データ・タイプ XML のパラメーターおよび変数に対する割り当ては、値によって行われます。値によって XML 値の割り当てを行うときにはコピーが作成されます。これは、シーケンス内の各項目ごとに以下のアクションを効果的に実行することによって行われます。

- ノード・ツリー (ある場合) を、文書の順序を維持しつつ、元の XML 値からコピーする。
- 新しいノード ID を各ノードに割り当てる。
- ルート・ノードの親プロパティ (ある場合) を空に設定する。

CALL ステートメント内でデータ・タイプ XML のパラメーターを SQL プロシージャーに渡すことは、参照によって行われます。参照によって XML 値を渡すときには、入力ノード・ツリー (ある場合) が XML 引数から直接に使用されます。その際、文書の順序、元のノード ID、およびすべての親プロパティを含むすべてのプロパティが保持されます。

例

以下の処置を実行するコンパウンド・ステートメントが含まれている、プロシージャー本体を作成します。

1. SQL 変数を宣言します。
2. IN パラメーターによって判別される部門の従業員の給与を戻すカーソルを宣言します。SELECT ステートメントで、*salary* 列のデータ・タイプを DECIMAL から DOUBLE にキャストします。
3. 条件 NOT FOUND (ファイル終わり) に EXIT ハンドラーを宣言します。これにより、値 '6666' が OUT パラメーター *medianSalary* に割り当てられます。
4. 指定された部門の従業員の数を選択して SQL 変数 *numRecords* に入れます。
5. 50% + 1 の従業員が検索されるまで、WHILE ループのカーソルから行を取り出します。
6. 給与の中央値を戻します。

```
CREATE PROCEDURE DEPT_MEDIAN
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
LANGUAGE SQL
BEGIN
  DECLARE v_numRecords INTEGER DEFAULT 1;
  DECLARE v_counter INTEGER DEFAULT 0;
  DECLARE c1 CURSOR FOR
    SELECT CAST(salary AS DOUBLE) FROM staff
      WHERE DEPT = deptNumber
      ORDER BY salary;
  DECLARE EXIT HANDLER FOR NOT FOUND
    SET medianSalary = 6666;
  -- initialize OUT parameter
  SET medianSalary = 0;
  SELECT COUNT(*) INTO v_numRecords FROM staff
    WHERE DEPT = deptNumber;
  OPEN c1;
  WHILE v_counter < (v_numRecords / 2 + 1) DO
    FETCH c1 INTO medianSalary;
    SET v_counter = v_counter + 1;
  END WHILE;
  CLOSE c1;
END
```

以下の例は、RESIGNAL の結果として別の条件から UNDO ハンドラーが活動化される場合を仮定して、実行の流れを図示しています。

```
CREATE PROCEDURE A()
LANGUAGE SQL
CS1: BEGIN ATOMIC
```

コンパウンド SQL (プロシージャー)

```
DECLARE C CONDITION FOR SQLSTATE '12345';
DECLARE D CONDITION FOR SQLSTATE '23456';

DECLARE UNDO HANDLER FOR C
H1: BEGIN
-- Rollback after error, perform final cleanup, and exit
-- procedure A.

-- ...

-- When this handler completes, execution continues after
-- compound statement CS1; procedure A will terminate.
END;

-- Perform some work here ...
CS2: BEGIN
DECLARE CONTINUE HANDLER FOR D
H2: BEGIN
-- Perform local recovery, then forward the error
-- condition to the outer handler for additional
-- processing.

-- ...

RESIGNAL C; -- will activate UNDO handler H1; execution
-- WILL NOT return here. Any local cursors
-- declared in H2 and CS2 will be closed.
END;

-- Perform some more work here ...

-- Simulate raising of condition D by some SQL statement
-- in compound statement CS2:
SIGNAL D; -- will activate H2
END;
END
```

CONNECT (タイプ 1)

CONNECT (タイプ 1) ステートメントは、リモート作業単位の規則に従って、指定したアプリケーション・サーバーにアプリケーション・プロセスを接続します。

1 つのアプリケーション・プロセスは、一時点で 1 つのアプリケーション・サーバーにのみ接続できます。これは、*現行サーバー* と呼ばれます。デフォルトのアプリケーション・サーバーは、アプリケーション・リクエスターの初期設定時に確立されます。暗黙接続が使用可能な場合にアプリケーション・プロセスが開始されると、暗黙でデフォルトのアプリケーション・プロセスに接続されます。そのアプリケーション・プロセスで CONNECT ステートメントを発行することによって、それとは別のアプリケーション・サーバーに明示的に接続することもできます。接続は、CONNECT RESET ステートメント、または DISCONNECT ステートメントが出されるまで、あるいは別の CONNECT ステートメントによってアプリケーション・サーバーが変更されるまで継続します。

呼び出し

対話式 SQL 機能には外見上対話式的の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

許可

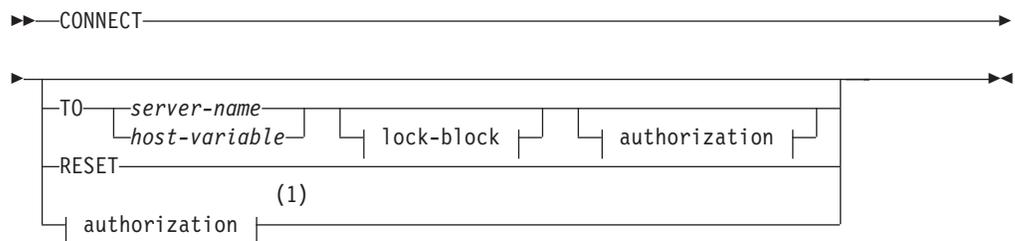
CONNECT 処理では、2 つのレベルのアクセス制御を通過します。接続を成功させるには、両方のレベルで条件を満たす必要があります。

第 1 レベルのアクセス制御は認証です。ここで、接続に関連付けられたユーザー ID が、サーバーにセットアップされた認証方式にしたがって、正常に認証されなければなりません。認証に成功すると、サーバーで有効な認証プラグインにしたがって、接続ユーザー ID から DB2 許可 ID が導出されます。続いて、DB2 許可 ID は、この接続における第 2 レベルのアクセス制御、すなわち許可をパスしなければなりません。パスするには、以下の権限のうち少なくとも 1 つが、この許可 ID に付与されている必要があります。

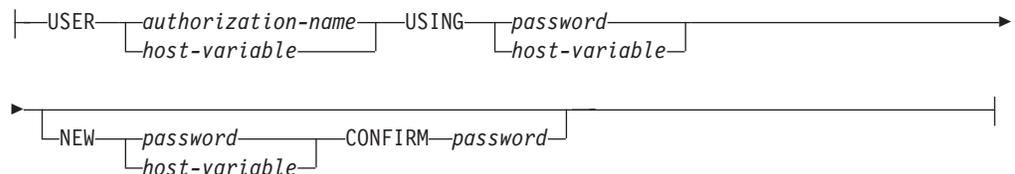
- CONNECT 権限
- DBADM 権限
- SYSADM 権限
- SYSCTRL 権限
- SYSMANT 権限

注: パーティション・データベースの場合、ユーザーとグループの定義は、データベース・パーティションのすべてにわたって同一である必要があります。

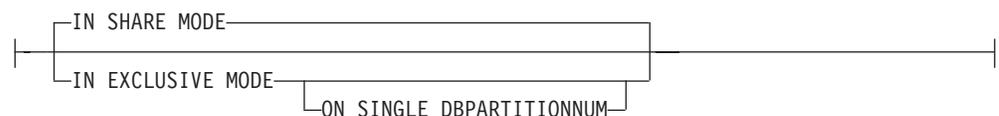
構文



許可:



lock-block:



注:

- 1 この形式は、暗黙の接続が使用可能な場合にのみ有効です。

説明

CONNECT (オペランドなし)

現行サーバーに関する情報を戻します。この情報は、「正常に接続された場合」の項に説明されているとおり、SQLCA の SQLERRP フィールドに戻されません。

接続状態が存在する場合、許可 ID とデータベース別名が、SQLCA の SQLERRMC フィールドに入れられます。権限 ID が 8 バイトを超える場合、これは 8 バイトに切り捨てられ、SQLCA の SQLWARN0 および SQLWARN1 フィールドにそれぞれ W と A のフラグが付きます。データベース構成パラメーター `dyn_query_mgmt` が使用可能に設定されている場合、SQLCA の SQLWARN0 および SQLWARN7 フィールドにはそれぞれ W と E のフラグが付きます。

接続が存在せず、暗黙接続が可能な場合は、暗黙接続が試みられます。暗黙接続が使用可能でない場合、この試みはエラーになります (既存の接続がない)。接続がない場合、SQLERRMC フィールドはブランクになります。

アプリケーション・サーバーのテリトリー・コードとコード・ページは、SQLERRMC フィールドに入れられます (正常に実行される CONNECT ステートメントの場合と同じ)。

この形式の CONNECT を使用する場合、

- アプリケーション・プロセスが接続可能状態である必要はありません。
- 接続されている場合、接続状態は変わりません。
- 接続されておらず、暗黙接続が使用可能な場合は、デフォルトのアプリケーション・サーバーとの接続が行われます。この場合、正常に実行される CONNECT ステートメントの場合と同様に、アプリケーション・サーバーの国または地域別コードとコード・ページが、SQLERRMC フィールドに入れられます。
- 未接続で暗黙的接続が不能な場合、アプリケーション・プロセスは未接続のままになります。
- カーソルをクローズしません。

TO *server-name* または *host-variable*

server-name (サーバー名) またはそのサーバー名が入る *host-variable* (ホスト変数) を指定することによって、アプリケーション・サーバーを指定します。

host-variable (ホスト変数) を指定する場合、それは、長さ属性が 8 以下の文字ストリング変数でなければならず、標識変数を含めることはできません。その *host-variable* に入る *server-name* は、左寄せする必要があり、引用符で区切ることはできません。

server-name は、アプリケーション・サーバーを指定するデータベース別名である点に注意してください。この名前は、アプリケーション・リクエストのローカル・ディレクトリーにリストされている必要があります。

CONNECT ステートメントが実行される時点で、アプリケーション・プロセスは接続可能状態でなければなりません。

正常に接続された場合

CONNECT ステートメントが正常に実行された場合、次のようになります。

- オープンされていたカーソルはすべてクローズされ、準備済みのステートメントはすべて破棄されて、以前のアプリケーション・サーバーからのすべてのロックが解放されます。
- アプリケーション・プロセスはそれ以前のアプリケーション・サーバー (ある場合) から切断され、指定されたアプリケーション・サーバーに接続されます。
- そのアプリケーション・サーバーの実際の名前 (別名ではなく) が CURRENT SERVER 特殊レジスターに入れられます。
- アプリケーション・サーバーに関する情報が、SQLCA の SQLERRP フィールドに入れられます。アプリケーション・サーバーが IBM 製品の場合、その情報は *pppvrrm* の形式です。ここで、
 - *ppp* は、以下のように製品を示します。
 - DB2 for z/OS の場合は DSN
 - DB2 Server for VSE & VM の場合は ARI
 - DB2 for i5/OS の場合は QSQ
 - DB2 Database for Linux, UNIX, and Windows の場合は SQL
 - *vv* は、2 桁のバージョン ID です (08 など)。
 - *rr* は、2 桁のリリース ID です (01 など)。
 - *m* は、1 文字の修正レベル ID です (0 など)。

DB2 Database for Linux, UNIX, and Windows のこのリリース (バージョン 9.5) は、SQL09050 と表記されます。

- SQLCA の SQLERRMC フィールドは、次の値を含む (X'FF' で区切って) ように設定されます。
 1. アプリケーション・サーバーの国または地域コード (DB2 Connect を使用している場合は空白)。
 2. アプリケーション・サーバーのコード・ページ (DB2 Connect を使用している場合は CCSID)。
 3. 権限 ID (最初の 8 バイトまで)。
 4. データベース別名。
 5. アプリケーション・サーバーのプラットフォーム・タイプ。現在識別される値は次のとおりです。

トークン

サーバー

QAS DB2 for System i

QDB2 DB2 for z/OS

QDB2/6000

DB2 Database for AIX

QDB2/HPUX

DB2 Database for HP-UX

QDB2/LINUX

DB2 Database for Linux

QDB2/NT

DB2 Database for Windows

QDB2/SUN

DB2 Database for Solaris Operating System

QSQLDS/VM

DB2 Server for VM

QSQLDS/VSE

DB2 Server for VSE

6. エージェント ID。これは、アプリケーションに代わってデータベース・マネージャー内で実行されるエージェントを指定します。このフィールドは、データベース・モニターによって戻される **agent_id** エレメントと同じです。
 7. エージェント索引。これは、エージェントの索引を識別し、サービスに使用されます。
 8. データベース・パーティション番号。非パーティション・データベースの場合は、この値は常に 0 です (存在する場合)。
 9. アプリケーション・クライアントのコード・ページ。
 10. パーティション・データベースのデータベース・パーティションの数。データベースを分散できない場合、値は 0 (ゼロ) になります。トークンは、バージョン 5 またはそれ以降の場合にのみ存在します。
- SQLCA の SQLERRD(1) フィールドは、アプリケーション・コード・ページからデータベース・コード・ページへの変換が行われる際に見込まれる混合文字データ (CHAR データ・タイプ) の長さの最大差を示します。0 または 1 の値は、長さが変わらないことを示し、1 より大きい値は長さが長くなることを、負の値は切り捨てが生じることを示します。
 - SQLCA の SQLERRD(2) フィールドは、データベース・コード・ページからアプリケーション・コード・ページへの変換が行われる際に見込まれる混合文字データ (CHAR データ・タイプ) の長さの最大差を示します。0 または 1 の値は、長さが変わらないことを示し、1 より大きい値は長さが長くなることを、負の値は切り捨てが生じることを示します。
 - SQLCA の SQLERRD(3) フィールドは、接続されているデータベースが更新可能か否かを示します。データベースは、最初は更新可能ですが、その許可 ID で更新を実行できないことが作業単位で明らかになると、読み取り専用に変更されます。この値は、次のいずれかです。
 - 1 - 更新可能
 - 2 - 読み取り専用
 - SQLCA の SQLERRD(4) フィールドは、接続の特定の特性を戻します。この値は、次のいずれかです。
 - 0** なし (1 フェーズ・コミットで更新元でもある下位レベル・クライアントから実行している場合のみ可能)。
 - 1** 1 フェーズ・コミット。
 - 2** 1 フェーズ・コミット。読み取り専用 (TP モニター環境にある DRDA1 データベースとの接続にのみ適用可能)。

- 3 2 フェーズ・コミット。
- SQLCA の SQLERRD(5) フィールドは、接続のための認証タイプを戻します。この値は、次のいずれかです。
 - 0 サーバーで認証された。
 - 1 クライアントで認証された。
 - 2 DB2 Connect を使用して認証された。
 - 4 暗号化を伴ってサーバーで認証された。
 - 5 暗号化を伴い、DB2 Connect を使用して認証された。
 - 7 外部 Kerberos セキュリティー機構を使用して認証された。
 - 9 外部 GSS API プラグイン・セキュリティ機構を使用して認証された。
 - 11 暗号化されたデータを受け入れるサーバーで認証された。
 - 255 認証は指定されていない。
- SQLCA の SQLERRD(6) フィールドは、データベースが分散されている場合に、接続されたデータベース・パーティションのデータベース・パーティション番号を戻します。区分化されていない場合、値 0 が戻されます。
- 正常に行われた接続の許可 ID の長さが 8 バイトを超える場合には、SQLCA の SQLWARN1 フィールドは A に設定されます。これは、切り捨てが発生したことを示します。SQLCA にある SQLWARN0 フィールドは、W に設定されて警告を示します。
- データベースでデータベース構成パラメーター `dyn_query_mgmt` が使用可能になっている場合には、SQLCA の SQLWARN7 フィールドは E に設定されます。SQLCA にある SQLWARN0 フィールドは、W に設定されて警告を示します。

正常に接続されなかった場合

CONNECT ステートメントが正常に実行されなかった場合、次のようになります。

- SQLCA の SQLERRP フィールドは、エラーを検出したアプリケーション・リクエスターのモジュール名に設定されます。モジュール名の最初の 3 文字は、製品を識別します。
- アプリケーション・プロセスが接続可能状態でないために CONNECT ステートメントがエラーになった場合、アプリケーション・プロセスの接続状態は変更されません。
- `server-name` がローカル・ディレクトリーのリストにないために CONNECT ステートメントがエラーになった場合は、エラー・メッセージ (SQLSTATE 08001) が出され、アプリケーション・プロセスの接続状態は変更されません。
 - アプリケーション・リクエスターがアプリケーション・サーバーに接続されなかった場合、アプリケーション・プロセスは接続されないままです。
 - アプリケーション・リクエスターがアプリケーション・サーバーにすでに接続されていた場合、アプリケーション・プロセスはそのアプリケーション

CONNECT (タイプ 1)

ン・サーバーに接続されたままです。それ以降のステートメントは、そのアプリケーション・サーバーで実行されます。

- その他の理由で CONNECT ステートメントが正常に実行されない場合、アプリケーション・プロセスは接続されていない状態になります。

IN SHARE MODE

データベースへの他の同時接続を可能にし、他のユーザーがデータベースに排他モードで接続しないようにします。

IN EXCLUSIVE MODE

排他ロックを保持するユーザーと同じ許可 ID を持つ場合を除き、複数の並行アプリケーション・プロセスがアプリケーション・サーバーで何らかの操作を実行するのを防止します。このオプションは DB2 Connect ではサポートされません。

ON SINGLE DBPARTITIONNUM

コーディネーターのデータベース・パーティションを排他モードで接続し、その他のすべてのデータベース・パーティションを共用モードで接続することを指定します。このオプションは、パーティション・データベースでのみ有効です。

RESET

アプリケーション・プロセスを現行サーバーから切断します。コミット操作が行われます。暗黙接続が使用可能な場合、アプリケーション・プロセスは SQL ステートメントが発行されるまで未接続のままになります。

USER *authorization-name* または *host-variable*

アプリケーション・サーバーに接続するユーザー ID を指定します。
host-variable (ホスト変数) を指定する場合、それは、標識変数を含まない文字ストリング変数でなければなりません。*host-variable* に入るユーザー ID は、左寄せする必要があり、引用符で区切ってはなりません。

USING *password* または *host-variable*

アプリケーション・サーバーに接続しようとするユーザー ID のパスワードを識別します。*password* または *host-variable* は最大 14 バイトまでの長さにすることができます。ホスト変数を指定する場合、それは、長さ属性が 14 以下の文字ストリング変数でなければならず、標識変数を含めることはできません。

NEW *password/host-variable* CONFIRM *password*

USER オプションによって識別されるユーザー ID に割り当てられる新規パスワードを識別します。*password* または *host-variable* は最大 14 バイトまでの長さにすることができます。ホスト変数を指定する場合、それは、長さ属性が 14 以下の文字ストリング変数でなければならず、標識変数を含めることはできません。パスワードが変更されるシステムは、ユーザー認証がセットアップされた方法によって異なります。この文節を使用して新規パスワードを割り当てることができるのは、次に挙げるサーバーの、次に示すリリース以降においてです。DB2® Universal Database™ バージョン 8 (AIX および Windows オペレーティング・システム上)、DB2 バージョン 9.1 フィックスパック 3 以降 (Linux オペレーティング・システム上)、DB2 for z/OS バージョン 7、および DB2 for i5/OS V6R1。Linux 上の DB2 データベース製品においてパスワード変更をサポ

ートするには、セキュリティー・プラグイン IBMOSchgpwdclient および IBMOSchgpwdserver を使用するように、DB2 インスタンスを構成する必要があります。

注

- アプリケーション・プロセスによって実行される最初の SQL ステートメントを CONNECT ステートメントにすることは望ましいことです。
- 異なるユーザー ID およびパスワードを用いて CONNECT ステートメントを現行のアプリケーション・サーバーに出すと、会話が割り振り解除され、割り振りし直されます。すべてのカーソルは、データベース・マネージャーによってクローズされます (WITH HOLD オプションが使用された場合は、カーソル位置は失われます)。
- 同じユーザー ID およびパスワードを使用して、CONNECT ステートメントが現行のアプリケーション・サーバーに出された場合、会話の割り振り解除および再割り振りは実行されません。この場合、カーソルはクローズされません。
- 複数パーティションのパーティション・データベース環境を使用するには、ユーザーまたはアプリケーションは db2nodes.cfg ファイルにリストされたデータベース・パーティションのいずれかに接続する必要があります。コーディネーターのパーティションと同じデータベース・パーティションをすべてのユーザーが使用しないようにする必要があります。
- *authorization-name* SYSTEM を CONNECT ステートメント内で明示的に指定することはできません。ただし、Windows オペレーティング・システムでは、ローカル・システム・アカウントの下で稼働しているローカル・アプリケーションは暗黙的にデータベースに接続することができ、その場合、ユーザー ID が SYSTEM になります。
- Windows Server に明示的に接続するときは、Microsoft® Windows Security Account Manager (SAM) 互換名を使用して、*authorization-name* またはユーザーの *host-variable* を指定することができます。修飾子は NetBIOS スタイル名でなければならず、最大長は 15 バイトです。たとえば「Domain¥User」などです。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。

例

例 1: C プログラムで、データベース別名 TOROLAB、ユーザー ID FERMAT、およびパスワード THEOREM を使用して、アプリケーション・サーバー TOROLAB に接続します。

```
EXEC SQL CONNECT TO TOROLAB USER FERMAT USING THEOREM;
```

例 2: C プログラムで、データベース別名がホスト変数 APP_SERVER (varchar(8)) に入っているアプリケーション・サーバーに接続します。正常に接続された後、アプリケーション・サーバーの 3 文字の製品 ID を、変数 PRODUCT (char(3)) にコピーします。

```
EXEC SQL CONNECT TO :APP_SERVER;
if (strncmp(SQLSTATE,'00000',5))
  strncpy(PRODUCT,sqlca.sqlerrp,3);
```

CONNECT (タイプ 2)

CONNECT (タイプ 2) ステートメントは、指定したアプリケーション・サーバーにアプリケーション・プロセスを接続し、アプリケーション制御の分散作業単位の規則を確立します。このサーバーは、そのプロセスの現行サーバーになります。

CONNECT (タイプ 1) ステートメントのほとんどの性質は、CONNECT (タイプ 2) ステートメントにも適用されます。この項では、それらを繰り返して説明するのではなく、タイプ 2 のエレメントのうちタイプ 1 とは異なる部分だけを説明します。

呼び出し

対話式 SQL 機能には外見上対話式的の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

許可

CONNECT 処理では、2 つのレベルのアクセス制御を通過します。接続を成功させるには、両方のレベルで条件を満たす必要があります。

第 1 レベルのアクセス制御は認証です。ここで、接続に関連付けられたユーザー ID が、サーバーにセットアップされた認証方式にしたがって、正常に認証されなければなりません。認証に成功すると、サーバーで有効な認証プラグインにしたがって、接続ユーザー ID から DB2 許可 ID が導出されます。続いて、DB2 許可 ID は、この接続における第 2 レベルのアクセス制御、すなわち許可をパスしなければなりません。パスするには、以下の権限のうち少なくとも 1 つが、この許可 ID に付与されている必要があります。

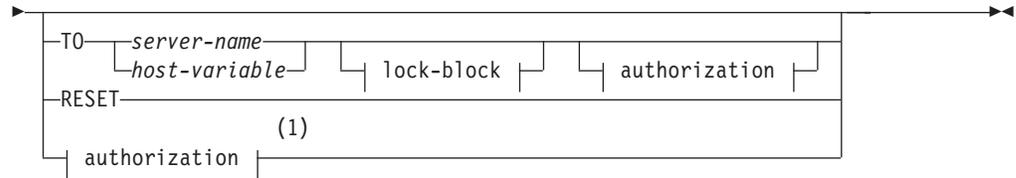
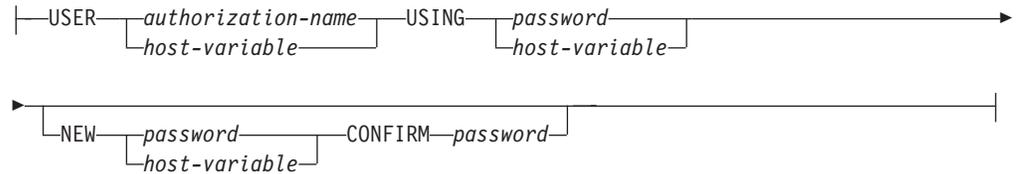
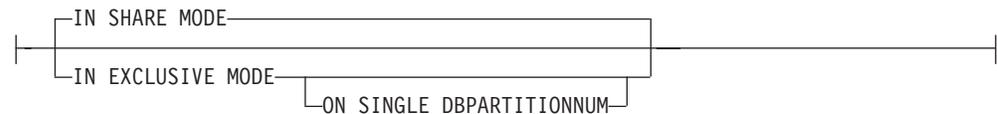
- CONNECT 権限
- DBADM 権限
- SYSADM 権限
- SYSCTRL 権限
- SYSMANT 権限

注: パーティション・データベースの場合、ユーザーとグループの定義は、データベース・パーティションのすべてにわたって同一である必要があります。

構文

タイプ 1 とタイプ 2 のどちらを選択するかは、プリコンパイラー・オプションによって決められます。それらのオプションの概要については、『分散リレーショナル・データベースへの接続』を参照してください。

▶▶—CONNECT—▶▶

**許可:****lock-block:****注:**

- 1 この形式は、暗黙の接続が使用可能な場合にのみ有効です。

説明**TO** *server-name/host-variable*

サーバーの名前のコーディング規則は、タイプ 1 と同じです。

SQLRULES(STD) オプションが有効な場合、*server-name* は、アプリケーション・プロセスの既存の接続を指定するものであってはなりません。そのように指定すると、エラー (SQLSTATE 08002) になります。

SQLRULES(DB2) オプションが有効で、*server-name* がアプリケーション・プロセスの既存の接続を指定している場合、その接続が現行接続になり、古い接続は休止状態になります。つまり、この状況での CONNECT ステートメントの効果は、SET CONNECTION ステートメントの効果と同じです。

SQLRULES の指定の詳細に関しては、『分散作業単位のセマンティクスを制御するオプション』を参照してください。

正常に接続された場合

CONNECT ステートメントが正常に実行された場合、次のようになります。

- アプリケーション・サーバーとの接続は作成されるか、または休止でない状態になり、現行接続かつ保留状態になります。
- CONNECT TO が現行サーバー以外のサーバーに出されると、現行の接続は休止状態になります。
- CURRENT SERVER 特殊レジスターと SQLCA は、CONNECT (タイプ 1) の場合と同じ方法で更新されます。

正常に接続されなかった場合

CONNECT (タイプ 2)

CONNECT ステートメントが正常に実行されなかった場合、次のようになります。

- エラーの理由に関係なく、アプリケーション・プロセスの接続状態とその接続の状態は変更されません。
- 失敗したタイプ 1 の CONNECT の場合と同様に、SQLCA の SQLERRP フィールドは、エラーを検出したアプリケーション・リクエスターまたはサーバーのモジュール名に設定されます。

CONNECT (オペランドなし)、IN SHARE/EXCLUSIVE MODE、USER、および USING

接続が存在する場合、タイプ 2 の動作はタイプ 1 と同様です。許可 ID とデータベース別名が、SQLCA の SQLERRMC フィールドに入れられます。接続が存在しない場合、暗黙接続の試みは行われず、SQLERRP および SQLERRMC の各フィールドはブランクを戻します。(アプリケーションでは、これらのフィールドを調べることによって、現行接続が存在しているか否かの検査を行うことができます。)

USER と USING を含むオペランドのない CONNECT は、DB2DBDFT 環境変数を使用することによって、アプリケーション・プロセスをデータベースに接続することができます。この方法は、タイプ 2 の CONNECT RESET に相当しますが、ユーザー ID とパスワードの使用が可能です。

RESET

デフォルトのデータベースが使用可能な場合、そのデータベースへの明示接続と同等です。デフォルトのデータベースが使用できない場合、アプリケーション・プロセスの接続状態とその接続の状態は変更されません。

デフォルトのデータベースが使用可能か否かは、インストール・オプション、環境変数、および認証設定値によって決まります。

規則

- 『分散作業単位のセマンティクスを制御するオプション』で概略を説明するように、一連の接続オプションによって、接続管理のセマンティクスが制御されます。すべてのプリプロセス済みソース・ファイルには、デフォルト値が割り当てられます。1 つのアプリケーションが、異なるさまざまな接続オプションでプリコンパイルされた複数のソース・ファイルで構成されている場合もあります。

SET CLIENT コマンドまたは API を最初に実行しない限り、実行時に実行される最初の SQL ステートメントを含むソース・ファイルのプリプロセスに使用された接続オプションが、有効な接続オプションになります。

ソース・ファイルの 1 つの CONNECT ステートメントが、異なる接続オプションで次々にプリプロセスされ、その間に SET CLIENT コマンドまたは API を実行することなく実行されると、エラー (SQLSTATE 08001) が戻されます。SET CLIENT コマンドまたは API を実行すると、アプリケーション内のすべてのソース・ファイルをプリプロセスするために使用された接続オプションは無視されます。

このステートメントの『例』セクションの例 1 では、こうした規則について説明されています。

- CONNECT ステートメントを使用して接続を確立したり切り替えたりすることができますが、USER/USING 節を使用した CONNECT は、指定したサーバーとの現行接続や休止接続がない場合にしか受け入れられません。USER/USING 節によって同じサーバーとの接続を発行するには、その前に接続を解放する必要があります。そうでない場合、リジェクトされます (SQLSTATE 51022)。接続を解放するには、DISCONNECT ステートメントまたは RELEASE ステートメントを出し、次に COMMIT ステートメントを出します。

注

- 暗黙接続は、タイプ 2 の接続を行うアプリケーションの最初の SQL ステートメントでサポートされます。SQL ステートメントをデフォルトのデータベースに対して実行するには、まず CONNECT RESET ステートメントまたは CONNECT USER/USING ステートメントを使用して、接続を確立する必要があります。オペランドのない CONNECT ステートメントでは、現行接続があればそれに関する情報が表示されますが、現行接続がない場合にはデフォルトのデータベースには接続しません。
- *authorization-name* SYSTEM を CONNECT ステートメント内で明示的に指定することはできません。ただし、Windows オペレーティング・システムでは、ローカル・システム・アカウントの下で稼働しているローカル・アプリケーションは暗黙的にデータベースに接続することができ、その場合、ユーザー ID が SYSTEM になります。
- Windows Server に明示的に接続するときは、Microsoft Windows Security Account Manager (SAM) 互換名を使用して、*authorization-name* またはユーザーの *host-variable* を指定することができます。修飾子は NetBIOS スタイル名でなければならず、最大長は 15 バイトです。たとえば「Domain¥User」などです。

タイプ 1 とタイプ 2 の CONNECT ステートメントの比較

CONNECT ステートメントのセマンティクスは、CONNECT プリコンパイラ・オプションまたは SET CLIENT API によって決まります (『分散作業単位のセマンティクスを制御するオプション』を参照)。CONNECT タイプ 1 または CONNECT タイプ 2 は指定することができ、それらのプログラムの CONNECT ステートメントは、それぞれタイプ 1 およびタイプ 2 の CONNECT ステートメントと呼ばれます。それらのセマンティクスについて、以下に説明します。

CONNECT の使用

タイプ 1

各作業単位は、1 つのアプリケーション・サーバーに対してのみ接続を確立できます。

他のアプリケーション・サーバーと接続するためには、その前に、現行の作業単位をコミットまたはロールバックする必要があります。

CONNECT ステートメントは、現行接続を確立します。後続の SQL 要求は、他の CONNECT によって変更されるまで、この接続に送られます。

タイプ 2

各作業単位は、複数のアプリケーション・サーバーとの接続を確立することができます。

他のアプリケーション・サーバーと接続する前に、現行の作業単位をコミットまたはロールバックする必要はありません。

最初の接続を確立する場合はタイプ 1 の CONNECT と同じです。休止接続に切り替える際に SQLRULES が STD に設定されている場合には、SET CONNECTION ステートメントを使用する必要があります。

CONNECT (タイプ 2)

タイプ 1

現行接続への接続が有効であり、現行接続を変更しません。

他のアプリケーション・サーバーに接続すると、現行接続が切断されます。新しい接続が現行接続になります。1つの作業単位で維持される接続は1つだけです。

SET CONNECTION ステートメントはタイプ 1 の接続でサポートされていますが、有効な接続先は現行接続だけです。

タイプ 2

SQLRULES プリコンパイラ・オプションが DB2 に設定されている場合は、タイプ 1 の CONNECT と同じです。SQLRULES が STD に設定されている場合、SET CONNECTION ステートメントを使用する必要があります。

別のアプリケーション・サーバーに接続すると、現行接続は休止状態 になります。新しい接続が現行接続になります。1つの作業単位で複数の接続を維持できます。

CONNECT が休止接続のアプリケーション・サーバーに対するものである場合、それが現行接続になります。

CONNECT を使用した休止接続への接続は、SQLRULES(DB2) が指定されている場合にのみ可能です。SQLRULES(STD) が指定されている場合は、SET CONNECTION ステートメントを使用する必要があります。

タイプ 2 の接続では、接続状態を休止から現行に変更する SET CONNECTION ステートメントがサポートされています。

CONNECT...USER...USING の使用

タイプ 1

USER...USING 節を使用した接続では、現行接続が切断され、指定した許可名とパスワードで新しい接続が確立されます。

タイプ 2

USER/USING 節を使用した接続は、指定したその同じサーバーに対する現行接続も休止接続もない場合にのみ、受け入れられます。

暗黙の CONNECT、CONNECT RESET の使用、および切断

タイプ 1

CONNECT RESET を使用して、現行接続を切断することができます。

CONNECT RESET を使用して現行接続を切断した場合、その次の SQL ステートメントが CONNECT ステートメントでなく、デフォルトのアプリケーション・サーバーがシステムに定義されていれば、それに対する暗黙接続が行われます。

連続して CONNECT RESET を出すと、エラーになります。

CONNECT RESET では、現行の作業単位のコミットも暗黙のうちに行われます。

何らかの理由で既存の接続がシステムによって切断された場合、このデータベースに対して CONNECT 以外の SQL ステートメントを連続して出すと、08003 という SQLSTATE を受け取るようになります。

アプリケーション・プロセスが正常に終了した時点で、作業単位が暗黙のうちにコミットされます。

アプリケーション・プロセスが終了すると、すべての接続 (1 つだけ) が切断されます。

タイプ 2

CONNECT RESET は、デフォルトのアプリケーション・サーバーがシステムに定義されている場合、それに対する明示的接続に相当します。

接続は、COMMIT の正常実行時にアプリケーションによって切断できます。コミットの前には、RELEASE ステートメントを使用して、接続を解放ペンディングにします。このような接続はすべて、その次の COMMIT 時に切断されます。

あるいは RELEASE ステートメントの代わりに、プリコンパイラ・オプションの DISCONNECT(EXPLICIT)、DISCONNECT(CONDITIONAL)、DISCONNECT(AUTOMATIC)、または DISCONNECT の各ステートメントを使用することができます。

デフォルトのアプリケーション・サーバーがシステムに定義されている場合、CONNECT RESET はそれに対する明示接続に相当しません。

連続する CONNECT RESET を発行してエラーになるのは、SQLRULES(STD) が指定されている場合だけです。このオプションを指定すると既存の接続に対して CONNECT を使用できなくなります。

CONNECT RESET では、現行の作業単位はコミットされません。

既存の接続がシステムによって切断された場合でも、COMMIT、ROLLBACK、および SET CONNECTION の各ステートメントを使用することができます。

タイプ 1 と同じ。

アプリケーション・プロセスが終了すると、すべての接続 (現行、休止、および解放ペンディングのもの) が切断されます。

CONNECT (タイプ 2)

CONNECT のエラー

タイプ 1

ローカル・ディレクトリーにサーバー名が定義されていないというエラー以外で CONNECT がエラーになった場合、現行接続があるかどうかに関係なく、アプリケーション・プロセスは未接続状態になります。それ以降の CONNECT 以外のステートメントは、08003 の SQLSTATE を受け取ることになります。

タイプ 2

CONNECT がエラーになったときに現行接続があっても、その現行接続は影響を受けません。

CONNECT がエラーになったときに、現行接続がなかった場合、プログラムは未接続状態になります。それ以降の CONNECT 以外のステートメントは、08003 の SQLSTATE を受け取ることになります。

例

例 1:

この例では、複数のソース・プログラム (枠の中に示される) の使用法を示します。いくつかは異なる接続オプション (コードの上に示される) を指定してプリプロセスされ、そのうち 1 つは SET CLIENT API 呼び出しを含んでいます。

PGM1: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
exec sql CONNECT TO OTTAWA;
exec sql SELECT col1 INTO :hv1
FROM tb11;
...
```

PGM2: CONNECT(2) SQLRULES(STD) DISCONNECT(AUTOMATIC)

```
...
exec sql CONNECT TO QUEBEC;
exec sql SELECT col1 INTO :hv1
FROM tb12;
...
```

PGM3: CONNECT(2) SQLRULES(STD) DISCONNECT(EXPLICIT)

```
...
SET CLIENT CONNECT 2 SQLRULES DB2 DISCONNECT EXPLICIT 1
exec sql CONNECT TO LONDON;
exec sql SELECT col1 INTO :hv1
FROM tb13;
...
```

1 注: SET CLIENT API の実際の構文ではありません。

PGM4: CONNECT(2) SQLRULES(DB2) DISCONNECT(CONDITIONAL)

```
...
exec sql CONNECT TO REGINA;
exec sql SELECT col1 INTO :hv1
FROM tb14;
...
```

アプリケーションが PGM1 に続いて PGM2 を実行すると、次のようになります。

- OTTAWA への接続が実行されます。 connect=2、 sqlrules=DB2、 disconnect=CONDITIONAL

- QUEBEC への接続はエラー (SQLSTATE 08001) になります。これは、SQLRULES と DISCONNECT が共に異なっているためです。

アプリケーションが PGM1 に続いて PGM3 を実行すると、次のようになります。

- OTTAWA への接続が実行されます。 connect=2、 sqlrules=DB2、 disconnect=CONDITIONAL
- LONDON への接続が実行されます。 connect=2、 sqlrules=DB2、 disconnect=EXPLICIT

2 番目の CONNECT ステートメントの前に SET CLIENT API が実行されるため、問題ありません。

アプリケーションが PGM1 に続いて PGM4 を実行すると、次のようになります。

- OTTAWA への接続が実行されます。 connect=2、 sqlrules=DB2、 disconnect=CONDITIONAL
- REGINA への接続が実行されます。 connect=2 sqlrules=DB2、 disconnect=CONDITIONAL

これは、PGM1 のプリプロセッサ・オプションが PGM4 と同じなので、問題ありません。

例 2:

この例では、CONNECT (タイプ 2)、SET CONNECTION、RELEASE、および DISCONNECT の各ステートメントの相互関係を示します。S0、S1、S2、および S3 は 4 つのサーバーを示します。

順序	ステートメント	現行サーバー	休止接続	解放ペンディング
0	• ステートメントはない	• なし	• なし	• なし
1	• SELECT * FROM TBLA	• S0 (デフォルト値)	• なし	• なし
2	• CONNECT TO S1 • SELECT * FROM TBLB	• S1 • S1	• S0 • S0	• なし • なし
3	• CONNECT TO S2 • UPDATE TBLC SET ...	• S2 • S2	• S0, S1 • S0, S1	• なし • なし
4	• CONNECT TO S3 • SELECT * FROM TBLD	• S3 • S3	• S0, S1, S2 • S0, S1, S2	• なし • なし
5	• SET CONNECTION S2	• S2	• S0, S1, S3	• なし
6	• RELEASE S3	• S2	• S0, S1	• S3
7	• COMMIT	• S2	• S0, S1	• なし
8	• SELECT * FROM TBLE	• S2	• S0, S1	• なし
9	• DISCONNECT S1 • SELECT * FROM TBLF	• S2 • S2	• S0 • S0	• なし • なし

CREATE ALIAS

CREATE ALIAS ステートメントは、表、ビュー、ニックネーム、または他の別名に対する別名を定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する IMPLICIT_SCHEMA 権限 (別名の暗黙または明示のスキーマ名が存在しない場合)
- スキーマに対する CREATEIN 特権 (別名のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

別名によって参照されるオブジェクトを使用するために必要な特権は、直接そのオブジェクトを使用するために必要な特権と同一です。

構文

```

▶▶ CREATE ALIAS alias-name FOR
    table-name
    view-name
    nickname
    alias-name2
  
```

説明

alias-name

別名を指定します。この名前が、現行データベースに存在する表、ビュー、ニックネーム、または別名を指定してはなりません。

2 つの部分からなる名前を指定する場合、'SYS' で始まるスキーマ名は使用できません (SQLSTATE 42939)。

別名を定義する際の規則は、表名の定義に使用される規則と同じです。

FOR *table-name*, *view-name*, *nickname*, または *alias-name2*

alias-name を定義する対象の表名、ビュー名、ニックネーム、または別名を指定します。他の別名 (*alias-name2*) を指定する場合、その別名は、定義される新しい *alias-name* (完全修飾形式の) と同じであってはなりません。 *table-name* を宣言済み一時表にすることはできません (SQLSTATE 42995)。

注

- 新しく作成した別名の定義は、SYSCAT.TABLES に保管されます。

- 別名は、定義時に存在していないオブジェクトに対しても定義できます。オブジェクトが存在しない場合、警告が出されます (SQLSTATE 01522)。ただし、参照されるオブジェクトは、その別名を含む SQL ステートメントのコンパイル時には存在していなければなりません。そうでない場合、エラーになります (SQLSTATE 52004)。
- 他の別名を参照する別名を別名チェーンの一部として定義することは可能ですが、SQL ステートメントで使用する場合には、単一の別名と同じ制約がそのチェーンにも適用されます。別名チェーンは、単一の別名と同じ方法で解決されます。ビュー定義、パッケージ内のステートメント、または別名チェーンを指すトリガーで別名が使用された場合、そのビュー、パッケージ、またはチェーンの各別名についてのトリガーに従属関係が記録されます。別名チェーンの中に反復サイクルがあってはならず、それは別名定義時に検出されます。
- まだ存在していないスキーマ名を用いて別名を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- **互換性**
 - DB2 for z/OS との互換性:
 - ALIAS の代わりに SYNONYM を指定できます。

例

例 1: HEDGES は表 T1 に対して別名を作成します (どちらも修飾なし)。

```
CREATE ALIAS A1 FOR T1
```

HEDGES.T1 に対して別名 HEDGES.A1 が作成されます。

例 2 HEDGES は表に対して別名を作成します (どちらも修飾付き)。

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
```

MCKNIGHT.A1 に対して別名 HEDGES.T1 が作成されます。

例 3: HEDGES は表に対して別名を作成します (異なるスキーマ内の別名。HEDGES は DBADM ではなく、HEDGES はスキーマ MCKNIGHT に対して CREATEIN を持っていない)。

```
CREATE ALIAS MCKNIGHT.A1 FOR MCKNIGHT.T1
```

この例はエラーになります (SQLSTATE 42501)。

例 4: HEDGES は未定義の表に対して別名を作成します (どちらも修飾付き。FUZZY.WUZZY は存在しない)。

```
CREATE ALIAS HEDGES.A1 FOR FUZZY.WUZZY
```

このステートメントは成功しますが、警告 (SQLSTATE 01522) が出されます。

例 5: HEDGES は別名に対して別名を作成します (どちらも修飾付き)。

```
CREATE ALIAS HEDGES.A1 FOR MCKNIGHT.T1
CREATE ALIAS HEDGES.A2 FOR HEDGES.A1
```

CREATE ALIAS

最初のステートメントは成功します (例 2 と同じ)。

2 番目のステートメントも成功して、別名チェーンが作成されます。つまり、HEDGES.A2 が HEDGES.A1 を参照し、その HEDGES.A1 が MCKNIGHT.T1 を参照することになります。HEDGES に MCKNIGHT.T1 に対する特権があるかどうかは関係ありません。別名は、表の特権に関係なく作成されます。

例 6: ニックネーム FUZZYBEAR の別名として、A1 を指定します。

```
CREATE ALIAS A1 FOR FUZZYBEAR
```

例 7: ある大規模な組織に、D108 という会計部門と D577 という人事部門があります。D108 は、DB2 RDBMS に存在する表に、ある情報を保持しています。D577 は、Oracle RDBMS に存在する表に、いくつかのレコードを保持しています。DBA は、この 2 つの RDBMS をフェデレーテッド・システム内のデータ・ソースとして定義し、それぞれの表に DEPTD108 および DEPTD577 というニックネームを付けます。フェデレーテッド・システムのユーザーはこれらの表の結合を作成する必要がありますが、英数字のニックネームではなく、もっと意味のある名前前で参照できるようにしたいことがあります。それで、DEPTD108 の別名として FINANCE を定義し、DEPTD577 の別名として PERSONNEL を定義します。

```
CREATE ALIAS FINANCE FOR DEPTD108
CREATE ALIAS PERSONNEL FOR DEPTD577
```

CREATE AUDIT POLICY

CREATE AUDIT POLICY ステートメントは、現行サーバーの監査ポリシーを定義します。ポリシーは、監査するカテゴリーを決定します。その後このポリシーを他のデータベース・オブジェクトに適用して、それらのオブジェクトの使用を監査する方法を決めることができます。

呼び出し

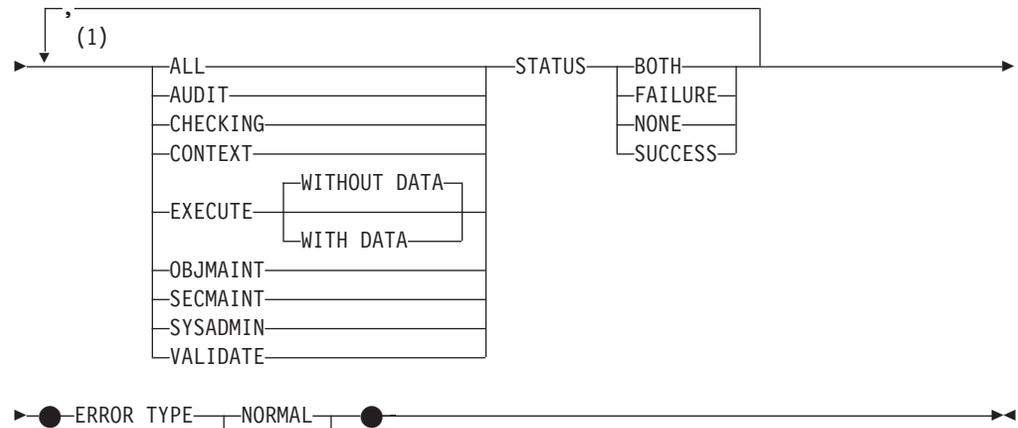
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

```
►► CREATE AUDIT POLICY policy-name ● CATEGORIES ◀◀
```

**注:**

- 1 カテゴリーはそれぞれ 2 度以上指定できず (SQLSTATE 42614)、ALL が指定されている場合にはその他のカテゴリーを指定できません (SQLSTATE 42601)。

説明*policy-name*

監査ポリシーの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *policy-name* はカタログで既に記述されている監査ポリシーを識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

CATEGORIES

状況が指定される 1 つ以上の監査カテゴリーのリスト。 ALL が指定されない場合、明示的に指定されないカテゴリーの STATUS は NONE に設定されます。

ALL

すべてのカテゴリーを同じ状況に設定します。 EXECUTE カテゴリーは WITHOUT DATA です。

AUDIT

監査設定が変更されたとき、または監査ログにアクセスされたときにレコードを生成します。

CHECKING

データベース・オブジェクトまたは関数へのアクセス試行またはその操作試行の許可検査中にレコードを生成します。

CONTEXT

データベース操作が実行されたときの操作コンテキストを示すレコードを生成します。

EXECUTE

SQL ステートメントの実行を示すレコードを生成します。

CREATE AUDIT POLICY

WITHOUT DATA または WITH DATA

任意のホスト変数およびパラメーター・マーカーに指定される入力データ値を EXECUTE カテゴリの一部としてログに記録するかどうかを指定します。

WITHOUT DATA

任意のホスト変数およびパラメーター・マーカーに指定される入力データ値は EXECUTE カテゴリの一部としてログに記録されません。WITHOUT DATA はデフォルトです。

WITH DATA

任意のホスト変数およびパラメーター・マーカーに指定される入力データ値は EXECUTE カテゴリの一部としてログに記録されます。すべての入力値がログに記録されるわけではありません。特に、LOB、LONG、XML、および構造化タイプのパラメーターは NULL 値となります。日付、時刻、およびタイム・スタンプの各フィールドは ISO 形式でログに記録されます。入力データ値はログに記録される前にデータベース・コード・ページに変換されます。コード・ページの変換に失敗してもエラーは戻されず、変換前のデータがログに記録されます。

OBJMAINT

データ・オブジェクトが作成またはドロップされたときにレコードを生成します。

SECMAINT

オブジェクト特権、データベース特権、または DBADM 権限が付与されたとき、または取り消されたときにレコードを生成します。データベース・マネージャーのセキュリティー構成パラメーター **sysadm_group**、**sysctrl_group**、または **sysmaint_group** が変更されたときにもレコードが生成されます。

SYSADMIN

SYSADM、SYSMAINT、または SYSCTRL 権限を必要とする操作が実行されたときにレコードを生成します。

VALIDATE

ユーザーが認証されたとき、またはユーザーに関連したシステム・セキュリティー情報が取得されたときにレコードを生成します。

STATUS

指定されたカテゴリの状況を指定します。

BOTH

成功イベントと失敗イベントが監査されます。

FAILURE

失敗イベントのみが監査されます。

SUCCESS

成功イベントのみが監査されます。

NONE

このカテゴリのイベントは監査されません。

ERROR TYPE

監査エラーを戻すか、無視するかを指定します。

NORMAL

監査によって生成されたエラーはすべて無視され、実行される操作に関連したエラーの SQLCODE だけがアプリケーションに戻されます。

AUDIT

監査機能自体で発生したエラーを含む、すべてのエラーがアプリケーションに戻されます。

規則

- **AUDIT 排他 SQL ステートメントの後**は、COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。AUDIT 排他 SQL ステートメントは次のとおりです。
 - AUDIT
 - CREATE AUDIT POLICY、ALTER AUDIT POLICY、または DROP (AUDIT POLICY)
 - DROP (監査ポリシーと関連付けられる場合は ROLE または TRUSTED CONTEXT)
- **AUDIT 排他 SQL ステートメントをグローバル・トランザクション** (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- データベース・パーティション全体を通じて、同時に実行できる非コミットの AUDIT 排他 SQL ステートメントは 1 つのみです。非コミットの AUDIT 排他 SQL ステートメントが実行されている場合、後続の AUDIT 排他 SQL ステートメントは、現行の AUDIT 排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

AUDIT および OBJMAINT カテゴリについての成功と失敗を監査し、SECMAINT、CHECKING、および VALIDATE カテゴリについては失敗だけを監査し、その他のカテゴリについてはどちらのイベントも監査しない監査ポリシーを作成します。

```
CREATE AUDIT POLICY DBAUDPRF
  CATEGORIES AUDIT STATUS BOTH,
              SECMAINT STATUS FAILURE,
              OBJMAINT STATUS BOTH,
              CHECKING STATUS FAILURE,
              VALIDATE STATUS FAILURE
  ERROR TYPE NORMAL
```

CREATE BUFFERPOOL

CREATE BUFFERPOOL ステートメントは、データベース・マネージャーにより使用される新しいバッファ・プールを定義します。

CREATE BUFFERPOOL

パーティション・データベースでは、特定のデータベース・パーティションのサイズを上書きできるように、デフォルトのバッファ・プール定義がそれぞれのデータベース・パーティションに対して指定されます。またパーティション・データベースでは、データベース・パーティション・グループを指定しない限り、すべてのデータベース・パーティションにバッファ・プールが定義されます。データベース・パーティション・グループを指定すると、そのデータベース・パーティション・グループのデータベース・パーティションにだけバッファ・プールが作成されます。

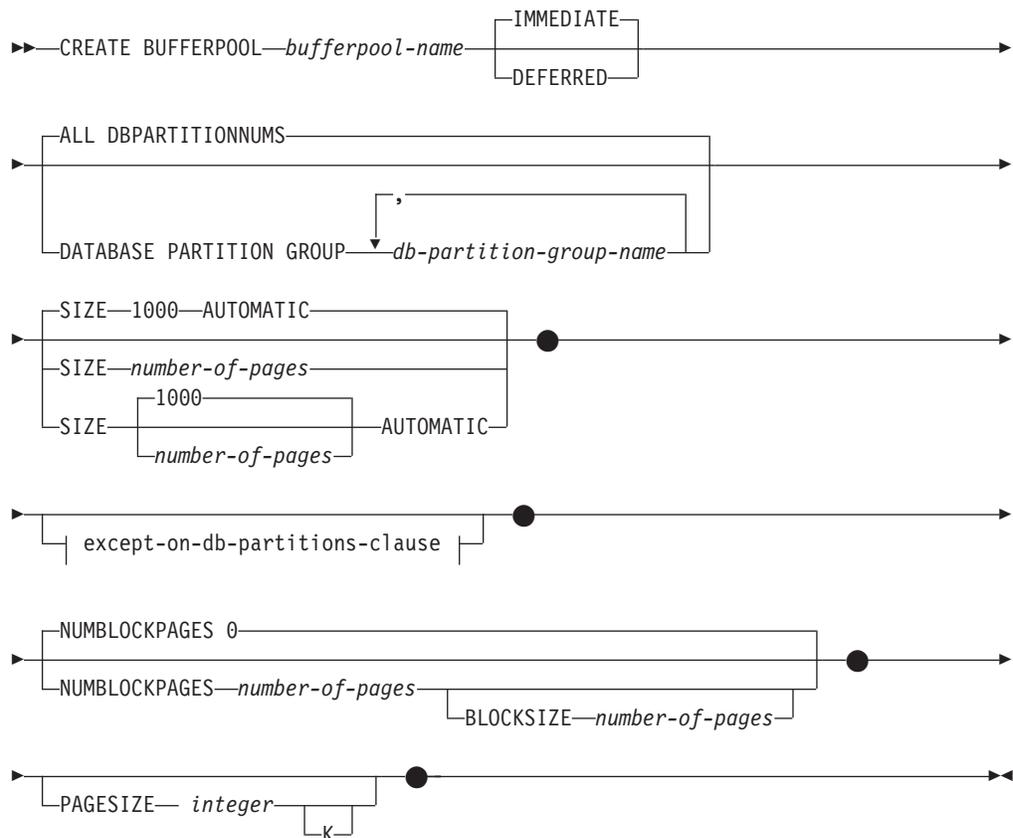
呼び出し

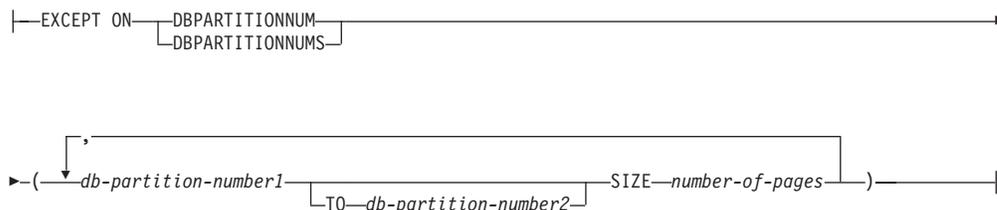
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SYSCCTRL または SYSADM 権限が含まれている必要があります。

構文



except-on-db-partitions-clause:**説明***bufferpool-name*

バッファークールの名前を指定します。これは、1部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *bufferpool-name* には、すでにカタログに存在するバッファークールを指定してはなりません (SQLSTATE 42710)。 *bufferpool-name* を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

IMMEDIATE または DEFERRED

バッファークールが直ちに作成されるようにするかどうかを指定します。

IMMEDIATE

バッファークールは直ちに作成されます。メモリーを共用するデータベース内に新規バッファークールを割り振るのに十分な予約済みスペースがない場合 (SQLSTATE 01657)、このステートメントは DEFERRED で実行されます。

DEFERRED

データベースが非活動状態になると、バッファークールが作成されます (すべてのアプリケーションがデータベースから切断される必要があります)。予約済みのメモリー・スペースは必要ありません。DB2 が、システムから必要なメモリーを割り振ります。

ALL DBPARTITIONNUMS

このバッファークールは、データベースのすべてのデータベース・パーティションに作成されます。

DATABASE PARTITION GROUP *db-partition-group-name, ...*

バッファークール定義を適用するデータベース・パーティション・グループ (単一または複数) を指定します。このパラメーターを指定すると、バッファークールはこれらのデータベース・パーティション・グループのデータベース・パーティションにだけ作成されます。それぞれのデータベース・パーティション・グループは、現在データベースに存在している必要があります (SQLSTATE 42704)。 DATABASE PARTITION GROUP 節を指定しないと、このバッファークールはすべてのデータベース・パーティション (およびその後データベースに追加されるあらゆるデータベース・パーティション) に作成されます。

SIZE

バッファークールのサイズを指定します。パーティション・データベースの場合、これはバッファークールが存在するすべてのデータベース・パーティションのデフォルトのサイズになります。デフォルトは 1000 ページです。

CREATE BUFFERPOOL

number-of-pages

新規のバッファークールに対応するページ数。

AUTOMATIC

このバッファークールのセルフチューニングを使用可能に設定します。データベース・マネージャーは、作業負荷の要件に応じてバッファークールのサイズを調整します。暗黙的または明示的に指定されたページ数が、バッファークールの初期サイズとして使用されます。

NUMBLOCKPAGES *number-of-pages*

ブロック・ベース域に存在していなければならないページ数を指定します。ページ数は、バッファークールのページ数の 98% より小さくしなければなりません (SQLSTATE 54052)。値 0 を指定すると、ブロック入出力は不可になります。使用されている NUMBLOCKPAGES の実際の値は、BLOCKSIZE の倍数になります。

BLOCKSIZE *number-of-pages*

ブロック内のページ数を指定します。ブロック・サイズの値は、2 から 256 でなければなりません (SQLSTATE 54053)。デフォルト値は 32 です。

except-on-db-partitions-clause

そのバッファークールのサイズをデフォルトのサイズとは異なるサイズにしたい 1 つまたは複数のデータベース・パーティションを指定します。この節の指定がない場合、すべてのデータベース・パーティションが、このバッファークールに対して指定したのと同じプール・サイズを持つこととなります。

EXCEPT ON DBPARTITIONNUMS

特定のデータベース・パーティションを指定することを示すキーワードです。DBPARTITIONNUM は DBPARTITIONNUMS の同義語です。

db-partition-number1

バッファークールが作成されるデータベース・パーティションに含まれるデータベース・パーティション番号を指定します。

TO *db-partition-number2*

データベース・パーティション番号の範囲を指定します。

db-partition-number2 の値は、*db-partition-number1* の値以上でなければなりません (SQLSTATE 428A9)。指定するデータベース・パーティション番号の範囲 (両端を含む) のすべてのデータベース・パーティションは、バッファークールを作成するデータベース・パーティションに含まれていなければなりません (SQLSTATE 42729)。

SIZE *number-of-pages*

バッファークールのサイズをページ数で指定します。

PAGESIZE *integer* [**K**]

バッファークールに使用されるページのサイズを定義します。接尾部 **K** を持たない *integer* の有効値は、4096、8192、16 384、または 32 768 です。接尾部 **K** を持つ *integer* の有効値は、4、8、16、または 32 です。*integer* と **K** の間には、任意の数のスペースを使用できます (スペースなしでも可)。ページ・サイズがこれらのいずれかの値でない場合は、エラーが生じます (SQLSTATE 428DE)。

デフォルト値は **pagesize** データベース構成パラメーターによって指定されます。これは、データベースの作成時に設定されます。

注

- DEFERRED オプションを使用してバッファークラスタが作成されると、このバッファークラスタ内に作成される任意の表スペースは、データベースが次に活動状態になるときまで、同じページ・サイズの小さいシステム・バッファークラスタを使用します。バッファークラスタを再アクティブ化して、新たなバッファークラスタに対する表スペースの割り当てを有効にするには、データベースを再始動する必要があります。デフォルト・オプションは IMMEDIATE です。
- すべてのバッファークラスタの合計と、その他のデータベース・マネージャーやアプリケーションの要件に合うように、マシンに十分な実メモリーが必要です。DB2 が正規バッファークラスタに必要な合計のメモリーを入手できない場合には、各ページ・サイズ (4K、8K、16K および 32K) に対して小さいシステム・バッファークラスタの始動を試みます。この場合、ユーザーに警告が出され (SQLSTATE 01626)、すべての表スペースからのページはシステム・バッファークラスタを使用します。
- 互換性
 - 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。
 - DBPARTITIONNUMS の代わりに NODES を指定できます。
 - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できません。

CREATE DATABASE PARTITION GROUP

CREATE DATABASE PARTITION GROUP ステートメントは、データベースに新しいデータベース・パーティション・グループを定義し、データベース・パーティションをデータベース・パーティション・グループに割り当て、データベース・パーティション・グループ定義をシステム・カタログに記録します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

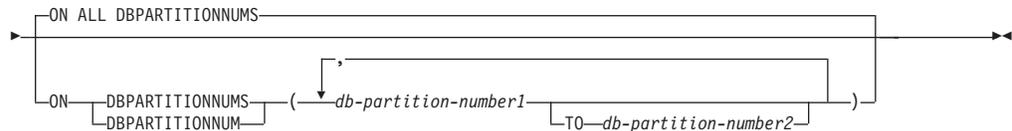
許可

このステートメントの許可 ID が持つ特権には、SYSCTRL または SYSADM 権限が含まれている必要があります。

構文

```
▶▶—CREATE DATABASE PARTITION GROUP—db-partition-group-name—▶▶
```

CREATE DATABASE PARTITION GROUP



説明

db-partition-group-name

データベース・パーティション・グループの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。

db-partition-group-name (データベース・パーティション・グループ名) は、すでにカタログに存在しているデータベース・パーティション・グループを指定するものであってはなりません (SQLSTATE 42710)。*db-partition-group-name* を文字 'SYS' または 'IBM' で始めることはできません (SQLSTATE 42939)。

ON ALL DBPARTITIONNUMS

データベース・パーティション・グループの作成時に、データベース (db2nodes.cfg ファイル) に定義されているすべてのデータベース・パーティションにわたってデータベース・パーティション・グループを定義することを指定します。

データベース・システムにデータベース・パーティションが追加された場合、ALTER DATABASE PARTITION GROUP ステートメントを実行して、この新しいデータベース・パーティションをデータベース・パーティション・グループ (IBMDEFAULTGROUP を含む) に組み込む必要があります。さらに、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを実行して、そのデータベース・パーティションにデータを移す必要があります。

ON DBPARTITIONNUMS

データベース・パーティション・グループに入れるデータベース・パーティションを指定します。DBPARTITIONNUM は DBPARTITIONNUMS の同義語です。

db-partition-number1

データベース・パーティション番号を指定します。(前のバージョンとの互換性を保つため、形式 NODEnnnnn の *node-name* も指定できます。)

TO db-partition-number2

データベース・パーティション番号の範囲を指定します。

db-partition-number2 の値は、*db-partition-number1* の値以上でなければなりません (SQLSTATE 428A9)。指定したデータベース・パーティション番号の範囲 (指定した番号を含む) のすべてのデータベース・パーティションが、データベース・パーティション・グループに入れられます。

規則

- 番号によって指定するそれぞれのデータベース・パーティションは、db2nodes.cfg ファイルに定義されていなければなりません (SQLSTATE 42729)。
- ON DBPARTITIONNUMS 節にリストするそれぞれの *db-partition-number* は、同じであってはなりません (SQLSTATE 42728)。
- 有効な *db-partition-number* は、0 から 999 (両端を含む) です (SQLSTATE 42729)。

注

- このステートメントは、データベース・パーティション・グループに対する分散マップを作成します。それぞれの分散マップごとに、分散マップ ID (PMAP_ID) が生成されます。この情報はカタログに記録され、SYSCAT.DBPARTITIONGROUPS と SYSCAT.PARTITIONMAPS から検索することができます。分散マップのそれぞれの項目は、ハッシュされたすべての行が常駐するターゲット・データベース・パーティションを指定します。単一パーティションのデータベース・パーティション・グループの場合、対応する分散マップの項目は 1 つだけです。複数パーティションのデータベース・パーティション・グループの場合、対応する分散マップには 4096 の項目があり、データベース・パーティション番号がマップ項目にラウンドロビン方式 (デフォルト) で割り当てられます。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。
 - DBPARTITIONNUMS の代わりに NODES を指定できます。
 - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。

例

0、1、2、5、7、および 8 として定義された 6 つのデータベース・パーティションを持つパーティション・データベースがあると想定します。

- 6 つのデータベース・パーティションすべてに対して、MAXGROUP という名前のデータベース・パーティション・グループを作成すると想定します。必要なステートメントは以下のようになります。

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- データベース・パーティション 0、1、2、5、および 8 に対して、MEDGROUP と呼ばれるデータベース・パーティション・グループを作成すると想定します。必要なステートメントは以下のようになります。

```
CREATE DATABASE PARTITION GROUP MEDGROUP
ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- データベース・パーティション 7 に対して、単一パーティションのデータベース・パーティション・グループ MINGROUP を作成すると想定します。必要なステートメントは以下のようになります。

```
CREATE DATABASE PARTITION GROUP MINGROUP
ON DBPARTITIONNUM (7)
```

CREATE EVENT MONITOR

CREATE EVENT MONITOR ステートメントは、データベースの使用中に発生する特定のイベントを記録するモニターを定義します。各イベント・モニターの定義には、データベースがイベントを記録するロケーションも指定します。

CREATE EVENT MONITOR

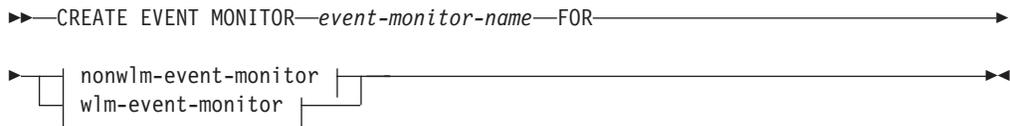
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

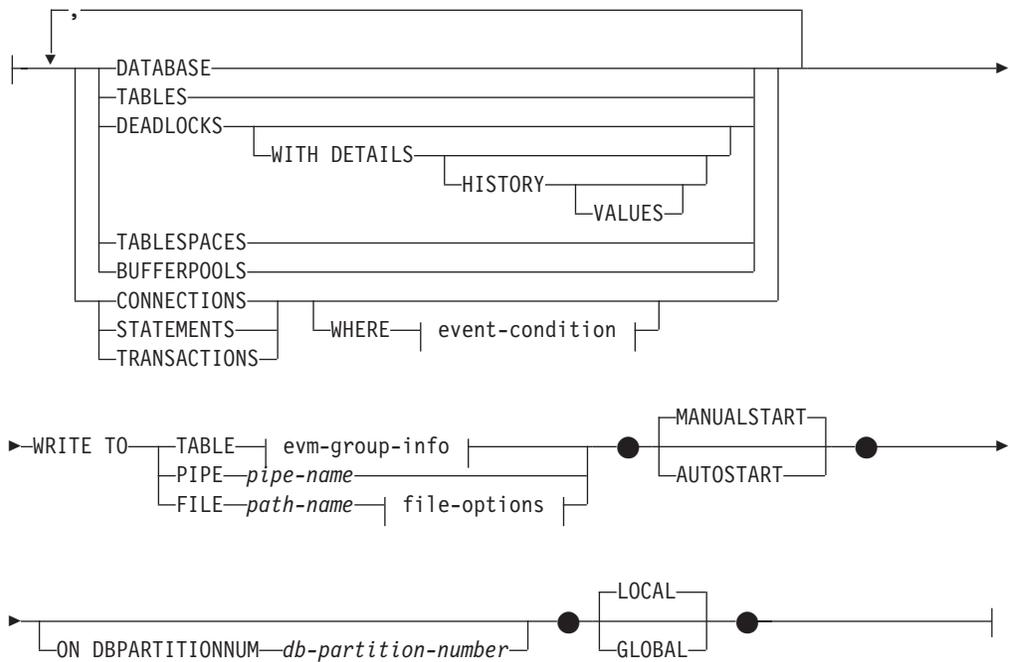
許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM のいずれかの権限が含まれている必要があります。

構文

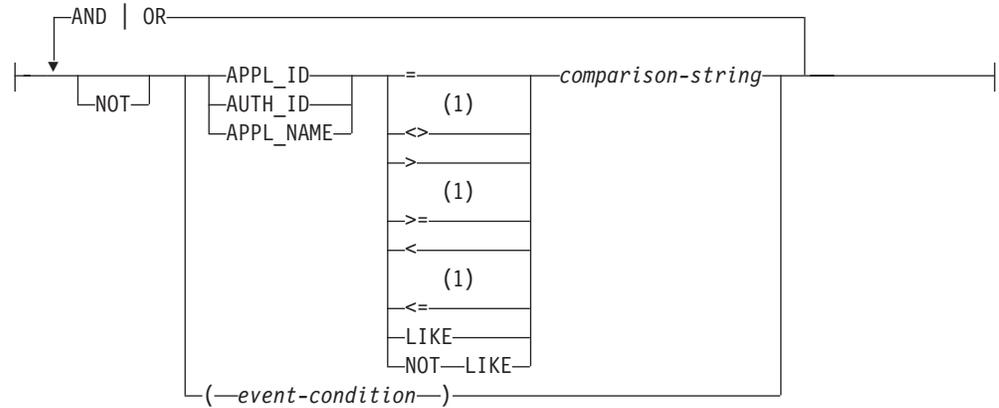


nonwlm-event-monitor:

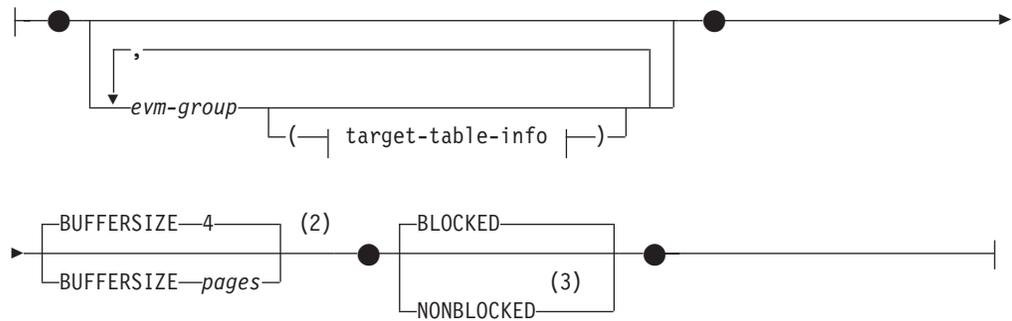


event-condition:

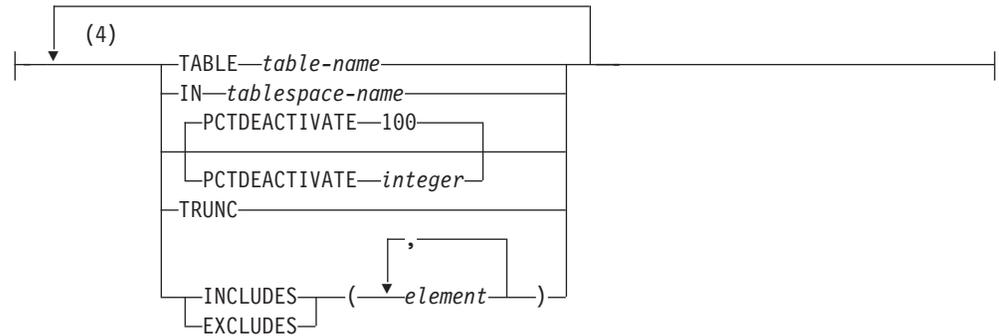
CREATE EVENT MONITOR



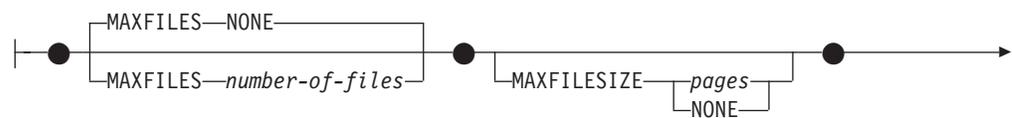
evm-group-info:



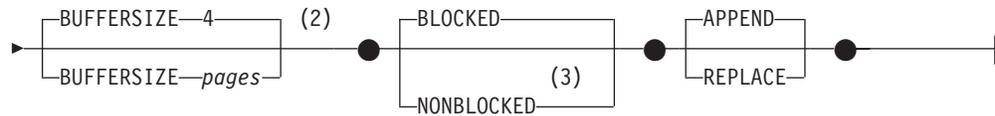
target-table-info:



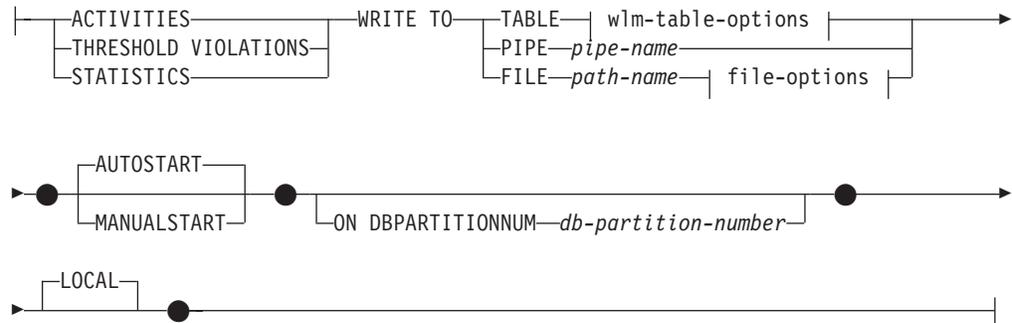
file-options:



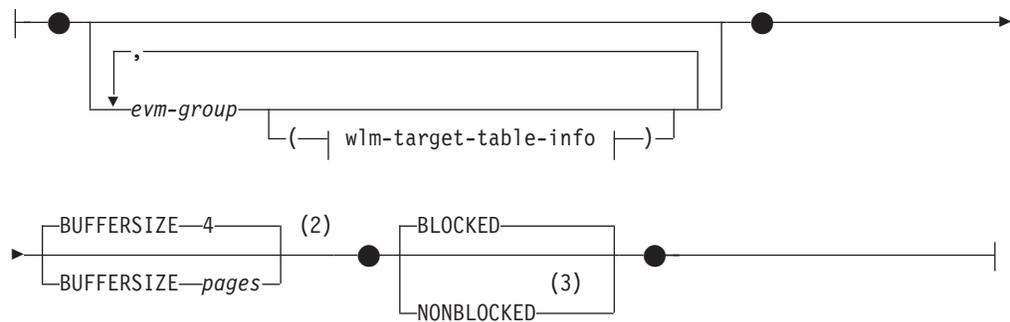
CREATE EVENT MONITOR



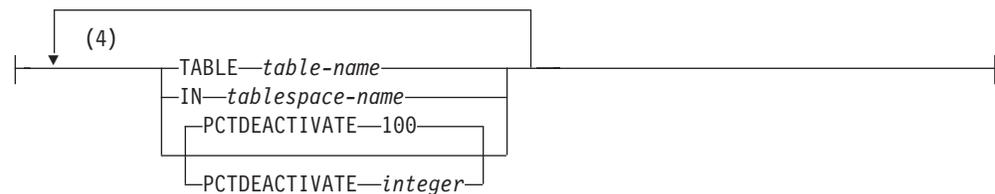
wlm-event-monitor:



wlm-table-options:



wlm-target-table-info:



注:

- 1 これらの演算子の他の形式もサポートされます。
- 2 BUFFER SIZE 節は、統計イベント・モニター、しきい値違反イベント・モニター、および非ワークロード管理 (WLM) イベント・モニターのみ適用できません (SQLSTATE 42613)。
- 3 NONBLOCKED キーワードは、統計イベント・モニター、しきい値違反イベント・モニター、および非 WLM イベント・モニターのみ適用できます (SQLSTATE 42613)。
- 4 各節は一度だけ指定できます。

説明*event-monitor-name*

イベント・モニターの名前。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *event-monitor-name* (イベント・モニター名) は、すでにカタログに存在するイベント・モニターを指定する名前ではありません (SQLSTATE 42710)。

FOR

記録するイベント・タイプをこの後に指定します。

DATABASE

最後のアプリケーションがデータベースから切断された場合に、イベント・モニターがそのデータベース・イベントを記録することを指定します。

TABLES

最後のアプリケーションがデータベースから切断された場合に、イベント・モニターがアクティブな各表の表イベントを記録することを指定します。パーティション表の場合は、アクティブな各表の各データ・パーティションごとに 1 つの表イベントが記録されます。アクティブな表とは、データベースに最初に接続した時点以降に変更が行われた表です。

DEADLOCKS

デッドロックが発生した場合に、イベント・モニターがデッドロック・イベントを記録することを指定します。

WITH DETAILS

イベント・モニターが、デッドロックに関係する各アプリケーションのより詳細なデッドロック接続イベントを生成することを指定します。この追加の詳細には、以下の情報が含まれます。

- デッドロックが生じたときにアプリケーションが実行していたステートメントに関する情報。例えば、ステートメント・テキスト。
- デッドロックが生じた際にアプリケーションによって保持されていたロック。パーティション・データベース環境では、デッドロックが生じたときにアプリケーションがロックを待機していたデータベース・パーティション上に保持されたロックだけが、これに含まれます。パーティション表の場合は、データ・パーティション ID が含まれません。

HISTORY

イベント・モニター・データに、以下のものも含まれることを指定します。

- 特定のノードの現行作業単位のすべてのステートメントの履歴 (直前の作業単位で開かれた WITH HOLD カーソルも含む)。非コミット読み取り (UR) 分離レベルで発行された SELECT ステートメントは、ステートメント履歴に含まれません。
- 各 SQL ステートメントのステートメント・コンパイル環境 (バイナリー・フォーマット) (使用可能な場合)

VALUES

イベント・モニター・データに、以下のものも含まれることを指定します。

CREATE EVENT MONITOR

- 各 SQL ステートメントの入力変数として使用されるデータ値。これらのデータ値には LOB データ、long データ、構造化タイプ・データ、または XML データは含まれません。

単一の CREATE EVENT MONITOR ステートメントで指定できるのは、DEADLOCKS、DEADLOCKS WITH DETAILS、DEADLOCKS WITH DETAILS HISTORY、または DEADLOCKS WITH DETAILS HISTORY VALUES のうちの 1 つだけです (SQLSTATE 42613)。

TABLESPACES

最後のアプリケーションがデータベースから切断された場合に、イベント・モニターが各表スペースに関する表スペース・イベントを記録することを指定します。

BUFFERPOOLS

最後のアプリケーションがデータベースから切断された場合に、イベント・モニターがバッファ・プール・イベントを記録することを指定します。

ACTIVITIES

アクティビティが実行を完了したとき、またはイベントが WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャによって起動された場合は実行完了前に、イベント・モニターがアクティビティ・イベントを記録することを指定します。アクティビティは、以下の要件のどちらかを満たさなければなりません。

- COLLECT ACTIVITY DATA セットを持つサービス・クラスまたはワークロードに属する
- 関連した作業アクションが COLLECT ACTIVITY DATA である作業クラスに属する
- COLLECT ACTIVITY DATA 節が指定されたしきい値に違反するアクティビティであると識別される
- 完了前に WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャの呼び出し時に識別されている

THRESHOLD VIOLATIONS

しきい値違反が起きた場合にイベント・モニターがしきい値違反イベントを記録することを指定します。こうしたイベントは、完了時だけでなく、アクティビティのファイル内のどのポイントでも記録されます。

STATISTICS

イベント・モニターが以下のタイミングでサービス・クラス、ワークロード、または作業クラスのイベントを記録することを指定します。

- *period* 分おき (*period* は **wlm_collect_int** データベース構成パラメータの値)
- **wlm_collect_stats** プロシージャが呼び出されたとき

CONNECTIONS

アプリケーションがデータベースから切断された場合に、イベント・モニターが接続イベントを記録することを指定します。

STATEMENTS

SQL ステートメントの実行が完了した時点で、イベント・モニターがステートメント・イベントを記録することを指定します。

TRANSACTIONS

トランザクションが完了した時点で (すなわち、コミットまたはロールバックの操作が行われた時点で)、イベント・モニターがトランザクション・イベントを記録することを指定します。

WHERE event-condition

どの接続が CONNECTION、STATEMENT、または TRANSACTION イベントを引き起こすかを判別するフィルターを定義します。特定の接続に関してイベント条件 (*event condition*) の結果が真の場合、その接続は要求されたイベントを生成します。

この節は、WHERE 節の特殊な形式です。標準的な検索条件と混同しないようにしてください。

アプリケーションが特定のイベント・モニターに対するイベントを生成するかどうかを判別するために、この WHERE 節は次のように評価されます。

- イベント・モニターが初めてオンになった時点でアクティブである各接続がまず評価されます。
- それ以後のデータベースへの新たな接続は、その接続時に評価されます。

WHERE 節は各イベントごとに評価されるわけではありません。

WHERE 節の指定がない場合、指定したイベント・タイプのイベントがすべてモニターされます。

event-condition は、データベース・コード・ページ内で、32 678 バイト以内の長さでなければなりません (SQLSTATE 22001)。

APPL_ID

該当の接続が CONNECTION、STATEMENT、または TRANSACTION のいずれのイベント (指定による) を生成する必要があるかどうかを判別するために各接続のアプリケーション ID と *comparison-string* (比較ストリング) とを比較しなければならないことを指定します。

AUTH_ID

該当の接続が CONNECTION、STATEMENT、または TRANSACTION のいずれのイベント (指定による) を生成する必要があるかどうかを判別するために各接続の許可 ID と *comparison-string* とを比較しなければならないことを指定します。

APPL_NAME

該当の接続が CONNECTION、STATEMENT、または TRANSACTION のいずれのイベント (指定による) を生成する必要があるかどうかを判別するために各接続のアプリケーション・プログラム名と *comparison-string* とを比較しなければならないことを指定します。

アプリケーション・プログラム名は、(最後のパス区切り記号の後の) アプリケーション・プログラム・ファイル名の最初の 20 バイトです。

comparison-string

データベースに接続する各アプリケーションの APPL_ID、AUTH_ID、または APPL_NAME と比較するストリングを指定します。

comparison-string (比較ストリング) は、ストリング定数でなければなりません (ホスト変数や他のストリング式は使用できません)。

CREATE EVENT MONITOR

WRITE TO

データの出力先をこの後に指定します。

TABLE

イベント・モニターのデータの出力先が一連のデータベース表であることを示します。イベント・モニターは、データ・ストリームを 1 つまたは複数の論理データ・グループに分離し、各グループを別個の表に挿入します。ターゲット表のあるグループのデータは保持されますが、ターゲット表のないグループのデータは破棄されます。グループに含まれる各モニター・エレメントは、同じ名前の表列にマップされます。対応する表列を持つエレメントだけが表に挿入されます。他のエレメントは破棄されます。

evm-group-info

論理データ・グループのターゲット表を定義します。この節は、記録される各グループごとに指定しなければなりません。しかし

evm-group-info 節が指定されない場合には、イベント・モニター・タイプのすべてのグループが記録されます。

evm-group

ターゲット表が定義する論理データ・グループを指定します。以下の表に示されているように、値はイベント・モニターのタイプに基づきます。

イベント・モニターのタイプ	evm-group 値
データベース	<ul style="list-style-type: none">• DB• CONTROL¹• DBMEMUSE
表	<ul style="list-style-type: none">• TABLE• CONTROL¹
デッドロック	<ul style="list-style-type: none">• CONNHEADER• DEADLOCK• DLCONN• CONTROL¹
詳細なデッドロック	<ul style="list-style-type: none">• CONNHEADER• DEADLOCK• DLCONN²• DLLOCK³• CONTROL¹
詳細な履歴のあるデッドロック	<ul style="list-style-type: none">• CONNHEADER• DEADLOCK• DLCONN²• DLLOCK³• STMTHIST• CONTROL¹

イベント・モニターのタイプ	evm-group 値
詳細な履歴値のあるデッドロック	<ul style="list-style-type: none"> • CONNHEADER • DEADLOCK • DLCONN² • DLLOCK³ • STMT HIST • STMT VALS • CONTROL¹
表スペース	<ul style="list-style-type: none"> • TABLESPACE • CONTROL¹
バッファ・プール	<ul style="list-style-type: none"> • BUFFERPOOL • CONTROL¹
接続	<ul style="list-style-type: none"> • CONNHEADER • CONN • CONTROL¹ • CONNMEMUSE
ステートメント	<ul style="list-style-type: none"> • CONNHEADER • STMT • SUBSECTION⁴ • CONTROL¹
トランザクション	<ul style="list-style-type: none"> • CONNHEADER • XACT • CONTROL¹
アクティビティ	<ul style="list-style-type: none"> • ACTIVITY • ACTIVITYSTMT • ACTIVITYVALS • CONTROL
統計	<ul style="list-style-type: none"> • QSTATS • SCSTATS • WCSTATS • WLSTATS • HISTOGRAMBIN • CONTROL
しきい値違反	<ul style="list-style-type: none"> • THRESHOLDVIOLATIONS • CONTROL

イベント・モニターのタイプ evm-group 値

¹ 論理データ・グループ dbheader (conn_time エレメントのみ)、開始およびオーバーフローは、CONTROL グループに書き込まれます。イベント・モニターがブロック化されておらず、イベントが破棄された場合に、オーバーフロー・グループが書き込まれます。

² DETAILED_DLCONN イベントに相当します。

³ 各 DETAILED_DLCONN イベント内で発生する LOCK 論理データに相当します。

⁴ パーティション・データベース環境に対してのみ作成されます。

target-table-info

グループのターゲット表を指定します。 *target-table-info* の値が指定されないと、CREATE EVENT MONITOR の処理は以下のように続行されます。

- 派生した表名が使用されます (以下で説明します)。
- デフォルトの表スペースが選択されます (以下で説明します)。
- すべてのエレメントを含めます。
- PCTDEACTIVATE および TRUNC は指定されません。

TABLE table-name

ターゲット表の名前を指定します。ターゲット表は、非パーティション表でなければなりません。名前が修飾なしの場合には、現行の許可 ID のスキーマが表スキーマのデフォルトになります。名前が指定されない場合、以下のように非修飾名は *evm-group* および *event-monitor-name* から派生します。

```
substring(evm-group CONCAT " "
         CONCAT event-monitor-name,1,128)
```

IN tablespace-name

表を作成する表スペースの名前を指定します。表スペース名が指定されない場合には、以下のように表スペースが選択されます。

```
IF ユーザーが USE 特権を持っている
   表スペース IBMDEFAULTGROUP が存在する場合には
   THEN それを選択します。
ELSE IF ユーザーが USE 特権をもっている
   表スペース IBMDEFAULTGROUP が存在する場合には
   THEN それを選択します。
ELSE エラーを出します (SQLSTATE 42727)
```

PCTDEACTIVATE integer

表が DMS 表スペースに作成される場合には、PCTDEACTIVATE パラメーターは、どの程度表スペースが満たされた時点でイベント・モニターが非活動化されるかを指定します。パーセンテージを表す値は、0 から 100 の範囲で指定可能です。デフォルト値は 100 です (表スペース

が完全にいっぱいになるときにイベント・モニターが非活動化されることを意味します)。このオプションは、SMS 表スペースでは指定できません。

ターゲット表スペースで自動サイズ変更が使用可能になっている場合には、PCTDEACTIVATE を 100 に設定することをお勧めします。

TRUNC

STMT_TEXT および STMT_VALUE_DATA 列を VARCHAR(*n*) で定義することを指定します。ここで *n* は、表行に収めることができる最大サイズです。この場合、*n* バイトより大きいデータは切り捨てられます。以下の例は、*n* の値の計算方法を例示しています。次のような前提があります。

- 表は、32K ページを使用する表スペースに作成されます。
- 表内の他のすべての列の合計は、357 バイトです。

この場合、表の最大行サイズは 32677 バイトです。ですから、エレメントは VARCHAR(32316) に定義されます (つまり、32677 - 357 - 4)。TRUNC が指定されない場合には、列は CLOB(64K) に定義されます。STMT_TEXT が STMT イベント・グループ、STMT_HISTORY イベント・グループ、および DLCONN イベント・グループで見つかることに注意してください (詳細なイベント・モニターのあるデッドロックの場合)。STMT_VALUE_DATA は DATA_VALUE イベント・グループで見つかります。

INCLUDES

続くエレメントが表に含まれるように指定します。

EXCLUDES

続くエレメントが表に含まれないように指定します。

element

モニター・エレメントを指定します。エレメント情報は、以下の書式で提供されます。

- エレメント情報を指定しない。この場合、すべてのエレメントは CREATE TABLE ステートメントに組み込まれます。
- 書式 INCLUDES (*element1*, *element2*, ..., *elementn*) のエレメントが組み込まれるように指定する。こうしたエレメントの表列だけが作成されます。
- 書式 EXCLUDES (*element1*, *element2*, ..., *elementn*) のエレメントを除外するよう指定する。指定されたすべてのエレメントを除外した表列だけが作成されます。

db2evtbl コマンドを使用して、グループのエレメントの完全なリストを含む CREATE EVENT MONITOR ステートメントを構築します。

BUFFERSIZE *pages*

イベント・モニター・バッファのサイズを指定します (4K ページ単位)。表イベント・モニターはバッファからのすべてのデータを挿入し、バッファが処理されると **COMMIT** を発行します。バッファが大きいほど、イベント・モニターによって使用されるコミット有効範囲は大きくなります。活動頻度の高いイベント・モニターには、比較的活動頻度の低いイベント・モニターよりも大きいバッファを用意する必要があります。モニターが開始されると、指定したサイズの 2 つのバッファが割り振られます。イベント・モニターは、二重バッファリングを使用して、非同期入出力を可能にします。

各バッファのデフォルト・サイズは 4 ページです (16K バッファが 2 個割り振られます)。最小サイズは 1 ページです。バッファはヒープから割り振られるので、バッファの最大サイズはモニター・ヒープのサイズによって制約されます。多くのイベント・モニターを同時に使用する場合には、**mon_heap_sz** データベース構成パラメーターのサイズを大きくします。

BLOCKED

エージェントが 2 つのイベント・バッファがいっぱいであると判断した場合、イベントを生成するそのエージェントはイベント・バッファがディスクへ書き込まれるのを待機しなければならないことを指定します。イベント・データが失われるのを防止する場合には、**BLOCKED** を選択する必要があります。これはデフォルトのオプションです。

NONBLOCKED

エージェントが 2 つのイベント・バッファがいっぱいであると判断した場合、イベントを生成するそのエージェントはイベント・バッファがディスクへ書き込まれるのを待機しないことを指定します。**NONBLOCKED** の指定を伴うイベント・モニターは、**BLOCKED** の指定を伴うイベント・モニターほどには、データベース操作の速度を低下させません。ただし、**NONBLOCKED** のイベント・モニターは、アクティブ度の高いシステムではデータの消失の可能性が高くなります。

PIPE

イベント・モニター・データの出力先が名前付きパイプであることを指定します。イベント・モニターは、データを単一のストリーム (単一の無限に長いファイルであるかのように) でパイプに書き込みます。データをパイプに書き込む時点で、イベント・モニターはブロック化書き込みを行いません。パイプ・バッファに余地がない場合、イベント・モニターはそのデータを廃棄します。データを失いたくない場合、モニターするアプリケーション側でデータを迅速に読み取る必要があります。

pipe-name

イベント・モニターがデータを書き込むパイプの名前 (AIX では FIFO) を指定します。

パイプの命名規則は、プラットフォームごとに異なります。UNIX オペレーティング・システムでは、パイプ名はファイル名と同様に扱われ

ます。したがって、相対パイプ名を使用することができ、相対パス名と同様に扱われます (下記の *path-name* を参照)。ただし、Windows では、パイプ名に関して特殊な構文があるので、絶対パイプ名が必要です。

パイプの存在は、イベント・モニターの作成時には検査されません。モニター・アプリケーションは、イベント・モニターが活動化された時点で、読み取り用パイプを作成し、オープンしておく必要があります。この時点でパイプが使用不可である場合には、イベント・モニターはオフになり、エラーがログに記録されます。(つまり、AUTOSTART オプションの結果としてイベント・モニターがデータベースの開始時に活動化された場合、イベント・モニターはエラーをシステム・エラー・ログに記録します。) SET EVENT MONITOR STATE SQL ステートメントによってイベント・モニターが活動化された場合、そのステートメントはエラーになります (SQLSTATE 58030)。

FILE

イベント・モニターのデータの出力先がファイル (または一連のファイル) であることを示します。イベント・モニターは、拡張子 “*evt*” が付いた一連の 8 文字の番号のファイル (例えば、00000000.*evt*、00000001.*evt*、および 00000002.*evt*) に、データ・ストリームを書き出します。データが細かく分割されている場合でも、データは 1 つの論理ファイルと見なす必要があります (つまり、データ・ストリームの最初はファイル 00000000.*evt* の最初のバイトであり、データ・ストリームの最後は、ファイル *nnnnnnnnn.evt* の最後のバイトになります)。

各ファイルの最大サイズとファイルの最大数とを指定することができます。イベント・モニターは、1 つのイベント・レコードを 2 つのファイルに分割することはありません。ただしイベント・モニターは、互いに関連する複数のレコードを 2 つの異なるファイルに記録する場合があります。そのデータを使用するアプリケーションでは、イベント・ファイルの処理時にこのような関連する情報を追跡する必要があります。

path-name

イベント・モニターがイベント・ファイルのデータを書き込む先のディレクトリーの名前を指定します。パスはサーバーにおいて既知である必要があります。ただし、パス自体は別のデータベース・パーティションにある可能性があります (たとえば、UNIX システムでは、NFS にマウントされたファイルである場合もあります)。 *path-name* (パス名) の指定には、ストリング定数を使用する必要があります。

ディレクトリーは、CREATE EVENT MONITOR の時に存在している必要はありません。ただし、イベント・モニターが活動化される時点で、ターゲット・パスの存否の検査が行われます。その時点で、ターゲット・パスが存在しない場合は、エラー (SQLSTATE 428A3) になります。

絶対パス (AIX の場合にルート・ディレクトリーで始まるパス、または Windows の場合にディスク ID で始まるパス) を指定すると、指定したパスが使用されます。相対パス (ルートから始まっていないパス) が指定されている場合は、データベース・ディレクトリーの DB2EVENT ディレクトリーからの相対パスが使用されます。

相対パスが指定されている場合、それを絶対パスに変換するために DB2EVENT ディレクトリーが使用されます。したがって、絶対パスと相対パスの間に区別はありません。絶対パスは SYSCAT.EVENTMONITORS カタログ・ビューに保管されます。

複数のイベント・モニターに指定するターゲット・パスを同じパスにすることはできません。ただし、イベント・モニターの 1 つが最初に活動化されると、ターゲット・ディレクトリーが空でないかぎり、他のイベント・モニターはいずれも活動化することはできなくなります。

file-options

ファイル形式のオプションを指定します。

MAXFILES NONE

イベント・モニターが作成するイベント・ファイルの数の制限がないことを指定します。これはデフォルトです。

MAXFILES *number-of-files*

特定の 1 つのイベント・モニターについて、1 時点で存在するイベント・モニター・ファイルの数の限界があることを指定します。イベント・モニターがファイルをもう 1 つ作成しなければならない場合、ディレクトリー内の .evt ファイルの数が *number-of-files* よりも少ないかどうかを検査されます。すでにこの限界に達している場合、イベント・モニターはオフになります。

書き込み済みのイベント・ファイルを、アプリケーションがディレクトリーから削除した場合は、イベント・モニターが作成するファイルの合計数が *number-of-files* を超えることがあります。このオプションの使用によって、ユーザーはイベント・データによるディスク・スペースの消費量が指定量を超えないようにすることができます。

MAXFILESIZE *pages*

各イベント・モニター・ファイルのサイズに限界があることを指定します。イベント・モニターは、新しいイベント・レコードをファイルに書き込む場合、そのファイルが *pages* (4K ページ単位のページ数) を超えないかどうかを調べます。結果のファイルが大きすぎる場合、イベント・モニターはその次のファイルに切り替えます。このオプションのデフォルト値は次のとおりです。

- Windows - 200 個の 4K ページ
- UNIX - 1000 個の 4K ページ

ページ数は、少なくともイベント・バッファのサイズ (ページ数) よりも大きくなければなりません。この要件が満たされていない場合、エラー (SQLSTATE 428A4) になります。

MAXFILESIZE NONE

ファイルのサイズに限界を設定しないことを指定します。MAXFILESIZE NONE を指定すると、MAXFILES 1 も指定する必要があります。このオプションは、特定のイベント・モニターのイベント・データすべてを 1 つのファイルに入れることを示します。このような場合、イベント・ファイルは 00000000.evt だけになります。

BUFFERSIZE *pages*

イベント・モニター・バッファのサイズを指定します (4K ページ単位)。イベント・モニターのパフォーマンスを向上させるために、すべてのイベント・モニターのファイル入出力はバッファに入れます。バッファが大きいくほど、イベント・モニターによって行われる入出力は少なくなります。活動頻度の高いイベント・モニターには、比較的活動頻度の低いイベント・モニターよりも大きいバッファを用意する必要があります。モニターが開始されると、指定したサイズの 2 つのバッファが割り振られます。イベント・モニターは、二重バッファリングを使用して、非同期入出力を可能にします。

各バッファのデフォルト・サイズは 4 ページです (16K バッファが 2 個割り振られます)。最小サイズは 1 ページです。バッファの最大サイズは、MAXFILESIZE パラメータの値の他に、モニター・ヒープのサイズによっても制約されます。バッファはそのヒープから割り振られるからです。多くのイベント・モニターを同時に使用する場合には、`mon_heap_sz` データベース構成パラメータのサイズを大きくします。

データをパイプに書き込むイベント・モニターにも、それぞれサイズが 1 ページの 2 つの内部 (構成不能) バッファがあります。これらのバッファも、モニター・ヒープ (MON_HEAP) から割り振られます。出力先がパイプである各アクティブ・イベント・モニターごとに、データベース・ヒープのサイズを 2 ページ分大きくしてください。

BLOCKED

エージェントが 2 つのイベント・バッファがいっぱいであると判断した場合、イベントを生成するそのエージェントはイベント・バッファがディスクへ書き込まれるのを待機しなければならないことを指定します。イベント・データが失われるのを防止する場合には、BLOCKED を選択する必要があります。これはデフォルトのオプションです。

NONBLOCKED

エージェントが 2 つのイベント・バッファがいっぱいであると判断した場合、イベントを生成するそのエージェントはイベント・バッファがディスクへ書き込まれるのを待機しないことを指定します。NONBLOCKED の指定を伴うイベント・モニターは、BLOCKED の指定を伴うイベント・モニターほどには、データベース操作の速度を低下させません。ただし、NONBLOCKED のイベント・モニターは、アクティブ度の高いシステムではデータの消失の可能性が高くなります。

APPEND

イベント・モニターがオンになった時点でイベント・データ・ファイルがすでに存在する場合、そのイベント・モニターは新しいイベント・データをデータ・ファイルの既存のストリームに付加するように指定します。イベント・モニターが再活動化されると、それは

CREATE EVENT MONITOR

オフにならなかったかのように、イベント・ファイルへの書き込みを再開します。 APPEND はデフォルトのオプションです。

新しく作成されたイベント・モニターがイベント・データを書き込むディレクトリーに既存のイベント・データがない場合、 CREATE EVENT MONITOR 時に APPEND オプションは適用されません。

REPLACE

イベント・モニターがオンになった時点でイベント・データ・ファイルがすでに存在する場合、そのイベント・モニターが、イベント・ファイルをすべて削除して、ファイル 00000000.evt へのデータの書き込みを開始するように指定します。

MANUALSTART

SET EVENT MONITOR STATE ステートメントを使用してイベント・モニターを手動で活動化することを指定します。活動化された MANUALSTART イベント・モニターは、SET EVENT MONITOR STATE ステートメントを使用するか、インスタンスを停止することによってのみ非活動状態にできます。これは、非 WLM イベント・モニター (DB2DETAILDEADLOCK イベント・モニターを除く) のデフォルトの動作です。

AUTOSTART

イベント・モニターを実行するデータベース・パーティションを活動化した時点で、イベント・モニターも自動的に活動化することを指定します。これは、WLM イベント・モニターおよび DB2DETAILDEADLOCK イベント・モニターのデフォルトの動作です。

ON DBPARTITIONNUM *db-partition-number*

ファイルまたはパイプのイベント・モニターを実行するデータベース・パーティションを指定します。モニター有効範囲が LOCAL と定義された場合、指定されたパーティションについてのみデータが収集されます。モニター有効範囲が GLOBAL と定義された場合、すべてのデータベース・パーティションがデータを収集し、指定の番号のデータベース・パーティションに報告を行います。I/O コンポーネントは指定のデータベース・パーティションで物理的に稼働し、指定されたファイルまたはパイプにレコードを書き込みます。

この節は、表イベント・モニターには無効です。パーティション・データベース環境では、write-to-table イベント・モニターは、ターゲット表のための表スペースが定義されているすべてのデータベース・パーティションで、イベントの実行と書き込みを行います。

この節を指定しない場合は、現在の (アプリケーションの) 接続先のデータベース・パーティション番号が使用されます。

LOCAL

イベント・モニターはモニター稼働しているデータベース・パーティションについてのみ報告します。この報告は、データベース活動の部分的なトレースです。これはデフォルトです。

この節は、表イベント・モニターには無効です。

GLOBAL

イベント・モニターはすべてのデータベース・パーティションについて報告しま

す。パーティション・データベースの場合、GLOBAL と定義できるのは DEADLOCKS イベント・モニターのみです。

この節は、表イベント・モニターには無効です。

規則

- 各イベント・タイプ (DATABASE、TABLES、DEADLOCK、...) は、特定のイベント・モニターの定義に 1 回だけ指定できます。

注

- イベント・モニターの定義は、SYSCAT.EVENTMONITORS カタログ・ビューに記録されます。イベント自体は、SYSCAT.EVENTS カタログ・ビューに記録されます。ターゲット表の名前は、SYSCAT.EVENTTABLES カタログ・ビューに記録されます。
- DEADLOCKS ではなく DEADLOCKS WITH DETAILS を使用すると、パフォーマンスに影響します。デッドロックが生じると、データベース・マネージャーは、追加のデッドロック情報を記録する余分の時間を必要とします。
- 通常 CONNHEADER イベントは、接続が確立されるといつでも書き込まれます。しかしイベント・モニターが DEADLOCKS WITH DETAILS に対してのみ作成される場合には、CONNHEADER イベントは、接続が初めてデッドロックになったときだけ書き込まれます。
- 複数のデータベース・パーティションを持つデータベースでは、DEADLOCKS イベント・タイプを持つ FILE および PIPE イベント・モニターで ON DBPARTITIONNUM 節を使用し、イベント・モニター自体が存在する場所を示すことができます。そうすれば、他のデータベース・パーティションからの情報が関連のあるものである場合に、その情報はこの場所に送られて、処理されます。
- 複数のデータベース・パーティションを持つデータベースの場合、デッドロック・イベント・モニターは、デッドロックに関係するロックを持つアプリケーションに関する情報を、関係するそのロックが存在していたすべてのデータベース・パーティションから受信します。アプリケーションの接続先データベース・パーティション (アプリケーション・コーディネーター・パーティション) が、関係しているデータベース・パーティションの 1 つではない場合、そのデータベース・パーティションからはデッドロック・イベントに関する情報を受信しません。
- BUFFERSIZE パラメーターは、STMT、STMT_HISTORY、DATA_VALUE、および DETAILED_DLCONN イベントのサイズを制約します。STMT または STMT_HISTORY イベントがバッファの大きさに合わない場合、ステートメント・テキストを切り捨ててそのイベントを切り捨てます。DETAILED_DLCONN イベントがバッファの大きさに合わない場合、ロックを除去してそのイベントを切り捨てます。それでもまだ大きさが合わない場合には、ステートメント・テキストを切り捨てます。DATA_VAL イベントがバッファの大きさに合わない場合、データ値を切り捨てます。

イベント・モニター WITH DETAILS HISTORY VALUES (および、程度はそれほどではないものの WITH DETAILS HISTORY) は、ステートメントおよびそれらのデータ値を追跡するために、モニター・ヒープ・スペースのかなりの量を使用します。詳しくは、**mon_heap_sz** データベース・マネージャー構成パラメーターの説明を参照してください。

CREATE EVENT MONITOR

- イベント・モニターを実行するデータベース・パーティションがアクティブでない場合は、次回そのデータベース・パーティションを活動化した時点で、イベント・モニターも活動化されます。
- イベント・モニターは、活動化の後に、明示的に非活動化されるか、インスタンスがリサイクルされるまで、自動始動のイベント・モニターのように動作します。つまり、データベース・パーティションが非活動化された時点でイベント・モニターがアクティブであれば、そのデータベース・パーティションがそれ以降に再び活動化された時点で、イベント・モニターも明示的に再活動化されます。
- **表書き込みイベント・モニター:**
 - 一般的な注:
 - すべてのターゲット表は、CREATE EVENT MONITOR ステートメントの実行時に作成されます。
 - 何らかの理由により表の作成に失敗すると、エラーがアプリケーション・プログラムに戻され、CREATE EVENT MONITOR ステートメントは失敗します。
 - 1 つのターゲット表は、1 つのイベント・モニターでのみ使用可能です。CREATE EVENT MONITOR 処理時に、別のイベント・モニターによって既に定義されているターゲット表が検出されると、CREATE EVENT MONITOR ステートメントは失敗し、エラーがアプリケーション・プログラムに戻されます。表名が SYSCAT.EVENTTABLES カタログ・ビューの値と一致する場合には、その表は別のイベント・モニターによって使用されるよう定義されています。
 - CREATE EVENT MONITOR 処理時に、表が既に存在するものの別のイベント・モニターによって使用されるよう定義されていない場合には、表は作成されず、処理は続行されます。警告が、アプリケーション・プログラムに出されます。
 - CREATE EVENT MONITOR ステートメントが実行される前に、すべての表スペースが存在しなければなりません。CREATE EVENT MONITOR ステートメントは、表スペースを作成しません。
 - 指定されている場合、LOCAL および GLOBAL キーワードは無視されます。WRITE TO TABLE イベント・モニターの場合、イベント・モニターの出力処理またはスレッドは、インスタンスの各データベース・パーティションで開始され、そうした個々の処理は実行しているデータベース・パーティションのデータのみを報告します。
 - フラット・モニター・ログ・ファイルからの以下のイベント・タイプまたはパイプ・フォーマットは、表書き込みイベント・モニターにより記録されません。
 - LOG_STREAM_HEADER
 - LOG_HEADER
 - DB_HEADER (エレメント db_name および db_path は記録されません。エレメント conn_time は CONTROL に記録されます。)
 - パーティション・データベース環境では、表スペースが存在するデータベース・パーティション上のターゲット表だけにデータが書き込まれます。ターゲット表のための表スペースがいずれかのデータベース・パーティションに存在しない場合は、そのターゲット表についてのデータは無視されます。こ

の動作によってユーザーは、特定のデータベース・パーティション上のみ存在する表スペースを作成して、モニターするデータベース・パーティションのサブセットを選択できます。

パーティション・データベース環境で、いくつかのターゲット表がデータベース・パーティションに存在しないものの、その同じデータベース・パーティションに他のターゲット表がある場合には、そのデータベース・パーティションにあるターゲット表についてのデータだけが記録されます。

- ユーザーは、すべてのターゲット表を手動で整理する必要があります。
- 表列:

- 表の列名は、イベント・モニター・エレメント ID と一致します。タイプ `sqlm_time` (経過時間) のモニター変数は、例外です。このタイプの列名は `TYPE_NAME_S` および `TYPE_NAME_MS` であり、それぞれ秒単位およびマイクロ秒単位で時間を保管する列を表しています。対応するターゲット表列のないイベント・モニター・エレメントは、無視されます。
- `db2evtbl` コマンドを使用して、グループのエレメントの完全なリストを含む `CREATE EVENT MONITOR` コマンドを構築します。
- モニター・エレメントに使用されている列のタイプは、以下のマッピングに related します。

<code>SQLM_TYPE_STRING</code>	<code>CHAR[n]</code> , <code>VARCHAR[n]</code> or <code>CLOB(n)</code> (イベント・モニター内のデータが <code>n</code> バイトを超えた場合切り捨てが行われる。)
<code>SQLM_TYPE_U8BIT</code> and <code>SQLM_TYPE_8BIT</code>	<code>SMALLINT</code> , <code>INTEGER</code> or <code>BIGINT</code>
<code>SQLM_TYPE_16BIT</code> and <code>SQLM_TYPE_U16BIT</code>	<code>SMALLINT</code> , <code>INTEGER</code> or <code>BIGINT</code>
<code>SQLM_TYPE_32BIT</code> and <code>SQLM_TYPE_U32BIT</code>	<code>INTEGER</code> or <code>BIGINT</code>
<code>SQLM_TYPE_U64BIT</code> and <code>SQLM_TYPE_64BIT</code>	<code>BIGINT</code>
<code>sqlm_timestamp</code>	<code>TIMESTAMP</code>
<code>sqlm_time(elapsed time)</code>	<code>BIGINT</code>
<code>sqlca:</code>	
<code>sqlerrmc</code>	<code>VARCHAR[72]</code>
<code>sqlstate</code>	<code>CHAR[5]</code>
<code>sqlwarn</code>	<code>CHAR[11]</code>
other fields	<code>INTEGER</code> or <code>BIGINT</code>

- 列は、`NOT NULL` になるよう定義されています。
- `CLOB` 列のある表のパフォーマンスは `VARCHAR` 列を含む表より劣るので、`STMT evm-group` 値 (または、`DEADLOCKS WITH DETAILS` イベント・タイプを使用する場合に `DLCONN evm-group` 値) を指定する場合には、`TRUNC` キーワードの使用を考慮してください。
- 他のターゲット表とは異なり、`CONTROL` 表の列はモニター・エレメント ID と一致しません。列は、以下のように定義されます。

列名	データ・タイプ	NULL 可能	説明
-----	-----	-----	-----
<code>PARTITION_KEY</code>	<code>INTEGER</code>	N	分散キー (パーティション・データベースのみ)
<code>PARTITION_NUMBER</code>	<code>INTEGER</code>	N	データベース・パーティション番号 (パーティション・データベースのみ)
<code>EVMONNAME</code>	<code>VARCHAR(128)</code>	N	イベント・モニターの名前
<code>MESSAGE</code>	<code>VARCHAR(128)</code>	N	<code>MESSAGE_TIME</code> 列の性質を記述します。 これは、次のいずれかになります。 - <code>FIRST_CONNECT</code> (活動化の前にデータベースに最初に接

い場合は、そのターゲット表にはフラグが立てられ、そのターゲット表を宛先とするデータが実行時処理中にドロップされるようにします。

- イベント・モニターが活動化する際にターゲット表が存在しない場合 (またはパーティション・データベース環境で、表スペースがパーティション・データベースにない場合) には、活動化は続行され、この表に挿入されるはずであったデータは無視されます。
- 活動化処理は、各ターゲット表の妥当性検査をします。妥当性検査がうまくいかないと、イベント・モニターの活動化は失敗し、管理ログにメッセージが書き込まれます。
- パーティション・データベース環境における活動化の際に、FIRST_CONNECT および EVMON_START の CONTROL 表行は、カタログ・データベース・パーティションでのみ挿入されます。これには、コントロール表の表スペースがカタログ・データベース・パーティションに存在することが必要です。カタログ・データベース・パーティションにない場合には、挿入は実行されません。
- パーティション・データベース環境では、表書き込みイベント・モニターがアクティブであるときにパーティションがまだアクティブでない場合には、イベント・モニターは次にそのパーティションがアクティブ化されたときにアクティブ化されます。
- ランタイム:
 - イベント・モニターは DBADM 権限で実行されます。
 - イベント・モニターがアクティブであるときに、ターゲット表への挿入操作が失敗すると、
 - コミットされていない変更がロールバックされます。
 - メッセージが、管理ログに書き込まれます。
 - イベント・モニターが非活動になります。
 - イベント・モニターがアクティブである場合には、イベント・モニター・バッファの処理を終えるとローカル COMMIT が実行されます。
 - パーティション・データベース環境では、アプリケーション・コーディネーターのデータベース・パーティションで実行されているイベント・モニター処理によって、長さが 65 535 バイトまで可能な実際のステートメント・テキストだけが保管されます (STMT または DLCONN 表に)。他のデータベース・パーティションでは、この値の長さはゼロです。
 - 非パーティションのデータベース環境では、最後のアプリケーションが終了すると (それまでにデータベースが明示的に活動化されていないと)、表書き込みイベント・モニターはすべて非活動化されます。パーティション・データベース環境では、カタログ・パーティションが非活動化されると表書き込みイベント・モニターが非活動化されます。
 - DROP EVENT MONITOR ステートメントはターゲット表をドロップしません。
 - 表書き込みイベント・モニターがアクティブになると、イベント・モニターは、それがアクティブである間にターゲット表が変更されることを防ぐため、常に各ターゲット表に対する IN 表ロックを獲得します。表ロックは、イベント・モニターがアクティブである間は、すべての表において維持されます。ターゲット表のいずれかにおいて排他的アクセスが必要である場合

CREATE EVENT MONITOR

(例えば、ユーティリティが実行される場合)には、まずイベント・モニターを非アクティブにして、そのようなアクセスを試行する前に表ロックを解放します。

• 互換性

- 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。
 - *target-table-info* 節または *wlm-target-table-info* 節では、コンマを使って複数のオプションを分離できます。

例

例 1: 次の例では、SMITHPAY と呼ばれるイベント・モニターを作成します。このイベント・モニターは、データベースと、JSMITH 許可 ID が所有する PAYROLL アプリケーションによって実行される SQL ステートメントに関するイベント・データを収集します。データは、絶対パス/home/jsmith/event/smithpay/ に付加されます。最大 25 のファイルが作成されます。各ファイルの最大長は 4K ページ 1 024 個分です。ファイル入出力は非ブロック化されます。

```
CREATE EVENT MONITOR SMITHPAY
FOR DATABASE, STATEMENTS
WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
WRITE TO FILE '/home/jsmith/event/smithpay'
MAXFILES 25
MAXFILESIZE 1024
NONBLOCKED
APPEND
```

例 2: 次の例では、DEADLOCKS_EVTS と呼ばれるイベント・モニターを作成します。このイベント・モニターは、デッドロック・イベントを収集して、それらを相対パス DLOCKS に書き込みます。1 つのファイルに書き込まれ、ファイル・サイズに上限はありません。イベント・モニターが活動化されるたびに、ファイル 00000000.evt が存在する場合にはそこにイベント・データが付加されます。このイベント・モニターは、データベースが開始されるたびに開始されます。入出力はデフォルト解釈によりブロック化されます。

```
CREATE EVENT MONITOR DEADLOCK_EVTS
FOR DEADLOCKS
WRITE TO FILE 'DLOCKS'
MAXFILES 1
MAXFILESIZE NONE
AUTOSTART
```

例 3: この例では、DB_APPLS と呼ばれるイベント・モニターを作成します。このイベント・モニターは、接続イベントを収集し、それらのデータを名前付きパイプ /home/jsmith/applpipe に書き込みます。

```
CREATE EVENT MONITOR DB_APPLS
FOR CONNECTIONS
WRITE TO PIPE '/home/jsmith/applpipe'
```

例 4: この例では、パーティション・データベース環境であることが前提で、FOO と呼ばれるイベント・モニターを作成します。このイベント・モニターは SQL ステートメント・イベントを収集し、以下の導出された名前で SQL 表に書き込みます。

- CONNHEADER_FOO

- STMT_FOO
- SUBSECTION_FOO
- CONTROL_FOO

表スペース情報が提供されていないので、IN *tablespace-name* 節で説明されていた規則に基づいて、システムが選択した表スペースにすべての表が作成されます。すべての表がこれらのグループのすべてのエレメントを含みます (つまり、列名がエレメント名と同じになるように列は定義されます。)

```
CREATE EVENT MONITOR FOO
FOR STATEMENTS
WRITE TO TABLE
```

例 5 この例では、パーティション・データベース環境であることが前提で、BAR と呼ばれるイベント・モニターを作成します。このイベント・モニターは SQL ステートメントおよびトランザクション・イベントを収集し、以下のように表に書き込みます。

- STMT グループからのすべてのデータは、MYDEPT.MYSTMTINFO 表に書き込まれます。この表は、MYTABLESPACE 表スペースに作成されます。ROWS_READ、ROWS_WRITTEN、および STMT_TEXT エレメントに対してのみ列を作成します。グループの他のエレメントは破棄されます。
- SUBSECTION グループからのすべてのデータは、MYDEPT.MYSUBSECTIONINFO 表に書き込まれます。この表は、MYTABLESPACE 表スペースに作成されます。この表は、START_TIME、STOP_TIME、および PARTIAL_RECORD 以外のすべての列を含みます。
- XACT グループからのすべてのデータは、XACT_BAR 表に書き込まれます。表スペース情報が提供されていないので、IN *tablespace-name* 節で説明されていた規則に基づいて、システムが選択した表スペースに表が作成されます。この表には、XACT グループにあるすべてのエレメントが含まれます。
- CONNHEADER または CONTROL に対しては表は作成されません。これらのグループのすべてのデータは破棄されます。

```
CREATE EVENT MONITOR BAR
FOR STATEMENTS, TRANSACTIONS
WRITE TO TABLE
STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
INCLUDES(ROWS_READ, ROWS_WRITTEN, STMT_TEXT)),
STMT(TABLE MYDEPT.MYSTMTINFO IN MYTABLESPACE
EXCLUDES(START_TIME, STOP_TIME, PARTIAL_RECORD)),
XACT
```

CREATE FUNCTION

CREATE FUNCTION ステートメントは、ユーザー定義の関数または関数テンプレートを、現行サーバーで登録または定義する場合に使用されます。

このステートメントを使用して作成できる関数には、異なる 5 つのタイプがあります。これらのそれぞれについて、個々に説明します。

- 外部スカラー。この関数はプログラミング言語で書かれ、1 つのスカラー値を戻します。外部の実行可能プログラムが、関数の種々の属性を伴ってデータベースに登録されます。

CREATE FUNCTION

- 外部表。この関数はプログラミング言語で書かれ、完全な表を戻します。外部の実行可能プログラムが、関数の種々の属性を伴ってデータベースに登録されます。
- OLE DB 外部表。ユーザー定義の OLE DB 外部表関数はデータベースに登録されて、OLE DB Provider からデータをアクセスします。
- ソースまたはテンプレート。ソース関数は、データベースにすでに登録されている他の関数（組み込み、外部、SQL、またはソースのいずれか）を呼び出すことによってインプリメントされます。

関数テンプレート という部分関数を作成し、戻される値のタイプを定義することができますが、実行可能コードを含めることはできません。ユーザーはこれをフェデレーテッド・システム内のデータ・ソース関数にマップし、フェデレーテッド・データベースからそのデータ・ソース関数を呼び出せるようにします。関数テンプレートは、フェデレーテッド・サーバーとして指定されたアプリケーション・サーバーにだけ登録できます。

- SQL スカラー、表、または行。関数本体は SQL で書かれ、データベースに登録で定義されます。これは、スカラー値、表、または単一の行を戻します。

CREATE FUNCTION (外部スカラー)

CREATE FUNCTION (外部スカラー) ステートメントは、ユーザー定義の外部スカラー関数を現行サーバーに登録する場合に使用されます。スカラー関数は、呼び出されるたびに 1 つの値を返し、SQL 式が使用可能な個所であれば一般に使用することができます。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する CREATE_EXTERNAL_ROUTINE 権限、および以下の少なくとも 1 つ。
 - データベースに対する IMPLICIT_SCHEMA 権限 (関数のスキーマ名が既存のスキーマを指していない場合)
 - スキーマに対する CREATEIN 特権 (関数のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

非 fenced の関数を作成するには、ステートメントの許可 ID の特権に、以下の特権の少なくとも 1 つが含まれている必要があります。

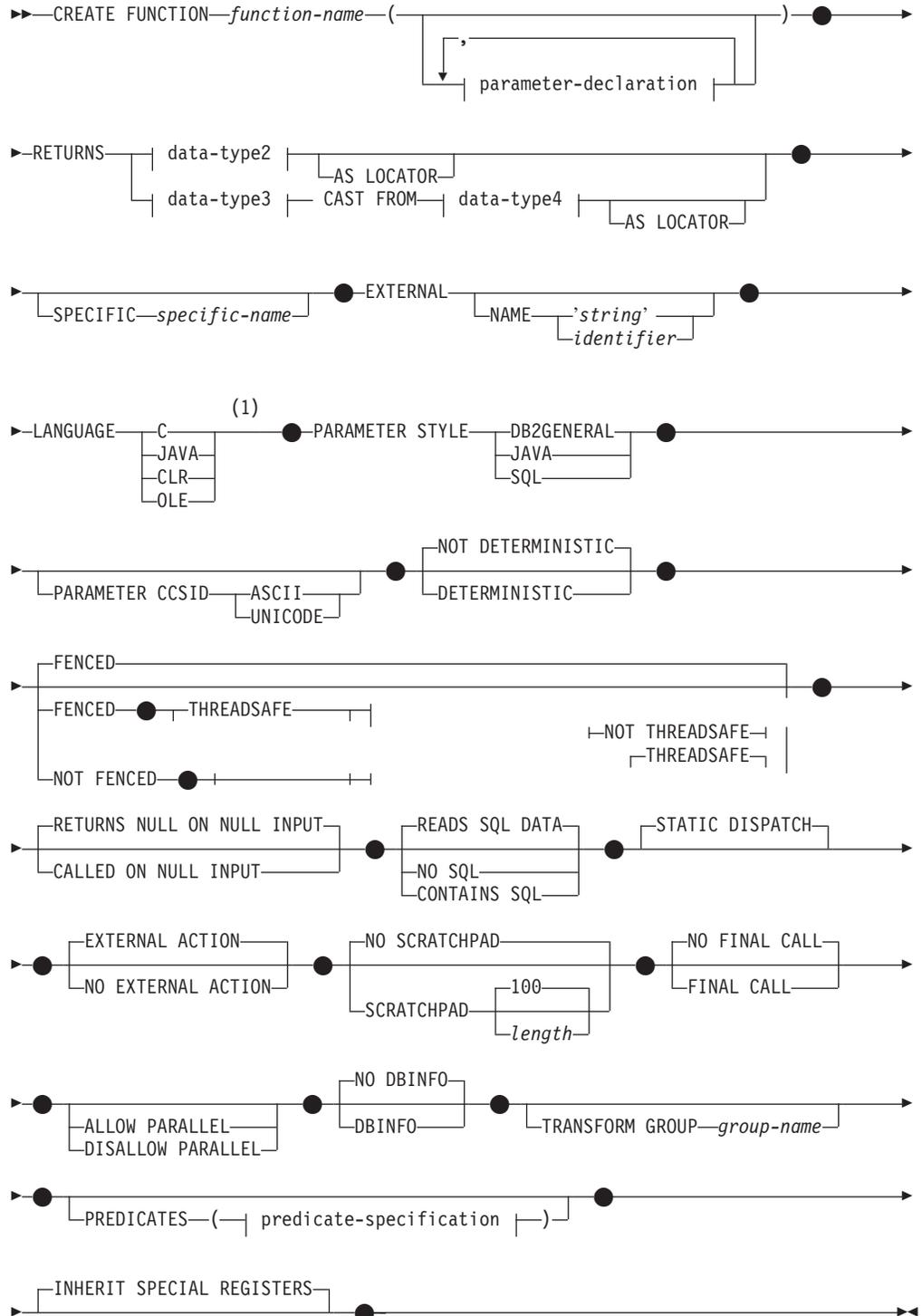
- データベースに対する CREATE_NOT_FENCED_ROUTINE 権限

CREATE FUNCTION (外部スカラー)

- SYSADM または DBADM 権限

fenced 関数を作成する場合には、さらに別の権限や特権は必要ありません。

構文



CREATE FUNCTION (外部スカラー)

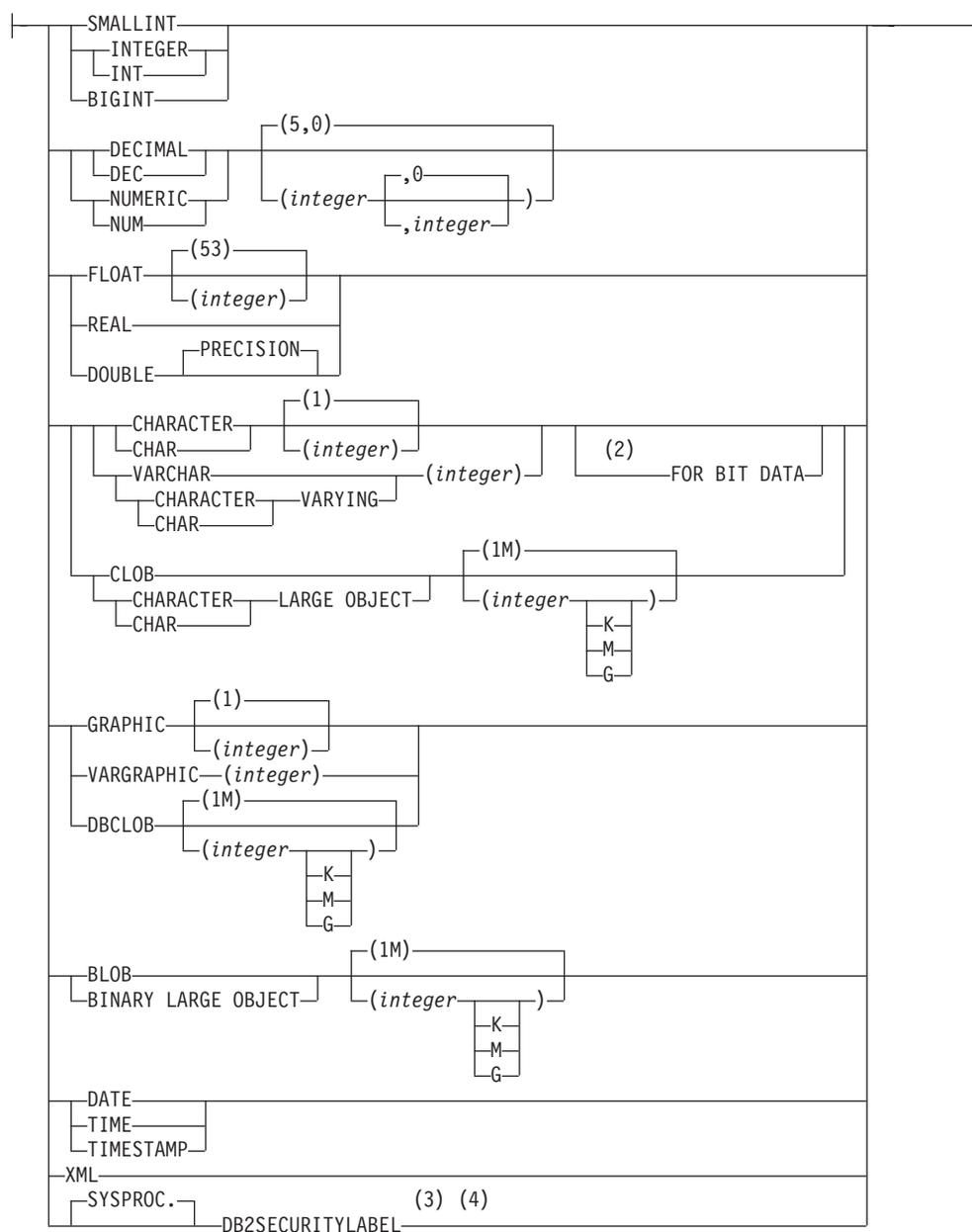
parameter-declaration:



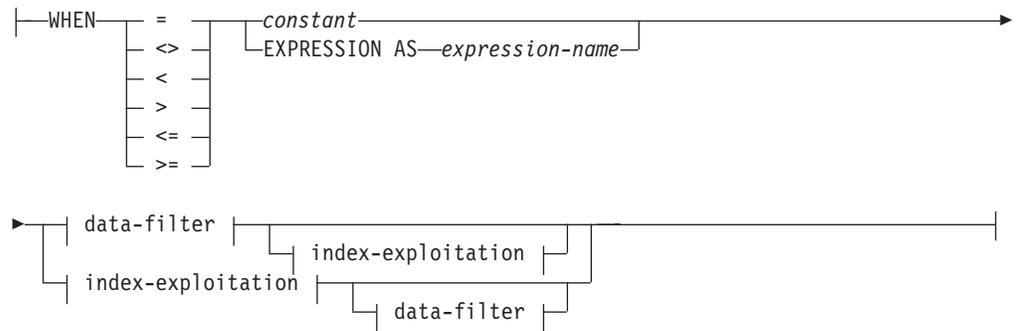
data-type1, data-type2, data-type3, data-type4:



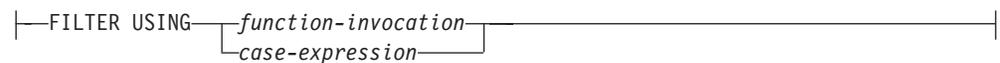
built-in-type:



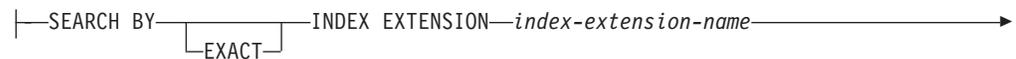
predicate-specification:



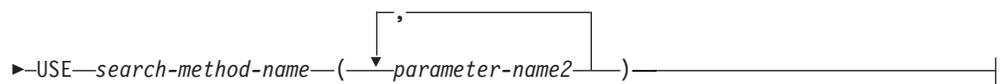
data-filter:



index-exploitation:



exploitation-rule:



注:

- 1 LANGUAGE SQL もサポートされています。
- 2 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- 3 DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- 4 タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE 42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。

CREATE FUNCTION (外部スカラー)

説明

function-name

定義する関数の名前を指定します。これは、関数を指定する修飾または非修飾の名前です。 *function-name* (関数名) の非修飾形式は SQL ID です (最大長 128)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。最初のパラメーターが構造化タイプの場合、修飾名は、最初のパラメーターのデータ・タイプと同じであってはなりません。

暗黙または明示の修飾子を含む名前、およびパラメーターの数と各パラメーターのデータ・タイプ (データ・タイプの長さ、精度、または位取りの各属性には関係なく) は、カタログに記述されている関数またはメソッドを指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数およびデータ・タイプとの組み合わせは、そのスキーマ内では当然ユニークですが、複数のスキーマ間でユニークである必要はありません。

2 つの部分から成る名前を指定する場合、“SYS” で始まる *schema-name* (スキーマ名) は使用できません。使用した場合、エラー (SQLSTATE 42939) になります。

述部のキーワードとして使用される多くの名前は、システム使用として予約されており、*function-name* として使用することはできません。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。この規則に違反すると、エラーになります (SQLSTATE 42939)。

一般に、関数のシグニチャーに何らかの差異がある場合には、同じ名前を複数の関数に使用することができます。

禁止されてはいませんが、意図的にオーバーライドを行う場合を除き、外部ユーザー定義関数の名前として、組み込み関数と同じ名前を指定するのは避けるべきです。異なる意味を持つ関数に組み込みのスカラー関数または列関数と同じ名前 (例えば、LENGTH、VALUE、MAX など) を与えることは、たとえその引数が一致していたとしても、動的 SQL ステートメントの過程で、あるいは静的 SQL アプリケーションの再バインド時に問題が生じます。すなわち、アプリケーションが失敗することがあり、また、さらに悪いケースとして、外見上は正常に実行されていても、結果が異なる場合があります。

(parameter-declaration,...)

関数の入力パラメーターの数を指定するとともに、各パラメーターのデータ・タイプを指定します。このリストには、関数が受け取ることを予期している各パラメーターごとに 1 つの項目を指定する必要があります。パラメーターの数は 90 を超えることはできません (SQLSTATE 54023)。

パラメーターのない関数も登録可能です。この場合、指定するデータ・タイプがない場合でも、括弧はコーディングする必要があります。以下に例を示します。

```
CREATE FUNCTION WOOFER() ...
```

対応するすべてのパラメーターで、1 つのスキーマ内で名前が同じ 2 つの関数が、まったく同じタイプを持つことはできません。このタイプの比較では長さ、

精度、および位取りは考慮されません。したがって、CHAR(8) と CHAR(35)、また DECIMAL(11,2) と DECIMAL (4,3) は、それぞれ同じタイプと見なされません。Unicode データベースの場合には、CHAR(13) と GRAPHIC(8) は、それぞれ同じタイプと見なされます。さらに、DECIMAL と NUMERIC などのように、この目的で複数のタイプが同じタイプとして扱われることがあります。シングニチャーが重複していると、エラー (SQLSTATE 42723) を戻します。

parameter-name

入力パラメーターにオプションの名前を指定します。パラメーター名は、述部指定の *index-exploitation* 節にある関数のパラメーターを参照するのに必要です。この名前は、パラメーター・リスト内の他のすべての *parameter-name* と同じにすることはできません (SQLSTATE 42734)。

data-type1

入力パラメーターのデータ・タイプを指定します。データ・タイプは、組み込みデータ・タイプ、特殊タイプ、構造化タイプ、または参照タイプにすることができます。各組み込みデータ・タイプの詳細な説明は、『CREATE TABLE』を参照してください。一部のデータ・タイプは、すべての言語でサポートされません。SQL データ・タイプとホスト言語データ・タイプの対応関係の詳細については、「組み込み SQL アプリケーション内で SQL データ・タイプにマップするデータ・タイプ」を参照してください。

- 日時タイプのパラメーターは文字データ・タイプとして受け渡され、そのデータは ISO 形式で受け渡されます。
- DECIMAL (および NUMERIC) は、LANGUAGE C と OLE では無効です (SQLSTATE 42815)。
- DECFLOAT は、LANGUAGE C、COBOL、CLR、JAVA、および OLE では無効です (SQLSTATE 42815)。
- XML は、LANGUAGE OLE では無効です。
- 関数内での XML 値の表現は、関数呼び出しでパラメーターとして渡される XML 値をシリアライズしたバージョンなので、タイプ XML のパラメーターは構文 XML AS CLOB(*n*) を使用して宣言する必要があります。
- CLR は 28 より大きい DECIMAL スケールをサポートしていません (SQLSTATE 42613)。
- 配列タイプを指定することはできません (SQLSTATE 42815)。

ユーザー定義特殊タイプの場合、パラメーターの長さ、精度、または桁数の属性は、特殊タイプのソース・タイプのもの (CREATE TYPE で指定されたもの) になります。特殊タイプのパラメーターは、特殊タイプのソース・タイプとして受け渡されます。特殊タイプの名前が修飾されていない場合は、データベース・マネージャーによって SQL パス内のスキーマが検索され、スキーマ名が解決されます。

ユーザー定義構造化タイプの場合、適切なトランスフォーム関数が関連するトランスフォーム・グループに存在する必要があります。

参照タイプの場合、パラメーターが有効範囲を指定されていない場合は、パラメーターを REF(*type-name*) で指定できます。

AS LOCATOR

実際の値の代わりにパラメーターの値へのロケーターを関数に受け渡す

CREATE FUNCTION (外部スカラー)

ことを指定します。LOB データ・タイプを持つパラメーター、またはLOB データ・タイプに基づく特殊タイプにのみ AS LOCATOR を指定します (SQLSTATE 42601)。値の代わりにロケーターを受け渡すと、関数に受け渡されるバイト数を特にパラメーターの値が非常に大きい場合に少なくできます。

AS LOCATOR 節は、データ・タイプをプロモート可能かどうかの判別に効果はなく、また関数解決で使用される関数シグニチャーにも影響を与えません。

関数が FENCED で NO SQL オプションを持っている場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

RETURNS

これは必須の節であり、関数の出力を指定します。

data-type2

出力のデータ・タイプを指定します。

この場合、上記の関数パラメーター *data-type1* で説明した外部関数のパラメーターと同じ考慮事項が適用されます。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。これは、実際の値の代わりにLOB ロケーターが UDF から渡されることを示します。

data-type3 **CAST FROM** *data-type4*

出力のデータ・タイプを指定します。

この形式の RETURNS 節は、関数コードから戻されたデータ・タイプとは異なるデータ・タイプを、呼び出しステートメントに戻すのに使用されます。例えば、以下の例で、

```
CREATE FUNCTION GET_HIRE_DATE(CHAR(6))
  RETURNS DATE CAST FROM CHAR(10)
  ...
```

CHAR(10) の値が関数コードからデータベース・マネージャーに戻され、データベース・マネージャーは、その値を DATE に変換して、変換された値を呼び出し側ステートメントに渡します。 *data-type4* は、 *data-type3* パラメーターにキャスト可能でなければなりません。キャスト可能でない場合、エラー (SQLSTATE 42880) になります。

data-type3 の長さ、精度または位取りは、 *data-type4* から推断することができるので、 *data-type3* に指定されるパラメーター化タイプの長さ、精度、または位取りを指定する必要はありません (指定は可能です)。代わりに、空の括弧を使用できます (例えば、VARCHAR() など)。パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

特殊タイプ、配列タイプ、および構造化タイプは、 *data-type4* に指定するタイプとしては無効です (SQLSTATE 42815)。

キャスト操作は、変換エラーになる可能性があるランタイム・チェックの対象にもなります。

AS LOCATOR

data-type4 の指定が、LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。これは、実際の値の代わりに LOB ロケーターが UDF から戻されることを示します。

SPECIFIC *specific-name*

定義する関数のインスタンスに対するユニーク名を指定します。この特定名は、この関数をソース関数として使用する場合、この関数をドロップする場合、またはこの関数にコメントを付ける場合に使用することができます。これは、関数の呼び出しには使用できません。*specific-name* の非修飾形式は SQL ID です (最大長 128)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別の関数インスタンスまたはメソッド指定を識別するものであってはなりません。そうでない場合、エラー (SQLSTATE 42710) になります。

specific-name は、既存の *function-name* (関数名) と同じでも構いません。

修飾子を指定しない場合、*function-name* に使用された修飾子を使用されます。修飾子を指定する場合は、*function-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラー (SQLSTATE 42882) になります。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成されるユニーク名は、SQL の後に文字のタイム・スタンプが続く名前です (SQLyymmddhhmmssxxx)。

EXTERNAL

この節は、CREATE FUNCTION ステートメントが、外部プログラミング言語で書かれたコードに基づく新しい関数を登録するのに使用されており、文書化されたリンケージの規則とインターフェースに準拠していることを示します。

NAME 節を指定しない場合、"NAME *function-name*" が想定されます。

NAME '*string*'

この節は、定義している関数をインプリメントするユーザー作成コードの名前を指定します。

'*string*' オプションは、最大 254 バイトのストリング定数です。ストリングに使用される形式は、指定した LANGUAGE によって異なります。

- LANGUAGE C の場合:

指定する *string* (ストリング) は、作成しているユーザー定義関数を実行するためにデータベース・マネージャーが呼び出すライブラリー名と、そのライブラリー中の関数名です。ライブラリー (およびそのライブラリー中の関数) は、CREATE FUNCTION ステートメントの実行時に存在している必要はありません。ただし、関数が SQL ステートメントで使用される時点では、そのライブラリーとそのライブラリー中の該当の関数は存在していなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42724)。

string は、以下のように指定できます。

CREATE FUNCTION (外部スカラー)

```
▶─' library_id ─┬─┬─▶  
    └─┬─ absolute_path_id ─┬─┬─▶  
      └─┬─ !─ func_id ─┬─▶
```

単一引用符内に、余分なブランクを使用することはできません。

library_id

関数を含むライブラリー名を指定します。データベース・マネージャーは、次のようにしてこのライブラリーを特定します。

- UNIX システムの場合、*library_id* が 'myfunc' と指定されており、データベース・マネージャーが /u/production から実行されていると、データベース・マネージャーはライブラリー /u/production/sqllib/function/myfunc で関数を特定します。
- Windows オペレーティング・システムの場合、データベース・マネージャーは LIBPATH または PATH 環境変数に指定されているディレクトリー・パスから関数を特定します。

absolute_path_id

関数を含んでいるファイルの絶対パス名を指定します。

例えば、UNIX システムの場合、'/u/jchui/mylib/myfunc' を指定すると、データベース・マネージャーは /u/jchui/mylib を調べて myfunc 共用ライブラリーを探します。

Windows オペレーティング・システムの場合、'd:¥mylib¥myfunc.dll' を指定すると、データベース・マネージャーは d:¥mylib ディレクトリーからダイナミック・リンク・ライブラリー myfunc.dll をロードします。絶対パス ID がルーチン本体の識別に使用されている場合は、.dll 拡張子を必ず付加してください。

! func_id

呼び出される関数の入り口点名を指定します。! は、ライブラリー ID と関数 ID との間の区切り文字です。

例えば、UNIX システムで 'mymod!func8' と指定すると、データベース・マネージャーはライブラリー \$inst_home_dir/sqllib/function/mymod を調べて、そのライブラリー内の入り口点 func8 を使用します。

Windows オペレーティング・システムの場合 'mymod!func8' を指定すると、データベース・マネージャーは mymod.dll ファイルをロードして、そのダイナミック・リンク・ライブラリー (DLL) の func8() 関数を呼び出します。

ストリングの形式が正しくない場合には、エラーが戻されます (SQLSTATE 42878)。

すべての外部関数の本体は、すべてのデータベース・パーティションで使用可能なディレクトリーにある必要があります。

- LANGUAGE JAVA の場合:

指定する *string* には、作成中のユーザー定義関数を実行するためにデータベース・マネージャーが呼び出す、任意指定の jar ファイル、クラス ID、およびメソッド ID が含まれています。クラス ID とメソッド ID

は、CREATE FUNCTION ステートメントの実行時には存在している必要はありません。 *jar_id* を指定する場合、ID は、CREATE FUNCTION ステートメントの実行時に存在していなければなりません。ただし、関数が SQL ステートメントで使用される時点では、メソッド ID は存在していなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されず (SQLSTATE 42724)。

string は、以下のように指定できます。

```

▶▶ ' _____class_id_____ . _____method_id_____ '
    |_____jar_id :_____ |_____!_____
  
```

単一引用符内に、余分なブランクを使用することはできません。

jar_id

jar の集合をデータベースへインストールしたときに、その jar の集合に付けられた jar ID を指定します。これは、単純 ID またはスキーマ修飾 ID のいずれかにすることができます。例えば、'myJar' や 'mySchema.myJar' のようになります。

class_id

Java オブジェクトのクラス ID を指定します。クラスがパッケージの一部である場合、クラス ID の部分に完全なパッケージ接頭部 (例: 'myPacks.UserFuncs') が含まれている必要があります。Java 仮想マシンは、ディレクトリー './myPacks/UserFuncs/' 中のクラスを探します。Windows オペレーティング・システムでは、Java 仮想マシンはディレクトリー './myPacks\UserFuncs\' を探索します。

method_id

呼び出す Java オブジェクトのメソッド名を指定します。

• LANGUAGE CLR の場合:

指定された *string* は、作成する関数を実行するためにデータベース・マネージャーが呼び出す .NET アセンブリー (ライブラリーまたは実行可能モジュール)、そのアセンブリー内のクラス、およびそのクラス内のメソッドを表します。モジュール、クラス、およびメソッドは、CREATE FUNCTION ステートメントの実行時に存在している必要はありません。ただし、関数を SQL ステートメントで使用するときには、モジュール、クラス、およびメソッドは存在していなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42724)。

'clr' コンパイラー・オプションで管理対象コード拡張を指定してコンパイルされている C++ ルーチンは、'LANGUAGE C' ではなく 'LANGUAGE CLR' としてカタログする必要があります。DB2 は、必要な実行時の決定を行えるようにするために、.NET インフラストラクチャーがユーザー定義関数内で使用されていることを認識する必要があります。 .NET インフラストラクチャーを使用するすべてのユーザー定義関数は、'LANGUAGE CLR' としてカタログする必要があります。

CREATE FUNCTION (外部スカラー)

string は、以下のように指定できます。

▶ '—assembly—:—class_id—!—method_id—' ◀

名前は、単一引用符で囲む必要があります。余分なブランクを使用することはできません。

assembly

クラスを含む DLL ファイルまたは他のアセンブリー・ファイルを指定します。ファイル拡張子 (.dll など) まで指定します。絶対パス名を指定しない場合、ファイルは DB2 インストール・パスの関数ディレクトリー (例えば、c:\sqlib¥function) にあるものとされます。ファイルがインストール関数ディレクトリーのサブディレクトリーにある場合は、絶対パスを指定せずに、ファイル名の前にサブディレクトリーを指定します。例えば、インストール・ディレクトリーが c:\sqlib であり、アセンブリー・ファイルが

c:\sqlib¥function¥myprocs¥mydotnet.dll であるなら、アセンブリーの指定は 'myprocs¥mydotnet.dll' とするだけで十分です。このパラメーターの大文字小文字が区別されるかどうかは、ファイル・システムの設定と同じです。

class_id

呼び出すメソッドが属するアセンブリー内のクラスの名前を指定します。クラスがネーム・スペース内にある場合は、クラスだけでなく絶対ネーム・スペースも指定することが必要です。例えば、クラス EmployeeClass がネーム・スペース MyCompany.ProcedureClasses にあるのであれば、MyCompany.ProcedureClasses.EmployeeClass をクラスとして指定します。一部の .NET 言語用のコンパイラーはクラスのネーム・スペースとしてプロジェクト名を追加するため、コマンド行コンパイラーと GUI コンパイラーのどちらを使用するかで動作が異なってくるので注意してください。このパラメーターには、大文字と小文字の区別があります。

method_id

指定したクラス内で呼び出されるメソッドを指定します。このパラメーターには、大文字と小文字の区別があります。

• LANGUAGE OLE の場合:

指定する *string* は、作成中のユーザー定義関数を実行するためにデータベース・マネージャーが呼び出す、OLE のプログラム ID (progid) またはクラス ID (clsid)、およびメソッド ID です。プログラム ID またはクラス ID、およびメソッド ID は、CREATE FUNCTION ステートメントの実行時に存在している必要はありません。ただし、関数が SQL ステートメントで使用される時点では、メソッド ID は存在していなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42724)。

string は、以下のように指定できます。

CREATE FUNCTION (外部スカラー)

定義関数は、「*OLE Automation Programmer's Reference*」に説明されている、OLE 自動化データ・タイプと呼び出しメカニズムに準拠している必要があります。

LANGUAGE OLE は、Windows オペレーティング・システム用の DB2 で保管されたユーザー定義関数に対してのみサポートされます。

LANGUAGE OLE を指定した UDF には、THREADSAFE は指定できません (SQLSTATE 42613)。

PARAMETER STYLE

この節は、関数にパラメーターを渡し、関数から値を戻すのに用いる規則を指定するために使用します。

DB2GENERAL

Java クラスのメソッドとして定義された外部関数との間で、パラメーターを渡し、値を戻す場合に用いる規則を指定します。これは、LANGUAGE JAVA を使用する場合にだけ指定する必要があります。

DB2GENERAL の同義語として値 DB2GENRL が使用可能です。

JAVA

関数は、Java 言語および SQLJ ルーチンの仕様に準拠する、パラメーターの受け渡し規則を使用します。これを指定できるのは、LANGUAGE JAVA を使用し、パラメーターとして構造化データ・タイプを指定せず、戻りタイプとして CLOB、BLOB、DBCLOB のいずれかのデータ・タイプを指定していない場合だけです (SQLSTATE 429B8)。PARAMETER STYLE JAVA 関数は、FINAL CALL、SCRATCHPAD または DBINFO 節をサポートしていません。

SQL

C 言語の呼び出しとリンケージの規則、OLE 自動化オブジェクトによって公開されたメソッド、または .NET オブジェクトの共用静的メソッドに準拠する規則を、この外部メソッドとの間でパラメーターを渡し、値を戻す場合の規則として指定します。これは、LANGUAGE C、LANGUAGE CLR、または LANGUAGE OLE を使用する場合に指定する必要があります。

PARAMETER CCSID

関数とやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。関数が呼び出されるときのアプリケーション・コード・ページはデータベース・コード・ページです。

UNICODE

ストリング・データが Unicode でエンコードされることを指定します。データベースが Unicode データベースの場合、文字データは UTF-8、GRAPHIC データは UCS-2 になります。データベースが Unicode データベ

ースでない場合は、文字データは UTF-8 になります。いずれの場合も、関数が呼び出される時のアプリケーション・コード・ページは 1208 です。

データベースが Unicode データベースではないのに、PARAMETER CCSID UNICODE を指定した関数を作成すると、その関数は GRAPHIC タイプ、XML タイプ、またはユーザー定義タイプを取ることはできません (SQLSTATE 560C1)。

データベースが Unicode ではなく、データベース構成に代替照合シーケンスが指定されている場合、PARAMETER CCSID ASCII または PARAMETER CCSID UNICODE を指定した関数を作成できます。関数とやり取りされるすべてのストリング・データは、適切なコード・ページに変換されます。

この節を LANGUAGE OLE、LANGUAGE JAVA、または LANGUAGE CLR とともに指定することはできません (SQLSTATE 42613)。

DETERMINISTIC または NOT DETERMINISTIC

この節は任意指定で、特定の引数の値に対して関数が常に同じ結果を戻すか (DETERMINISTIC)、それとも状態値に依存して関数の結果が影響を受けるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC を伴う関数は、連続で同じ入力を指定して呼び出しが行われた場合に常に同じ結果を戻します。NOT DETERMINISTIC を指定すると、同じ入力によって常に同じ結果が生じる利点に基づく最適化ができなくなります。乱数を生成する関数は、NOT DETERMINISTIC 関数の例です。入力の平方根を求める関数は、DETERMINISTIC 関数の例です。

FENCED または NOT FENCED

この節は、関数をデータベース・マネージャーの操作環境のプロセスまたはアドレス・スペースで実行しても“安全”かどうかを指定します。

関数が FENCED として登録されると、データベース・マネージャーは、その内部リソース (データ・バッファなど) を保護して、その関数からアクセスされないようにします。多くの関数は、FENCED または NOT FENCED のどちらかで実行するように選択することができます。一般に、FENCED として実行される関数は、NOT FENCED として実行されるものと同じようには実行されません。

注意:

適切にコード化、検討、およびテストされていない関数に NOT FENCED を使用すると、DB2 の整合性に危険を招く場合があります。DB2 では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられていますが、NOT FENCED ユーザー定義関数が使用される場合には、完全な整合性を確保できません。

LANGUAGE OLE または NOT THREADSAFE を指定した関数には、FENCED のみを指定できます (SQLSTATE 42613)。

関数が FENCED で NO SQL オプションを持っている場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

ユーザー定義関数を NOT FENCED として登録するには、SYSADM 権限、DBADM 権限、または特殊権限 (CREATE_NOT_FENCED_ROUTINE) が必要です。

CREATE FUNCTION (外部スカラー)

NOT FENCED 節を指定している場合は、LANGUAGE CLR ユーザー定義関数を作成できません (SQLSTATE 42601)。

THREADSAFE または NOT THREADSAFE

関数を他のルーチンと同じプロセスで実行しても安全か (THREADSAFE)、そうでないか (NOT THREADSAFE) を指定します。

関数が OLE 以外の LANGUAGE で定義される場合:

- 関数が THREADSAFE に定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスで関数を呼び出すことができます。一般に、スレッド・セーフにするには、関数はどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。FENCED および NOT FENCED 関数の両方が THREADSAFE になることが可能です。
- 関数が NOT THREADSAFE と定義される場合には、データベース・マネージャーが関数を他のルーチンと同じプロセスで同時に呼び出すことは決してありません。

FENCED 関数の場合、LANGUAGE が JAVAまたは CLR なら THREADSAFE がデフォルトです。これ以外のすべての言語の場合は、NOT THREADSAFE がデフォルトです。関数が LANGUAGE OLE に定義される場合には、THREADSAFE は指定できません (SQLSTATE 42613)。

NOT FENCED 関数の場合には、THREADSAFE がデフォルトです。NOT THREADSAFE を指定することはできません (SQLSTATE 42613)。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT

このオプション節を使用すると、引数のいずれかが NULL 値の場合に、外部関数を呼び出さないようにすることができます。ユーザー定義関数がパラメーターなしで定義されている場合、この NULL 引数条件は引き起こされることはないため、この仕様のコーディング方法はそれほど重要ではなくなります。

RETURNS NULL ON NULL INPUT が指定されており、実行時に関数の引数のいずれかが NULL 値の場合、このユーザー定義関数は呼び出されず、結果は NULL 値になります。

CALLED ON NULL INPUT が指定されると、引数が NULL 値か否かに関係なくユーザー定義関数が呼び出されます。これは、NULL 値を戻す場合も、通常の (NULL 以外の) 値を戻す場合もあります。ただし、NULL の引数値の有無のテストは UDF が行う必要があります。

値 NULL CALL は、後方互換性またはファミリーの互換性のために、CALLED ON NULL INPUT の同義語として使うことができます。同様に、NOT NULL CALL は、RETURNS NULL ON NULL INPUT の同義語として使用できます。

NO SQL、CONTAINS SQL、READS SQL DATA

関数から SQL ステートメントが発行されるかどうかと、もし発行されればどのタイプかを示します。

NO SQL

関数はどの SQL ステートメントも実行できないことを指示します (SQLSTATE 38001)。

CONTAINS SQL

SQL データの読み取りも変更も行わない SQL ステートメントを、関数で実行できることを指定します (SQLSTATE 38004 または 42985)。どの関数でもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

READS SQL DATA

SQL データを変更しない SQL ステートメントを、関数で実行できることを指定します (SQLSTATE 38002 または 42985)。どの関数でもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

STATIC DISPATCH

このオプション節は、関数解決時に DB2 が関数のパラメーターの静的タイプ (宣言済みタイプ) に基づいて関数を選択するよう指示します。

EXTERNAL ACTION または NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るかどうかを指定します。外部アクションの例としては、メッセージの送信やファイルへのレコードの書き込みがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取ることを指定します。

外部アクションが指定された関数は、関数が並列タスクによって実行されると、不正確な結果を戻す場合があります。例えば、初期呼び出しを受けるたびに注釈を送信する関数の場合、関数ごとに 1 つの注釈が送信されるのではなく、並列タスクごとに 1 回ずつ送信されることになります。並列処理を正しく扱うことのできない関数については、DISALLOW PARALLEL 節を指定します。

NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取らないことを指定します。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。

NO SCRATCHPAD または SCRATCHPAD *length*

この節はオプションであり、この外部関数に対してスクラッチパッドを用意するか否かを指定するのに使用することができます。(ユーザー定義関数を再入可能にすることを強くお勧めします。再入可能にすると、スクラッチパッドによってある呼び出しと次の呼び出しとの間に関数が“状態を保管する”手段が用意されます。)

SCRATCHPAD を指定すると、ユーザー定義関数の最初の呼び出し時に、その外部関数によって使用されるスクラッチパッドにメモリーが割り振られます。このスクラッチパッドには、次の特性があります。

- *length* を指定すると、スクラッチパッドのサイズをバイト単位で設定できます。この値は 1 から 32 767 の範囲で指定する必要があります (SQLSTATE 42820)。デフォルト・サイズは 100 バイトです。
- すべて X'00' に初期化されます。

CREATE FUNCTION (外部スカラー)

- その有効範囲は、該当の SQL ステートメントです。SQL ステートメントでの外部関数に対する参照ごとに 1 つのスクラッチパッドがあります。したがって、以下のステートメントの関数 UDFX が SCRATCHPAD キーワードを指定して定義されている場合、3 つのスクラッチパッドが割り当てられます。

```
SELECT A, UDFX(A) FROM TABLEB
WHERE UDFX(A) > 103 OR UDFX(A) < 19
```

ALLOW PARALLEL が指定されているか、またはデフォルト値として使用された場合、その有効範囲は上記とは異なります。関数が複数のデータベース・パーティションで実行される場合、関数が処理されるそれぞれのデータベース・パーティションにおいて、SQL ステートメントでの関数へのそれぞれの参照ごとにスクラッチパッドが割り当てられます。同様に、パーティション内並列処理をオンにして照会が実行される場合、3 より多くのスクラッチパッドが割り当てられることがあります。

- スクラッチパッドは持続します。その内容は、外部関数のある呼び出しから次の呼び出しになっても持続します。外部関数のある呼び出しによってスクラッチパッドに対して行われた変更はいずれも、次の呼び出し時に持続しています。データベース・マネージャーは、各 SQL ステートメントの実行開始時に、スクラッチパッドを初期設定します。各副照会の実行開始時には、データベース・マネージャーによってスクラッチパッドがリセットされます。FINAL CALL オプションが指定されている場合、システムは、スクラッチパッドのリセットに先立って、最終呼び出しを行います。
- これは、外部関数が獲得するシステム・リソース (メモリーなどの) の中央点として使用することもできます。関数は、最初の呼び出しでメモリーを獲得し、そのアドレスをスクラッチパッドに保管して、後の呼び出しでそれを参照することができます。

(このようにシステム・リソースが獲得される場合、FINAL CALL キーワードも指定する必要があります。これにより、ステートメントの最後で特殊な呼び出しが行われ、外部関数は獲得したシステム・リソースをすべて解放することができます。)

SCRATCHPAD を指定すると、ユーザー定義関数を呼び出すたびに、スクラッチパッドをアドレッシングする外部関数に追加の引数が渡されます。

NO SCRATCHPAD を指定すると、外部関数に対してスクラッチパッドは割り振られず、渡されません。

SCRATCHPAD は、PARAMETER STYLE JAVA 関数ではサポートされていません。

FINAL CALL または NO FINAL CALL

この節はオプションであり、外部関数に対する最終呼び出しが行われるか否かを指定します。このような最終呼び出しの目的は、外部関数が、獲得したシステム・リソースすべてを解放できるようにすることです。外部関数がメモリーなどのシステム・リソースを獲得し、それをスクラッチパッドに固定するような状況では、これを SCRATCHPAD キーワードと共に使用すると便利です。FINAL CALL を指定した場合、実行時点で以下が行われます。

- 呼び出しのタイプを指定する追加の引数が外部関数に渡されます。呼び出しのタイプは次のとおりです。

- 通常呼び出し。SQL 引数が渡され、結果が戻されることが予期されます。
- 最初の呼び出し。この SQL ステートメントのユーザー定義関数に対する参照に対応する外部関数の最初の呼び出し。最初の呼び出しは通常呼び出しです。
- 最終呼び出し。外部関数がリソースを解放できるようにするための、その関数に対する最終呼び出し。最終呼び出しは、通常呼び出しではありません。この最終呼び出しは、以下の時点で行われます。
 - ステートメント終了時。これは、カーソル指向型のステートメントでカーソルがクローズされた場合、あるいはステートメントが実行を終了した場合に発生します。
 - 並列タスクの終了時。これは、関数が並列タスクで実行された場合に発生します。
 - トランザクション終了時または割り込み時。これは、通常のステートメント終了が発生しなかった場合に発生します。例えば、何らかの理由で、アプリケーションのロジックが、カーソルをクローズしないようになっている場合があります。このタイプの最終呼び出しの際、CLOSE カーソル以外は、SQL ステートメントが発行されない可能性があります (SQLSTATE 38505)。このタイプの最終呼び出しは、「呼び出しタイプ」の引数の特殊値で指示されます。

WITH HOLD として定義されたカーソルがオープンされている間に、コミット操作が発生すると、それ以降のカーソルのクローズ時、またはアプリケーションの終了時に最終呼び出しが行われます。

NO FINAL CALL を指定すると、“呼び出しタイプ”の引数は外部関数に渡されず、最終呼び出しは行われません。

FINAL CALL は、PARAMETER STYLE JAVA 関数ではサポートされていません。

ALLOW PARALLEL または DISALLOW PARALLEL

この節はオプションで、関数への 1 つの参照に対して、関数の呼び出しを並列化できるか否かを指定します。一般に、ほとんどのスカラー関数の呼び出しは並列化が可能ですが、並列化できない関数 (1 つのスクラッチパッドのコピーに依存する関数など) もあります。スカラー関数に対して ALLOW PARALLEL または DISALLOW PARALLEL を指定すると、DB2 はその指定を受け入れません。関数にどちらのキーワードが当てはまるかを判別するには、以下の点について検討する必要があります。

- UDF のすべての呼び出しが、互いに完全に独立していますか? YES の場合には、ALLOW PARALLEL を指定します。
- UDF を呼び出すごとに、次の呼び出しに関する値を提供するスクラッチパッドが更新されますか? (例えば、カウンターの増分など。) YES の場合には、DISALLOW PARALLEL を指定するか、またはデフォルトを受け入れません。
- 1 つのデータベース・パーティションでのみ起こる必要のある外部アクションが UDF によって実行されますか? YES の場合には、DISALLOW PARALLEL を指定するか、またはデフォルトを受け入れません。

CREATE FUNCTION (外部スカラー)

- コストのかかる初期化処理の実行回数を最小にするためだけに、スクラッチパッドを使用していますか? YES の場合には、ALLOW PARALLEL を指定します。

いずれの場合も、すべての外部関数の本体は、すべてのデータベース・パーティションで使用可能なディレクトリーにある必要があります。

ステートメントで以下の 1 つ以上のオプションが指定されている場合以外は、デフォルト値は ALLOW PARALLEL です。

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

これらのオプションのいずれかが指定または暗黙指定されている場合は、デフォルト値は DISALLOW PARALLEL です。

INHERIT SPECIAL REGISTERS

このオプション節は、関数の更新可能な特殊レジスターが、呼び出しステートメントの環境からの初期値を継承することを指定します。カーソルの選択ステートメントで呼び出される関数の場合、初期値はカーソルがオープンした際の環境から継承します。ネストされたオブジェクト (例えば、トリガーまたはビュー) に呼び出されるルーチンの場合、初期値は (オブジェクト定義から継承するのではなく) ランタイム環境から継承します。

特殊レジスターに対する変更が、関数の呼び出し側に戻されることはありません。

更新不能の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、デフォルト値に設定されます。

NO DBINFO または DBINFO

この節はオプションで、DB2 において既知である特定の情報を追加の呼び出し時引数として UDF に渡すか (DBINFO)、または渡さないか (NO DBINFO) を指定します。NO DBINFO がデフォルト値です。DBINFO は、LANGUAGE OLE ではサポートされません (SQLSTATE 42613)。また、PARAMETER STYLE JAVA でもサポートされません。

DBINFO を指定すると、以下の情報を含む構造が UDF に渡されます。

- データベース名 - 現在接続されているデータベースの名前。
- アプリケーション ID - データベースへの接続ごとに確立された、固有アプリケーション ID。
- アプリケーション許可 ID - アプリケーション実行時の許可 ID。この UDF とアプリケーションとの間でネストされている UDF は無関係。
- コード・ページ - データベースのコード・ページを識別します。
- スキーマ名 - 表名とまったく同じ条件のもとでは、スキーマの名前が入ります。その他の場合はブランクです。
- 表名 - UDF 参照が UPDATE ステートメントの SET 節の右側にある場合、または INSERT ステートメントの VALUES リストの項目である場合のいずれかに限り、更新または挿入される表の非修飾名が入ります。その他の場合はブランクです。

- 列名 - 表名とまったく同じ条件で、更新または挿入される列の名前が入りません。その他の場合はブランクです。
- データベースのバージョン/リリース - UDF を呼び出すデータベース・サーバーのバージョン、リリースおよび修正レベルを識別します。
- プラットフォーム - サーバーのプラットフォーム・タイプが入ります。
- 表関数の結果の列番号 - 外部スカラー関数には当てはまりません。

TRANSFORM GROUP *group-name*

関数を呼び出す際のユーザー定義の構造化タイプのトランスフォーメーションに使用するトランスフォーム・グループを指定します。関数定義にパラメーターまたは RETURNS データ・タイプとしてユーザー定義の構造化タイプが含まれている場合、トランスフォームが必要になります。この節が指定されない場合には、デフォルトのグループ名 DB2_FUNCTION が使用されます。指定した (またはデフォルトの) *group-name* が、参照された構造化タイプに定義されていない場合、エラーになります (SQLSTATE 42741)。指定した *group-name* または構造化タイプに必須の FROM SQL または TO SQL トランスフォーム関数が定義されていない場合には、エラーになります (SQLSTATE 42744)。

トランスフォーム関数は、FROM SQL および TO SQL の両方とも、指定された場合も暗黙的に指定されている場合でも、構造化タイプと組み込みタイプ属性とのトランスフォームを適切に実行する SQL 関数でなければなりません。

PREDICATES

述部でこの関数が使用されるときに実行される、フィルター操作や索引拡張の活用を定義します。述部仕様では、検索条件のオプションの SELECTIVITY 節を指定できます。PREDICATES 節が指定された場合、関数は NO EXTERNAL ACTION を指定した DETERMINISTIC として定義しなければなりません (SQLSTATE 42613)。PREDICATES 節が指定されており、データベースが Unicode データベースでない場合は、PARAMETER CCSID UNICODE は指定できません (SQLSTATE 42613)。

WHEN *comparison-operator*

比較演算子 ("=", "<", ">", ">=", "<=", "<>") を使用した述部での、関数の特定の使用を導入します。

constant

関数の RETURNS タイプに比較可能なデータ・タイプを使用して、定数値を指定します (SQLSTATE 42818)。述部が同じ比較演算子とこの定数でこの関数を使用する場合、指定されたフィルターおよび索引の活用がオプティマイザーにより考慮されます。

EXPRESSION AS *expression-name*

式に名前を提供します。述部が同じ比較演算子と式でこの関数を使用する場合、指定されたフィルターおよび索引の活用が行われます。この式には、式名が割り当てられ、検索関数の引数として使用できるようになっています。 *expression-name* は、作成されている関数のいずれかの *parameter-name* と同じにすることはできません (SQLSTATE 42711)。式が指定される際に、その式のタイプが識別されます。

FILTER USING

結果表をさらにフィルター操作する際に使用する、外部関数またはケース式の指定を許可します。

CREATE FUNCTION (外部スカラー)

function-invocation

結果表の追加のフィルター操作の実行に使用できるフィルター関数を指定します。これは定義された関数のバージョンであり (述部で使用)、ユーザー定義述部で実行される行数を減らし、行を限定するかどうかを判別します。索引により生成される結果が、ユーザー定義述部に期待される結果に近い場合には、フィルター関数を適用する効果はあまりありません。これを指定しない場合は、データのフィルター操作は実行されません。

この関数は、任意の *parameter-name*、*expression-name*、または定数を引数として使用でき (SQLSTATE 42703)、整数を戻します (SQLSTATE 428E4)。戻り値 1 の場合は行が保持され、その他の場合は破棄されます。

この関数は、以下の要件を満たしていなければなりません。

- LANGUAGE SQL で定義されていないこと (SQLSTATE 429B4)
- NOT DETERMINISTIC または EXTERNAL ACTION で定義されていないこと (SQLSTATE 42845)
- いずれかのパラメーターのデータ・タイプとして構造化データ・タイプがないこと (SQLSTATE 428E3)
- 副照会が含まれていないこと (SQLSTATE 428E4)
- XMLQUERY または XMLEXISTS 式が含まれていないこと (SQLSTATE SQLSTATE 428E4)

引数が他の関数またはメソッドを呼び出す場合、このネストされた関数またはメソッドにもこれらの規則が課されます。ただし、引数が組み込みデータ・タイプに評価されるかぎり、システム生成の *observer* メソッドをフィルター関数 (または、引数として使用される任意の関数またはメソッド) への引数として使用することができます。

関数の定義者は、指定されたフィルター関数に対して EXECUTE 特権を持っていないければなりません。

function-invocation 節は、データベース・コード・ページ内で、65 536 バイト以内の長さでなければなりません (SQLSTATE 22001)。

case-expression

結果表をさらにフィルター操作するためのケース式を指定します。

searched-when-clause および *simple-when-clause* では、*parameter-name*、*expression-name*、または定数を使用できます (SQLSTATE 42703)。FILTER USING *function-invocation* に指定された規則を使って、外部関数を結果式として使用することができます。*case-expression* で参照される関数またはメソッドはすべて、*function-invocation* にリストされている 4 つの規則にも適合していなければなりません。

case-expression の中では副照会および XMLQUERY または XMLEXISTS 式は使用できません (SQLSTATE 428E4)。

ケース式は整数を戻さなければなりません (SQLSTATE 428E4)。結果式で戻り値が 1 の場合は行が保持され、その他の場合は破棄されます。

case-invocation 節は、データベース・コード・ページ内で、65 536 バイト以内の長さでなければなりません (SQLSTATE 22001)。

index-exploitation

索引を活用するために使用する索引拡張の検索メソッドによって、規則のセットを定義します。

SEARCH BY INDEX EXTENSION *index-extension-name*

索引拡張を指定します。 *index-extension-name* は、既存の索引拡張を指定する必要があります。

EXACT

述部評価の時に索引検索が厳密に行われることを指定します。索引検索後、オリジナルのユーザー定義の述部関数も、フィルターも適用する必要がないことを DB2 に指示するのに EXACT を使用します。 EXACT 述部は、索引検索が述部と同じ結果を戻す場合に便利です。

EXACT が指定されない場合には、索引検索後、オリジナルのユーザー定義述部が適用されます。索引が類似した述部を提供するのにとどまると思われる場合には、 EXACT オプションは指定しないでください。

索引検索が使用されない場合には、フィルター関数とオリジナルの述部を適用する必要があります。

exploitation-rule

検索ターゲットおよび検索指数を記述し、さらにこれらを使用して索引拡張で定義した検索メソッドを介して索引検索を実行する方法を記述します。

WHEN KEY (*parameter-name1*)

検索ターゲットを定義します。 1 つのキーにつき 1 つしか、探索ターゲットを指定できません。 *parameter-name1* 値は、定義された関数のパラメーター名を指定します (SQLSTATE 42703 または 428E8)。

parameter-name1 のデータ・タイプは、索引拡張で指定したソース・キーのデータ・タイプに適合しなければなりません (SQLSTATE 428EY)。この適合は、組み込みおよび特殊データ・タイプで厳密に一致しなければならず、構造化タイプの同じタイプ階層内になければなりません。

指定されたパラメーターの値が、指定された索引拡張に基づく索引により網羅される列である場合、この節は真となります。

USE *search-method-name*(*parameter-name2*,...)

検索指数を定義します。索引拡張で定義されている検索メソッドから、使用する検索メソッドを指定します。 *search-method-name* は、索引拡張で定義される検索メソッドと適合しなければなりません (SQLSTATE 42743)。 *parameter-name2* 値は、定義された関数のパラメーター名、または EXPRESSION AS 節の *expression-name* を指定します (SQLSTATE 42703)。これは、検索ターゲットに指定したパラメーター名と異ならなければなりません (SQLSTATE 428E9)。パラメーターの数と各 *parameter-name2* のデータ・タイプは、索引拡張の検索メソッドに定義されるパラメーターに適合しなければなりません (SQLSTATE

CREATE FUNCTION (外部スカラー)

42816)。この適合は、組み込みおよび特殊データ・タイプで厳密に一致しなければならず、構造化タイプの同じタイプ階層内になければなりません。

注

- あるデータ・タイプが他のデータ・タイプにキャスト可能かどうかの判別では、CHAR や DECIMAL などのパラメーター化データ・タイプの長さまたは精度と位取りは考慮されません。したがって、ソース・データ・タイプの値をターゲット・データ・タイプの値にキャストしようとする、関数の使用時にエラーになる可能性があります。例えば、VARCHAR は DATE にキャストできますが、実際にはソース・タイプが VARCHAR(5) と定義されている場合には、関数の使用時にエラーになります。
- ユーザー定義関数のパラメーターのデータ・タイプを選択する場合は、入力値に影響を与えるプロモーションの規則を考慮してください (『データ・タイプのプロモーション』を参照してください)。例えば、入力値として使用できる定数のデータ・タイプは、予期される以外の組み込みデータ・タイプである可能性があり、さらには、予期されるデータ・タイプにプロモートできない場合があります。プロモーションの規則に従って、一般にパラメーターには次のデータ・タイプを使用するようにしてください。
 - SMALLINT ではなく INTEGER
 - REAL ではなく DOUBLE
 - CHAR ではなく VARCHAR
 - GRAPHIC ではなく VARGRAPHIC
- プラットフォーム間での UDF の移植性を保つためには、以下のデータ・タイプは使用しないようにする必要があります。
 - FLOAT- 代わりに DOUBLE または REAL を使用してください。
 - NUMERIC- 代わりに DECIMAL を使用してください。
 - LONG VARCHAR- 代わりに CLOB (または BLOB) を使用してください。
- 関数とメソッドは、オーバーライド関係にはなりません (SQLSTATE 42745)。オーバーライドについての詳細は、『CREATE TYPE (構造化)』を参照してください。
- 関数のシグニチャーは、メソッドのシグニチャーと同じではありません (関数の最初の *parameter-type* と、メソッドの *subject-type* を比較) (SQLSTATE 42723)。
- まだ存在していないスキーマ名を用いて関数を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- パーティション・データベース環境では、外部ユーザー定義関数またはメソッドでの SQL の使用はサポートされていません (SQLSTATE 42997)。
- 索引拡張を定義するには、NO SQL として定義されたルーチンしか使用できません (SQLSTATE 428F8)。
- 関数が SQL を許可する場合、外部プログラムは、フェデレーテッド・オブジェクトへのアクセスを試行してはなりません (SQLSTATE 55047)。

- NOT FENCED として定義される Java ルーチンは、FENCED THREADSAFE として定義されているかのように呼び出されます。
- PARAMETER STYLE DB2GENERAL 節が指定されている場合、XML パラメータは、LANGUAGE JAVA 外部関数でのみサポートされます。
- **表アクセスの制限**

関数が READS SQL DATA に定義されている場合には、関数のいかなるステートメントも、関数を呼び出したステートメントによって変更されている表にはアクセスできません (SQLSTATE 57053)。例えば、ユーザー定義関数 BONUS() が READS SQL DATA に定義されているとします。ステートメント UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO) が呼び出される場合、BONUS 関数の SQL ステートメントは、EMPLOYEE 表からの読み取りを行えません。

- **特権**

- 関数の定義者は、関数に対する WITH GRANT OPTION 付きの EXECUTE 特権と、関数をドロップする権利を常に与えられます。
- 関数を SQL ステートメントで使用する時点で、関数の定義者はその関数によって使用されるすべてのパッケージに対して EXECUTE 特権を持っていないければなりません。

- **互換性**

- DB2 for z/OS との互換性:
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - ASUTIME NO LIMIT
 - NO COLLID
 - PROGRAM TYPE SUB
 - STAY RESIDENT NO
 - Unicode データベースでの CCSID UNICODE
 - PARAMETER CCSID UNICODE が指定されていない場合、非 Unicode データベース内での CCSID ASCII
- 以前のバージョンの DB2 との互換性:
 - PARAMETER STYLE SQL の代わりに PARAMETER STYLE DB2SQL を指定できます。
 - DETERMINISTIC の代わりに NOT VARIANT を、また NOT DETERMINISTIC の代わりに VARIANT を指定することができます。
 - CALLED ON NULL INPUT の代わりに NULL CALL を、また RETURNS NULL ON NULL INPUT の代わりに NOT NULL CALL を指定できます。

例

例 1: Pellow は、自身の PELLOW スキーマに CENTRE 関数を登録します。デフォルト値のあるキーワードはデフォルト値を使い、関数特定名はシステムに生成させることにします。

```
CREATE FUNCTION CENTRE (INT,FLOAT)
  RETURNS FLOAT
  EXTERNAL NAME 'mod!middle'
  LANGUAGE C
```

CREATE FUNCTION (外部スカラー)

```
PARAMETER STYLE SQL
DETERMINISTIC
NO SQL
NO EXTERNAL ACTION
```

例 2: ここで、McBride (DBADM 権限を持つ) が PELLOW スキーマに別の CENTRE 関数を登録し、関数にデータ定義言語でその後使用するための明示的な特定名を付け、すべてのキーワード値を明示的に指定します。また、この関数はスクラッチパッドを使用し、おそらく後続の結果に影響するデータをスクラッチパッドに累積します。DISALLOW PARALLEL が指定されているので、関数への参照は並列化されず、したがって 1 つのスクラッチパッドを使用して一度限りの初期化と結果の保管が行われます。

```
CREATE FUNCTION PELLOW.CENTRE (FLOAT, FLOAT, FLOAT)
RETURNS DECIMAL(8,4) CAST FROM FLOAT
SPECIFIC FOCUS92
EXTERNAL NAME 'effects!focalpt'
LANGUAGE C PARAMETER STYLE SQL
DETERMINISTIC FENCED NOT NULL CALL NO SQL NO EXTERNAL ACTION
SCRATCHPAD NO FINAL CALL
DISALLOW PARALLEL
```

例 3: 以下の例は、次の規則をインプリメントするために書かれた、C 言語のユーザー定義関数プログラムです。

```
output = 2 * input - 4
```

入力が NULL 値の場合には (そしてその場合のみ)、NULL 値を戻します。これは、CREATE FUNCTION ステートメントで NOT NULL CALL を指定することにより、より簡単に (つまり NULL 値チェックを行わずに) 作成することができます。CREATE FUNCTION ステートメントは、次のとおりです。

```
CREATE FUNCTION ntest1 (SMALLINT)
RETURNS SMALLINT
EXTERNAL NAME 'ntest1!nudft1'
LANGUAGE C PARAMETER STYLE SQL
DETERMINISTIC NOT FENCED NULL CALL
NO SQL NO EXTERNAL ACTION
```

プログラム・コードは、次のとおりです。

```
#include "sqlsystem.h"
/* NUDFT1 IS A USER_DEFINED SCALAR FUNCTION */
/* udft1 accepts smallint input
and produces smallint output
implementing the rule:
if (input is null)
set output = null;
else
set output = 2 * input - 4;
*/
void SQL_API_FN nudft1
(short *input, /* ptr to input arg */
short *output, /* ptr to where result goes */
short *input_ind, /* ptr to input indicator var */
short *output_ind, /* ptr to output indicator var */
char sqlstate[6], /* sqlstate, allows for null-term */
char fname[28], /* fully qual func name, nul-term */
char finst[19], /* func specific name, null-term */
char msgtext[71]) /* msg text buffer, null-term */
{
/* first test for null input */
if (*input_ind == -1)
```

```

{
/* input is null, likewise output */
*output_ind = -1;
}
else
{
/* input is not null. set output to 2*input-4 */
*output = 2 * (*input) - 4;
/* and set out null indicator to zero */
*output_ind = 0;
}
/* signal successful completion by leaving sqlstate as is */
/* and exit */
return;
}
/* end of UDF: NUDFT1 */

```

例 4: 次の例では、ストリングの中で最初に現れる母音の位置を戻す Java UDF を登録します。UDF は Java で書かれており、fenced して実行されるクラス javaUDFs の findvwl メソッドです。

```

CREATE FUNCTION findv ( CLOB(100K)
    RETURNS INTEGER
    FENCED
    LANGUAGE JAVA
    PARAMETER STYLE JAVA
    EXTERNAL NAME 'javaUDFs.findvwl'
    NO EXTERNAL ACTION
    CALLED ON NULL INPUT
    DETERMINISTIC
    NO SQL

```

例 5: この例では、タイプ SHAPE の 2 つのパラメーター g1 および g2 を入力として取るユーザー定義述部 WITHIN を概説します。

```

CREATE FUNCTION within (g1 SHAPE, g2 SHAPE)
    RETURNS INTEGER
    LANGUAGE C
    PARAMETER STYLE SQL
    NOT VARIANT
    NOT FENCED
    NO SQL
    NO EXTERNAL ACTION
    EXTERNAL NAME 'db2sefn!SDESpatialRelations'
    PREDICATES
    WHEN = 1
    FILTER USING mbrOverlap(g1..xmin, g1..ymin, g1..xmax, g1..max,
        g2..xmin, g2..ymin, g2..xmax, g2..ymax)
    SEARCH BY INDEX EXTENSION gridIndex
    WHEN KEY(g1) USE withinExp1Rule(g2)
    WHEN KEY(g2) USE withinExp1Rule(g1)

```

WITHIN 関数の記述は、任意のユーザー定義の関数の記述に類似しているものの、以下を追加することにより、この関数がユーザー定義の述部で使用できることを指定することができます。

- **PREDICATES WHEN = 1** は、DML ステートメントの WHERE 節でこの関数が

```
within(g1, g2) = 1
```

と表されるときに、述部はユーザー定義の述部として扱われ、索引拡張 *gridIndex* で定義される索引は、この述部に適合する行を検索するのに使用されるように指定します。定数が指定される場合には、DML ステートメントで指定される定数

CREATE FUNCTION (外部スカラー)

は、索引の作成ステートメントで指定される定数と完全に一致していなければなりません。この条件は、主に、結果タイプが 1 または 0 のいずれかになるブール式に対応するように提供されています。他の場合には、EXPRESSION 節を選択するとよいでしょう。

- **FILTER USING mbrOverlap** は、フィルター関数 `mbrOverlap` を参照します。これは、WITHIN 述部の低コスト・バージョンです。上の例では、`mbrOverlap` 関数は入力として最小の境界長方形を使用し、これらがオーバーラップするかどうかを素早く判別します。2 つの入力の形の最小の境界長方形がオーバーラップしない場合、`g1` が `g2` に含まれることはありません。このようにして、タプルを安全に廃棄でき、コストの高い WITHIN 述部のアプリケーションを避けることができます。
- **SEARCH BY INDEX EXTENSION** 節は、索引拡張と検索ターゲットの組み合わせをこのユーザー定義の述部で使用できることを指定します。

例 6: この例では、タイプ POINT の 2 つのパラメーター P1 および P2 を入力として取るユーザー定義述部 DISTANCE を概説します。

```
CREATE FUNCTION distance (P1 POINT, P2 POINT)
  RETURNS INTEGER
  LANGUAGE C
  PARAMETER STYLE SQL
  NOT VARIANT
  NOT FENCED
  NO SQL
  NO EXTERNAL ACTION
  EXTERNAL NAME 'db2sefn!SDEDistances'
  PREDICATES
  WHEN > EXPRESSION AS distExpr
  SEARCH BY INDEX EXTENSION gridIndex
  WHEN KEY(P1) USE distanceGrRule(P2, distExpr)
  WHEN KEY(P2) USE distanceGrRule(P1, distExpr)
```

DISTANCE 関数の記述は、任意のユーザー定義関数の記述に類似しているものの、以下の追加により、この関数が述部で使用される場合に、この述部がユーザー定義述部であることを指定します。

- **PREDICATES WHEN > EXPRESSION AS distExpr** も、有効な述部指定です。WHEN 節で式が指定されると、この述部が DML ステートメントのユーザー定義述部であるかどうかを判別するために、この式の結果タイプが使用されます。以下に例を示します。

```
SELECT T1.C1
  FROM T1, T2
 WHERE distance (T1.P1, T2.P1) > T2.C2
```

述部指定 `distance` は、2 つのパラメーターを入力として使用し、タイプ INTEGER の `T2.C2` を使用して結果を比較します。(特定の定数を使用する場合とは異なり) 式の右辺のデータ・タイプのみ問題となるため、CREATE FUNCTION DDL にある EXPRESSION 節を選択して、比較値としてワイルドカードを指定するとよいでしょう。

別の方法として、以下のものも有効なユーザー定義述部です。

```
SELECT T1.C1
  FROM T1, T2
 WHERE distance(T1.P1, T2.P1) > distance (T1.P2, T2.P2)
```

現在のところ、右辺しか式として扱われないという制限があります。左辺の項は、ユーザー定義述部用のユーザー定義関数です。

- **SEARCH BY INDEX EXTENSION** 節は、索引拡張と検索ターゲットの組み合わせをこのユーザー定義の述部に使用できることを指定します。 `distance` 関数の場合、`distExpr` として指定された式も範囲生成関数 (索引拡張の一部として定義) に渡される検索引数の 1 つです。式の ID は、式の名前を定義するのに使用され、引数として範囲生成関数に渡されます。

CREATE FUNCTION (外部表)

CREATE FUNCTION (外部表) ステートメントは、ユーザー定義の外部表関数を現行サーバーに登録する場合に使用されます。

表関数は、SELECT の FROM 節で使用することができ、行を一度に 1 行戻すことによって、SELECT に表を戻します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する CREATE_EXTERNAL_ROUTINE 権限、および以下の少なくとも 1 つ。
 - データベースに対する IMPLICIT_SCHEMA 権限 (関数の暗黙または明示のスキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (関数のスキーマ名が存在する場合)
- SYSADM または DBADM 権限

非 fenced の関数を作成するには、ステートメントの許可 ID の特権に、以下の特権の少なくとも 1 つが含まれている必要があります。

- データベースに対する CREATE_NOT_FENCED_ROUTINE 権限
- SYSADM または DBADM 権限

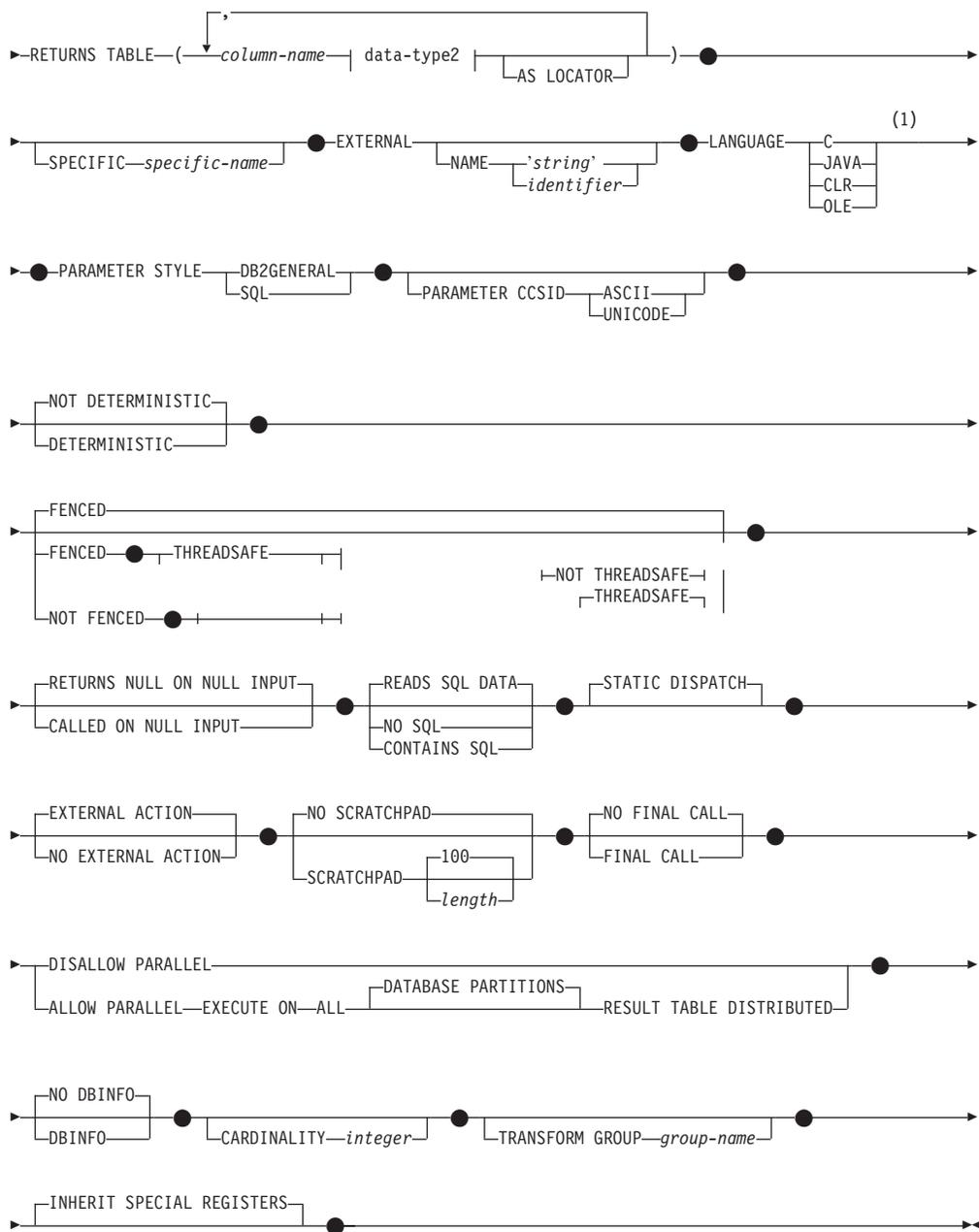
fenced 関数を作成する場合には、さらに別の権限や特権は必要ありません。

構文

```

▶▶ CREATE FUNCTION function-name (
    ┌──────────────────────────────────────────────────────────────────────────┐
    │ parameter-declaration ───────────────────────────────────────────────────┘
    └──────────────────────────────────────────────────────────────────────────┘
)
  
```

CREATE FUNCTION (外部表)



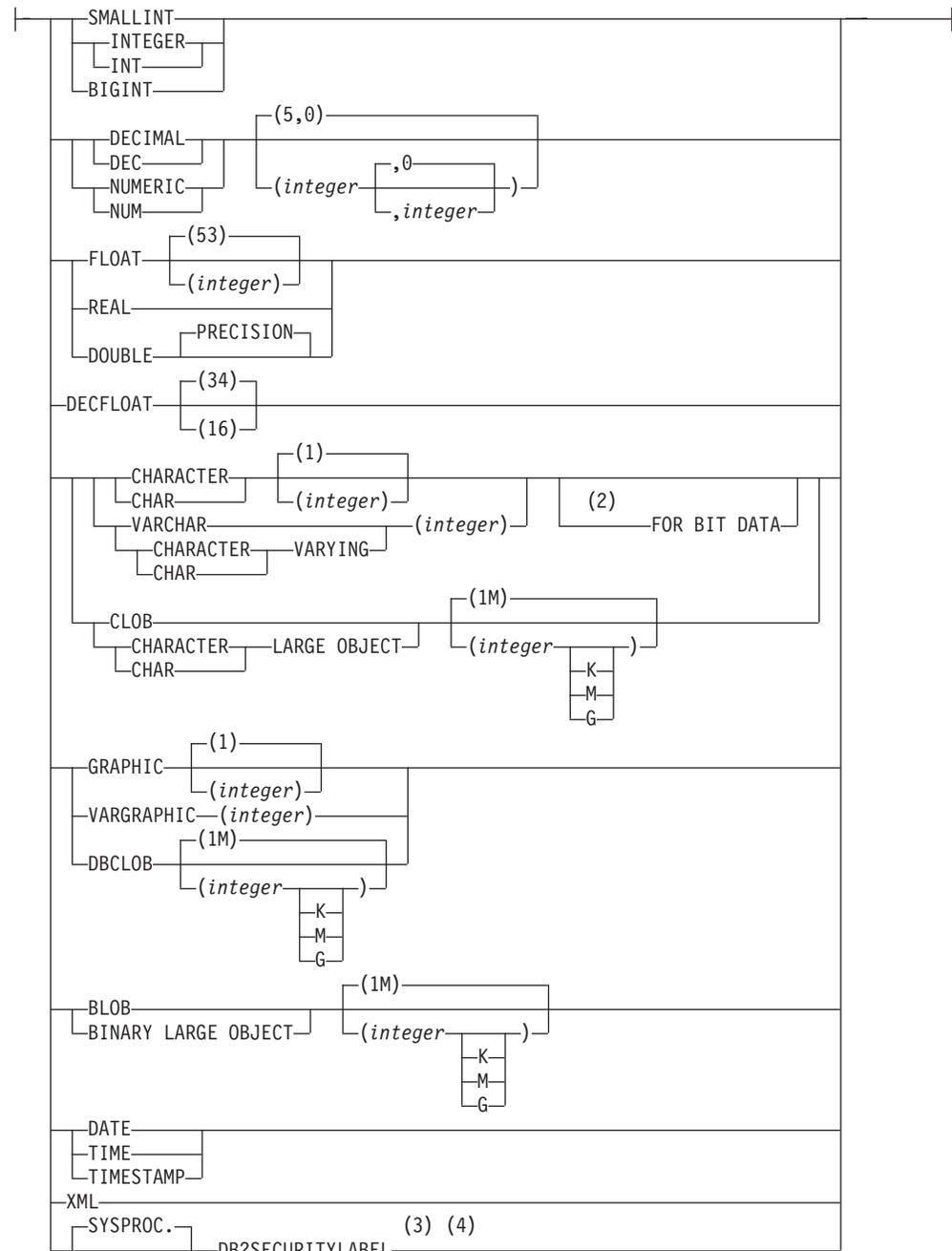
parameter-declaration:

parameter-name data-type1 AS LOCATOR

data-type1、data-type2:

built-in-type
 distinct-type-name
 structured-type-name
 REF (*type-name*)

built-in-type:



注:

- 1 LANGUAGE OLE DB 外部表関数の作成の詳細は、『CREATE FUNCTION (OLE DB 外部表)』を参照してください。LANGUAGE SQL 表関数の作成の詳細は、「CREATE FUNCTION (SQL スカラー、表、または行)」を参照してください。
- 2 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。

CREATE FUNCTION (外部表)

- DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE 42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。

説明

function-name

定義する関数の名前を指定します。これは、関数を指定する修飾または非修飾の名前です。 *function-name* (関数名) の非修飾形式は SQL ID です (最大長 128)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。最初のパラメーターが構造化タイプの場合、修飾名は、最初のパラメーターのデータ・タイプと同じであってはなりません。

暗黙または明示の修飾子を含む名前、およびパラメーターの数と各パラメーターのデータ・タイプ (データ・タイプの長さ、精度、または位取りの各属性には関係なく) は、カタログに記述されている関数を指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数およびデータ・タイプとの組み合わせは、そのスキーマ内では当然ユニークですが、複数のスキーマ間でユニークである必要はありません。

2 つの部分からなる名前を指定する場合、'SYS' で始まる *schema-name* (スキーマ名) は使用できません (SQLSTATE 42939)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*function-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

関数のシグニチャーに何らかの差異があれば、同じ名前を複数の関数に使用することができます。禁止されてはいませんが、外部ユーザー定義表関数の名前として、組み込み関数と同じ名前を指定すべきではありません。

(parameter-declaration,...)

関数の入力パラメーターの数を指定するとともに、各パラメーターのデータ・タイプを指定します。このリストには、関数が受け取ることを予期している各パラメーターごとに 1 つの項目を指定する必要があります。パラメーターの数は 90 を超えることはできません (SQLSTATE 54023)。

パラメーターのない関数も登録可能です。この場合、指定するデータ・タイプがない場合でも、括弧はコーディングする必要があります。以下に例を示します。

```
CREATE FUNCTION WOOFER() ...
```

対応するすべてのパラメーターで、1 つのスキーマ内で名前が同じ 2 つの関数が、まったく同じタイプを持つことはできません。このタイプの比較では長さ、精度、および位取りは考慮されません。したがって、CHAR(8) と CHAR(35)、

また DECIMAL(11,2) と DECIMAL (4,3) は、それぞれ同じタイプと見なされません。Unicode データベースの場合には、CHAR(13) と GRAPHIC(8) は、それぞれ同じタイプと見なされます。さらに、DECIMAL と NUMERIC などのように、この目的で複数のタイプが同じタイプとして扱われることがあります。シングニチャーが重複していると、エラー (SQLSTATE 42723) を戻します。

parameter-name

入力パラメーターにオプションの名前を指定します。この名前は、パラメーター・リスト内の他のすべての *parameter-name* と同じにすることはできません (SQLSTATE 42734)。

data-type1

入力パラメーターのデータ・タイプを指定します。データ・タイプは、組み込みデータ・タイプ、特殊タイプ、構造化タイプ、または参照タイプにすることができます。各組み込みデータ・タイプの詳細な説明は、『CREATE TABLE』を参照してください。一部のデータ・タイプは、すべての言語でサポートされません。SQL データ・タイプとホスト言語データ・タイプの対応関係の詳細については、「組み込み SQL アプリケーション内で SQL データ・タイプにマップするデータ・タイプ」を参照してください。

- 日時タイプのパラメーターは文字データ・タイプとして受け渡され、そのデータは ISO 形式で受け渡されます。
- DECIMAL (および NUMERIC) は、LANGUAGE C と OLE では無効です (SQLSTATE 42815)。
- XML は、LANGUAGE OLE では無効です。
- 関数内での XML 値の表現は、関数呼び出しでパラメーターとして渡される XML 値をシリアライズしたバージョンなので、タイプ XML のパラメーターは構文 XML AS CLOB(*n*) を使用して宣言する必要があります。
- CLR は 28 より大きい DECIMAL スケールをサポートしていません (SQLSTATE 42613)。
- 配列タイプを指定することはできません (SQLSTATE 42815)。

ユーザー定義特殊タイプの場合、パラメーターの長さ、精度、または桁数の属性は、特殊タイプのソース・タイプのもの (CREATE TYPE で指定されたもの) になります。特殊タイプのパラメーターは、特殊タイプのソース・タイプとして受け渡されます。特殊タイプの名前が修飾されていない場合は、データベース・マネージャーによって SQL パス内のスキーマが検索され、スキーマ名が解決されます。

ユーザー定義構造化タイプの場合、適切なトランスフォーム関数が関連するトランスフォーム・グループに存在する必要があります。

参照タイプの場合、パラメーターが有効範囲を指定されていない場合は、パラメーターを REF(*type-name*) で指定できます。

AS LOCATOR

実際の値の代わりにパラメーターの値へのロケーターを関数に受け渡すことを指定します。LOB データ・タイプを持つパラメーター、または LOB データ・タイプに基づく特殊タイプにのみ AS LOCATOR を指定します (SQLSTATE 42601)。値の代わりにロケーターを受け渡すと、関数に受け渡されるバイト数を特にパラメーターの値が非常に大きい場合に少なくできます。

CREATE FUNCTION (外部表)

AS LOCATOR 節は、データ・タイプをプロモート可能かどうかの判別に効果はなく、また関数解決で 사용되는関数シグニチャーにも影響を与えません。

関数が FENCED で NO SQL オプションを持っている場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

RETURNS TABLE

関数の出力が表であることを指定します。このキーワードに続く括弧は、表の列の名前とタイプのリストを区切るもので、他の指定 (例えば、制約) のない単純な CREATE TABLE ステートメントの形式と類似しています。255 列以内が許可されます (SQLSTATE 54011)。

column-name

この列の名前を指定します。名前を修飾することはできず、表の複数の列に対して同じ名前を使用することはできません。

data-type2

列のデータ・タイプを指定します。構造化タイプ以外であれば、特定言語において、UDF 作成のパラメーターとしてサポートされるどのようなデータ・タイプでも構いません (SQLSTATE 42997)。

AS LOCATOR

data-type2 が LOB タイプまたは LOB タイプに基づく特殊タイプの場合、このオプションを使用すると、関数は結果表でインスタンス化される LOB 値のロケーターを戻します。

この節で使用できる有効なタイプについては、『CREATE FUNCTION (外部スカラー)』で説明されています。

SPECIFIC *specific-name*

定義する関数のインスタンスに対するユニーク名を指定します。この特定名は、この関数をソース関数として使用する場合、この関数をドロップする場合、またはこの関数にコメントを付ける場合に使用することができます。これは、関数の呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 128)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別の関数インスタンスを指定するものであってはなりません。そうでない場合、エラー (SQLSTATE 42710) になります。

specific-name は、既存の *function-name* (関数名) と同じでも構いません。

修飾子を指定しない場合、*function-name* に使用された修飾子が使用されます。修飾子を指定する場合は、*function-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラー (SQLSTATE 42882) になります。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成されるユニーク名は、SQL の後に文字のタイム・スタンプが続く名前です (SQLyymmddhhmmssxxx)。

EXTERNAL

この節は、CREATE FUNCTION ステートメントが、外部プログラミング言語で書かれたコードに基づく新しい関数を登録するのに使用されており、文書化されたリンケージの規則とインターフェースに準拠していることを示します。

NAME 節を指定しない場合、"NAME *function-name*" が想定されます。

NAME 'string'

この節は、定義する関数をインプリメントするためのユーザー作成コードを指定します。

'string' オプションは、最大 254 バイトのストリング定数です。ストリングに使用される形式は、指定した LANGUAGE によって異なります。

• LANGUAGE C の場合:

指定する *string* (ストリング) は、作成しているユーザー定義関数を実行するためにデータベース・マネージャーが呼び出すライブラリー名と、そのライブラリー中の関数名です。ライブラリー (およびそのライブラリー中の関数) は、CREATE FUNCTION ステートメントの実行時に存在している必要はありません。ただし、関数が SQL ステートメントで使用される時点では、そのライブラリーとそのライブラリー内の関数が存在していなければならない、しかもデータベース・サーバーのマシンからアクセス可能でなければなりません。

string は、以下のように指定できます。

```
▶─' ┌── library_id ───┐ ┌── !-func_id ───┐ ───▶
    └── absolute_path_id ───┘ └──┘
```

単一引用符内に、余分なブランクを使用することはできません。

library_id

関数を含むライブラリー名を指定します。データベース・マネージャーは、次のようにしてこのライブラリーを特定します。

- UNIX システムの場合、*library_id* が 'myfunc' と指定されており、データベース・マネージャーが /u/production から実行されていると、データベース・マネージャーはライブラリー /u/production/sql/lib/function/myfunc で関数を特定します。
- Windows オペレーティング・システムの場合、データベース・マネージャーは LIBPATH または PATH 環境変数に指定されているディレクトリー・パスから関数を特定します。

absolute_path_id

関数を含んでいるファイルの絶対パス名を指定します。

例えば、UNIX システムの場合、'/u/jchui/mylib/myfunc' を指定すると、データベース・マネージャーは /u/jchui/mylib を調べて myfunc 共用ライブラリーを探します。

Windows オペレーティング・システムの場合、'd:¥mylib¥myfunc.dll' を指定すると、データベース・マネージャーは d:¥mylib ディレクトリーからダイナミック・リンク・ライブラリー myfunc.dll をロードします。絶対パス ID がルーチン本体の識別に使用されている場合は、.dll 拡張子を必ず付加してください。

CREATE FUNCTION (外部表)

! func_id

呼び出される関数の入り口点名を指定します。! は、ライブラリー ID と関数 ID との間の区切り文字です。

例えば、UNIX システムで 'mymod!func8' と指定すると、データベース・マネージャーはライブラリー \$inst_home_dir/sql/lib/function/mymod を調べて、そのライブラリー内の入り口点 func8 を使用します。

Windows オペレーティング・システムの場合 'mymod!func8' を指定すると、データベース・マネージャーは mymod.dll ファイルをロードして、そのダイナミック・リンク・ライブラリー (DLL) の func8() 関数を呼び出します。

ストリングの形式が正しくない場合には、エラーが戻されます (SQLSTATE 42878)。

いずれの場合も、すべての外部関数の本体は、すべてのデータベース・パーティションで使用可能なディレクトリーにある必要があります。

• LANGUAGE JAVA の場合:

指定する *string* には、作成中のユーザー定義関数を実行するためにデータベース・マネージャーが呼び出す、任意指定の jar ファイル、クラス ID、およびメソッド ID が含まれています。クラス ID とメソッド ID は、CREATE FUNCTION ステートメントの実行時には存在している必要はありません。jar_id を指定する場合、ID は、CREATE FUNCTION ステートメントの実行時に存在していなければなりません。ただし、関数を SQL ステートメントで使用する時点で、メソッド ID は存在しなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。

string は、以下のように指定できます。

```
→ ' _____class_id_____ . _____method_id_____ '
    |_____jar_id :_____| |_____!_____|
```

単一引用符内に、余分なブランクを使用することはできません。

jar_id

jar の集合をデータベースへインストールしたときに、その jar の集合に付けられた jar ID を指定します。これは、単純 ID またはスキーマ修飾 ID のいずれかにすることができます。例えば、'myJar' や 'mySchema.myJar' のようになります。

class_id

Java オブジェクトのクラス ID を指定します。クラスがパッケージの一部である場合、クラス ID の部分に完全なパッケージ接頭部 (例: 'myPacks.UserFuncs') が含まれている必要があります。Java 仮想マシンは、ディレクトリー './myPacks/UserFuncs/' 中のクラスを探し

ます。Windows 32 ビット・オペレーティング・システム では、Java 仮想マシンはディレクトリー '`...¥myPacks¥UserFuncs¥`' を探索します。

method_id

呼び出す Java オブジェクトのメソッド名を指定します。

- LANGUAGE CLR の場合:

指定された *string* は、作成する関数を実行するためにデータベース・マネージャーが呼び出す .NET アセンブリー (ライブラリーまたは実行可能モジュール)、そのアセンブリー内のクラス、およびそのクラス内のメソッドを表します。モジュール、クラス、およびメソッドは、CREATE FUNCTION ステートメントの実行時に存在している必要はありません。ただし、関数を SQL ステートメントで使用する時点では、モジュール、クラス、およびメソッドは存在していなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42724)。

'clr' コンパイラー・オプションで管理対象コード拡張を指定してコンパイルされている C++ ルーチンは、'LANGUAGE C' ではなく 'LANGUAGE CLR' としてカタログする必要があります。DB2 は、必要な実行時の決定を行えるようにするために、.NET インフラストラクチャーがユーザー定義関数内で使用されていることを認識している必要があります。.NET インフラストラクチャーを使用するすべてのユーザー定義関数は、'LANGUAGE CLR' としてカタログする必要があります。

string は、以下のように指定できます。

►► '—assembly—:—class_id—!—method_id—' ◀◀

名前は、単一引用符で囲む必要があります。余分なブランクを使用することはできません。

assembly

クラスを含む DLL ファイルまたは他のアセンブリー・ファイルを指定します。ファイル拡張子 (.dll など) まで指定します。絶対パス名を指定しない場合、ファイルは DB2 インストール・パスの関数ディレクトリー (例えば、`c:¥sqllib¥function`) にあるものとされます。ファイルがインストール関数ディレクトリーのサブディレクトリーにある場合は、絶対パスを指定せずに、ファイル名の前にサブディレクトリーを指定します。例えば、インストール・ディレクトリーが `c:¥sqllib` であり、アセンブリー・ファイルが `c:¥sqllib¥function¥myprocs¥mydotnet.dll` であるなら、アセンブリーの指定は 'myprocs¥mydotnet.dll' とするだけで十分です。このパラメーターの大文字小文字が区別されるかどうかは、ファイル・システムの設定と同じです。

class_id

呼び出すメソッドが属するアセンブリー内のクラスの名前を指定します。クラスがネーム・スペース内にある場合は、クラスだけでなく絶

NAME *identifier*

この節は、定義している関数をインプリメントするユーザー作成コードの名前を指定します。指定する *identifier* は SQL ID です。SQL ID は、ストリングの *library-id* として使用されます。区切られた ID でない場合、ID は大文字に変換されます。ID がスキーマ名で修飾されている場合、スキーマ名の部分は無視されます。この形式の NAME は、LANGUAGE C でのみ使用可能です。

LANGUAGE

この節は必須で、ユーザー定義関数の本体が準拠している言語インターフェース規則を指定するのに使用します。

C これは、データベース・マネージャーが、ユーザー定義関数を C の関数であるかのように呼び出すことを意味します。ユーザー定義関数は、標準 ANSI C プロトタイプで定義されている C 言語の呼び出しおよびリンケージの規則に準拠していなければなりません。

JAVA データベース・マネージャーは、Java クラスのメソッドとしてユーザー定義関数を呼び出します。

CLR データベース・マネージャーは、.NET クラスのメソッドとしてユーザー定義関数を呼び出します。LANGUAGE CLR は、Windows オペレーティング・システム上で実行するユーザー定義機能のみサポートされます。NOT FENCED は CLR ルーチンに指定できません (SQLSTATE 42601)。

OLE データベース・マネージャーは、OLE 自動化オブジェクトによって公開されたメソッドとして、ユーザー定義関数を呼び出します。ユーザー定義関数は、「*OLE Automation Programmer's Reference*」に説明されている、OLE 自動化データ・タイプと呼び出しメカニズムに準拠している必要があります。

LANGUAGE OLE は、Windows 32 ビット・オペレーティング・システム用の DB2 で保管されたユーザー定義関数に対してのみサポートされます。

LANGUAGE OLE DB 外部表関数の作成の詳細は、『CREATE FUNCTION (OLE DB 外部表)』を参照してください。

PARAMETER STYLE

この節は、関数にパラメーターを渡し、関数から値を戻すのに用いる規則を指定するために使用します。

DB2GENERAL

Java クラスのメソッドとして定義された外部関数との間で、パラメーターを渡し、値を戻す場合に用いる規則を指定します。これは、LANGUAGE JAVA を使用する場合にだけ指定する必要があります。

SQL

C 言語の呼び出しとリンケージの規則、OLE 自動化オブジェクトによって公開されたメソッド、または .NET オブジェクトの共用静的メソッドに準拠する規則を、この外部メソッドとの間でパラメーターを渡し、値を戻す場合の規則として指定します。これは、LANGUAGE C、LANGUAGE CLR、または LANGUAGE OLE を使用する場合に指定する必要があります。

CREATE FUNCTION (外部表)

PARAMETER CCSID

関数とやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。関数が呼び出される時のアプリケーション・コード・ページはデータベース・コード・ページです。

UNICODE

ストリング・データが Unicode でエンコードされることを指定します。データベースが Unicode データベースの場合、文字データは UTF-8、GRAPHIC データは UCS-2 になります。データベースが Unicode データベースでない場合は、文字データは UTF-8 になります。いずれの場合も、関数が呼び出される時のアプリケーション・コード・ページは 1208 です。

データベースが Unicode データベースではないのに、PARAMETER CCSID UNICODE を指定した関数を作成すると、その関数は GRAPHIC タイプやユーザー定義タイプを取ることができません (SQLSTATE 560C1)。

データベースが Unicode データベースでない場合、表関数を PARAMETER CCSID UNICODE を指定して作成できますが、以下の規則が適用されます。

- 表関数を作成するより前に、代替照合シーケンスをデータベース構成に指定する必要があります (SQLSTATE 56031)。PARAMETER CCSID UNICODE 表関数は、データベース構成に指定されている代替照合シーケンスと照合されます。
- CCSID ASCII を指定して作成された表または表関数と、CCSID UNICODE を指定して作成された表または表関数とを、1 つの SQL ステートメント内で両方とも使用することはできません (SQLSTATE 53090)。このことは、ステートメント内で直接参照されている表および表関数、および間接的に (例えば、参照整合性制約、トリガー、マテリアライズ照会表、およびビューの本体内の表によって) 参照されている表および表関数に適用されます。
- PARAMETER CCSID UNICODE を指定して作成された表関数は、SQL 関数または SQL メソッド内では参照できません (SQLSTATE 560C0)。
- PARAMETER CCSID UNICODE を指定して作成された表関数を参照する SQL ステートメントは、SQL 関数または SQL メソッドを呼び出すことができません (SQLSTATE 53090)。
- GRAPHIC タイプ、XML タイプ、およびユーザー定義タイプは、PARAMETER CCSID UNICODE 表関数へのパラメーターとしては使用できません (SQLSTATE 560C1)。
- PARAMETER CCSID UNICODE 表関数を参照するステートメントは、DB2 バージョン 8.1 以降のクライアントからのみ呼び出すことができません (SQLSTATE 42997)。

- SQL ステートメントは常にデータベース・コード・ページで解釈されます。特にこのことは、リテラル、16 進数リテラル、および区切り ID 内のすべての文字がデータベース・コード・ページで表記されていないと意味します。そうでないと、文字は置換文字によって置き換えられてしまいます。

データベースが Unicode ではなく、データベース構成に代替照合シーケンスが指定されている場合、PARAMETER CCSID ASCII または PARAMETER CCSID UNICODE を指定した関数を作成できます。関数とやり取りされるすべてのストリング・データは、適切なコード・ページに変換されます。

この節を LANGUAGE OLE、LANGUAGE JAVA、または LANGUAGE CLR とともに指定することはできません (SQLSTATE 42613)。

DETERMINISTIC または NOT DETERMINISTIC

この節は任意指定で、特定の引数の値に対して関数が常に同じ結果を返すか (DETERMINISTIC)、それとも状態値に依存して関数の結果が影響を受けるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC 関数は、同一の入力で連続で呼び出しが行われたとき、常に同じ表を返します。NOT DETERMINISTIC を指定すると、同じ入力によって常に同じ結果が生じる利点に基づく最適化ができなくなります。非 deterministic である表関数の例として、特殊レジスター、グローバル変数、非 deterministic 関数、またはシーケンスを参照する際に表関数結果表に影響を与えるような表関数があります。

FENCED または NOT FENCED

この節は、関数をデータベース・マネージャーの操作環境のプロセスまたはアドレス・スペースで実行しても「安全」か (NOT FENCED)、そうでないか (FENCED) を指定します。

関数が FENCED として登録されると、データベース・マネージャーは、その内部リソース (データ・バッファーなど) を保護して、その関数からアクセスされないようにします。多くの関数は、FENCED または NOT FENCED のどちらかで実行するように選択することができます。一般に、FENCED として実行される関数は、NOT FENCED として実行されるものと同じようには実行されません。

注意:

適切にコード化、検討、およびテストされていない関数に NOT FENCED を使用すると、DB2 の整合性に危険を招く場合があります。DB2 では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられていますが、NOT FENCED ユーザー定義関数が使用される場合には、完全な整合性を確保できません。

LANGUAGE OLE または NOT THREADSAFE を指定した関数には、FENCED のみを指定できます (SQLSTATE 42613)。

関数が FENCED で NO SQL オプションを持っている場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

ユーザー定義関数を NOT FENCED として登録するには、SYSADM 権限、DBADM 権限、または特殊権限 (CREATE_NOT_FENCED_ROUTINE) が必要です。

CREATE FUNCTION (外部表)

NOT FENCED 節を指定している場合は、LANGUAGE CLR ユーザー定義関数を作成できません (SQLSTATE 42601)。

THREADSAFE または NOT THREADSAFE

関数を他のルーチンと同じプロセスで実行しても安全か (THREADSAFE)、そうでないか (NOT THREADSAFE) を指定します。

関数が OLE 以外の LANGUAGE で定義される場合:

- 関数が THREADSAFE に定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスで関数を呼び出すことができます。一般に、スレッド・セーフにするには、関数はどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。FENCED および NOT FENCED 関数の両方が THREADSAFE になることが可能です。
- 関数が NOT THREADSAFE と定義される場合には、データベース・マネージャーが関数を他のルーチンと同じプロセスで同時に呼び出すことは決してありません。

FENCED 関数の場合、LANGUAGE が JAVA または CLR なら THREADSAFE がデフォルトです。これ以外のすべての言語の場合は、NOT THREADSAFE がデフォルトです。関数が LANGUAGE OLE に定義される場合には、THREADSAFE は指定できません (SQLSTATE 42613)。

NOT FENCED 関数の場合には、THREADSAFE がデフォルトです。NOT THREADSAFE を指定することはできません (SQLSTATE 42613)。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT

このオプション節を使用すると、引数のいずれかが NULL 値の場合に、外部関数を呼び出さないようにすることができます。ユーザー定義関数がパラメーターなしで定義されている場合、この NULL 引数条件は引き起こされることはないため、この仕様のコーディング方法はそれほど重要ではなくなります。

RETURNS NULL ON NULL INPUT が指定されており、表関数 OPEN が実行されるたびに、関数の引数のいずれかが NULL 値の場合、ユーザー定義関数は呼び出されません。試行した表関数スキャンの結果は、空の表 (行のない表) になります。

CALLED ON NULL INPUT が指定されると、引数が NULL 値か否かに関係なくユーザー定義関数が呼び出されます。これは、NULL 値を戻す場合も、通常の (NULL 以外の) 値を戻す場合もあります。ただし、NULL の引数値の有無のテストは UDF が行う必要があります。

値 NULL CALL は、後方互換性またはファミリーの互換性のために、CALLED ON NULL INPUT の同義語として使うことができます。同様に、NOT NULL CALL は、RETURNS NULL ON NULL INPUT の同義語として使用できます。

NO SQL、CONTAINS SQL、READS SQL DATA

関数から SQL ステートメントが発行されるかどうかと、もし発行されればどのタイプかを示します。

NO SQL

関数はどの SQL ステートメントも実行できないことを指示します (SQLSTATE 38001)。ALLOW PARALLEL、EXECUTE ON ALL

DATABASE PARTITIONS、および RESULT TABLE DISTRIBUTED 節がすべて指定されている場合、許容されるオプションは NO SQL のみとなります。

CONTAINS SQL

SQL データの読み取りも変更も行わない SQL ステートメントを、関数で実行できることを指定します (SQLSTATE 38004 または 42985)。どの関数でもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

READS SQL DATA

SQL データを変更しない SQL ステートメントを、関数で実行できることを指定します (SQLSTATE 38002 または 42985)。どの関数でもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

STATIC DISPATCH

このオプション節は、関数解決時に DB2 が関数のパラメーターの静的タイプ (宣言済みタイプ) に基づいて関数を選択するよう指示します。

EXTERNAL ACTION または NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るかどうかを指定します。外部アクションの例としては、メッセージの送信やファイルへのレコードの書き込みがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取ることを指定します。

外部アクションが指定された関数は、関数が並列タスクによって実行されると、不正確な結果を戻す場合があります。例えば、初期呼び出しを受けるたびに注釈を送信する関数の場合、関数ごとに 1 つの注釈が送信されるのではなく、並列タスクごとに 1 回ずつ送信されることになります。並列処理を正しく扱うことのできない関数については、DISALLOW PARALLEL 節を指定します。

NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取らないことを指定します。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。

NO SCRATCHPAD または SCRATCHPAD *length*

この節はオプションであり、この外部関数に対してスクラッチパッドを用意するか否かを指定するのに使用することができます。(ユーザー定義関数を再入可能にすることを強くお勧めします。再入可能にすると、スクラッチパッドによってある呼び出しと次の呼び出しとの間に関数が“状態を保管する”手段が用意されます。)

SCRATCHPAD を指定すると、ユーザー定義関数の最初の呼び出し時に、その外部関数によって使用されるスクラッチパッドにメモリーが割り振られます。このスクラッチパッドには、次の特性があります。

CREATE FUNCTION (外部表)

- *length* を指定すると、スクラッチパッドのサイズをバイト単位で設定できます。この値は 1 から 32 767 の範囲で指定する必要があります (SQLSTATE 42820)。デフォルト値は 100 です。
- すべて 'X'00' に初期化されます。
- その有効範囲は、該当の SQL ステートメントです。SQL ステートメントでの外部関数に対する参照ごとに 1 つのスクラッチパッドがあります。したがって、以下のステートメントの UDFX 関数が、SCRATCHPAD キーワードを使用して定義されると、2 つのスクラッチパッドが割り当てられます。

```
SELECT A.C1, B.C2
FROM TABLE (UDFX(:hv1)) AS A,
TABLE (UDFX(:hv1)) AS B
WHERE ...
```

- スクラッチパッドは持続します。スクラッチパッドは、ステートメントの実行開始時に初期化され、ある呼び出しから次の呼び出しにスクラッチパッドの状態を保存するために、外部表関数で使用することができます。UDF に FINAL CALL キーワードも指定されている場合、DB2 がスクラッチパッドを変更することはありません。また、特殊 FINAL 呼び出しがなされると、スクラッチパッドに固定されていたすべてのリソースが解放されます。

NO FINAL CALL が指定またはデフォルト指定されている場合は、DB2 が OPEN 呼び出しごとにスクラッチパッドを初期化し直すので、外部表関数は CLOSE 呼び出し時に、スクラッチパッドに固定されているすべてのリソースに対して終結処理を行います。FINAL CALL または NO FINAL CALL の判別、およびスクラッチパッドの関連する動作は、重要な考慮事項です。表関数が副照会または結合で使用されるときは、ステートメントの実行中に複数の OPEN 呼び出しが生じ得るので、特に重要です。

- これは、外部関数が獲得するシステム・リソース (メモリーなどの) の中央点として使用することもできます。関数は、最初の呼び出しでメモリーを獲得し、そのアドレスをスクラッチパッドに保管して、後の呼び出しでそれを参照することができます。

(上で概説したように、FINAL CALL/NO FINAL CALL キーワードは、スクラッチパッドの再初期化を制御するために使用され、スクラッチパッドに固定されているリソースを外部表関数が解放する時期を指示します。)

SCRATCHPAD を指定すると、ユーザー定義関数を呼び出すたびに、スクラッチパッドをアドレッシングする外部関数に追加の引数が渡されます。

NO SCRATCHPAD を指定すると、外部関数に対してスクラッチパッドは割り振られず、渡されません。

FINAL CALL または NO FINAL CALL

この節はオプションであり、外部関数に対する最終呼び出し (および別個の最初の呼び出し) が行われるか否かを指定します。この節は、スクラッチパッドが再初期化される時期も制御します。NO FINAL CALL が指定されている場合は、DB2 はオープン、取り出しおよびクローズの 3 つのタイプの表関数の呼び出ししか行うことができません。しかし、FINAL CALL が指定されている場合は、オープン、取り出しおよびクローズに加えて、表関数に対して最初の呼び出しと最終呼び出しを行うことができます。

外部表関数の場合、どのオプションが選択されたかにかかわらず、呼び出しタイプ引数は常に存在します。

割り込みかトランザクションの終了のために最終呼び出しが行われると、UDF は CLOSE カーソル以外の SQL ステートメントを発行できません (SQLSTATE 38505)。こうした最終呼び出しの状況の場合には、「呼び出しタイプ」の引数に特殊値が渡されます。

DISALLOW PARALLEL または ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

関数への単一の参照に対して、関数の呼び出しを並列化するかどうかを指定します。

DISALLOW PARALLEL

関数を呼び出すごとに、DB2 が単一のデータベース・パーティションに対して関数を呼び出すように指定します。

ALLOW PARALLEL EXECUTE ON ALL DATABASE PARTITIONS RESULT TABLE DISTRIBUTED

関数を呼び出すごとに、DB2 がすべてのデータベース・パーティションに対して関数を呼び出すように指定します。各データベース・パーティションで取得された結果セットの和集合が戻されます。この関数は SQL ステートメントを実行できません (NO SQL 節も指定する必要があります)。

NO DBINFO または DBINFO

この節はオプションで、DB2 において既知である特定の情報を追加の呼び出し時引数として関数に渡すか (DBINFO)、または渡さないか (NO DBINFO) を指定します。NO DBINFO がデフォルト値です。DBINFO は、LANGUAGE OLE ではサポートされません (SQLSTATE 42613)。

DBINFO を指定すると、以下の情報を含む構造が関数に渡されます。

- データベース名 - 現在接続されているデータベースの名前。
- アプリケーション ID - データベースへの接続ごとに確立された、ユニークなアプリケーション ID。
- アプリケーション許可 ID - アプリケーション実行時の許可 ID。この関数とアプリケーションとの間でネストされている関数は無関係です。
- コード・ページ - データベースのコード・ページ。
- スキーマ名 - 外部表関数には適用されません。
- 表名 - 外部表関数には適用されません。
- 列名 - 外部表関数には適用されません。
- データベースのバージョンまたはリリース - 関数を呼び出すデータベース・サーバーのバージョン、リリース、および修正レベル。
- プラットフォーム - サーバーのプラットフォーム・タイプ。
- 表関数の結果の列番号 - この関数を参照しているステートメントによって使用される、結果の列番号の配列。この情報により、関数はすべての列値を戻す代わりに必要な列値だけを戻すことができます。
- データベース・パーティション番号 - 外部表関数が呼び出されたデータベース・パーティションの番号。単一データベース・パーティション環境では、この値は 0 になります。

CREATE FUNCTION (外部表)

CARDINALITY *integer*

この節はオプションで、関数によって戻されると予想される行数の見積もりを最適化のために指定します。 *integer* の値の有効範囲は、0 から 9 223 372 036 854 775 807 (両端の値を含む) です。

表関数に対して CARDINALITY 節の指定がない場合、DB2 はデフォルト値として有限の値を想定します (RUNSTATS ユーティリティが統計を収集していない表に対して想定される値と同じ)。

警告: 関数が事実上無限のカーディナリティーを持っている (すなわち、呼び出されるといつでも行を戻し、"end-of-table" 条件を戻さない) 場合、"end-of-table" 条件を必要とする照会は無限に実行されるので、照会を中断させる必要があります。このような照会の例としては、GROUP BY 節や ORDER BY 節を含む照会があります。このような UDF を作成することは推奨されていません。

TRANSFORM GROUP *group-name*

関数を呼び出す際のユーザー定義の構造化タイプのトランスフォーメーションに使用するトランスフォーム・グループを指定します。関数定義にパラメーター・データ・タイプとしてユーザー定義の構造化タイプが含まれている場合、トランスフォームが必要になります。この節が指定されない場合には、デフォルトのグループ名 DB2_FUNCTION が使用されます。参照された構造化タイプに、指定した (またはデフォルトの) グループ名が定義されていない場合には、エラーになります (SQLSTATE 42741)。指定した *group-name* または構造化タイプに必須の FROM SQL 変換関数が定義されていない場合には、エラーになります (SQLSTATE 42744)。

INHERIT SPECIAL REGISTERS

このオプション節は、関数の更新可能な特殊レジスターが、呼び出しステートメントの環境からの初期値を継承することを指定します。カーソルの選択ステートメントで呼び出される関数の場合、初期値はカーソルがオープンした際の環境から継承します。ネストされたオブジェクト (例えば、トリガーまたはビュー) に呼び出されるルーチンの場合、初期値は (オブジェクト定義から継承するのではなく) ランタイム環境から継承します。

特殊レジスターに対する変更が、関数の呼び出し側に戻されることはありません。

更新不能の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、デフォルト値に設定されます。

規則

- パーティション・データベース環境では、外部ユーザー定義関数またはメソッドでの SQL の使用はサポートされていません (SQLSTATE 42997)。
- 索引拡張を定義するには、NO SQL として定義されたルーチンしか使用できません (SQLSTATE 428F8)。
- 関数が SQL を許可する場合、外部プログラムは、フェデレーテッド・オブジェクトへのアクセスを試行してはなりません (SQLSTATE 55047)。
- 表アクセスの制限** 関数が READS SQL DATA に定義されている場合には、関数のいかなるステートメントも、関数を呼び出したステートメントによって変更されている表にはアクセスできません (SQLSTATE 57053)。例えば、ユーザー定義

関数 BONUS() が READS SQL DATA に定義されているとします。ステートメント UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO) が呼び出される場合、BONUS 関数の SQL ステートメントは、EMPLOYEE 表からの読み取りを行えません。

注

- ユーザー定義関数のパラメーターのデータ・タイプを選択する場合は、入力値に影響を与えるプロモーションの規則を考慮してください。例えば、入力値として使用できる定数のデータ・タイプは、予期される以外の組み込みデータ・タイプである可能性があり、さらには、予期されるデータ・タイプにプロモートできない場合があります。プロモーションの規則に従って、一般にパラメーターには次のデータ・タイプを使用するようにしてください。
 - SMALLINT ではなく INTEGER
 - REAL ではなく DOUBLE
 - CHAR ではなく VARCHAR
 - GRAPHIC ではなく VARGRAPHIC
- プラットフォーム間での UDF の移植性を保つためには、以下のデータ・タイプの使用をお勧めします。
 - FLOAT ではなく DOUBLE または REAL
 - NUMERIC ではなく DECIMAL
 - LONG VARCHAR ではなく CLOB (または BLOB)
- まだ存在していないスキーマ名を用いて関数を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- NOT FENCED として定義される Java ルーチンは、FENCED THREADSAFE として定義されているかのように呼び出されます。
- **特権**
 - 関数の定義者は、関数に対する WITH GRANT OPTION 付きの EXECUTE 特権と、関数をドロップする権利を常に与えられます。
 - 関数を SQL ステートメントで使用する時点で、関数の定義者はその関数によって使用されるすべてのパッケージに対して EXECUTE 特権を持っていないければなりません。
- **互換性**
 - DB2 for z/OS との互換性:
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - ASUTIME NO LIMIT
 - NO COLLID
 - PROGRAM TYPE SUB
 - STAY RESIDENT NO
 - Unicode データベースでの CCSID UNICODE
 - PARAMETER CCSID UNICODE が指定されていない場合、非 Unicode データベース内での CCSID ASCII

CREATE FUNCTION (外部表)

- 以前のバージョンの DB2 との互換性:
 - PARAMETER STYLE SQL の代わりに PARAMETER STYLE DB2SQL を指定できます。
 - DETERMINISTIC の代わりに NOT VARIANT を指定できます。
 - NOT DETERMINISTIC の代わりに VARIANT を指定できます。
 - CALLED ON NULL INPUT の代わりに NULL CALL を指定できます。
 - RETURNS NULL ON NULL INPUT の代わりに NOT NULL CALL を指定できます。
 - DB2GENERAL の代わりに DB2GENRL を指定できます。

例

例 1: 以下の例では、テキスト管理システムにおいて既知の各文書の 1 つの文書 ID 列からなる行を戻す表関数を登録しています。最初のパラメーターは指定された対象領域をマッチングし、2 番目パラメーターには指定されたストリングが入ります。

単一セッションのコンテキスト内では UDF は常に同じ表を戻すため、UDF は DETERMINISTIC として定義されています。DOCMATCH からの出力を定義する RETURNS 節に注意してください。それぞれの表関数に対して、FINAL CALL を指定する必要があります。さらに、この表関数は並列して実行できないので、DISALLOW PARALLEL キーワードが追加されています。DOCMATCH の出力のサイズは大きく変動しますが、DB2 オプティマイザーにとって有用な CARDINALITY 20 が代表値として指定されています。

```
CREATE FUNCTION DOCMATCH (VARCHAR(30), VARCHAR(255))
  RETURNS TABLE (DOC_ID CHAR(16))
  EXTERNAL NAME '/common/docfuncs/rajiv/udfmatch'
  LANGUAGE C
  PARAMETER STYLE SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
  NOT FENCED
  SCRATCHPAD
  FINAL CALL
  DISALLOW PARALLEL
  CARDINALITY 20
```

例 2: 以下の例では、Microsoft Exchange のメッセージのメッセージ・ヘッダー情報と、部分的なメッセージ・テキストの検索に使用する OLE 表関数を登録しています。

```
CREATE FUNCTION MAIL()
  RETURNS TABLE (TIMERECEIVED DATE,
                 SUBJECT VARCHAR(15),
                 SIZE INTEGER,
                 TEXT VARCHAR(30))
  EXTERNAL NAME 'tfmail.header!list'
  LANGUAGE OLE
  PARAMETER STYLE SQL
  NOT DETERMINISTIC
  FENCED
  CALLED ON NULL INPUT
  SCRATCHPAD
```

FINAL CALL
NO SQL
EXTERNAL ACTION
DISALLOW PARALLEL

CREATE FUNCTION (OLE DB 外部表)

CREATE FUNCTION (OLE DB 外部表) ステートメントは、OLE DB Provider からデータをアクセスするための、ユーザー定義の OLE DB 外部表関数をアプリケーション・サーバーに登録する場合に使用します。

表関数 は、SELECT の FROM 節で使用できます。

呼び出し

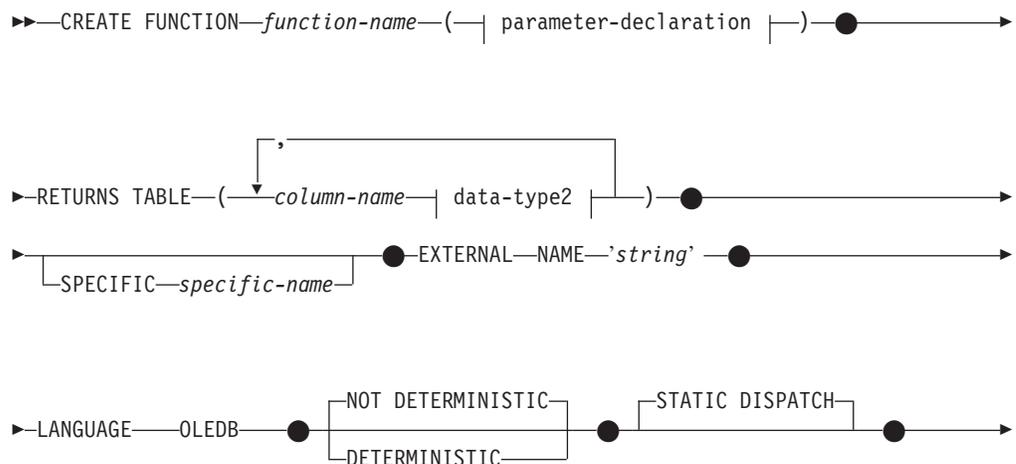
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

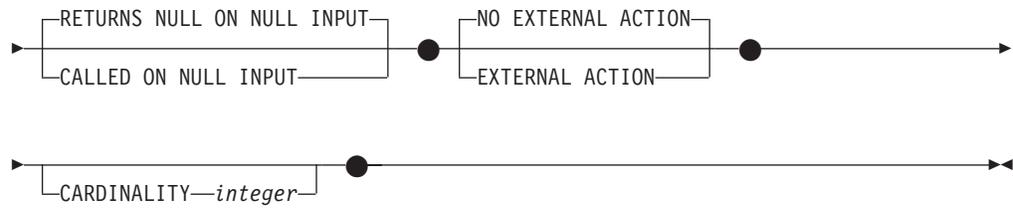
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する CREATE_EXTERNAL_ROUTINE 権限、および以下の少なくとも 1 つ。
 - データベースに対する IMPLICIT_SCHEMA 権限 (関数の暗黙または明示のスキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (関数のスキーマ名が存在する場合)
- SYSADM または DBADM 権限

構文



CREATE FUNCTION (OLE DB 外部表)



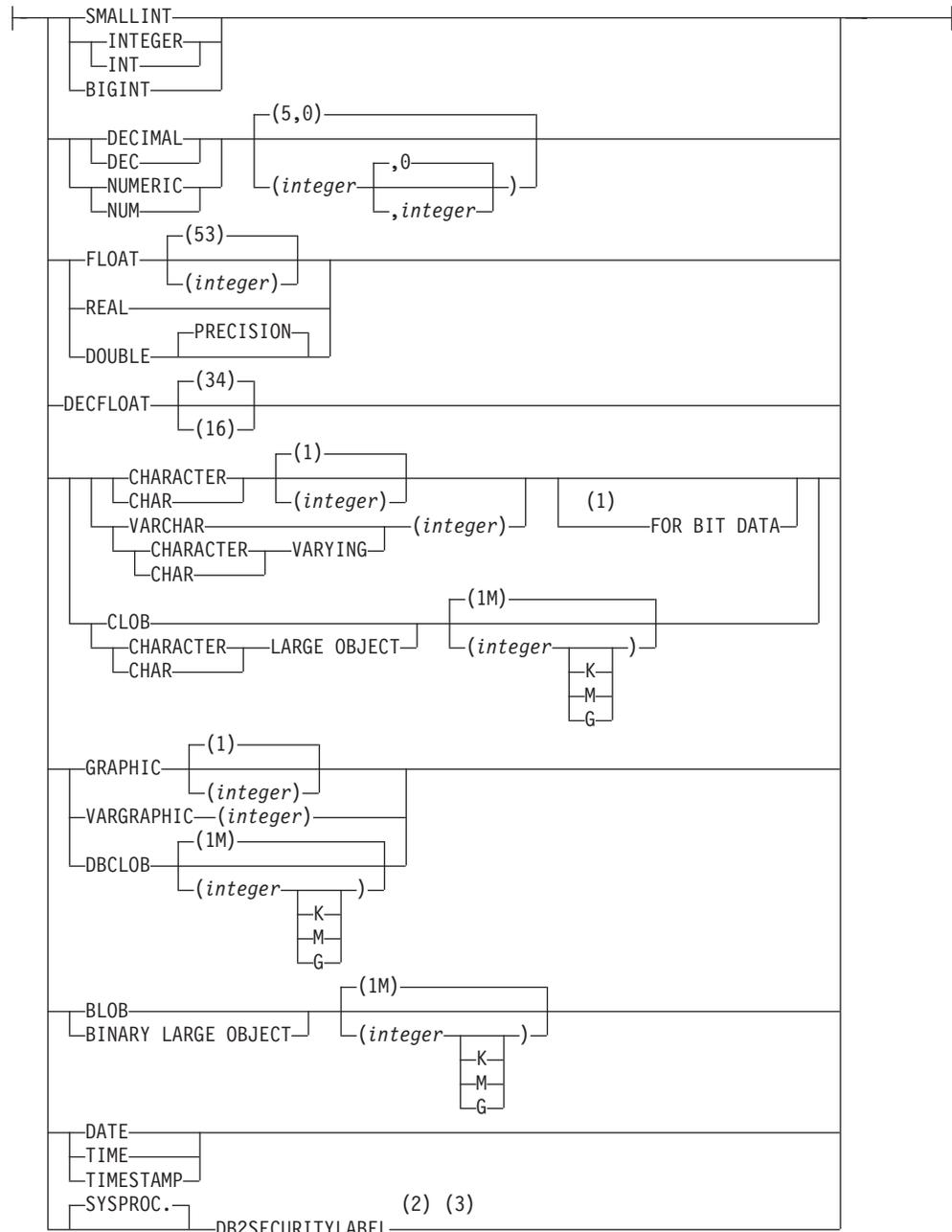
parameter-declaration:



data-type1、data-type2:



built-in-type:



注:

- 1 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- 2 DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- 3 タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE 42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。

説明

function-name

定義する関数の名前を指定します。これは、関数を指定する修飾または非修飾の名前です。 *function-name* (関数名) の非修飾形式は SQL ID です (最大長 18)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

暗黙または明示の修飾子を含む名前、およびパラメーターの数と各パラメーターのデータ・タイプ (データ・タイプの長さ、精度、または位取りの各属性には関係なく) は、カタログに記述されている関数を指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数およびデータ・タイプとの組み合わせは、そのスキーマ内では当然ユニークですが、複数のスキーマ間でユニークである必要はありません。

2 つの部分からなる名前を指定する場合、'SYS' で始まる *schema-name* (スキーマ名) は使用できません (SQLSTATE 42939)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*function-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

関数のシグニチャーに何らかの差異があれば、同じ名前を複数の関数に使用することができます。禁止されてはいませんが、外部ユーザー定義表関数の名前として、組み込み関数と同じ名前を指定すべきではありません。

(parameter-declaration,...)

関数の入力パラメーターを指定するとともに、そのパラメーターのデータ・タイプを指定します。入力パラメーターを指定しないと、データは、外部ソースから取り出されます (多くの場合、照会最適化によってサブセット化されます)。入力パラメーターにより、コマンド・テキストが OLE DB Provider に受け渡されます。

パラメーターのない関数も登録可能です。この場合、指定するデータ・タイプがない場合でも、括弧はコーディングする必要があります。以下に例を示します。

```
CREATE FUNCTION WOOFER() ...
```

対応するすべてのパラメーターで、1 つのスキーマ内で名前が同じ 2 つの関数が、まったく同じタイプを持つことはできません。このタイプの比較では長さ、精度、および位取りは考慮されません。したがって、CHAR(8) と CHAR(35) は、それぞれ同じタイプと見なされます。Unicode データベースの場合には、CHAR(13) と GRAPHIC(8) は、それぞれ同じタイプと見なされます。シグニチャーが重複していると、エラー (SQLSTATE 42723) を戻します。

parameter-name

入力パラメーターにオプションの名前を指定します。

data-type1

入力パラメーターのデータ・タイプを指定します。データ・タイプは、任意

の文字または GRAPHIC ストリング・データ・タイプ、あるいは文字または GRAPHIC ストリング・データ・タイプに基づく特殊タイプにすることができます。タイプ XML のパラメーターはサポートされていません (SQLSTATE 42815)。

各組み込みデータ・タイプの詳細な説明は、『CREATE TABLE』を参照してください。

ユーザー定義特殊タイプの場合、パラメーターの長さ、精度、または桁数の属性は、特殊タイプのソース・タイプのもの (CREATE TYPE で指定されたもの) になります。特殊タイプのパラメーターは、特殊タイプのソース・タイプとして受け渡されます。特殊タイプの名前が修飾されていない場合は、データベース・マネージャーによって SQL パス内のスキーマが検索され、スキーマ名が解決されます。

RETURNS TABLE

関数の出力が表であることを指定します。このキーワードに続く括弧は、表の列の名前とタイプのリストを区切るもので、他の指定 (例えば、制約) のない単純な CREATE TABLE ステートメントの形式と類似しています。

column-name

列の名前を指定します。これは、対応する rowset の列名と同じでなければなりません。名前を修飾することはできず、表の複数の列に対して同じ名前を使用することはできません。

data-type2

列のデータ・タイプを指定します。XML は無効です (SQLSTATE 42815)。

SPECIFIC *specific-name*

定義する関数のインスタンスに対するユニーク名を指定します。この特定名は、この関数をソース関数として使用する場合、この関数をドロップする場合、またはこの関数にコメントを付ける場合に使用することができます。これは、関数の呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 18)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別の関数インスタンスを指定するものであってはなりません。そうでない場合、エラー (SQLSTATE 42710) になります。

specific-name は、既存の *function-name* (関数名) と同じでも構いません。

修飾子を指定しない場合、*function-name* に使用された修飾子を使用されます。修飾子を指定する場合は、*function-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラー (SQLSTATE 42882) になります。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成されるユニーク名は、SQL の後に文字のタイム・スタンプが続く名前です (SQLyymmddhhmmssxxx)。

EXTERNAL NAME '*string*'

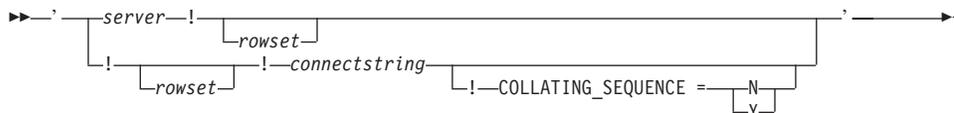
この節は、外部表と OLE DB Provider を指定します。

'*string*' オプションは、最大 254 バイトのストリング定数です。

CREATE FUNCTION (OLE DB 外部表)

指定される文字列は、OLE DB Provider との接続およびセッションを確立し、rowset からデータを取り出すときに使われます。OLE DB Provider とデータ・ソースは、CREATE FUNCTION ステートメントの実行時には存在している必要はありません。

string は、以下のように指定できます。



server

「CREATE SERVER」で定義されているように、データ・ソースのローカル名を指定します。

rowset

OLE DB Provider によって公開された rowset (表) を指定します。カタログまたはスキーマ名をサポートする OLE DB Provider の、完全修飾表名を指定する必要があります。

connectstring

データ・ソースへ接続するときに必要な、初期化プロパティの文字列・バージョン。接続文字列の基本形式は、ODBC 接続文字列に基づいています。この文字列には、セミコロンで区切られた、一連のキーワード/値の対が含まれています。等号 (=) により、各キーワードとその値を区切ります。キーワードは、OLE DB 初期化プロパティ (プロパティ・セット DBPROPSET_DBINIT) の記述か、プロバイダー固有のキーワードです。

COLLATING_SEQUENCE

データ・ソースが、DB2 Database for Linux, UNIX, and Windows と同じ照合順序を使うかどうかを指定します。詳細については、『CREATE SERVER』を参照してください。有効な値は、以下のとおりです。

- Y = 同じ照合順序
- N = 異なる照合順序

COLLATING_SEQUENCE を指定しない場合、データ・ソースと DB2 Database for Linux, UNIX, and Windows の照合順序は異なるものと見なされます。

server を指定する場合、外部名として *connectstring* または COLLATING_SEQUENCE を使うことはできません。それらは、サーバー・オプション CONNECTSTRING および COLLATING_SEQUENCE として定義されています。*server* を指定しないのであれば、*connectstring* を指定する必要があります。*rowset* を指定しないのであれば、表関数には、コマンド・テキストを OLE DB Provider に渡すための入力パラメーターが必要です。

LANGUAGE OLEDB

これを指定すると、データベース・マネージャーは、組み込まれた汎用 OLE DB の消費者情報を配置し、OLE DB Provider からデータを取り出します。開発者側で表関数をインプリメントする必要はありません。

LANGUAGE OLEDB 表関数は、任意のプラットフォームで作成できますが、Microsoft OLE DB によってサポートされているプラットフォーム上でのみ実行できます。

DETERMINISTIC または NOT DETERMINISTIC

この節は任意指定で、特定の引数の値に対して関数が常に同じ結果を戻すか (DETERMINISTIC)、それとも状態値に依存して関数の結果が影響を受けるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC 関数は、同一の入力で連続で呼び出しが行われたとき、常に同じ表を返します。NOT DETERMINISTIC を指定すると、同じ入力によって常に同じ結果が生じる利点に基づく最適化ができなくなります。

STATIC DISPATCH

このオプション節は、関数解決時に DB2 が関数のパラメーターの静的タイプ (宣言済みタイプ) に基づいて関数を選択するよう指示します。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT

このオプション節を使用すると、引数のいずれかが NULL 値の場合に、外部関数を呼び出さないようにすることができます。ユーザー定義関数がパラメーターなしで定義される場合、この NULL 引数条件が引き起こされることはありません。

RETURNS NULL ON NULL INPUT が指定されており、実行時に関数の引数のいずれかが NULL 値の場合、ユーザー定義関数は呼び出されず、結果は空の表、すなわち行のない表になります。

CALLED ON NULL INPUT が指定されると、引数が NULL 値か否かに関係なく実行時にユーザー定義関数が呼び出されます。関数の論理によって、空の表を戻すことも戻さないこともあります。ただし、NULL の引数値の有無のテストは UDF が行う必要があります。

値 NULL CALL は、後方互換性またはファミリーの互換性のために、CALLED ON NULL INPUT の同義語として使うことができます。同様に、NOT NULL CALL は、RETURNS NULL ON NULL INPUT の同義語として使用できます。

NO EXTERNAL ACTION または EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るかどうかを指定します。外部アクションの例としては、メッセージの送信やファイルへのレコードの書き込みがあります。デフォルトは NO EXTERNAL ACTION です。

NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取らないことを指定します。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。

EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取ることを指定します。

CREATE FUNCTION (OLE DB 外部表)

CARDINALITY *integer*

この節はオプションで、関数によって戻されると予想される行数の見積もりを最適化のために指定します。 *integer* の値の有効範囲は、0 から 2 147 483 647 (両端の値を含む) です。

表関数に対して CARDINALITY 節の指定がない場合、DB2 はデフォルト値として有限の値を想定します (RUNSTATS ユーティリティーが統計を収集していない表に対して想定される値と同じ)。

警告: 関数が事実上無限のカーディナリティーを持っている (すなわち、呼び出されるといつでも行を戻し、"end-of-table" 条件を戻さない) 場合、"end-of-table" 条件を必要とする照会は無限に実行されるので、照会を中断させる必要があります。このような照会の例としては、GROUP BY 節や ORDER BY 節を含む照会があります。このような UDF を作成することは推奨されていません。

注

- FENCED、FINAL CALL、SCRATCHPAD、PARAMETER STYLE SQL、DISALLOW PARALLEL、NO DBINFO、NOT THREADSAFE、および NO SQL は、ステートメントでは暗黙的であり、指定できます。
- ユーザー定義関数のパラメーターのデータ・タイプを選択する場合は、入力値に影響を与えるプロモーションの規則を考慮してください。例えば、入力値として使用できる定数のデータ・タイプは、予期される以外の組み込みデータ・タイプである可能性があり、さらには、予期されるデータ・タイプにプロモートできない場合があります。プロモーションの規則に従って、一般にパラメーターには次のデータ・タイプを使用するようにしてください。
 - CHAR ではなく VARCHAR
 - GRAPHIC ではなく VARGRAPHIC
- プラットフォーム間での UDF の移植性を保つためには、以下のデータ・タイプの使用をお勧めします。
 - FLOAT ではなく DOUBLE または REAL
 - NUMERIC ではなく DECIMAL
 - LONG VARCHAR ではなく CLOB (または BLOB)
- まだ存在していないスキーマ名を用いて関数を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- **特権**

関数の定義者は、関数に対する WITH GRANT OPTION 付きの EXECUTE 特権と、関数をドロップする権利を常に与えられます。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - DETERMINISTIC の代わりに NOT VARIANT を指定できます。
 - NOT DETERMINISTIC の代わりに VARIANT を指定できます。
 - CALLED ON NULL INPUT の代わりに NULL CALL を指定できます。

- RETURNS NULL ON NULL INPUT の代わりに NOT NULL CALL を指定できます。

例

例 1: 次の例では、OLE DB 表関数を登録し、Microsoft Access データベースから受注情報を取り出します。外部名として接続ストリングが定義されています。

```
CREATE FUNCTION orders ()
  RETURNS TABLE (orderid INTEGER,
                  customerid CHAR(5),
                  employeeid INTEGER,
                  orderdate TIMESTAMP,
                  requireddate TIMESTAMP,
                  shippeddate TIMESTAMP,
                  shipvia INTEGER,
                  freight dec(19,4))

LANGUAGE OLEDB
EXTERNAL NAME '!orders!Provider=Microsoft.Jet.OLEDB.3.51;
              Data Source=c:\sql\lib\samples\%oledb%\nwind.mdb
!COLLATING_SEQUENCE=Y';
```

例 2: 次の例では、OLE DB 表関数を登録し、Oracle データベースから顧客情報を取り出します。接続ストリングは、サーバー定義で指定されています。外部名では、表名は完全に修飾されたものです。ローカル・ユーザーである john が、リモート・ユーザーの dave にマップされます。他のユーザーは、接続ストリングでゲスト・ユーザー ID を使用します。

```
CREATE SERVER spirit
  WRAPPER OLEDB
  OPTIONS (CONNECTSTRING 'Provider=MSDAORA;Persist Security Info=False;
                          User ID=guest;password=pwd;Locale Identifier=1033;
                          OLE DB Services=CLIENTCURSOR;Data Source=spirit');

CREATE USER MAPPING FOR john
  SERVER spirit
  OPTIONS (REMOTE_AUTHID 'dave', REMOTE_PASSWORD 'mypwd');

CREATE FUNCTION customers ()
  RETURNS TABLE (customer_id INTEGER,
                  name VARCHAR(20),
                  address VARCHAR(20),
                  city VARCHAR(20),
                  state VARCHAR(5),
                  zip_code INTEGER)

LANGUAGE OLEDB
EXTERNAL NAME 'spirit!demo.customer';
```

例 3: 次の例では、OLE DB 表関数を登録し、MS SQL Server 7.0 データベースから店舗についての情報を取り出します。外部名として接続ストリングが指定されています。表関数には、コマンド・テキストを OLE DB Provider に渡すための入力パラメーターがあります。外部名として rowset 名を指定する必要はありません。照会例では、SQL ステートメント・テキストを渡し、上位 3 店舗についての情報を取り出します。

```
CREATE FUNCTION favorites (varchar(600))
  RETURNS TABLE (store_id CHAR (4),
                  name VARCHAR (41),
                  sales INTEGER)

SPECIFIC favorites
LANGUAGE OLEDB
EXTERNAL NAME '!!Provider=SQLOLEDB.1;Persist Security Info=False;
```

CREATE FUNCTION (OLE DB 外部表)

```
User ID=sa;Initial Catalog=pubs;Data Source=WALTZ;  
Locale Identifier=1033;Use Procedure for Prepare=1;  
Auto Translate=False;Packet Size=4096;Workstation ID=WALTZ;  
OLE DB Services=CLIENTCURSOR;';
```

```
SELECT *  
FROM TABLE (favorites  
(' select top 3 sales.stor_id as store_id, ' CONCAT  
  ' stores.stor_name as name, ' CONCAT  
  ' sum(sales.qty) as sales ' CONCAT  
  ' from sales, stores ' CONCAT  
  ' where sales.stor_id = stores.stor_id ' CONCAT  
  ' group by sales.stor_id, stores.stor_name ' CONCAT  
  ' order by sum(sales.qty) desc ')) as f;
```

CREATE FUNCTION (ソース派生またはテンプレート)

CREATE FUNCTION (ソース派生またはテンプレート) ステートメントは、以下の目的で使用します。

- 他の既存のスカラー関数または列関数に基づくユーザー定義関数を、現行サーバーに登録する。
- フェデレーテッド・サーバーとして指定されたアプリケーション・サーバーに、関数テンプレートを登録する。関数テンプレートとは、実行可能コードを含まない部分関数のことです。ユーザーは、データ・ソース関数へマッピングする目的でこれを作成します。マッピングを作成したら、フェデレーテッド・サーバーへサブミットする照会に、その関数テンプレートを指定できます。そのような照会を処理する場合、フェデレーテッド・サーバーは、テンプレートのマップ先のデータ・ソース関数を呼び出し、値を戻します。この値のデータ・タイプは、テンプレートの定義の RETURNS 部分にある値に対応するものです。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する IMPLICIT_SCHEMA 権限 (関数の暗黙または明示のスキーマ名が存在しない場合)
- スキーマに対する CREATEIN 特権 (関数のスキーマ名が存在する場合)
- SYSADM または DBADM 権限

このステートメントの許可 ID が持つ特権に SYSADM または DBADM 権限が含まれておらず、かつ SOURCE 節が指定されている場合、ステートメントの許可 ID が保持する特権にはソース関数に対する EXECUTE 特権も含まれている必要があります。

CREATE FUNCTION (ソース派生またはテンプレート)

構文

▶ CREATE FUNCTION *function-name* (*parameter-declaration*) ●

▶ RETURNS *data-type2* ● [SPECIFIC *specific-name*] ● SOURCE

▶ *function-name* [SPECIFIC *specific-name*] *function-name* (

[*data-type*]) [PARAMETER CCSID [ASCII] [UNICODE]] AS TEMPLATE ●

[NOT DETERMINISTIC] [DETERMINISTIC] ● [EXTERNAL ACTION] [NO EXTERNAL ACTION] ●

parameter-declaration:

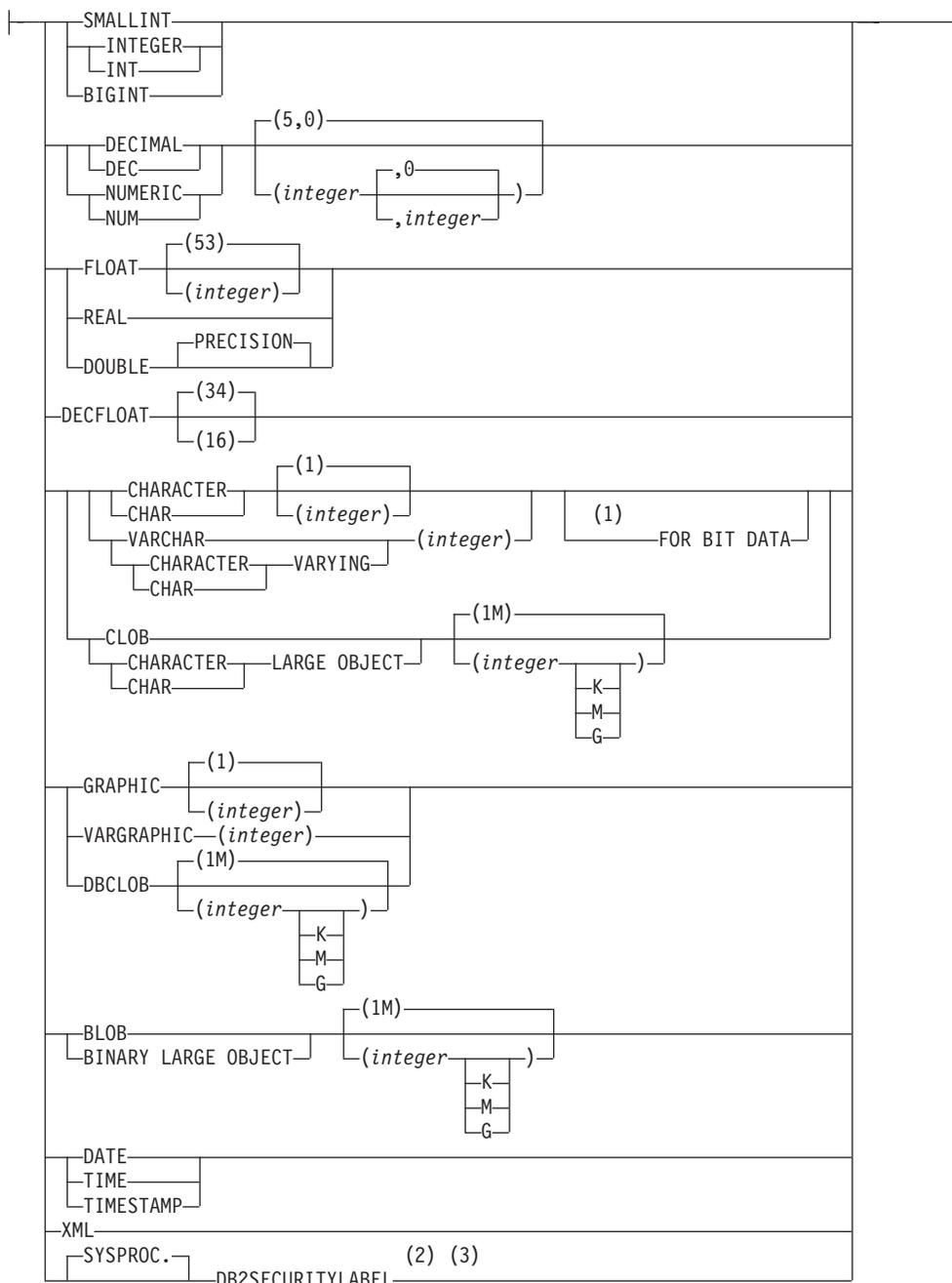
[*parameter-name*] *data-type1*

data-type1、data-type2:

[built-in-type] [*distinct-type-name*] [*structured-type-name*]

built-in-type:

CREATE FUNCTION (ソース派生またはテンプレート)



注:

- FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE 42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。

説明

function-name

定義する関数または関数テンプレートを指定します。これは、関数を指定する修飾または非修飾の名前です。 *function-name* (関数名) の非修飾形式は SQL ID です (最大長 18)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

暗黙または明示の修飾子を含む名前、およびパラメーターの数と各パラメーターのデータ・タイプ (データ・タイプの長さ、精度、または位取りの各属性には関係なく) は、カタログに記述されている関数または関数テンプレートを指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数およびデータ・タイプとの組み合わせは、そのスキーマ内では当然ユニークですが、複数のスキーマ間でユニークである必要はありません。

2 つの部分からなる名前を指定する場合、‘SYS’ で始まる *schema-name* (スキーマ名) は使用できません (SQLSTATE 42939)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*function-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

ユーザー定義特殊タイプと同じ機能をサポートする目的で、既存の関数に基づくユーザー定義関数に名前を付ける場合は、元の関数と同じ名前を使用することができます。これにより、ユーザーは、追加定義の必要性を特に意識することなく、同じ関数のユーザー定義特殊タイプ版を使用することができます。一般に、関数のシグニチャーに何らかの差異がある場合には、同じ名前を複数の関数に使用することができます。

(parameter-declaration,...)

関数または関数テンプレートの入力パラメーターの数を指定するとともに、各パラメーターのデータ・タイプを指定します。このリストには、関数または関数テンプレートが受け取ることを予期している各パラメーターごとに、1 つの項目を指定する必要があります。パラメーターの数は 90 を超えることはできません (SQLSTATE 54023)。

パラメーターのない関数も登録可能です。この場合、指定するデータ・タイプがない場合でも、括弧はコーディングする必要があります。以下に例を示します。

```
CREATE FUNCTION WOOFER() ...
```

対応するすべてのパラメーターで、1 つのスキーマ内で名前が同じ 2 つの関数が、まったく同じタイプを持つことはできません。この制限は、同じ名前を持つ同じスキーマ内の関数および関数テンプレートにも適用されます。このタイプの比較では長さ、精度、および位取りは考慮されません。したがって、CHAR(8) と CHAR(35)、また DECIMAL(11,2) と DECIMAL (4,3) は、それぞれ同じタイプと見なされます。Unicode データベースの場合には、CHAR(13) と GRAPHIC(8) は、それぞれ同じタイプと見なされます。さらに、DECIMAL と

CREATE FUNCTION (ソース派生またはテンプレート)

NUMERIC などのように、この目的で複数のタイプが同じタイプとして扱われることがあります。シグニチャーが重複していると、エラー (SQLSTATE 42723) を戻します。

parameter-name

入力パラメーターにオプションの名前を指定します。この名前は、パラメーター・リスト内の他のすべての *parameter-name* と同じにすることはできません (SQLSTATE 42734)。

data-type1

入力パラメーターのデータ・タイプを指定します。データ・タイプは、組み込みデータ・タイプ、特殊タイプ、または構造化タイプにすることができます。

SOURCE 節で指定された関数の対応するパラメーターのタイプにキャスト可能な場合は、任意の有効な SQL データ・タイプを使用できます (詳細は「データ・タイプ間のキャスト」を参照)。ただし、この検査によって関数の呼び出し時にエラーが発生しないことが保証されるわけではありません。

各組み込みデータ・タイプの詳細な説明は、『CREATE TABLE』を参照してください。

- 日時タイプのパラメーターは文字データ・タイプとして受け渡され、そのデータは ISO 形式で受け渡されます。
- 配列タイプを指定することはできません (SQLSTATE 42879)。
- REF(*type-name*) で指定された参照タイプを指定することはできません (SQLSTATE 42879)。

ユーザー定義特殊タイプの場合、パラメーターの長さ、精度、または桁数の属性は、特殊タイプのソース・タイプのもの (CREATE TYPE で指定されたもの) になります。特殊タイプのパラメーターは、特殊タイプのソース・タイプとして受け渡されます。特殊タイプの名前が修飾されていない場合は、データベース・マネージャーによって SQL パス内のスキーマが検索され、スキーマ名が解決されます。

ユーザー定義構造化タイプの場合、適切なトランスフォーム関数に関連するトランスフォーム・グループに存在する必要があります。

関数がソース関数から派生するので、パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません (指定は可能です)。その代わりに、CHAR() のように空の括弧を使用できます。特定の長さ、位取り、または精度を指定して定義可能なデータ・タイプのことを、パラメーター化データ・タイプといいます。パラメーター化データ・タイプは、ストリング・データ・タイプと 10 進データ・タイプです。

関数テンプレートを使用すると、パラメーター化データ・タイプに長さ、精度、または位取りを指定する代わりに、空の括弧も使用できます。パラメーター化データ・タイプには空の括弧を使用することが推奨されています。空の括弧を使用する場合、長さ、精度、または位取りはリモート関数のものと同じです。それらは関数マッピング作成時に関数テンプレートがリモート関数にマップされるときに決定されます。括弧をすべて省略した場合は、データ・タイプのデフォルト長が使用されます (「CREATE TABLE」を参照)。

RETURNS

この節は必須で、関数または関数テンプレートの出力を指定します。

data-type2

出力のデータ・タイプを指定します。

ソース・スカラー関数では、ソース関数の結果のタイプからキャスト可能であれば、任意の有効な SQL データ・タイプ (特殊タイプも同様) を指定できます。配列タイプをパラメーターのデータ・タイプとして指定することはできません (SQLSTATE 42879)。

上記のような、ソース関数のパラメーターの場合、パラメーター化タイプのパラメーターを指定する必要はありません。その代わりに、VARCHAR() のように、空の括弧を使用できます。

関数が他の関数に基づいている場合に RETURNS 節のデータ・タイプの指定に適用される考慮事項と規則については、このステートメントの『規則』セクションを参照してください。

関数テンプレートを使用する場合は、空の括弧を使用できません (SQLSTATE 42611)。パラメーター化データ・タイプには、長さ、精度、または位取りを指定する必要があります。リモート関数と同じ長さ、精度、または位取りを指定することが推奨されています。

SPECIFIC *specific-name*

定義する関数のインスタンスに対するユニーク名を指定します。この特定名は、この関数をソース関数として使用する場合、この関数をドロップする場合、またはこの関数にコメントを付ける場合に使用することができます。これは、関数の呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 18)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別の関数インスタンスを指定するものであってはなりません。そうでない場合、エラー (SQLSTATE 42710) が戻されます。

specific-name は、既存の *function-name* (関数名) と同じでも構いません。

修飾子を指定しない場合、*function-name* に使用された修飾子が使用されます。修飾子を指定する場合は、*function-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラー (SQLSTATE 42882) が戻されます。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成されるユニーク名は、SQL の後に文字のタイム・スタンプが続く名前です (SQLyymmddhhmmssxxx)。

SOURCE

作成する関数が、データベース・マネージャーにとって既知の他の関数 (ソース関数) によってインプリメントされる関数であることを指定します。ソース関数は、組み込み関数 (COALESCE、DATAPARTITIONNUM、DBPARTITIONNUM、GREATEST、HASHEDVALUE、LEAST、MAX (スカラー)、MIN (スカラー)、NULLIF、NVL、RAISE_ERROR、RID、RID_BIT、TYPE_ID、TYPE_NAME、TYPE_SCHEMA、または VALUE を除く) か、以前に作成したユーザー定義のスカラー関数のいずれかになります。

CREATE FUNCTION (ソース派生またはテンプレート)

SOURCE 節は、スカラー関数または列関数に対してのみ指定が可能で、表関数には指定できません。

SOURCE 節は、このような他の関数の識別情報を示します。

function-name

ソースとして使用される特定の関数を指定します。ステートメントの許可 ID が EXECUTE 特権を持ち、この *function-name* を名前とする特定の関数が、スキーマに実際に 1 つだけ存在している場合にのみ有効です。この構文変形は、組み込み関数であるソース関数に対しては無効です。

非修飾名を指定すると、現行 SQL パス (CURRENT PATH 特殊レジスタの値) がその関数を見つけるのに使用されます。EXECUTE 特権のあるステートメントの許可 ID の名前を持つ関数が含まれている、SQL パスの最初のスキーマが選択されます。

指定されたスキーマにこの名前の関数が見つからないか、あるいは名前が修飾されておらず、この名前の関数が SQL パスにない場合は、エラー (SQLSTATE 42704) が戻されます。指定したスキーマまたは見つかったスキーマに、この関数の許可された特定インスタンスが複数個ある場合には、エラー (SQLSTATE 42725) が戻されます。その名前の関数が存在し、ステートメントの許可 ID が関数に対して EXECUTE 特権を持っていない場合には、エラー (SQLSTATE 42501) が戻されます。

SPECIFIC *specific-name*

関数の作成時に指定されたか、またはデフォルト値として使用された *specific-name* (特定名) を使用して、ソースとして使用する特定のユーザー定義関数を指定します。この構文変形は、組み込み関数であるソース関数に対しては無効です。

非修飾名を指定すると、現行 SQL パスがその関数を見つけるのに使用されます。EXECUTE 特権のあるステートメントの許可 ID の特定の名前を持つ関数が含まれている、SQL パスの最初のスキーマが選択されます。

指定されたスキーマにこの *specific-name* の関数が見つからないか、あるいは名前が修飾されておらず、この *specific-name* の関数が SQL パスにない場合は、エラー (SQLSTATE 42704) が戻されます。*specific-name* の関数が存在し、ステートメントの許可 ID が関数に対して EXECUTE 特権を持っていない場合には、エラー (SQLSTATE 42501) が出されます。

function-name (data-type,...)

ソース関数を固有に指定する関数シグニチャーを指定します。組み込み関数であるソース関数に対しては、これが唯一有効な構文変形です。

同じ関数名と SOURCE 節に指定されたデータ・タイプを持つ複数の関数の中から 1 つの関数を選択するために、関数解決の規則が適用されます。ただし、選択された関数の各パラメーターのデータ・タイプは、ソース関数に指定された対応するデータ・タイプと、まったく同じタイプでなければなりません。

function-name

ソース関数の関数名を指定します。非修飾名を指定すると、ユーザーの SQL パスのスキーマが考慮されます。

CREATE FUNCTION (ソース派生またはテンプレート)

data-type

これは、CREATE FUNCTION ステートメントで対応する位置 (コンマで区切られた) に指定されたデータ・タイプに一致していなければなりません。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。代わりに、空の括弧をコーディングすることによって、データ・タイプの一致を調べる際にそれらの属性を無視するように指定することができます。例えば、DECIMAL() は、データ・タイプが DECIMAL(7,2) として定義されているパラメーターと一致します。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE FUNCTION ステートメントにおける指定に完全に一致していなければなりません。これは、意図したとおりの関数が確実に使用されるようにする場合に便利です。また、データ・タイプの同義語は一致と見なされることにも注意してください (例えば、DEC と NUMERIC は一致します)。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(n) のタイプは、n に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つ関数がない場合は、エラー (SQLSTATE 42883) が戻されます。

PARAMETER CCSID

関数とやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。関数が呼び出されるときのアプリケーション・コード・ページはデータベース・コード・ページです。

UNICODE

ストリング・データが Unicode でエンコードされることを指定します。データベースが Unicode データベースの場合、文字データは UTF-8、GRAPHIC データは UCS-2 になります。データベースが Unicode データベースでない場合は、文字データは UTF-8 になります。いずれの場合も、関数が呼び出されるときのアプリケーション・コード・ページは 1208 です。

PARAMETER CCSID 節には、ソース関数と同じコード化スキームを指定することが必要です (SQLSTATE 53090)。

CREATE FUNCTION (ソース派生またはテンプレート)

AS TEMPLATE

このステートメントが、実行可能コードを含む関数ではなく、関数テンプレートを作成するために使われることを示しています。

NOT DETERMINISTIC または DETERMINISTIC

関数が同一の入力引数に同じ結果を戻すかどうかを指定します。デフォルトは NOT DETERMINISTIC です。

NOT DETERMINISTIC

同じ入力引数を指定して呼び出しても、関数が同じ結果を戻さない場合があることを指定します。関数は、結果に影響する状態値によって左右されます。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。deterministic ではない (常には同じ結果が戻されない) 関数の例としては、乱数を生成する関数があります。

deterministic ではない関数は、並列タスクによって実行されると、不正確な結果を受け取る場合があります。

DETERMINISTIC

同じ入力引数を指定して呼び出すと、関数から常に同じ結果が戻されることを指定します。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。deterministic な (常に同じ結果を戻す) 関数の例としては、入力引数の平方根を計算する関数があります。

EXTERNAL ACTION または NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るかどうかを指定します。外部アクションの例としては、メッセージの送信やファイルへのレコードの書き込みがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取ることを指定します。SQL ルーチン本体が EXTERNAL ACTION を指定して定義されている関数を呼び出す場合は、EXTERNAL ACTION を暗黙的または明示的に指定することが必要です (SQLSTATE 428C2)。

外部アクションが指定された関数は、関数が並列タスクによって実行されると、不正確な結果を戻す場合があります。例えば、初期呼び出しを受けるときに注釈を送信する関数の場合、関数ごとに 1 つの注釈が送信されるのではなく、並列タスクごとに 1 回ずつ送信されることになります。

NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取らないことを指定します。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。

規則

- 便宜上、この項では作成する関数を CF と呼び、SOURCE 節で指定する関数を SF と呼びます (許される 3 つの構文のどれが SF の指定に使用されたかは関係ありません)。
 - CF と SF のそれぞれの非修飾名が異なる名前であっても構いません。
 - 他の関数のソースとして指定された関数自体が、別の関数をソースとして使用した関数であっても構いません。間接的に呼び出された関数がエラーを戻すと、アプリケーションをデバッグすることが極めて難しくなるので、この機能を使用する場合には細心の注意を払う必要があります。
 - 以下の節は、SOURCE 節と共に指定した場合には無効になります (CF はこれらの属性を SF から継承するからです)。
 - CAST FROM ...
 - EXTERNAL ...
 - LANGUAGE ...
 - PARAMETER STYLE ...
 - DETERMINISTIC / NOT DETERMINISTIC
 - FENCED / NOT FENCED
 - RETURNS NULL ON NULL INPUT / CALLED ON NULL INPUT
 - EXTERNAL ACTION / NO EXTERNAL ACTION
 - NO SQL / CONTAINS SQL / READS SQL DATA
 - SCRATCHPAD / NO SCRATCHPAD
 - FINAL CALL / NO FINAL CALL
 - RETURNS TABLE (...)
 - CARDINALITY ...
 - ALLOW PARALLEL / DISALLOW PARALLEL
 - DBINFO / NO DBINFO
 - THREADSAFE / NOT THREADSAFE
 - INHERIT SPECIAL REGISTERS

これらの規則に違反すると、エラー (SQLSTATE 42613) になります。

- CF の入力パラメーターの数は、SF のパラメーターの数と同じでなければなりません。異なる場合には、エラー (SQLSTATE 42624) が戻されます。
- 次の場合には、CF にパラメーター化データ・タイプの長さ、精度、または位取りを指定する必要がありません。
 - 関数の入力パラメーター
 - その RETURNS パラメーター

代わりに、VARCHAR() のように、データ・タイプの一部として空の括弧を使用することにより、長さ、精度、および位取りがソース関数と同じであるか、あるいはキャストによって決定されるように指定することができます。

ただし、長さ、精度、または位取りを指定した場合には、CF における値と SF の対応する値とが比較検査されます。これについては、以下で入力パラメーターと戻り値とに分けて説明します。

CREATE FUNCTION (ソース派生またはテンプレート)

- CF の入力パラメーターの指定は、SF の入力パラメーターの指定と比較検査されます。CF の各パラメーターのデータ・タイプは、SF の対応するパラメーターのデータ・タイプと同じであるか、あるいはキャスト可能でなければなりません。同じタイプでないか、あるいはキャスト可能ではないパラメーターがある場合には、エラー (SQLSTATE 42879) が戻されます。

この規則は、CF の使用時に発生し得るエラーに対して何らかの保証を与えるものではありません。CF パラメーターのデータ・タイプや長さ、または精度属性に一致する引数は、対応する SF パラメーターの方が長さが短かったり精度が劣る場合には、割り当てることができません。一般に、CF のパラメーターの長さ属性や精度属性は、対応する SF パラメーターのそれよりも大きくしてはなりません。

- CF の RETURNS データ・タイプの指定は、SF のそれと比較検査されます。キャスト後の SF の最終の RETURNS データ・タイプは、CF の RETURNS のデータ・タイプと同じか、あるいはそれにキャスト可能でなければなりません。そうでない場合、エラー (SQLSTATE 42866) が戻されます。

この規則は、CF の使用時に発生し得るエラーに対して何らかの保証を与えるものではありません。SF の RETURNS データ・タイプのデータ・タイプや長さもしくは精度属性に一致する結果の値は、CF の RETURNS データ・タイプの方が長さが短かったり精度が劣る場合には、割り当てることができません。長さもしくは精度属性が SF の RETURNS データ・タイプのそれよりも小さい CF の RETURNS データ・タイプを指定することにした場合は、十分に注意を払ってください。

注

- あるデータ・タイプが他のデータ・タイプにキャスト可能かどうかの判別では、CHAR や DECIMAL などのパラメーター化データ・タイプの長さまたは精度と位取りは考慮されません。したがって、ソース・データ・タイプの値をターゲット・データ・タイプの値にキャストしようとする、関数の使用時にエラーになる可能性があります。例えば、VARCHAR は DATE にキャストできますが、実際にはソース・タイプが VARCHAR(5) と定義されている場合には、関数の使用時にエラーになります。
- ユーザー定義関数のパラメーターのデータ・タイプを選択する場合は、入力値に影響を与えるプロモーションの規則を考慮してください (『データ・タイプのプロモーション』を参照してください)。例えば、入力値として使用できる定数のデータ・タイプは、予期される以外の組み込みデータ・タイプである可能性があります。さらには、予期されるデータ・タイプにプロモートできない場合があります。プロモーションの規則に従って、一般にパラメーターには次のデータ・タイプを使用するようにしてください。
 - SMALLINT ではなく INTEGER
 - REAL ではなく DOUBLE
 - CHAR ではなく VARCHAR
 - GRAPHIC ではなく VARGRAPHIC
- まだ存在していないスキーマ名を用いて関数を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙に作

CREATE FUNCTION (ソース派生またはテンプレート)

成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。

- データ・ソース関数を認識するフェデレーテッド・サーバーの場合、この関数はフェデレーテッド・データベースにあるもう一方の関数にマップする必要があります。データベースに対となる関数がない場合、ユーザーがそれを作成してマッピングしなければなりません。

対となるもう一方は、関数 (スカラーまたはソース) か、関数テンプレートとすることができます。ユーザーが関数を作成して必要なマッピングを行うと、その関数を指定する照会を処理するたびに、DB2 は、(1) その関数を呼び出すときの戦略と、データ・ソース関数を呼び出すときの戦略を比べ、(2) オーバーヘッドが少ないと思われる関数を呼び出します。

ユーザーが関数テンプレートとマッピングを作成すると、そのテンプレートを指定する照会を処理するたびに、DB2 は、マッピング先のデータ・ソース関数を呼び出します (ただし、この関数を呼び出すためのアクセス・プランが存在する場合)。

• 特権

関数の定義者は、関数に対する EXECUTE 特権と、関数をドロップする権利を常に与えられます。以下のいずれかが真の場合には、関数の定義者には WITH GRANT OPTION も与えられます。

- ソース関数が、組み込み関数である。
- 関数の定義者が、ソース関数に対して WITH GRANT OPTION 付きの EXECUTE 特権を持っている。
- 関数が、テンプレートである。

例

例 1: Pellow がオリジナルの CENTRE 外部スカラー関数を作成した後に、別のユーザーがその関数に基づいて関数を作成します。この関数は、整数引数のみを受け入れるという点だけが異なります。

```
CREATE FUNCTION MYCENTRE (INTEGER, INTEGER)
  RETURNS FLOAT
  SOURCE PELLOW.CENTRE (INTEGER, FLOAT)
```

例 2: 組み込みの INTEGER データ・タイプに基づく特殊タイプ HATSIZE が作成済みです。それぞれの部門の平均の hat size を計算するには、AVG 関数を使用すると便利です。これは、以下のようにして簡単に実行できます。

```
CREATE FUNCTION AVG (HATSIZE) RETURNS HATSIZE
  SOURCE SYSIBM.AVG (INTEGER)
```

特殊タイプの作成により必要な cast 関数が生成され、引数の場合は HATSIZE から INTEGER に、関数の結果の場合は INTEGER から HATSIZE にキャストすることが可能です。

例 3: フェデレーテッド・システムで、表統計を倍精度浮動小数点付きの値で戻す Oracle UDF を起動します。フェデレーテッド・サーバーは、この関数とフェデレーテッド・データベース側の対となる関数との間でマッピングが行われる場合にだけ、この関数を認識することができます。ところが、そのような対となる関数は存

CREATE FUNCTION (ソース派生またはテンプレート)

在していません。それで、対となる関数を関数テンプレートの形で指定して、このテンプレートを NOVA というスキーマに割り当ててることを決定します。次のコードを使用して、テンプレートをフェデレーテッド・サーバーに登録します。

```
CREATE FUNCTION NOVA.STATS (DOUBLE, DOUBLE)
  RETURNS DOUBLE
  AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

例 4: フェデレーテッド・システムで、特定の組織の従業員に対して支給されるボーナスの総額 (ドル) を戻す Oracle UDF を呼び出します。フェデレーテッド・サーバーは、この関数とフェデレーテッド・データベース側の対となる関数との間でマッピングが行われる場合にだけ、この関数を認識することができます。ところがそのような対となる関数は存在しないため、ユーザーが関数テンプレートの形で作成します。次のコードを使用して、このテンプレートをフェデレーテッド・サーバーに登録します。

```
CREATE FUNCTION BONUS ()
  RETURNS DECIMAL (8,2)
  AS TEMPLATE DETERMINISTIC NO EXTERNAL ACTION
```

CREATE FUNCTION (SQL スカラー、表、または行)

CREATE FUNCTION (SQL スカラー、表、または行) ステートメントは、ユーザー定義の SQL スカラー、表、または行関数を定義するのに使用されます。スカラー関数は、呼び出されるたびに 1 つの値を返し、SQL 式が有効な個所であればどこでも有効です。表関数は、FROM 節で使用でき、表を返します。行関数は、トランスフォーム関数として使用でき、行を返します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 全選択に指定された表、ビュー、またはニックネームのそれぞれに対して、
 - その表、ビュー、またはニックネームに対する CONTROL 特権、または
 - その表、ビュー、またはニックネームに対する SELECT 特権および以下の少なくとも 1 つ
 - データベースに対する IMPLICIT_SCHEMA 権限 (関数の暗黙または明示のスキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (関数のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

PUBLIC 以外のグループ特権は、CREATE FUNCTION ステートメントで指定された表やビューに対しては考慮されません。

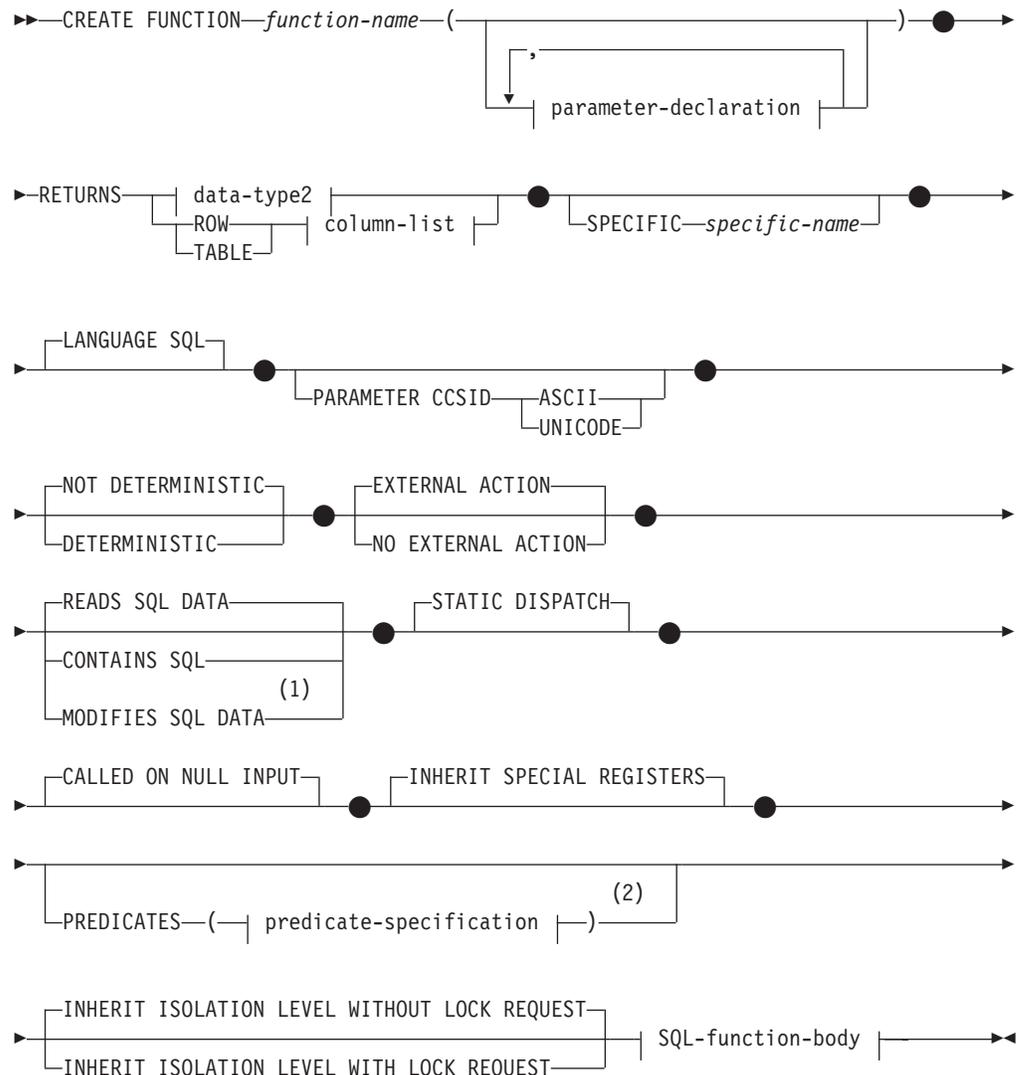
CREATE FUNCTION (SQL スカラー、表、または行)

このニックネームで示されている表またはビューのデータ・ソースの許可要件は、関数が呼び出される時に適用されます。接続の許可 ID は、別のリモート許可 ID へマップできます。

関数の定義者が SYSADM 権限を持つために、関数の作成しかできない場合、関数作成のため、その定義者には暗黙的に DBADM 権限が付与されます。

このステートメントの許可 ID に SYSADM 権限または DBADM 権限がない場合には、ステートメントの許可 ID が持つ特権に、関数本体で指定される SQL ステートメントを呼び出すのに必要なすべての特権が含まれていることも必要です。

構文



parameter-declaration:

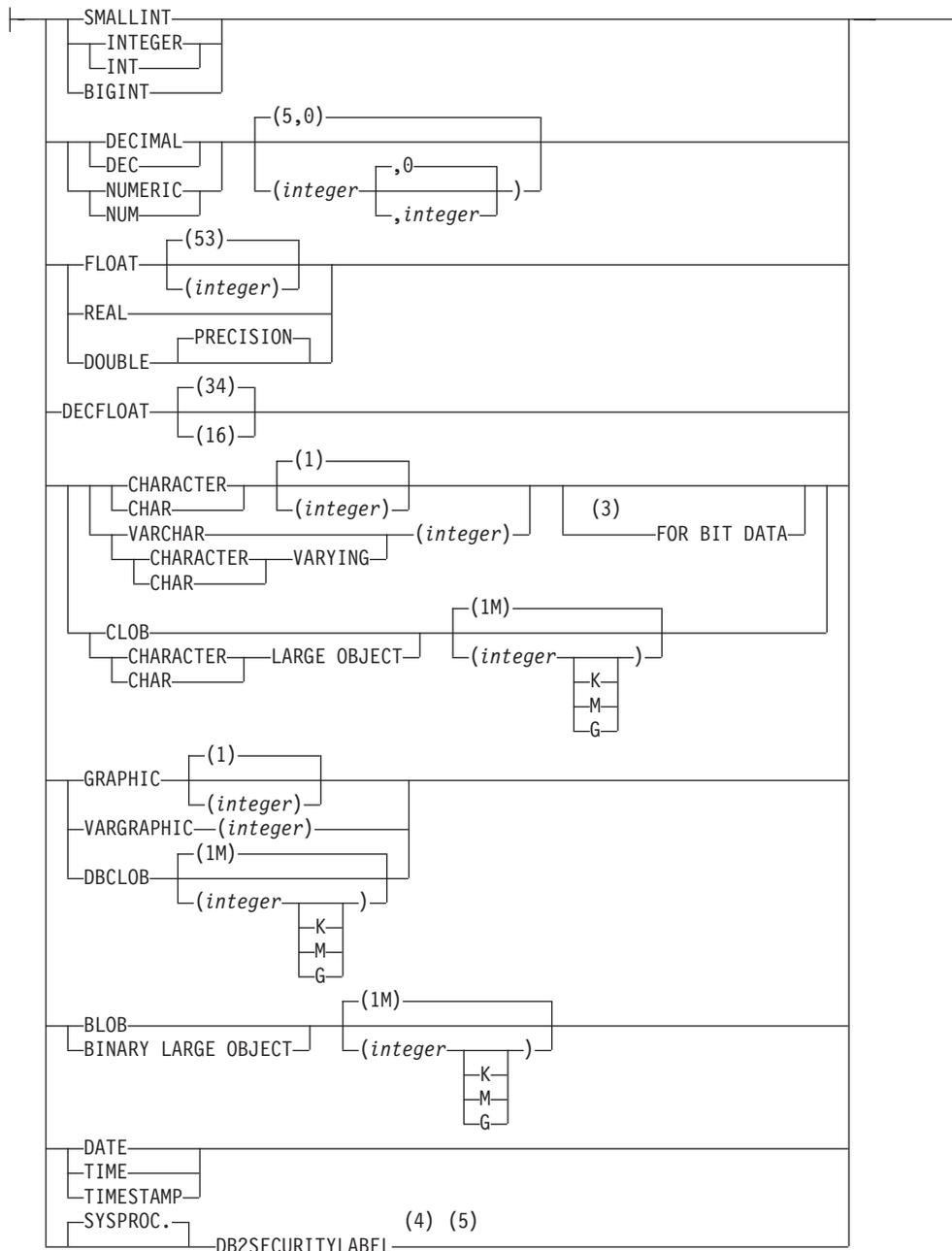
| *parameter-name* | data-type1 |

CREATE FUNCTION (SQL スカラー、表、または行)

data-type1、data-type2、data-type3:

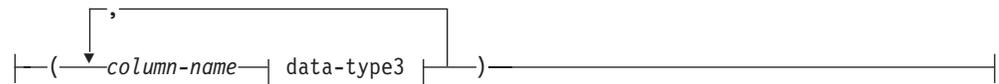


built-in-type:



CREATE FUNCTION (SQL スカラー、表、または行)

column-list:



SQL-function-body:



注:

- 1 RETURNS が表 (TABLE *column-list*) を指定している場合にのみ有効
- 2 RETURNS がスカラー結果 (*data-type2*) を指定している場合にのみ有効です。
- 3 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- 4 DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- 5 タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE 42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。

説明

function-name

定義する関数の名前を指定します。これは、関数を指定する修飾または非修飾の名前です。 *function-name* (関数名) の非修飾形式は SQL ID です (最大長 18)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

暗黙または明示の修飾子を含む名前、およびパラメーターの数と各パラメーターのデータ・タイプ (データ・タイプの長さ、精度、または位取りの各属性には関係なく) は、カタログに記述されている関数を指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数およびデータ・タイプとの組み合わせは、そのスキーマ内では当然ユニークですが、複数のスキーマ間でユニークである必要はありません。

2 つの部分からなる名前を指定する場合、'SYS' で始まる *schema-name* (スキーマ名) は使用できません (SQLSTATE 42939)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*function-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

CREATE FUNCTION (SQL スカラー、表、または行)

関数のシグニチャーに何らかの差異があれば、同じ名前を複数の関数に使用することができます。禁止されてはいませんが、外部ユーザー定義表関数の名前として、組み込み関数と同じ名前を指定すべきではありません。

(*parameter-declaration*,...)

関数の入力パラメーターの数を指定するとともに、各パラメーターのデータ・タイプを指定します。このリストには、関数が受け取ることを予期している各パラメーターごとに 1 つの項目を指定する必要があります。パラメーターの数は 90 を超えることはできません (SQLSTATE 54023)。

パラメーターのない関数も登録可能です。この場合、指定するデータ・タイプがない場合でも、括弧はコーディングする必要があります。以下に例を示します。

```
CREATE FUNCTION WOOFER() ...
```

対応するすべてのパラメーターで、1 つのスキーマ内で名前が同じ 2 つの関数が、まったく同じタイプを持つことはできません。このタイプの比較では長さ、精度、および位取りは考慮されません。したがって、CHAR(8) と CHAR(35)、DECIMAL(11,2) と DECIMAL (4,3)、および DECFLOAT(16) と DECFLOAT(34) は、それぞれ同じタイプと見なされます。Unicode データベースの場合には、CHAR(13) と GRAPHIC(8) は、それぞれ同じタイプと見なされます。さらに、DECIMAL と NUMERIC などのように、この目的で複数のタイプが同じタイプとして扱われることがあります。シグニチャーが重複していると、エラー (SQLSTATE 42723) を戻します。

parameter-name

入力パラメーターの名前を指定します。この名前は、パラメーター・リスト内の他のすべての *parameter-name* と同じにすることはできません (SQLSTATE 42734)。

data-type1

入力パラメーターのデータ・タイプを指定します。データ・タイプは、組み込みデータ・タイプ、特殊タイプ、構造化タイプ、または参照タイプにすることができます。

各組み込みデータ・タイプの詳細な説明は、『CREATE TABLE』を参照してください。

- 配列または XML データ・タイプは指定できません (SQLSTATE 42815)。
- 日時タイプのパラメーターは文字データ・タイプとして受け渡され、そのデータは ISO 形式で受け渡されます。

ユーザー定義特殊タイプの場合、パラメーターの長さ、精度、または桁数の属性は、特殊タイプのソース・タイプのもの (CREATE TYPE で指定されたもの) になります。特殊タイプのパラメーターは、特殊タイプのソース・タイプとして受け渡されます。特殊タイプの名前が修飾されていない場合は、データベース・マネージャーによって SQL パス内のスキーマが検索され、スキーマ名が解決されます。

参照タイプの場合、パラメーターが有効範囲を指定されていない場合は、パラメーターを REF(*type-name*) で指定できます。システムがパラメーターまたは結果の有効範囲を推測することはしません。関数本体では、最初に参照タイプをキャストして有効範囲を指定してからでなければ、間接参照操作は

CREATE FUNCTION (SQL スカラー、表、または行)

使用できません。同様に、SQL 関数により戻される参照も、最初にキャストしてこれに有効範囲を指定してからでなければ間接参照操作は使用できません。

RETURNS

これは必須の節であり、関数の出力のタイプを指定します。

data-type2

出力のデータ・タイプを指定します。

このステートメントでは、前述の関数パラメーター *data-type1* で説明した SQL 関数のパラメーターと同じ考慮事項が適用されます。

ROW *column-list*

関数の出力が単一の行であることを指定します。関数が複数の行を出力する場合、エラーになります (SQLSTATE 21505)。 *column-list* には、少なくとも 2 つの列を組み込まなければなりません (SQLSTATE 428F0)。

行関数は、構造化タイプの変換関数としてのみ使用できます (1 つの構造化タイプをパラメーターとして使用し、基本タイプのみ戻します)。

TABLE *column-list*

関数の出力が表であることを指定します。

column-list

ROW または TABLE 関数で戻される列名およびデータ・タイプのリスト。

column-name

この列の名前を指定します。名前を修飾することはできず、行の複数の列に対して同じ名前を使用することはできません。

data-type3

列のデータ・タイプを指定します。SQL 関数のパラメーターによりサポートされていれば、どのデータ・タイプでも構いません。

SPECIFIC *specific-name*

定義する関数のインスタンスに対するユニーク名を指定します。この特定名は、この関数をソース関数として使用する場合、この関数をドロップする場合、またはこの関数にコメントを付ける場合に使用することができます。これは、関数の呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 18)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別の関数インスタンスを識別するものであってはなりません。そうでない場合は、エラーになります (SQLSTATE 42710)。

specific-name は、既存の *function-name* (関数名) と同じでも構いません。

修飾子を指定しない場合、*function-name* に使用された修飾子が使用されます。修飾子を指定する場合は、*function-name* の明示または暗黙の修飾子と同じでなければなりません。そうでない場合は、エラーになります (SQLSTATE 42882)。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成されるユニーク名は、SQL の後に文字のタイム・スタンプが続く名前です (SQLyymmddhhmmssxxx)。

CREATE FUNCTION (SQL スカラー、表、または行)

LANGUAGE SQL

関数が SQL を使用して書かれていることを指定します。

PARAMETER CCSID

関数とやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。

UNICODE

文字データは UTF-8 で記述され、GRAPHIC データは UCS-2 で記述されることを指定します。データベースが Unicode データベースでない場合は、PARAMETER CCSID UNICODE は指定できません (SQLSTATE 56031)。

DETERMINISTIC または NOT DETERMINISTIC

この節は任意指定で、特定の引数の値に対して関数が常に同じ結果を戻すか (DETERMINISTIC)、それとも状態値に依存して関数の結果が影響を受けるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC 関数は、同一の入力で連続で呼び出しが行われたとき、常に同じ表を返します。NOT DETERMINISTIC を指定すると、同じ入力によって常に同じ結果が生じる利点に基づく最適化ができなくなります。

EXTERNAL ACTION または NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るかどうかを指定します。外部アクションの例としては、メッセージの送信やファイルへのレコードの書き込みがあります。デフォルトは EXTERNAL ACTION です。

EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取ることを指定します。

NO EXTERNAL ACTION

関数が、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取らないことを指定します。データベース・マネージャーは、SQL ステートメントの最適化中に、この情報を使用します。

CONTAINS SQL、READS SQL DATA、または MODIFIES SQL DATA

どのタイプの SQL ステートメントを実行できるかを指示します。

CONTAINS SQL

SQL データの読み取りも変更もしない SQL ステートメントを、関数により実行できるように指示します (SQLSTATE 42985)。

CREATE FUNCTION (SQL スカラー、表、または行)

READS SQL DATA

SQL データを変更しない SQL ステートメントを、関数により実行できるように指示します (SQLSTATE 42985)。

MODIFIES SQL DATA

dynamic-compound-statement でサポートされているすべての SQL ステートメントが関数によって実行可能であることを示します。

STATIC DISPATCH

このオプション節は、関数解決時に DB2 が関数のパラメーターの静的タイプ (宣言済みタイプ) に基づいて関数を選択するよう指示します。

CALLED ON NULL INPUT

この節は、なんらかの引数が NULL 値であるかどうかにかかわらず、関数が呼び出されることを指定します。これは、NULL 値を戻す場合も、NULL 以外の値を戻す場合もあります。NULL の引数値の有無のテストはユーザー定義関数が行う必要があります。

CALLED ON NULL INPUT の代わりに NULL CALL という句を使用できません。

INHERIT SPECIAL REGISTERS

このオプション節は、関数の更新可能な特殊レジスターが、呼び出しステートメントの環境からの初期値を継承するよう指示します。カーソルの選択ステートメントに呼び出された関数の場合、初期値はカーソルがオープンした際の環境から継承します。ネストされたオブジェクト (例えば、トリガーまたはビュー) に呼び出されるルーチンの場合、初期値はランタイム環境 (オブジェクト定義ではない) から継承します。

特殊レジスターに対する変更が、関数の呼び出し元に戻されることはありません。

一部の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、呼び出し元からの継承は行われません。

PREDICATES

この関数を使用する述部の場合、この節は索引拡張を活用できることを示し、オプションの SELECTIVITY 節を使用して述部の検索条件を指定できます。

PREDICATES 節が指定された場合、関数は NO EXTERNAL ACTION を指定した DETERMINISTIC として定義しなければなりません (SQLSTATE 42613)。

PREDICATES 節が指定されており、データベースが Unicode データベースでない場合は、PARAMETER CCSID UNICODE は指定できません (SQLSTATE 42613)。

predicate-specification

述部指定に関する詳細は、『CREATE FUNCTION (外部スカラー)』を参照してください。

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST または INHERIT ISOLATION LEVEL WITH LOCK REQUEST

関数が呼び出し元のステートメントの分離レベルを継承している場合に、ロック要求をステートメントの分離レベルに関連付けることができるかどうかを指定します。デフォルトは INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST です。

CREATE FUNCTION (SQL スカラー、表、または行)

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST

関数が呼び出し元のステートメントの分離レベルを継承している場合に、指定した `isolation-clause` (分離節) の一部として `lock-request-clause` (ロック要求節) が含まれている SQL ステートメントのコンテキストでそのメソッドを呼び出すことができません (SQLSTATE 42601)。

INHERIT ISOLATION LEVEL WITH LOCK REQUEST

関数が呼び出し元のステートメントの分離レベルを継承している場合に、指定した `lock-request-clause` (ロック要求節) をメソッドが継承することを指定します。

SQL-function-body

関数の本体を指定します。パラメーター名を `SQL-function-body` で参照することができます。あいまい参照を避けるため、パラメーター名は関数名で修飾できません。

`SQL-function-body` が動的コンパウンド・ステートメントであれば、ここには少なくとも 1 つの `RETURN` ステートメントが組み込まれていなければならず、また関数が呼び出されるときに `RETURN` ステートメントが実行されなければなりません (SQLSTATE 42632)。関数が表または行関数であれば、`RETURN` ステートメントは、動的コンパウンド・ステートメントの最後の関数でなければなりません。`RETURN` ステートメントを 1 つだけ組み込むことができます (SQLSTATE 429BD)。

注

- 関数本体内での関数呼び出しの解決は、`CREATE FUNCTION` ステートメントに対して有効な SQL パスに従って実行され、関数が作成された後も変更されません。
- SQL 関数に、何らかの日付または時刻の特殊レジスターへの参照が複数含まれている場合、すべての参照は同じ値を戻します。そして、関数を呼び出したステートメントでのレジスター呼び出しにより戻される値と同じになります。
- SQL 関数の本体には、これ自体または他の関数やこれを呼び出すメソッドに対する再帰呼び出しを組み込むことはできません。そのような関数は、呼び出しの対象として存在できないからです。
- 関数またはメソッドを作成するすべてのステートメントで、以下の規則が課されます。
 - 関数のシグニチャーは、メソッドのシグニチャーと同じであってはなりません (関数の最初の `parameter-type` と、メソッドの `subject-type` を比較)。
 - 関数とメソッドは、オーバーライド関係にあってはなりません。つまり、関数が、最初のパラメーターをサブジェクトとするメソッドである場合、これが他のメソッドをオーバーライドしたり、他のメソッドによりオーバーライドされたりすることはできません。オーバーライド・メソッドについての詳細は、『CREATE TYPE (構造化)』ステートメントを参照してください。
 - 関数にはオーバーライドが適用されないため、2 つの関数がメソッドである場合に、一方が他方をオーバーライドする状態で存在することは可能です。

上記の規則で `parameter-types` を比較する目的で、以下ようになります。

- パラメーター名、長さ、`AS LOCATOR`、および `FOR BIT DATA` は無視されます。

CREATE FUNCTION (SQL スカラー、表、または行)

- サブタイプとそのスーパータイプは異なるものと見なされます。

• 表アクセスの制限

関数が READS SQL DATA に定義されている場合には、関数のいかなるステートメントも、関数を呼び出したステートメントによって変更されている表にはアクセスできません (SQLSTATE 57053)。例えば、ユーザー定義関数 BONUS() が READS SQL DATA に定義されているとします。ステートメント UPDATE EMPLOYEE SET SALARY = SALARY + BONUS(EMPNO) が呼び出される場合、BONUS 関数の SQL ステートメントは、EMPLOYEE 表からの読み取りを行えません。

ネストされた CALL ステートメントが MODIFIES SQL DATA で定義された関数に含まれている場合、この関数によって (関数定義またはこの関数を呼び出すステートメントによって) 変更される表への読み取りアクセスは許可されません (SQLSTATE 57053)。

• 特権

関数の定義者は、関数に対する EXECUTE 特権と、関数をドロップする権利を常に与えられます。関数を定義するのに必要なすべての特権に対して定義者が WITH GRANT OPTION を持っている場合、または定義者が SYSADM か DBADM 権限を持っている場合には、関数の定義者には、関数に対する WITH GRANT OPTION も与えられます。

関数の定義者にそれらの特権が与えられるのは、それらの特権の派生元の特権が関数の作成時に存在している場合に限りです。定義者は、これらの特権を直接持っているか、または PUBLIC の特権として持っていることが必要です。関数の定義者がメンバーであるグループを持つ特権は考慮されません。関数を使用する場合、接続済みのユーザーの許可 ID には、そのデータ・ソースでニックネームが参照する表またはビューに対する適切な特権がなければなりません。

• 互換性

- DB2 for z/OS との互換性:

- 以下の構文はデフォルトの振る舞いとして受け入れられます。

- Unicode データベースでの CCSID UNICODE
- 非 Unicode データベースでの CCSID ASCII

- 以前のバージョンの DB2 との互換性:

- CALLED ON NULL INPUT の代わりに NULL CALL を指定できます。

例

例 1: 既存のサインおよびコサイン関数を使用して、値のタンジェントを戻すスカラー関数を定義します。

```
CREATE FUNCTION TAN (X DOUBLE)
  RETURNS DOUBLE
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN SIN(X)/COS(X)
```

例 2: 構造化タイプ PERSON のトランスフォーム関数を定義します。

CREATE FUNCTION (SQL スカラー、表、または行)

```
CREATE FUNCTION FROMPERSON (P PERSON)
  RETURNS ROW (NAME VARCHAR(10), FIRSTNAME VARCHAR(10))
  LANGUAGE SQL
  CONTAINS SQL
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN VALUES (P..NAME, P..FIRSTNAME)
```

例 3: 指定された部門番号の従業員を戻す表関数を定義します。

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  RETURN
  SELECT EMPNO, LASTNAME, FIRSTNAME
  FROM EMPLOYEE
  WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
```

例 4: スtringを反転するスカラー関数を定義します。

```
CREATE FUNCTION REVERSE(INSTR VARCHAR(4000))
  RETURNS VARCHAR(4000)
  DETERMINISTIC NO EXTERNAL ACTION CONTAINS SQL
  BEGIN ATOMIC
  DECLARE REVSTR, RESTSTR VARCHAR(4000) DEFAULT '';
  DECLARE LEN INT;
  IF INSTR IS NULL THEN
  RETURN NULL;
  END IF;
  SET (RESTSTR, LEN) = (INSTR, LENGTH(INSTR));
  WHILE LEN > 0 DO
  SET (REVSTR, RESTSTR, LEN)
  = (SUBSTR(RESTSTR, 1, 1) CONCAT REVSTR,
    SUBSTR(RESTSTR, 2, LEN - 1),
    LEN - 1);
  END WHILE;
  RETURN REVSTR;
END
```

例 5: 例 4 から監査付きで表関数を定義します。

```
CREATE FUNCTION DEPTEMPLOYEES (DEPTNO CHAR(3))
  RETURNS TABLE (EMPNO CHAR(6),
                 LASTNAME VARCHAR(15),
                 FIRSTNAME VARCHAR(12))
  LANGUAGE SQL
  MODIFIES SQL DATA
  NO EXTERNAL ACTION
  DETERMINISTIC
  BEGIN ATOMIC
  INSERT INTO AUDIT
  VALUES (USER,
          'Table: EMPLOYEE Prd: DEPTNO = ' CONCAT DEPTNO);
  RETURN
  SELECT EMPNO, LASTNAME, FIRSTNAME
  FROM EMPLOYEE
  WHERE EMPLOYEE.WORKDEPT = DEPTEMPLOYEES.DEPTNO
END
```

CREATE FUNCTION MAPPING

CREATE FUNCTION MAPPING ステートメントは、以下の目的で使用されます。

- フェデレーテッド・データベース関数 (または関数テンプレート) と、データ・ソース関数との間でマッピングを定義する。マッピングによって、フェデレーテッド・データベース関数またはテンプレートを以下に含まれている関数に関連付けることができます。
 - 指定したデータ・ソース
 - データ・ソースの範囲。たとえば、特定のタイプおよびバージョンのすべてのデータ・ソース
- フェデレーテッド・データベース関数とデータ・ソース関数との間での、デフォルトのマッピングを使用不可にする。

1 つの関数に対して複数の関数マッピングが適用可能である場合、最新のマッピングが適用されます。

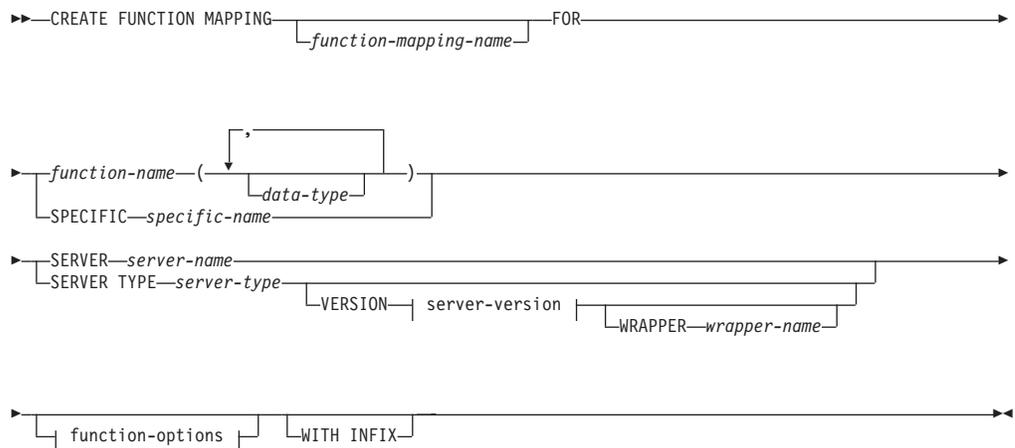
呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

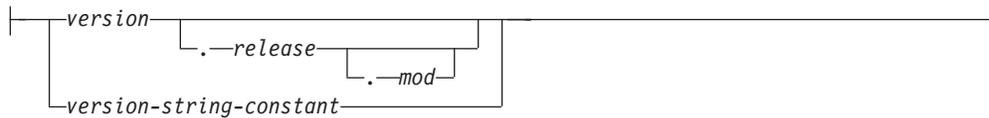
このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文

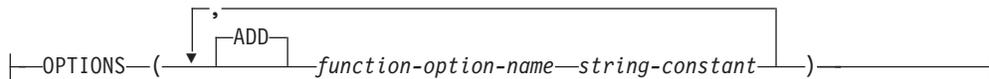


server-version:

CREATE FUNCTION MAPPING



function-options:



説明

function-mapping-name

関数マッピングを指定します。この名前は、カタログですでに記述されている関数マッピングを指定するものであってはなりません (SQLSTATE 42710)。

function-mapping-name を省略すると、システムが生成したユニークな名前が割り当てられます。

function-name

マップ元となるフェデレーテッド・データベース関数またはフェデレーテッド・データベース関数テンプレートの修飾名または非修飾名を指定します。

data-type

入力パラメーターのある関数または関数テンプレートの場合、*data-type* にそれらのパラメーターのデータ・タイプを指定します。*data-type* を、DECFLOAT、LONG VARCHAR、LONG VARGRAPHIC、XML、またはユーザー定義タイプにすることはできません。

パラメーター化データ・タイプに長さ、精度、または位取りを指定する代わりに、空の括弧を使用できます。パラメーター化データ・タイプには空の括弧を使用することが推奨されています。たとえば、CHAR() のようにします。特定の長さ、位取り、または精度を指定して定義可能なデータ・タイプのことを、パラメーター化データ・タイプといいます。パラメーター化データ・タイプは、ストリング・データ・タイプと 10 進データ・タイプです。指定する長さ、精度、または位取りは、関数テンプレートのもと同じでなければなりません。括弧をすべて省略した場合は、データ・タイプのデフォルト長が使用されます (CREATE TABLE ステートメントの説明を参照)。

SPECIFIC *specific-name*

マップ元の関数または関数テンプレートを指定します。*specific-name* を指定して、便利な関数名を作成します。

SERVER *server-name*

マップ元の関数を含むデータ・ソースを指定します。

SERVER TYPE *server-type*

マップ元の関数を含むデータ・ソースのタイプを指定します。

VERSION

server-type に示されたデータ・ソースのバージョンを指定します。

version

バージョン番号を指定します。値は整数でなければなりません。

release

version で示されたバージョンのリリース番号を指定します。値は整数でなければなりません。

mod

release で示されたリリースのモディフィケーション番号を指定します。値は整数でなければなりません。

version-string-constant

バージョンの正式名称を指定します。 *version-string-constant* は単一値 (例えば、'8i') にすることができます。あるいは、*version*、*release*、そして該当する場合は *mod* を連結した値にすることができます (例えば、'8.0.3')。

WRAPPER *wrapper-name*

server-type および *server-version* に示されたタイプおよびバージョンのデータ・ソースと対話するために、フェデレーテッド・サーバーが使用するラッパーの名前を指定します。

OPTIONS

使用可能にする関数マッピング・オプションを指定します。

ADD

1 つ以上の関数マッピング・オプションを使用可能にします。

function-option-name

関数マッピング、またはマッピングに含まれるデータ・ソース関数のいずれかに適用される、関数マッピング・オプションを指定します。

string-constant

function-option-name の設定を、文字ストリング定数として指定します。

WITH INFIX

データ・ソース関数が *infix* 形式で生成されることを指定します。フェデレーテッド・データベース・システムは接頭表記法をリモート・データ・ソースが使用する中置表記法に変換します。

注

- フェデレーテッド・データベースの関数または関数テンプレートは、以下の場合に、データ・ソース関数へマッピングすることができます。
 - フェデレーテッド・データベース関数またはテンプレートに、データ・ソース関数と同じ数の入力パラメーターがある場合。
 - フェデレーテッド・データベース関数またはテンプレートに定義されたデータ・タイプと、データ・ソース関数に定義された対応するデータ・タイプとが互換性のある場合。
- 分散要求が、データ・ソース関数へマップされる DB2 関数を参照する場合、オプティマイザーは、要求の処理時にいずれかの関数を呼び出すための戦略を開発します。DB2 関数を呼び出すときのオーバーヘッドが、データ・ソース関数を呼び出す場合より少なければ、DB2 関数が呼び出されます。しかし、DB2 関数の方がオーバーヘッドが大きいのであれば、データ・ソース関数が呼び出されません。

CREATE FUNCTION MAPPING

- 分散要求が、データ・ソース関数へマップされる DB2 関数テンプレートを参照する場合、要求の処理時には、データ・ソース関数だけ呼び出せます。テンプレートには実行可能コードがないので呼び出せません。
- デフォルトの関数マッピングを使用不可にして、操作できないよう設定することができます (ドロップすることはできません)。デフォルトの関数マッピングを使用不可にするには、`CREATE FUNCTION MAPPING` ステートメントをコーディングし、マッピング内に DB2 関数の名前を指定して `DISABLE` オプションを 'Y' に設定します。
- `SYSIBM` スキーマ内の関数には特定の名称はありません。 `SYSIBM` スキーマの関数のデフォルト関数マッピングをオーバーライドするには、明示修飾子 `SYSIBM` (`SYSIBM.LENGTH()` など) を使用して *function-name* を指定します。
- 所定の作業単位 (UOW) 内の `CREATE FUNCTION MAPPING` ステートメントは、以下のいずれかの条件の下では処理できません (SQLSTATE 55007)。
 - ステートメントが 1 つのデータ・ソースを参照していて、次のいずれかがすでに UOW に含まれている。
 - このデータ・ソース内の表またはビューのニックネームを参照する `SELECT` ステートメント。
 - このデータ・ソース内の表またはビューのニックネーム上のオープン・カーソル。
 - このデータ・ソース内の表またはビューのニックネームに対して発行された `INSERT`、`DELETE`、または `UPDATE` ステートメント。
 - ステートメントがデータ・ソースのカテゴリ (たとえば、特定のタイプおよびバージョンのすべてのデータ・ソースなど) を参照しており、次のいずれかがすでに UOW に含まれている。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネームを参照する `SELECT` ステートメント。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネーム上のオープン・カーソル。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネームに対して発行された `INSERT`、`DELETE`、または `UPDATE` ステートメント。

例

例 1: 関数テンプレートを、すべての Oracle データ・ソースでアクセスできる UDF にマップします。このテンプレートは `STATS` といい、`NOVA` というスキーマに属するものです。Oracle UDF は `STATISTICS` といい、`STAR` というスキーマに属しています。

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN1
FOR NOVA.STATS (DOUBLE, DOUBLE)
SERVER TYPE ORACLE
OPTIONS (REMOTE_NAME 'STAR.STATISTICS')
```

例 2: `BONUS` という関数テンプレートを、`ORACLE1` という Oracle データ・ソースで使われている UDF (これも `BONUS` という) へマップします。

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN2
FOR BONUS()
SERVER ORACLE1
OPTIONS (REMOTE_NAME 'BONUS')
```

例 3: フェデレーテッド・データベースに対して定義されている WEEK システム関数と、Oracle データ・ソースに対して定義されている同様の関数との間で、デフォルトの関数マッピングが行われているとします。Oracle データを要求し、WEEK を参照する照会が処理されると、オプティマイザーによるオーバーヘッドを少なくするための見積もりに応じて、WEEK 関数またはその Oracle 側の対となる関数のいずれかが呼び出されます。DBA は、そのときの照会で WEEK だけが呼び出された場合に、パフォーマンスがどの程度影響を受けるかを確認するようにしてください。WEEK が毎回確実に呼び出されるようにするには、DBA はマッピングを使用不可にしなければなりません。

```
CREATE FUNCTION MAPPING
FOR SYSFUN.WEEK(INT)
SERVER TYPE ORACLE
OPTIONS (DISABLE 'Y')
```

例 4: フェデレーテッド関数 UCASE(CHAR) を、ORACLE2 という Oracle データ・ソースで使っている UDF へマップします。Oracle UDF の呼び出しごとの見積命令数を組み込みます。

```
CREATE FUNCTION MAPPING MY_ORACLE_FUN4
FOR SYSFUN.UCASE(CHAR)
SERVER ORACLE2
OPTIONS
(REMOTE_NAME 'UPPERCASE',
INSTS_PER_INVOC '1000')
```

CREATE HISTOGRAM TEMPLATE

CREATE HISTOGRAM TEMPLATE ステートメントは、1 つ以上のサービス・クラスまたは作業クラスのデフォルト・ヒストグラムをオーバーライドするために使用できるヒストグラムのタイプを記述するテンプレートを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文

```
►►—CREATE HISTOGRAM TEMPLATE—template-name—HIGH BIN VALUE—bigint-constant—◀◀
```

説明

template-name

ヒストグラム・テンプレートの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。名前は、現行のサーバ

CREATE HISTOGRAM TEMPLATE

一上の既存のヒストグラム・テンプレートを識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

HIGH BIN VALUE *bigint-constant*

最後から 2 番目の bin の上限値を指定します (最後の bin の値には上限がありません)。単位はヒストグラムの使い方によって異なります。最大値は 268 435 456 です。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 全パーティションを通じて、同時に実行できる非コミットの WLM 排他 SQL ステートメントは 1 つのみです。非コミットの WLM 排他 SQL ステートメントが実行されている場合、後続の WLM 排他 SQL ステートメントは、現行の WLM 排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

サービス・スーパークラス ADMIN のサービス・クラス PAYROLL で LIFETIMETEMP という名前のヒストグラム・テンプレートを作成します。これは、デフォルトのアクティビティ存続時間ヒストグラム・テンプレートを新しい HIGH BIN VALUE 90 000 (90 000 マイクロ秒を表す) でオーバーライドします。これによりヒストグラムが生成され、その bin の範囲は急激に増大します。bin が 90 000 から無限大の範囲に入ると終了します。

```
CREATE HISTOGRAM TEMPLATE LIFETIMETEMP
HIGH BIN VALUE 90000
```

```
CREATE SERVICE CLASS PAYROLL
UNDER ADMIN ACTIVITY LIFETIME HISTOGRAM TEMPLATE LIFETIMETEMP
```

CREATE INDEX

CREATE INDEX ステートメントは、以下の目的で使用されます。

- DB2 表に対する索引を定義します。索引は、XML データまたはリレーショナル・データに関して定義できます。
- SPECIFICATION ONLY 指定の索引、つまり、データ・ソース表に索引があることをオプティマイザーに通知するメタデータを作成します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

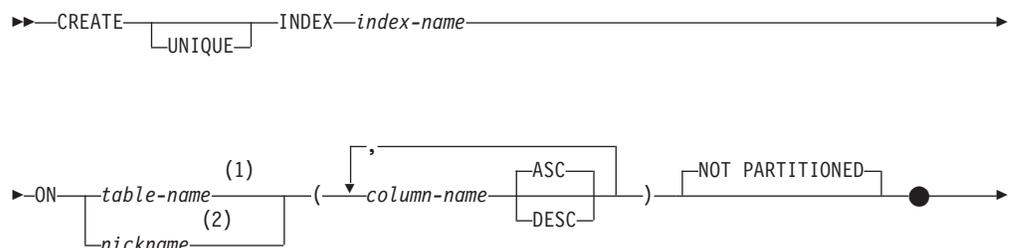
許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

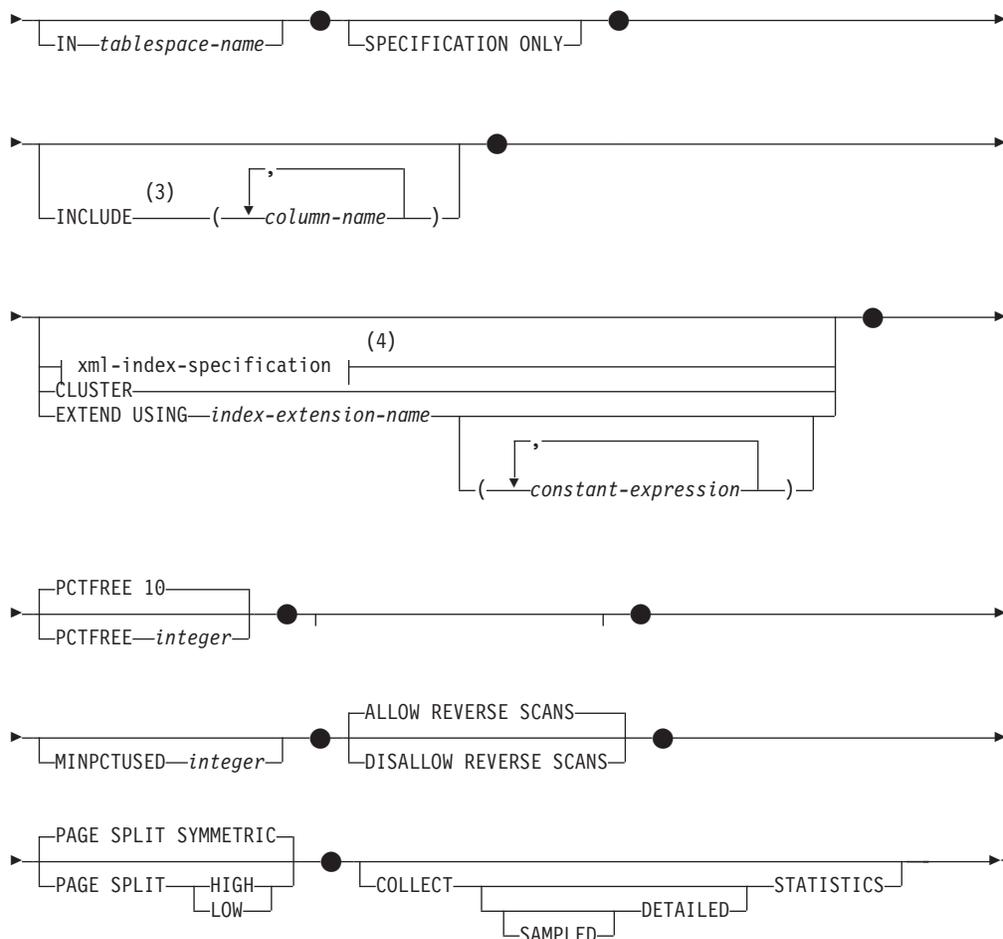
- 以下のいずれか
 - その索引の定義されている表またはニックネームに対する CONTROL 特権。
 - その索引の定義されている表またはニックネームに対する INDEX 特権。
- および以下のいずれか
 - データベースに対する IMPLICIT_SCHEMA 権限 (索引の暗黙または明示のスキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (索引のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

宣言済み一時表の索引を作成するのに、明示特権は必要ありません。

構文



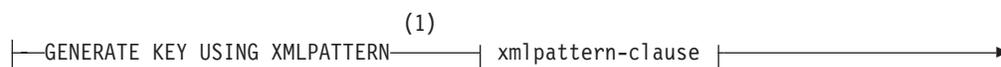
CREATE INDEX



注:

- 1 フェデレーテッド・システムでは、*table-name* には、フェデレーテッド・データベース内の表を指定します。データ・ソース表を指定することはできません。
- 2 *nickname* を指定すると、`CREATE INDEX` ステートメントは、`SPECIFICATION ONLY` 指定の索引を作成します。この場合、`INCLUDE`、*xml-index-specification*、`CLUSTER`、`EXTEND USING`、`PCTFREE`、`MINPCTUSED`、`DISALLOW REVERSE SCANS`、`ALLOW REVERSE SCANS`、`PAGE SPLIT`、または `COLLECT STATISTICS` は指定できません。
- 3 `INCLUDE` 節は `UNIQUE` が指定されている場合のみ指定できます。
- 4 *xml-index-specification* が指定された場合、*column-name* `DESC`、`INCLUDE`、または `CLUSTER` は指定できません。

xml-index-specification:



▶ | xmltype-clause |-----|

注:

1 代替構文 GENERATE KEYS USING XMLPATTERN が使用できます。

xmlpattern-clause:

| '-----|
 | namespace-declaration | pattern-expression | '-----|

namespace-declaration:

|-----|
 | DECLARE NAMESPACE—*namespace-prefix*=*namespace-uri* ; |-----|
 | DECLARE DEFAULT ELEMENT NAMESPACE—*namespace-uri* |-----|

pattern-expression:

|-----|
 | / | forward-axis | xmlname-test |-----|
 | // | | xmlkind-test |-----|

forward-axis:

|-----|
 | child:: |-----|
 | @ |-----|
 | attribute:: |-----|
 | descendant:: |-----|
 | self:: |-----|
 | descendant-or-self:: |-----|

xmlname-test:

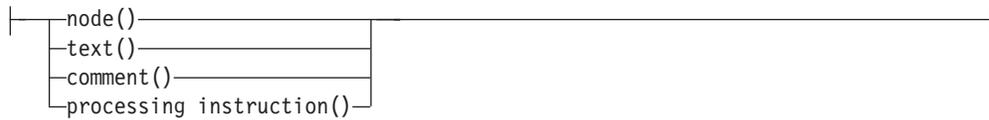
|-----|
 | *xml-qname* |-----|
 | xml-wildcard |-----|

xml-wildcard:

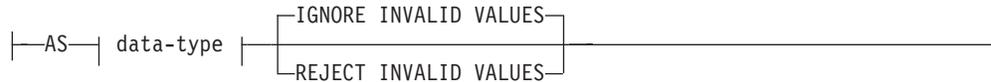
|-----|
 | * |-----|
 | *xml-namespace-prefix*:* |-----|
 | *:*xml-namespace-name* |-----|

xmlkind-test:

CREATE INDEX



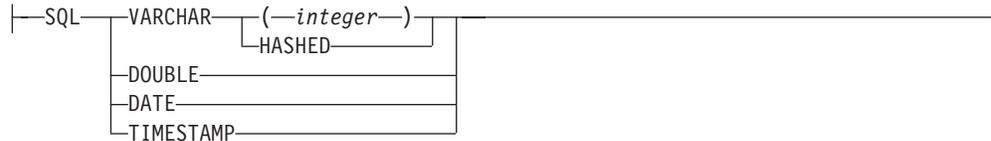
xmltype-clause:



data-type:



sql-data-type:



説明

UNIQUE

ON *table-name* を指定する場合、UNIQUE により、表には索引キーの値が同じである複数の行を含めることができなくなります。行の更新、または新しい行の挿入を行う SQL ステートメントの終了時に、固有性が確保されます。

この固有性は、CREATE INDEX ステートメントの実行の過程でも検査されます。重複するキー値を含む行がすでに表に含まれている場合、索引は作成されません。

索引が XML 列にあれば (索引が XML データについての索引であれば)、表のすべての行に関して指定された *pattern-expression* によって値に固有性が適用されます。固有性は、値が指定の *sql-data-type* に変換された後、各値に強制されます。指定の *sql-data-type* への変換により、精度または範囲の損失が起きる可能性があり、同じキー値に複数の異なる値がハッシュされる可能性もあるため、XML 文書の中で固有に見える複数の値が、重複キー・エラーを引き起こす恐れもあります。文字ストリングの固有性は、末尾空白が重要な XQuery セマンティクスに依存しています。したがって、SQL で複製されるものの、末尾空白では異なる値は、XML データに対する索引では固有値であると見なされます。

UNIQUE を使用する場合、NULL 値は他の値と同様に扱われます。たとえば、キーが NULL 可能な単一系列である場合、その列では 1 つの NULL 値しか含めることができません。

UNIQUE オプションの指定があり、しかも表に分散キーがある場合、索引キーの列は分散キーのスーパーセットである必要があります。つまり、ユニーク索引キーに対して指定される列は、分散キーのすべての列を含んでいる必要があります (SQLSTATE 42997)。

主キーまたはユニーク・キーは、ディメンションのサブセットにはなりません (SQLSTATE 429BE)。

ON *nickname* が指定されている場合、索引キーのデータ・ソース表の各行に固有値が含まれている場合に限り、UNIQUE を指定するようにします。固有性は検査されません。

XML データに対する索引の場合、UNIQUE は、指定の *pattern-expression* が単一の完全パスを指定し、*descendant* または *descendant-or-self* 軸、*//*、*xml-wildcard*、*node()*、または *processing-instruction()* を含まない場合にのみ指定できます (SQLSTATE 429BS)。

INDEX *index-name*

索引または SPECIFICATION ONLY 指定の索引を指定します。暗黙または明示の修飾子を含む名前は、カタログに記述されている索引または SPECIFICATION ONLY 指定の索引、または宣言済み一時表の既存の索引を識別するものであってはなりません (SQLSTATE 42704)。修飾子は、SYSIBM、SYSCAT、SYSFUN、または SYSSTAT であってはなりません (SQLSTATE 42939)。

宣言済みグローバル一時表の索引の暗黙または明示の修飾子は、SESSION でなければなりません (SQLSTATE 428EK)。

ON *table-name* または *nickname*

table-name には、索引を作成する表を識別します。表は、基本表 (ビューではない) か、カタログに記述されたマテリアライズ照会表か、または宣言済み一時表でなければなりません。宣言済み一時表の名前は、SESSION によって修飾される必要があります。 *table-name* に、カタログ表を指定することはできません (SQLSTATE 42832)。UNIQUE が指定されており、*table-name* が型付き表である場合には、副表を指定することはできません (SQLSTATE 429B3)。

nickname は、SPECIFICATION ONLY 指定の索引を作成するニックネームです。この *nickname* により、索引が SPECIFICATION ONLY 指定の索引によって記述されているデータ・ソース表、またはそのような表に基づくデータ・ソース・ビューのいずれかが参照されます。この *nickname* は、カタログにリストされていなければなりません。

column-name

索引の場合、*column-name* には索引キーを構成する列を指定します。SPECIFICATION ONLY 指定の索引の場合、*column-name* は、フェデレーテッド・サーバーがデータ・ソース表の列を参照するときの名前になります。

各 *column-name* は、表の列を指定する非修飾名でなければなりません。列は 64 個まで指定できます。 *table-name* が型付き表である場合は、63 列まで指定できます。 *table-name* が副表であるならば、副表内の少なくとも 1 つの *column-name* はスーパー表から継承するのではなく、新たに導入する必要があります (SQLSTATE 428DS)。同じ *column-name* を繰り返すことはできません (SQLSTATE 42711)。

指定された列の保管長の合計は、ページ・サイズに対する索引キーの長さの上限を超えてはなりません。キー長の制限については、『SQL の制限』を参照してください。 *table-name* が型付き表である場合は、索引キーの長さ制限は 4 バイト減ります。この長さの上限は、列のデータ・タイプや列が NULL 可能か否かによって変動するシステムのオーバーヘッドにより、より小さい値になることがあります。この制限に影響を与えるオーバーヘッドの詳細については、『CREATE TABLE』の『バイト・カウント』を参照してください。

この長さは、列のデータ・タイプや NULL 可能か否かによって変動するシステムのオーバーヘッドにより、より小さい値になることがあります。この制限に影響を与えるオーバーヘッドの詳細については、『CREATE TABLE』の『バイト・カウント』を参照してください。

列の長さ属性がページ・サイズの索引キーの長さの上限を超えない場合でも、LOB 列、LONG VARCHAR 列、LONG VARGRAPHIC 列、または、LOB、LONG VARCHAR、LONG VARGRAPHIC に基づく特殊タイプ列は、索引の一部として使用できません (SQLSTATE 54008)。構造化タイプ列は、EXTEND USING 節も指定されている場合にのみ指定できます (SQLSTATE 42962)。EXTEND USING 節が指定される場合、列は 1 つしか指定できず、列のタイプは構造化タイプか、あるいは LOB、LONG VARCHAR、LONG VARGRAPHIC を基にしていたのではない特殊タイプでなければなりません (SQLSTATE 42997)。

索引が 1 つの列しか持たず、その列が XML データ・タイプを持ち、GENERATE KEY USING XMLPATTERN 節も指定されている場合、索引は XML データに対する索引になります。XML データ・タイプを持つ列は、GENERATE KEY USING XMLPATTERN 節も同時に指定されている場合にのみ指定できます (SQLSTATE 42962)。GENERATE KEY USING XMLPATTERN 節が指定される場合、列は 1 つしか指定できず、列のタイプは XML でなければなりません。

ASC

索引項目が、列の値の昇順で保持されるように指定します。これがデフォルト設定です。ASC は、EXTEND USING で定義される索引に指定することはできません (SQLSTATE 42601)。

DESC

索引項目が、列の値の降順で保持されるように指定します。DESC は、EXTEND USING で定義される索引に指定することはできません。また、索引が XML データに対する索引である場合にも指定できません (SQLSTATE 42601)。

NOT PARTITIONED

単一の索引が、表に関して定義されたすべてのデータ・パーティションにわたって作成されることを指定します。 *table-name* は、データ・パーティションを伴って定義されている表を示していなければなりません (SQLSTATE 53036)。

IN *tablespace-name*

索引を作成する表スペースを指定します。この節は、パーティション表についての索引に関してのみサポートされます。この節は、INDEX IN 節が表作成時に指定された場合にも指定できます。この節は、その節よりも優先します。

tablespace-name で指定する表スペースは、表のデータ表スペースと同じデータベース・パーティション・グループになければならず、パーティション表の他の表スペースと同じスペース管理方式でなければなりません (SQLSTATE 42838)。ステートメントの許可 ID には、その表スペースに対する USE 特権が必要です。

IN 節が指定されなければ、索引は、CREATE TABLE ステートメントの INDEX IN 節に指定される表スペースに作成されます。INDEX IN 節が指定されなかった場合、表の、最初の可視の、またはアタッチされたデータ・パーティションの表スペースが使用されます。これは、範囲指定に基づいてソートされたデータ・パーティションのリスト中の最初のパーティションです。IN 節が指定されなければ、ステートメントの許可 ID にはデフォルトの表スペースに対する USE 特権は必要ありません。

SPECIFICATION ONLY

このステートメントが、*nickname* で参照するデータ・ソース表に適用される SPECIFICATION ONLY 指定の索引を作成するときに使われることを示します。SPECIFICATION ONLY は、*nickname* を指定した場合に、指定しなければなりません (SQLSTATE 42601)。*table-name* を指定した場合には、指定することはできません (SQLSTATE 42601)。

SPECIFICATION ONLY 指定の索引がユニーク索引に適用される場合、DB2 は、リモート表内の列値が固有であるかどうかを検査しません。リモート列値が固有でない場合、索引列を含むニックネームに対する照会が、誤ったデータやエラーを戻す可能性があります。

宣言済み一時表の索引が作成される場合には、この節は使用できません (SQLSTATE 42995)。

INCLUDE

このキーワードは、一連の索引キー列に付加する追加の列を指定する節を新たに指定します。この節によって組み込まれる列は、固有性を強制するために使用されることはありません。これらの組み込み列を使用して索引のみアクセスを実行することにより、一部の照会のパフォーマンスが向上する可能性があります。この列は、固有性を強制するために使用される列とは区別する必要があります (SQLSTATE 42711)。UNIQUE は、INCLUDE が指定された場合には指定する必要があります (SQLSTATE 42613)。列の数および長さ属性の合計に対する制限は、ユニーク・キーと索引にあるすべての列にも適用されます。

この節は、宣言済み一時表と共に使用できません (SQLSTATE 42995)。

column-name

索引には組み込まれているものの、ユニーク索引キーの一部ではない列を指定します。ユニーク索引キーの列に定義された規則と同様の規則が適用されます。*column-name* に続けてキーワード ASC または DESC を指定しても構いませんが、順序に影響はありません。

INCLUDE は、EXTEND USING で定義される索引に指定することはできません。*nickname* が指定されている場合、また索引が XML 値の索引である場合にも使用できません (SQLSTATE 42601)。

xml-index-specification

XML 列に保管された XML 文書からどのように索引キーが生成されるかを指定

します。 *xml-index-specification* は、索引列が複数存在する場合、または列が XML データ・タイプを持たない場合は、指定できません。

この節は、XML 列にのみ適用されます (SQLSTATE 429BS)。

GENERATE KEY USING XMLPATTERN *xmlpattern-clause*

索引の対象となる XML 文書の部分を指定します。XML パターン値は、*xmlpattern-clause* によって生成される索引付け対象値です。リスト・データ・タイプのノードは索引ではサポートされていません。ノードが *xmlpattern-clause* によって修飾され、そのノードがリスト・データ・タイプであることを示している XML スキーマが存在している場合、このリスト・データ・タイプのノードを索引付けすることはできません (CREATE INDEX ステートメントでは SQLSTATE 23526、または INSERT および UPDATE ステートメントでは SQLSTATE 23525)。

xmlpattern-clause

索引にされるノードを特定するパターン式を含みます。オプションの *namespace-declaration* と必須の *pattern-expression* で構成されます。

namespace-declaration

パターン式に修飾名が含まれている場合は、*namespace-declaration* を指定してネーム・スペース接頭部を定義する必要があります。非修飾名には、デフォルトのネーム・スペースを定義できます。

DECLARE NAMESPACE *namespace-prefix=namespace-uri*

NCName である *namespace-prefix* を、ストリング・リテラルである *namespace-uri* にマップします。*namespace-declaration* には、複数の *namespace-prefix* から *namespace-uri* へのマッピングを含めることができます。*namespace-prefix* は、*namespace-declaration* のリストの中で固有でなければなりません (SQLSTATE 10503)。

DECLARE DEFAULT ELEMENT NAMESPACE *namespace-uri*

非修飾の要素名またはタイプに対するデフォルトのネーム・スペース URI を宣言します。デフォルトのネーム・スペースが宣言されない場合、要素やタイプの非修飾名はどのネーム・スペースにも属さないこととなります。宣言できるデフォルトのネーム・スペースは 1 つだけです (SQLSTATE 10502)。

pattern-expression

XML 文書内で索引が作られているノードを指定します。*pattern-expression* には、パターン・マッチング文字 (*) を入れることができます。XQuery のパス式に似ていますが、DB2 によってサポートされる XQuery 言語のサブセットをサポートしています。

/ (スラッシュ)

パス式のステップを分離します。

// (二重スラッシュ)

これは、*/descendant-or-self::node()/* の短縮構文です。// (二重スラッシュ) は、UNIQUE も指定した場合は使用できません。

forward-axis

child::

コンテキスト・ノードの子を指定します。これは、他のフォワード軸が指定されない場合のデフォルトです。

@ コンテキスト・ノードの属性を指定します。これは、`attribute::` の短縮構文です。

attribute::

コンテキスト・ノードの属性を指定します。

descendant::

コンテキスト・ノードの子孫を指定します。 `descendant::` は、UNIQUE も指定した場合は使用できません。

self::

コンテキスト・ノード自体を単独で指定します。

descendant-or-self::

コンテキスト・ノードとそのコンテキスト・ノードの子孫を指定します。 `descendant-or-self::` は、UNIQUE も指定した場合は使用できません。

xmlname-test

パス内のステップに、修飾 XML 名 (`xml-qname`) またはワイルドカード (`xml-wildcard`) を使用してノード名を指定します。

xml-ncname

XML 1.0 で定義された XML 名。コロン文字を組み込むことはできません。

xml-qname

以下の 2 とおりの形式のいずれかで修飾 XML 名 (QName としても知られる) を指定します。

- `xml-namespace:xml-ncname`。 `xml-namespace` は、範囲内ネーム・スペースを特定する `xml-ncname`。
- `xml-ncname`。暗黙の `xml-namespace` としてデフォルトのネーム・スペースが適用されるよう指定する。

xml-wildcard

`xml-qname` を、以下の 3 とおりの形式のいずれかでワイルドカードとして指定します。

- `*` (単一のアスタリスク文字)。これは、あらゆる `xml-qname` に対応する。
- `xml-namespace:*`。これは、指定のネーム・スペース内のあらゆる `xml-ncname` に対応する。
- `*:xml-ncname`。これは、あらゆる範囲内ネーム・スペースの特定の XML 名に対応する。

`xml-wildcard` は、UNIQUE も指定した場合は使用できません。

xmlkind-test

これらのオプションは、パターン・マッチングするノードのタイプを指定するのに使用します。ユーザーは以下のオプションを使用できます。

node()

あらゆるノードに一致します。 *node()* は、UNIQUE も指定した場合は使用できません。

text()

あらゆるテキスト・ノードに一致します。

comment()

あらゆるコメント・ノードに一致します。

processing-instruction()

あらゆる処理命令ノードに一致します。

processing-instruction() は、UNIQUE も指定した場合は使用できません。

*xmltype-clause***AS data-type**

索引値を保管前に変換するデータ・タイプを指定します。値は、指定した索引 SQL データ・タイプに対応する索引 XML データ・タイプに変換されます。

表 17. 対応する索引データ・タイプ

索引 XML データ・タイプ	索引 SQL データ・タイプ
xs:string	VARCHAR(<i>integer</i>), VARCHAR HASHED
xs:double	DOUBLE
xs:date	DATE
xs:dateTime	TIMESTAMP

VARCHAR(*integer*) および VARCHAR HASHED の場合、値は、XQuery 関数 *fn:string* を使用して *xs:string* 値に変換されます。VARCHAR(*integer*) の長さ属性は、結果 *xs:string* 値への制約として適用されます。VARCHAR HASHED の索引 SQL データ・タイプは、ハッシュ・アルゴリズムを結果 *xs:string* 値に適用し、索引に挿入されるハッシュ・コードを生成します。

データ・タイプ DOUBLE、DATE、および TIMESTAMP を使用する索引の場合、この値は、XQuery キャスト式を使用して索引 XML データ・タイプに変換されます。

索引がユニークであれば、索引にされたタイプに値が変換された後、値の固有性が強制されることになります。

data-type

以下のデータ・タイプがサポートされています。

sql-data-type

サポートされる SQL データ・タイプは以下のとおりです。

VARCHAR(integer)

この書式の VARCHAR が指定されると、DB2 は *integer* を制約として使用します。索引にされる文書ノードに、*integer* より長い値があれば、索引がすでに存在する場合、文書は表に挿入されません。索引が存在しない場合、索引は作成されません。 *integer* は、1 とページ・サイズ依存の最大値の間の値です。ページ・サイズごとの最大値は、表 18 のようになります。

表 18. ページ・サイズごとの文書ノードの最大長

ページ・サイズ	文書ノードの最大長 (単位：バイト)
4KB	817
8KB	1841
16KB	3889
32KB	7985

XQuery セマンティクスが、ストリングの比較に使用されます。ここで、末尾ブランクは重要です。これは、SQL セマンティクスとは異なっています。その場合は、比較時に末尾ブランクは重要ではありません。

VARCHAR HASHED

任意の長さの文字ストリングの索引作成を扱う VARCHAR HASHED を指定します。索引にされるストリングの長さには制限はありません。DB2 は、ストリング全体に対応して 8 バイトのハッシュ・コードを生成します。これらのハッシュされた文字ストリングを使用する索引は、同等性検索にのみ使用できます。

XQuery セマンティクスが、ストリングの等価比較に使用されます。ここで、末尾ブランクは重要です。これは、SQL セマンティクスとは異なっています。その場合は、比較時に末尾ブランクは重要ではありません。ストリング上のハッシュは、等価の XQuery セマンティクスを保持しますが、SQL セマンティクスは保持しません。

DOUBLE

データ・タイプ DOUBLE が数値の索引作成に使用されるよう指定します。無制限の DECIMAL タイプと 64 ビットの整数は、DOUBLE 値として保存される場合、精度を失う場合があります。DOUBLE の値には、特別な数値 *NaN*、*INF*、*-INF*、*+0*、および *-0* を含むことができます。ただし、SQL データ・タイプ DOUBLE 自体はこれらの値をサポートしていません。

DATE

データ・タイプ DATE が XML 値の索引作成に使用されるよう指定します。xs:date の XML スキーマ・デー

タ・タイプは、SQL データ・タイプよりも精度を高めることができます。範囲外の値が検出されると、エラーが戻されます。

TIMESTAMP

データ・タイプ `TIMESTAMP` が XML 値の索引作成に使用されるよう指定します。 `xs:dateTime` の XML スキーマ・データ・タイプは、SQL データ・タイプよりも精度を高めることができます。範囲外の値が検出されると、エラーが戻されます。

IGNORE INVALID VALUES

ターゲット索引の XML データ・タイプで無効な XML パターン値は無視され、格納されている XML 文書に含まれている対応する値の索引が `CREATE INDEX` ステートメントによって生成されないことを指定します。デフォルトでは、無効値は無視されます。挿入と更新の操作では、無効な XML パターン値の索引は生成されませんが、XML 文書は表に挿入されます。このようなデータ・タイプを指定しても XML パターン値の制約とは見なされないため、エラーや警告にはなりません (特定の XML 索引データ・タイプを検索する XQuery 式は、それらの値を処理の対象にしません)。

索引で無視できるのは、索引の XML データ・タイプで無効な XML パターン値だけです。有効な値は、索引の XML データ・タイプの値の DB2 表記法に準拠していなければなりません。そうでない場合は、エラーが戻されます。索引の XML データ・タイプ `xs:string` に関連した XML パターン値は、常に有効です。ただし、関連する索引の SQL データ・タイプ `VARCHAR(integer)` データ・タイプの長さに関する追加の制約については、最大長を超えればエラーになります。エラーが戻される場合、索引がすでに存在していれば、XML データが表に挿入されたり、表の中で更新されたりすることはありません (SQLSTATE 23525)。索引が存在しない場合、索引は作成されません (SQLSTATE 23526)。

REJECT INVALID VALUES

索引の XML データ・タイプで、すべての XML パターン値が有効でなければならないことを指定します。いずれかの XML パターン値を索引の XML データ・タイプにキャストできない場合は、エラーが戻されます。索引がすでに存在していれば、XML データが表に挿入されたり、表の中で更新されたりすることはありません (SQLSTATE 23525)。索引が存在しない場合、索引は作成されません (SQLSTATE 23526)。

CLUSTER

当該の索引を表のクラスター索引として指定します。クラスター索引のクラスター係数は、関連する表にデータが挿入される時に、動的に維持または改善されます。これは、この索引のキー値が同じ範囲にある行と物理的に近い位置に、新しい行の挿入を試みることによって行われます。ただし、表のクラスター索引は 1 つだけなので、`CLUSTER` が表の既存の索引の定義に使用されていて、

CLUSTER が指定できないということもあります (SQLSTATE 55012)。追加モードを使用するように定義されている表では、クラスター索引を作成できない場合があります (SQLSTATE 428D8)。

CLUSTER は、*nickname* が指定されている場合、または索引が XML データに対する索引である場合は使用できません (SQLSTATE 42601)。この節は、宣言済み一時表 (SQLSTATE 42995) や範囲クラスター表 (SQLSTATE 429BG) では使用できません。

EXTEND USING *index-extension-name*

この索引を管理するのに使用する *index-extension* を指定します。この節を指定する場合、1 つだけ *column-name* を指定しなければならず、この列は構造化タイプまたは特殊タイプでなければなりません (SQLSTATE 42997)。

index-extension-name (索引拡張名) は、カタログに記述されている索引拡張を指定する名前ではなければなりません (SQLSTATE 42704)。特殊タイプの場合には、列が、索引拡張でソース・キー・パラメーターに対応するタイプと完全に一致していなければなりません。構造化タイプ列では、対応するソース・キー・パラメーターのタイプが、列タイプのタイプまたはスーパータイプと同じでなければなりません (SQLSTATE 428E0)。

この節は、宣言済み一時表と共に使用できません (SQLSTATE 42995)。

constant-expression

索引拡張に必要な引数の値を指定します。各式は、対応する索引拡張パラメーターの定義されたデータ・タイプ (長さまたは精度、およびスケールも含む) に完全に一致するデータ・タイプを持つ定数値でなければなりません (SQLSTATE 428E0)。この節は、データベース・コード・ページ内で、32 768 バイト以内の長さでなければなりません (SQLSTATE 22001)。

PCTFREE *integer*

索引を構築する際に、各索引ページに残すフリー・スペースのパーセンテージを指定します。ページの最初の項目は、制限なしで追加されます。索引ページに項目を追加する場合には、各ページに少なくとも *integer* パーセントをフリー・スペースとして残します。 *integer* の値は 0 から 99 です。10 よりも大きな値を指定しても、ノンリーフ・ページには 10% のフリー・スペースしか残されません。デフォルト値は 10 です。

PCTFREE は、*nickname* が指定されている場合は使用できません (SQLSTATE 42601)。この節は、宣言済み一時表と共に使用できません (SQLSTATE 42995)。

LEVEL2 PCTFREE *integer*

索引を作成する際に、索引レベル 2 の各ページで何 % をフリー・スペースとして残すかを指定します。 *integer* の値は 0 から 99 です。LEVEL2 PCTFREE が設定されない場合は、すべての非リーフ・ページに最低 10 % または PCTFREE で指定された分 (%) のフリー・スペースが残されます。LEVEL2 PCTFREE が設定された場合は、 *integer* で指定された分 (%) のフリー・スペースがレベル 2 の中間ページに残され、レベル 3 以上の中間ページには最低 10 % または *integer* で指定された分 (%) のフリー・スペースが残されます。

nickname が指定されている場合は、LEVEL2 PCTFREE は使用できません (SQLSTATE 42601)。この節は、宣言済み一時表と共に使用できません (SQLSTATE 42995)。

MINPCTUSED *integer*

索引のリーフ・ページをオンラインでマージするかどうか、および索引のリーフ・ページで使用されるスペースの最小パーセンテージの限界値を指定します。索引のリーフ・ページからキーを除去した後、そのページで使用されているスペースのパーセントが *integer* のパーセントを下回る場合、このページにある残りのキーを近隣のページのキーにマージするよう試行されます。いずれかのページに十分なスペースがあれば、マージが行われ、いずれかのページが削除されます。*integer* の値は 0 から 99 です。パフォーマンス上の理由のため、50 以下の値をお勧めします。このオプションを指定すると、更新および削除のパフォーマンスに影響があります。タイプ 2 の索引の場合、排他的表ロックがあると、更新および削除操作の実行中に限りマージされます。排他的表ロックがない場合には、更新および削除操作の間にキーは疑似的に削除されたものとしてマークされ、マージは実行されません。リーフ・ページをマージするには、CREATE INDEX の MINPCTUSED を使うのではなく、REORG INDEXES の CLEANUP ONLY ALL オプションを使用することを考慮してください。

MINPCTUSED は、*nickname* が指定されている場合は使用できません (SQLSTATE 42601)。この節は、宣言済み一時表と共に使用できません (SQLSTATE 42995)。

DISALLOW REVERSE SCANS

索引において、前方向スキャン、すなわち索引作成時に定義された順序でのスキャンだけをサポートすることを指定します。

DISALLOW REVERSE SCANS は *nickname* と同時には指定できません (SQLSTATE 42601)。

ALLOW REVERSE SCANS

索引が前方向スキャンと反対方向スキャンの両方、すなわち、索引作成時に定義された順序での索引のスキャンと、その反対の順序でのスキャンをサポートすることを指定します。

ALLOW REVERSE SCANS は *nickname* と同時には指定できません (SQLSTATE 42601)。

PAGE SPLIT

索引分割の振る舞いを指定します。デフォルトは SYMMETRIC です。

SYMMETRIC

ページをだまかに半分で分割するよう指定します。

HIGH

索引キーの値が特定のパターンに従って挿入されている場合に、索引ページのスペースを効率的に使う索引ページの分割の振る舞いを指定します。索引キー値のサブセットについては、索引の左端の列 (複数の場合もある) には常に同じ値が含まれていなければならない、索引の右端の列 (複数の場合もある) には挿入ごとに増加する値が含まれていなければならない。詳細は、『CREATE INDEX ステートメントのオプション』を参照してください。

LOW

索引キーの値が特定のパターンに従って挿入されている場合に、索引ページのスペースを効率的に使う索引ページの分割の振る舞いを指定します。索引キー値のサブセットについては、索引の左端の列 (複数の場合もある) には常に同じ値が含まれていなければならない、索引の右端の列 (複数の場合もある) には挿入ごとに減少する値が含まれていなければならない。詳細は、『CREATE INDEX ステートメントのオプション』を参照してください。

COLLECT STATISTICS

索引の作成時に基本索引統計が収集されるように指定します。

DETAILED

索引の作成時に、拡張索引統計 (CLUSTERFACTOR および PAGE_FETCH_PAIRS) も収集されるように指定します。

SAMPLED

拡張索引統計のコンパイル時に、サンプリングを使用できるように指定します。

規則

- 既存の索引に一致する索引を作成しようとする、CREATE INDEX ステートメントはエラーになります (SQLSTATE 01550)。

2 つの索引記述は、以下の場合に重複していると見なされます。

- 列セット (キーと組み込み列の両方) と、索引内でのそれらの順序が、既存の索引と同じであり、かつ
- 順序付け属性が同じであり、しかも
- 以前に存在していた索引と作成中の索引の両方がユニークでないか、または以前に存在していた索引がユニークであり、さらに
- 以前に存在していた索引と作成中の索引の両方がユニークである場合、作成中の索引のキー列は、以前に存在していた索引のキー列と同一であるか、またはそのスーパーセットになります。

XML データについての索引の場合、索引記述は、索引名が異なっていれば、たとえ索引にされる XML 列、XML パターン、およびオプションも含めたデータ・タイプが同一であっても、重複しているとはみなされません。

- システム保守 MQT 上のユニーク索引は、サポートされません (SQLSTATE 42809)。
- COLLECT STATISTICS オプションは、ニックネームが指定される場合にはサポートされません (SQLSTATE 42601)。

注

- XML データについての索引では、CREATE INDEX 実行時の並行書き込みアクセスはサポートされません。
- リレーショナル索引のみ: 索引が作成されている間は、表に同時に読み取り/書き込みアクセスできません。索引が作成されると、索引の作成時に表に加えられた変更は、新しい索引に送られ適用されます。表への書き込みアクセスは、新しい索引が使用できるようになってから、索引の作成が完了するまでの短時間ブロックされます。

CREATE INDEX

このデフォルトの動作を回避するには、LOCK TABLE ステートメントを使用して、CREATE INDEX ステートメントが発行される前に表を明示的にロックします。(表は SHARE か EXCLUSIVE モードのいずれかでロックできます。読み取りアクセスが許可されているかどうかによります。)

- 指定した表にすでにデータが含まれる場合、CREATE INDEX は、そのデータの索引項目を作成します。表にまだデータが含まれていない場合、CREATE INDEX は索引記述を作成します。索引項目は、データが表に挿入される時点で作成されます。
- 索引が作成され、データが表にロードされた時点で、RUNSTATS コマンドを実行することをお勧めします。RUNSTATS コマンドは、データベース表、列、および索引について収集された統計値を更新します。これらの統計値は、表への最適アクセス・パスを判別するために使用されます。RUNSTATS コマンドを実行することによって、データベース・マネージャーが新しい索引の特性を判別することができます。CREATE INDEX ステートメントが発行される前にデータをロードする場合には、CREATE INDEX ステートメントの COLLECT STATISTICS オプションを、RUNSTATS コマンドの代わりに使用することをお勧めします。
- まだ存在していないスキーマ名を用いて索引を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- オプティマイザーは、実際の索引を作成する前に、複数の索引を推奨することがあります。
- 索引のあるデータ・ソース表に SPECIFICATION ONLY 指定の索引を定義している場合、その SPECIFICATION ONLY 指定の索引の名前は索引の名前と一致していなくても構いません。
- オプティマイザーは SPECIFICATION ONLY 指定の索引を使用して、その指定を適用するデータ・ソース表へのアクセスを改善します。
- **互換性**
 - DB2 for z/OS との互換性:
 - 以下の構文は許容されますが、無視されます。
 - CLOSE
 - DEFINE
 - FREEPAGE
 - GBPCACHE
 - PIECESIZE
 - TYPE 2
 - using-block
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - COPY NO
 - DEFER NO

例

例 1: PROJECT 表に対して UNIQUE_NAM という名前の索引を作成します。この索引の目的は、プロジェクト名 (PROJNAME) の値が同じ 2 つの項目が表に作成されないようにすることです。索引項目は昇順に並べます。

```
CREATE UNIQUE INDEX UNIQUE_NAM
ON PROJECT (PROJNAME)
```

例 2: EMPLOYEE 表に対して JOB_BY_DPT という名前の索引を作成します。索引項目は、各部門 (WORKDEPT) の中ではジョブ名 (JOB) 順に昇順で並べます。

```
CREATE INDEX JOB_BY_DPT
ON EMPLOYEE (WORKDEPT, JOB)
```

例 3: ニックネーム EMPLOYEE は、CURRENT_EMP というデータ・ソース表を参照します。このニックネームを作成した後、索引が CURRENT_EMP で定義されます。索引キー用に選んだ列は WORKDEPT と JOB です。この索引を記述する SPECIFICATION ONLY 指定の索引を作成します。この指定を参照することにより、オプティマイザーは、索引が存在することと索引に含まれるキーを知ることになります。この情報を利用して、オプティマイザーは、表をアクセスするときの戦略を改善することができます。

```
CREATE UNIQUE INDEX JOB_BY_DEPT
ON EMPLOYEE (WORKDEPT, JOB)
SPECIFICATION ONLY
```

例 4: 構造化タイプ列の位置に、拡張索引タイプ SPATIAL_INDEX を作成します。索引拡張 GRID_EXTENSION の記述が SPATIAL_INDEX を保守するのに使用されます。リテラルが GRID_EXTENSION に指定されて、索引格子サイズを作成します。

```
CREATE INDEX SPATIAL_INDEX ON CUSTOMER (LOCATION)
EXTEND USING (GRID_EXTENSION ('x'000100100010001000400010'))
```

例 5: TAB1 という名前の表に IDX1 という名前の索引を作成し、索引 IDX1 の基本索引統計を収集します。

```
CREATE INDEX IDX1 ON TAB1 (col1) COLLECT STATISTICS
```

例 6: TAB1 という名前の表に IDX2 という名前の索引を作成し、索引 IDX2 の詳細な索引統計を収集します。

```
CREATE INDEX IDX2 ON TAB1 (col2) COLLECT DETAILED STATISTICS
```

例 7: TAB1 という名前の表に IDX3 という名前の索引を作成し、サンプリングを使用して索引 IDX3 の詳細な索引統計を収集します。

```
CREATE INDEX IDX3 ON TAB1 (col3) COLLECT SAMPLED DETAILED STATISTICS
```

例 8: 表スペース IDX_TBSP 内の MYNUMBERDATA というパーティション表に A_IDX というユニーク索引を作成します。

```
CREATE UNIQUE INDEX A_IDX ON MYNUMBERDATA (A) IN IDX_TBSP
```

例 9: 表スペース IDX_TBSP 内の MYNUMBERDATA というパーティション表に B_IDX という非ユニーク索引を作成します。

```
CREATE INDEX B_IDX ON MYNUMBERDATA (B)
NOT PARTITIONED IN IDX_TBSP
```

CREATE INDEX

例 10: COMPANYDOCS という XML 列を含む、COMPANYINFO という表に、XML データに対する索引を作成します。XML 列 COMPANYDOCS には、以下のような数多くの XML 文書が含まれます。

```
<company name="Company1">
  <emp id="31201" salary="60000" gender="Female">
    <name>
      <first>Laura</first>
      <last>Brown</last>
    </name>
    <dept id="M25">
      Finance
    </dept>
  </emp>
</company>
```

COMPANYINFO 表のユーザーは、頻繁に、従業員 ID を使用して従業員情報を検索する必要があります。以下のような索引を作成すると、そうした検索がより効率的になる可能性があります。

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
GENERATE KEY USING XMLPATTERN '/company/emp/@id'
AS SQL DOUBLE
```

例 11: 以下の索引は、論理的には前の例で作成したものと同等ですが、短縮していない構文を使用している点が異なります。

```
CREATE INDEX EMPINDEX ON COMPANYINFO(COMPANYDOCS)
GENERATE KEY USING XMLPATTERN '/child::company/child::emp/attribute::id'
AS SQL DOUBLE
```

例 12: 本のタイトルだけを VARCHAR(100) として索引にし、DOC という列に索引を作成します。本のタイトルは、どれも他のすべての本と異なる固有なものなので、索引もユニークでなければなりません。

```
CREATE UNIQUE INDEX MYDOCSIDX ON MYDOCS(DOC)
GENERATE KEY USING XMLPATTERN '/book/title'
AS SQL VARCHAR(100)
```

例 13: 章番号を DOUBLE として索引にし、DOC という列に索引を作成します。この例には、ネーム・スペース宣言が含まれます。

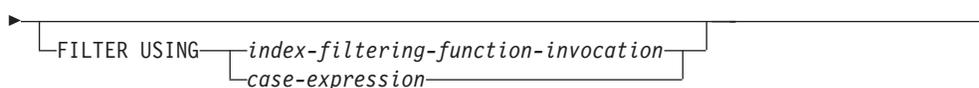
```
CREATE INDEX MYDOCSIDX ON MYDOCS(DOC)
GENERATE KEY USING XMLPATTERN
'declare namespace b="http://www.foobar.com/book/";
declare namespace c="http://acme.org/chapters";
/b:book/c:chapter/@number'
AS SQL DOUBLE
```

CREATE INDEX EXTENSION

CREATE INDEX EXTENSION ステートメントは、構造化タイプまたは特殊タイプ列のある表で索引を使用するための拡張オブジェクトを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。



説明

index-extension-name

索引拡張を指定します。暗黙または明示の修飾子を含む名前は、カタログに記述されている索引拡張を識別するものであってはなりません。2つの部分から成る *index-extension-name* を指定する場合、スキーマ名を 'SYS' で始めることはできません。違反すると、エラーが戻されます (SQLSTATE 42939)。

parameter-name1

CREATE INDEX 時に索引拡張に渡されるパラメーターを指定して、この索引拡張の実際の振る舞いを定義します。索引拡張に渡されるパラメーターは、インスタンス・パラメーターと呼ばれます。この値が索引拡張の新しいインスタンスを定義するためです。

parameter-name1 は、索引拡張の定義内でユニークでなければなりません。パラメーターの数は 90 を超えることはできません。この限界を超えると、エラー (SQLSTATE 54023) になります。

data-type1

各パラメーターのデータ・タイプを指定します。このリストには、索引拡張が受け取ることを予期している各パラメーターごとに、1つの項目を指定する必要があります。指定できる SQL データ・タイプは、VARCHAR、INTEGER、DECIMAL、DOUBLE、または VARGRAPHIC など、定数として使用できるタイプだけです (SQLSTATE 429B5)。10進浮動小数点データ・タイプは指定できません (SQLSTATE 429B5)。CREATE INDEX の索引拡張により受け取られるパラメーター値は、長さ、精度、およびスケールとも、*data-type1* に完全に一致していなければなりません (SQLSTATE 428E0)。

index-maintenance

構造化タイプまたは特殊タイプの列の索引キーを保守する方法を指定します。索引保守は、ソース列をターゲット・キーに変換するプロセスです。変換プロセスは、データベースで以前に定義されている表関数を使用して定義されます。

FROM SOURCE KEY (*parameter-name2 data-type2*)

この索引拡張によりサポートされるソース・キー列の、構造化データ・タイプまたは特殊タイプを指定します。

parameter-name2

ソース・キー列に関連するパラメーターを指定します。ソース・キー列は、*data-type2* と同じデータ・タイプの索引キー列です (CREATE INDEX で定義)。

data-type2

parameter-name2 のデータ・タイプを指定します。*data-type2* は、LOB、XML、DECFLOAT、LONG VARCHAR、または LONG VARGRAPHIC を基にしたのではないユーザー定義の構造化タイプまたは特殊タイプでなければなりません (SQLSTATE 42997)。CREATE

INDEX 時に索引拡張が索引に関連付けられる場合、索引キー列のデータ・タイプは以下のものでなければなりません。

- 特殊タイプの場合、*data-type2* に完全に一致しなければなりません。あるいは、
- 構造化タイプの場合、*data-type2* のタイプまたはサブタイプと同じななければなりません。

これ以外の場合には、エラーになります (SQLSTATE 428E0)。

GENERATE KEY USING *table-function-invocation*

ユーザー定義の表関数を使用して索引キーが生成される方法を指定します。単一のソース・キー・データ値に複数の索引項目を生成できます。単一のソース・キー・データ値から索引項目を複製することはできません (SQLSTATE 22526)。この関数は、引数として *parameter-name1*、*parameter-name2*、または定数を使用できます。データ・タイプ *parameter-name2* が構造化データ・タイプの場合、この引数では、この構造化タイプの *observer* メソッドしか使用できません (SQLSTATE 428E3)。TARGET KEY 指定では、GENERATE KEY 関数の出力を指定しなければなりません。関数の出力は、FILTER USING 節で指定される索引フィルター関数の入力としても使用できます。

table-function-invocation で使用される関数は、次のようであればなりません。

- 表関数に解決されること (SQLSTATE 428E4)
- このデータベースがユニコード・データベースではない場合、PARAMETER CCSID UNICODE で定義しないこと (SQLSTATE 428E4)
- LANGUAGE SQL で定義されていないこと (SQLSTATE 428E4)
- NOT DETERMINISTIC (SQLSTATE 428E4) または EXTERNAL ACTION (SQLSTATE 428E4) で定義されていないこと
- NO SQL で定義されていること (SQLSTATE 428E4)
- パラメーターのデータ・タイプに構造化データ・タイプ、LOB、XML、LONG VARCHAR、または LONG VARGRAPHIC がないこと (SQLSTATE 428E3)。ただし、システム生成のオブザーバー・メソッドだけは例外です。
- 副照会が含まれていないこと (SQLSTATE 428E3)
- XMLQUERY または XMLEXISTS 式が含まれていないこと (SQLSTATE 428E3)
- EXTEND USING 節なしで定義された索引の列のデータ・タイプの制限に従うデータ・タイプを持つ列を戻すこと

引数が他の操作またはルーチンを呼び出す場合、それはオブザーバー・メソッドでなければなりません (SQLSTATE 428E3)。

索引拡張の定義者は、この関数に対して EXECUTE 特権を持っている必要があります。

index-search

検索指数から検索範囲へのマッピングを提供することにより、検索の実行方法を指定します。

CREATE INDEX EXTENSION

WITH TARGET KEY

GENERATE KEY USING 節で指定されるキー生成関数の出力であるターゲット・キー・パラメーターを指定します。

parameter-name3

指定されるターゲット・キーに関連するパラメーターを指定します。

parameter-name3 は、GENERATE KEY USING 節の表関数で指定された RETURNS 表の列に対応します。指定されるパラメーターの数は、表関数で戻される列の数と一致しなければなりません (SQLSTATE 428E2)。

data-type3

それぞれの対応する *parameter-name3* のデータ・タイプを指定します。

data-type3 は、GENERATE KEY USING 節の表関数で指定されたように、RETURNS 表のそれぞれに対応する出力列のデータ・タイプに厳密に一致しなければなりません (SQLSTATE 428E2)。これには、長さ、精度、およびタイプが含まれます。

SEARCH METHODS

索引に定義される検索メソッドを導入します。

search-method-definition

索引検索のメソッドの詳細を指定します。これは、メソッド名、検索指数、範囲生成関数、およびオプションの索引フィルター関数で構成されます。

WHEN *method-name*

検索メソッドの名前。これは、索引活用規則 (ユーザー定義関数の PREDICATES 節にある) で指定されるメソッド名に関連する SQL ID です。検索メソッド定義で *search-method-name* を参照できる WHEN 節は 1 つだけです (SQLSTATE 42713)。

parameter-name4

検索指数のパラメーターを指定します。これらの名前は、RANGE THROUGH および FILTER USING 節で使用されます。

data-type4

検索パラメーターに関連付けられるデータ・タイプ。

RANGE THROUGH *range-producing-function-invocation*

検索範囲を生成する外部表関数を指定します。この関数は *parameter-name1*、*parameter-name4*、または定数を引数として使用し、検索範囲のセットを戻します。

range-producing-function-invocation で使用される表関数は、以下のようであればなりません。

- 表関数に解決されること (SQLSTATE 428E4)
- その引数に副照会 (SQLSTATE 428E3) または SQL 関数 (SQLSTATE 428E4) が含まれていないこと
- 引数内に XMLQUERY または XMLEXISTS 式が含まれていないこと (SQLSTATE 428E3)
- このデータベースがユニコード・データベースではない場合、PARAMETER CCSID UNICODE で定義しないこと (SQLSTATE 428E4)
- LANGUAGE SQL で定義されていないこと (SQLSTATE 428E4)

- NOT DETERMINISTIC または EXTERNAL ACTION で定義されていないこと (SQLSTATE 428E4)
- NO SQL で定義されていること (SQLSTATE 428E4)

この関数の結果の数およびタイプが、以下のように GENERATE KEY USING 節で指定した表関数の結果に関連していること (SQLSTATE 428E1)。

- キー・トランスフォーメーション関数で戻される数の 2 倍以内の数の列を戻す。
- 偶数の列があり、戻りコードの前半で範囲の開始 (開始キー値) を定義し、戻りコードの後半で範囲の終了 (停止キー値) を定義する。
- 対応する停止キー列と同じタイプの開始キー列がある。
- 対応するキー・トランスフォーメーション関数列と同じタイプの開始キー列がある。

厳密には、 $a_1:t_1, \dots, a_n:t_n$ を、関数結果列およびキー・トランスフォーメーション関数のデータ・タイプにします。 *range-producing-function-invocation* の関数結果列は、 $b_1:t_1, \dots, b_m:t_m, c_1:t_1, \dots, c_m:t_m$ でなければなりません。ここで、 $m \leq n$ および "b" 列は開始キー列で、"c" 列は停止キー列です。

range-producing-function-invocation が開始または停止キー値として NULL 値を戻す場合、セマンティクスは未定義です。

索引拡張の定義者は、この関数に対して EXECUTE 特権を持っている必要があります。

FILTER USING

範囲生成関数の適用後に戻された索引項目をフィルター操作する際に使用する、外部関数またはケース式の指定を許可します。

index-filtering-function-invocation

索引項目をフィルター操作するのに使用する外部関数を指定します。この関数は *parameter-name1*、*parameter-name3*、*parameter-name4*、または定数を引数として使用し (SQLSTATE 42703)、整数を戻します (SQLSTATE 428E4)。戻される値が 1 の場合、索引項目に対応する行が表から取り出されます。その他の場合、索引項目をさらに処理することはありません。

これを指定しない場合は、索引のフィルター操作は実行されません。

index-filtering-function-invocation で使用される関数は、以下のものでなければなりません。

- このデータベースがユニコード・データベースではない場合、PARAMETER CCSID UNICODE で定義しないこと (SQLSTATE 428E4)
- LANGUAGE SQL で定義されていないこと (SQLSTATE 429B4)
- NOT DETERMINISTIC または EXTERNAL ACTION で定義されていないこと (SQLSTATE 42845)
- NO SQL で定義されていること (SQLSTATE 428E4)
- どのパラメーターのデータ・タイプにも、構造化データ・タイプがないこと (SQLSTATE 428E3)
- 副照会が含まれていないこと (SQLSTATE 428E3)

CREATE INDEX EXTENSION

- XMLQUERY または XMLEXISTS 式が含まれていないこと (SQLSTATE 428E3)

引数が他の関数またはメソッドを呼び出す場合、このネストされた関数またはメソッドにもこれらの規則が課されます。ただし、引数が組み込みデータ・タイプになるかぎり、システム生成のオブザーバー・メソッドをフィルター関数 (または、引数として使用される任意の関数またはメソッド) への引数として使用することができます。

索引拡張の定義者は、この関数に対して EXECUTE 特権を持っている必要があります。

case-expression

索引項目をフィルター操作するためのケース式を指定します。

searched-when-clause および *simple-when-clause* では、*parameter-name1*、*parameter-name3*、*parameter-name4*、または定数を使用できます (SQLSTATE 42703)。FILTER USING *index-filtering-function-invocation* に指定された規則を使って、外部関数を *result-expression* として使用できます。

case-expression で参照される関数またはメソッドはすべて、*index-filtering-function-invocation* でリストされている規則に適合することも必要です。加えて、副照会および XMLQUERY または XMLEXISTS 式は、*case-expression* の中では使用できません (SQLSTATE 428E4)。ケース式は整数を戻さなければなりません (SQLSTATE 428E4)。*result-expression* で戻り値が 1 の場合は索引項目が保持され、その他の場合は索引項目は破棄されます。

注

- まだ存在していないスキーマ名を用いて索引拡張を作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。

例

例 1: ここでは、*gridEntry* という表関数で構造化タイプ SHAPE 列を使用する索引拡張 *grid_extension* を作成して、7 つの索引ターゲット・キーを生成します。この索引拡張は 2 つの索引検索メソッドも提供して、検索引数が指定される際の検索範囲を生成します。

```
CREATE INDEX EXTENSION GRID_EXTENSION (LEVELS VARCHAR(20) FOR BIT DATA)
FROM SOURCE KEY (SHAPECOL SHAPE)
GENERATE KEY USING GRIDENTRY(SHAPECOL..MBR..XMIN,
                              SHAPECOL..MBR..YMIN,
                              SHAPECOL..MBR..XMAX,
                              SHAPECOL..MBR..YMAX,
                              LEVELS)
WITH TARGET KEY (LEVEL INT, GX INT, GY INT,
                 XMIN INT, YMIN INT, XMAX INT, YMAX INT)
SEARCH METHODS
WHEN SEARCHFIRSTBYSECOND (SEARCHARG SHAPE)
RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
                          SEARCHARG..MBR..YMIN,
                          SEARCHARG..MBR..XMAX,
                          SEARCHARG..MBR..YMAX,
                          LEVELS)

FILTER USING
```

```

CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
SEARCHARG..MBR..YMAX < YMIN) THEN 0
ELSE CHECKDUPLICATE(LEVEL, GX, GY,
XMIN, YMIN, XMAX, YMAX,
SEARCHARG..MBR..XMIN,
SEARCHARG..MBR..YMIN,
SEARCHARG..MBR..XMAX,
SEARCHARG..MBR..YMAX,
LEVELS)

END
WHEN SEARCHSECONDBYFIRST (SEARCHARG SHAPE)
RANGE THROUGH GRIDRANGE(SEARCHARG..MBR..XMIN,
SEARCHARG..MBR..YMIN,
SEARCHARG..MBR..XMAX,
SEARCHARG..MBR..YMAX,
LEVELS)

FILTER USING
CASE WHEN (SEARCHARG..MBR..YMIN > YMAX) OR
SEARCHARG..MBR..YMAX < YMIN) THEN 0
ELSE MBROVERLAP(XMIN, YMIN, XMAX, YMAX,
SEARCHARG..MBR..XMIN,
SEARCHARG..MBR..YMIN,
SEARCHARG..MBR..XMAX,
SEARCHARG..MBR..YMAX)

END

```

CREATE METHOD

CREATE METHOD ステートメントは、すでにユーザー定義の構造化タイプの定義の一部となっているメソッド指定に、メソッド本体を関連付けるために使用されます。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- CREATE METHOD ステートメントで参照される構造化タイプのスキーマに対する CREATEIN 特権
- CREATE METHOD ステートメントで参照される構造化タイプの所有者
- SYSADM または DBADM 権限

外部メソッド本体をそのメソッド指定に関連付けるためには、ステートメントの許可 ID の特権に、以下の少なくとも 1 つが含まれている必要もあります。

- データベースに対する CREATE_EXTERNAL_ROUTINE 権限
- SYSADM または DBADM 権限

CREATE METHOD

SQL メソッドを作成する場合、全選択で識別される表、ビュー、またはニックネームのそれぞれに対して、ステートメントの許可 ID に以下の特権が含まれている必要があります。

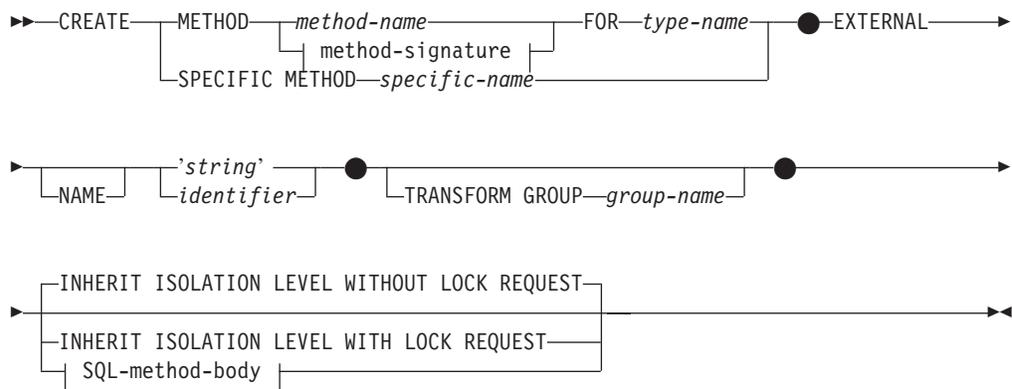
- その表、ビュー、またはニックネームに対する CONTROL 特権、または
- その表、ビュー、またはニックネームに対する SELECT 特権

SQL メソッドの定義者が SYSADM 権限を持つために、メソッドの作成しかできない場合、その定義者にメソッド作成のための DBADM 権限が暗黙的に付与されます。

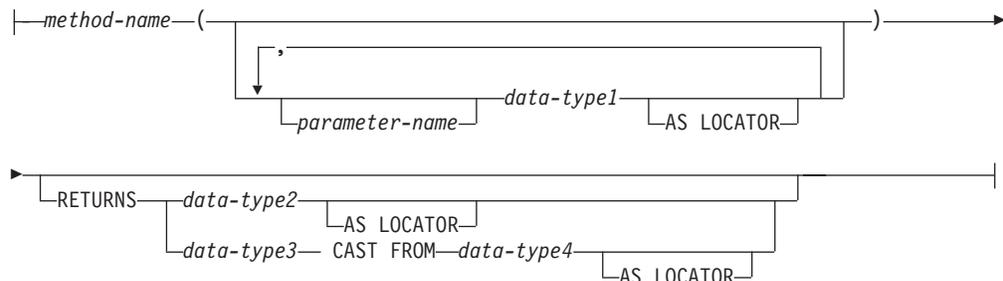
PUBLIC 以外のグループ特権は、CREATE METHOD ステートメントで指定された表やビューに対しては考慮されません。

このニックネームで示されている表またはビューのデータ・ソースの許可要件は、メソッドが呼び出される時に適用されます。接続の許可 ID は、別のリモート許可 ID へマップできます。

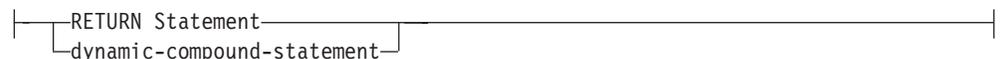
構文



method-signature:



SQL-method-body:



説明

METHOD

ユーザー定義の構造化タイプに関連付けられる既存のメソッド指定を識別します。メソッド指定は、以下のいずれかの方法で識別できます。

method-name

メソッド本体に対して定義するメソッド指定の名前を指定します。暗黙的スキーマは、サブジェクト・タイプ (*type-name*) のスキーマです。この *method-name* のある *type-name* には、1 つしかメソッドを指定できません (SQLSTATE 42725)。

method-signature

定義するメソッドを一意的に識別できるメソッド・シグニチャーを指定します。このメソッド・シグニチャーは、CREATE TYPE または ALTER TYPE ステートメントで提供されたメソッド指定と一致しなければなりません (SQLSTATE 42883)。

method-name

メソッド本体に対して定義するメソッド指定の名前を指定します。暗黙的スキーマは、サブジェクト・タイプ (*type-name*) のスキーマです。

parameter-name

パラメーター名を指定します。パラメーター名がメソッド・シグニチャーにより提供される場合、これらは適合するメソッド指定の対応する部分と全く同じでなければなりません。このステートメントでは、文書化だけのためにパラメーター名がサポートされています。

data-type1

各パラメーターのデータ・タイプを指定します。配列タイプはサポートされません (SQLSTATE 42815)。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。

RETURNS

この節は、メソッドの出力を指定します。RETURNS 節がメソッド・シグニチャーにより提供される場合、これは CREATE TYPE の対応するメソッド指定の対応する部分と全く同じでなければなりません。このステートメントでは、文書化だけのために RETURN 節がサポートされています。

data-type2

出力のデータ・タイプを指定します。配列タイプはサポートされません (SQLSTATE 42815)。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。これは、実際の値の代わりに LOB ロケーターが、メソッドにより戻されることを指定します。

CREATE METHOD

data-type3 **CAST FROM** *data-type4*

この形式の RETURNS 節は、関数コードから戻されたデータ・タイプとは異なるデータ・タイプを、呼び出しステートメントに戻すのに使用されます。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を使用して、LOB ロケーターが実際の値の代わりにメソッドから戻されるように指定できます。

FOR *type-name*

指定されたメソッドを関連付けるタイプを指定します。この名前は、カタログにすでに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

SPECIFIC METHOD *specific-name*

CREATE TYPE 時に指定されたか、デフォルト値として与えられた値を使用して、特定のメソッドを識別します。specific-name は、指定したスキーマまたは暗黙のスキーマのメソッド指定を識別しなければなりません。そうでない場合、エラーになります (SQLSTATE 42704)。

EXTERNAL

この節は、この CREATE METHOD ステートメントを使用して登録するメソッドが、外部プログラミング言語で作成されたコードに基づいており、文書化されたリンケージの規則とインターフェースに従っていることを示します。

CREATE TYPE で適合するメソッド指定は、SQL 以外の LANGUAGE を指定しなければなりません。このメソッドが呼び出されると、メソッドのサブジェクトが、暗黙の最初のパラメーターとしてインプリメンテーションに渡されます。

NAME 節の指定がない場合、"NAME *method-name*" が想定されます。

名前

この節は、定義するメソッドをインプリメントするユーザー作成コードの名前を指定します。

'string'

'string' オプションは、最大 254 バイトのストリング定数です。ストリングに使用される形式は、指定した LANGUAGE によって異なります。特定の言語規則に関する詳細は、『CREATE FUNCTION (外部スカラー) ステートメント』を参照してください。

identifier

指定する identifier は SQL ID です。SQL ID は、ストリングの library-id として使用されます。区切られた ID でない場合、ID は大文字に変換されます。ID がスキーマ名で修飾されている場合、スキーマ名の部分は無視されます。この形式の NAME は、LANGUAGE C でのみ使用可能です (CREATE TYPE のメソッド指定で定義)。

TRANSFORM GROUP *group-name*

メソッドを呼び出す際のユーザー定義の構造化タイプのトランスフォーメーション

ンに使用するトランスフォーム・グループを指定します。メソッド定義には、ユーザー定義の構造化タイプが含まれているため、トランスフォームが必要です。

ここで、トランスフォーム・グループ名を指定することを強くお勧めします。この節が指定されない場合、使用されるデフォルトのグループ名は DB2_FUNCTION です。参照された構造化タイプに、指定した (またはデフォルトの) グループ名が定義されていない場合には、エラーになります (SQLSTATE 42741)。同様に、指定したグループ名または構造化タイプに、必須の FROM SQL または TO SQL トランスフォーム関数が定義されていない場合には、エラーになります (SQLSTATE 42744)。

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST または INHERIT ISOLATION LEVEL WITH LOCK REQUEST

メソッドが呼び出し元のステートメントの分離レベルを継承している場合に、ロック要求をステートメントの分離レベルに関連付けることができるかどうかを指定します。デフォルトは INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST です。

INHERIT ISOLATION LEVEL WITHOUT LOCK REQUEST

メソッドが呼び出し元のステートメントの分離レベルを継承している場合に、指定した isolation-clause (分離節)の一部として lock-request-clause (ロック要求節) が含まれている SQL ステートメントのコンテキストでそのメソッドを呼び出すことができません (SQLSTATE 42601)。

INHERIT ISOLATION LEVEL WITH LOCK REQUEST

メソッドが呼び出し元のステートメントの分離レベルを継承している場合に、指定した lock-request-clause (ロック要求節) をメソッドが継承することを指定します。

SQL-method-body

CREATE TYPE のメソッド仕様が LANGUAGE SQL の場合、SQL-method-body は、どのようにメソッドがインプリメントされるかを定義します。

SQL-method-body は、以下のメソッド仕様のパーツに従っていなければなりません。

- DETERMINISTIC または NOT DETERMINISTIC (SQLSTATE 428C2)
- EXTERNAL ACTION または NO EXTERNAL ACTION (SQLSTATE 428C2)
- CONTAINS SQL または READS SQL DATA (SQLSTATE 42985)

パラメーター名を SQL-method-body で参照することができます。メソッドのサブジェクトは、暗黙的な最初のパラメーター SELF としてメソッド・インプリメンテーションに渡されます。

詳細については、『コンパウンド SQL (動的) ステートメント』および『RETURN ステートメント』を参照してください。

規則

- CREATE TYPE または ALTER TYPE ステートメントを使用して、前もってメソッド指定を定義していなければ、CREATE METHOD は使用できません (SQLSTATE 42723)。
- 作成されるメソッドがオーバーライド・メソッドの場合には、以下のメソッドに従属するパッケージは無効になります。

CREATE METHOD

- オリジナル・メソッド
 - 作成されるスーパータイプのメソッドをサブジェクトとして持つ、他のオーバーライド・メソッド
- XML データ・タイプは、メソッド内で使用できません。

注

- メソッドが SQL を許可する場合、外部プログラムは、フェデレーテッド・オブジェクトへのアクセスを試行してはなりません (SQLSTATE 55047)。
- **特権**

メソッドの定義者は、メソッドに対する EXECUTE 特権と、メソッドをドロップする権利を常に与えられます。

EXTERNAL メソッドが作成されると、メソッドの定義者は WITH GRANT OPTION 付きの EXECUTE 特権を常に受け取ります。

SQL メソッドが作成されると、メソッドの定義者がメソッドを定義するために必要なすべての特権に対して WITH GRANT OPTION を持っている場合、または定義者が SYSADM や DBADM 権限を持っている場合には、メソッドに対する WITH GRANT OPTION 付きの EXECUTE 特権のみが定義者に与えられます。SQL メソッドの定義者にそれらの特権が与えられるのは、それらの特権の派生元の特権がメソッドの作成時に存在している場合に限りです。定義者は、これらの特権を直接持っているか、または PUBLIC の特権として持っていることが必要です。メソッドの定義者がメンバーであるグループを持つ特権は考慮されません。メソッドを使用する場合、接続済みのユーザーの許可 ID には、そのデータ・ソースでニックネームが参照する表またはビューに対する適切な特権がなければなりません。

- **表アクセスの制限**

メソッドが READS SQL DATA として定義されている場合は、メソッド内のステートメントは、このメソッドを呼び出したステートメントによって変更される表にアクセスすることはできません (SQLSTATE 57053)。

例

例 1:

```
CREATE METHOD BONUS (RATE DOUBLE)
FOR EMP
RETURN SELF..SALARY * RATE
```

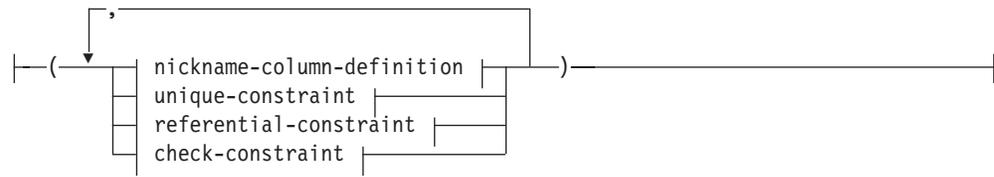
例 2:

```
CREATE METHOD SAMEZIP (addr address_t)
RETURNS INTEGER
FOR address_t
RETURN
(CASE
WHEN (self..zip = addr..zip)
THEN 1
ELSE 0
END)
```

例 3:

CREATE NICKNAME

nickname-column-list:



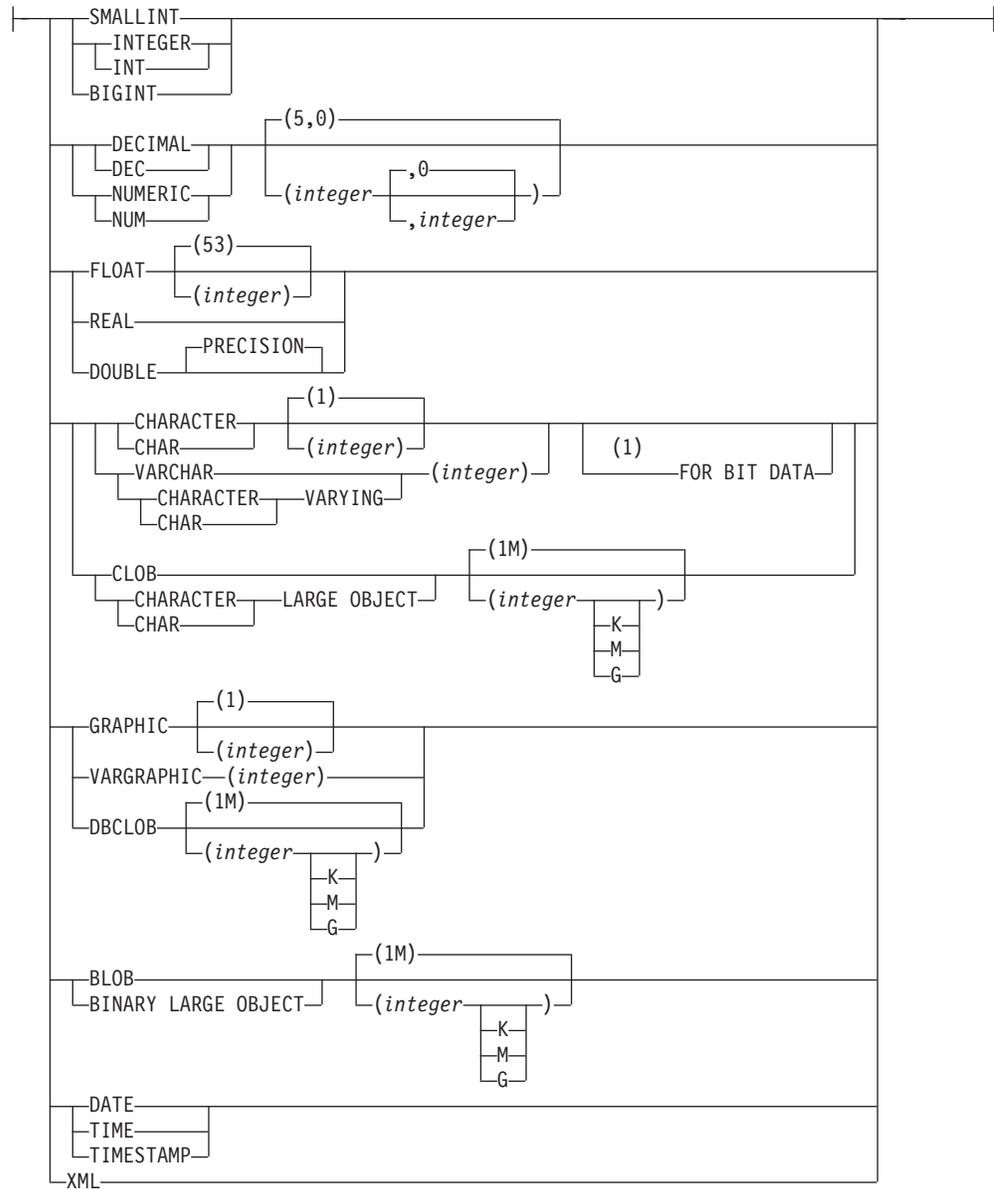
nickname-column-definition:



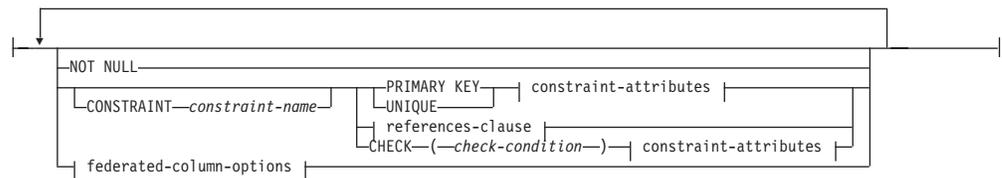
local-data-type:



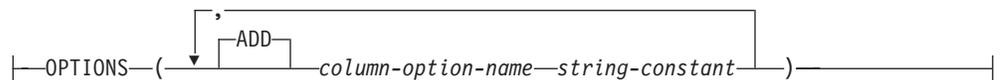
built-in-type:



nickname-column-options:

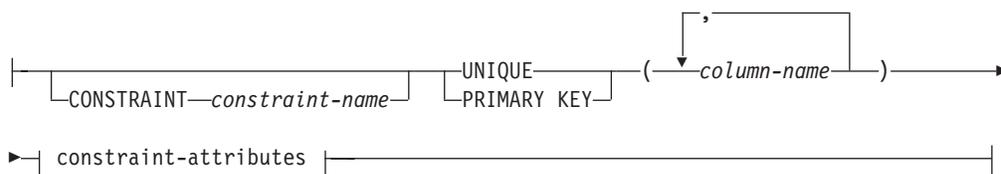


federated-column-options:



CREATE NICKNAME

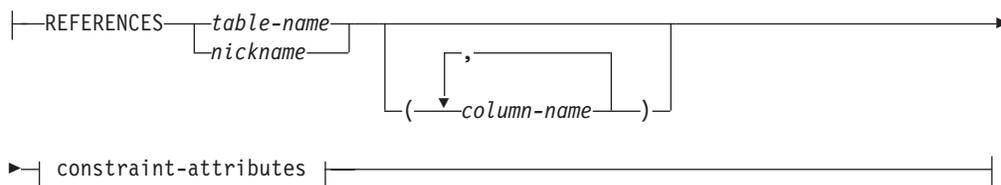
unique-constraint:



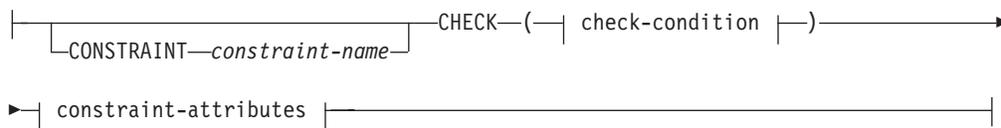
referential-constraint:



references-clause:



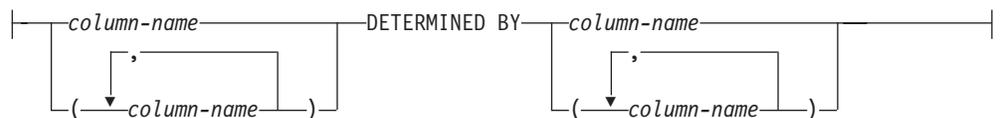
check-constraint:

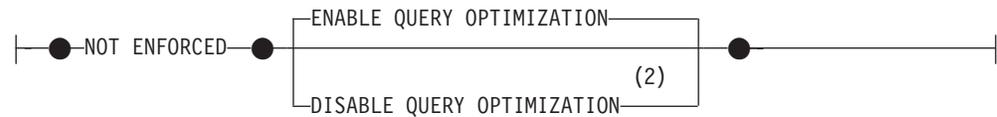


check-condition:



functional-dependency:



constraint-attributes:**注:**

- 1 FOR BIT DATA 節とその後が続く他の列制約とは、任意の順序で指定できます。
- 2 DISABLE QUERY OPTIMIZATION はユニーク制約または主キー制約をサポートしていません。

説明*nickname*

ニックネーム、つまりデータ・ソース・オブジェクト用にフェデレーテッド・サーバーが使用する ID を指定します。暗黙または明示の修飾子を含むニックネームは、カタログに記述されている表、ビュー、ニックネーム、または別名を指定するものであってはなりません。データ・ソース・オブジェクトを DB2 別名にすることはできません。スキーマ名を、'SYS' で始めることはできません (SQLSTATE 42939)。

FOR remote-object-name

ID を指定します。スキーマ名をサポートするデータ・ソースの場合、これは *data-source-name.remote-schema-name.remote-table-name* という形式の 3 つの部分からなる ID です。スキーマ名をサポートしないデータ・ソースの場合、これは *data-source-name.remote-table-name* という形式の 2 つの部分からなる ID です。

data-source-name

ニックネームを作成する対象となる表またはビューを含むデータ・ソースを指定します。 *data-source-name* は、CREATE SERVER ステートメント内の *server-name* に割り当てられた名前と同じです。

remote-schema-name

表またはビューが属するスキーマを指定します。リモート・スキーマ名に特殊文字や小文字が含まれる場合には、二重引用符で囲まなければなりません。

remote-table-name

ニックネームを作成している、特定のデータ・ソース・オブジェクト (表またはビューなど) の名前。表を宣言済み一時表 (SQLSTATE 42995) にすることはできません。リモート表名に特殊文字や小文字が含まれる場合には、二重引用符で囲まなければなりません。

non-relational-data-definition

非リレーショナル・ラッパーを通してアクセスするデータを定義します。

nickname-column-definition

ニックネームの列のローカル属性を定義します。一部のラッパーは属性を指定する必要がありますが、他のラッパーはデータ・ソースから属性を判別できます。

CREATE NICKNAME

column-name

列のローカル名を指定します。この名前は、対応する列の *remote-object-name* とは異なる可能性があります。

local-data-type

列のローカル・データ・タイプを指定します。一部のラッパーは、SQL データ・タイプのサブセットのみをサポートします。特定のデータ・タイプの説明については、『CREATE TABLE』を参照してください。

nickname-column-options

ニックネームの列に関連した追加オプションを指定します。

NOT NULL

列で NULL 値を許可しないように指定します。

CONSTRAINT *constraint-name*

制約の名前を指定します。 *constraint-name* (制約名) は、同じ CREATE NICKNAME ステートメントにすでに指定されている制約を指定するものであってはなりません (SQLSTATE 42710)。

この節が省略された場合は、ニックネームに定義されている既存の制約の ID の中でユニークな 18 バイトの長さの ID がシステムによって生成されます。(ID は、'SQL' と、タイム・スタンプ関数によって生成される一連の 15 の数字で構成されます。)

PRIMARY KEY 制約またはユニーク制約とともに使用した場合、この *constraint-name* は、制約をサポートするために作成される SPECIFICATION ONLY 指定の索引の名前として使用されます。

PRIMARY KEY

これは、1 つの列からなる主キーを定義する簡単な方法です。つまり、PRIMARY KEY が列 C の定義で指定されている場合、その効果は、PRIMARY KEY(C) 節を独立した節として指定する場合と同じです。

下記の *unique-constraint* の『PRIMARY KEY』を参照してください。

UNIQUE

これは、1 つの列からなるユニーク・キーを定義する簡単な方法です。すなわち、UNIQUE を列 C の定義に指定すると、UNIQUE(C) 節を独立した節として指定した場合と同じ結果になります。

下記の *unique-constraint* の『UNIQUE』を参照してください。

references-clause

これは、1 つの列からなる外部キーを定義する簡単な方法です。つまり、*references-clause* が列 C の定義に指定されている場合、その効果は、列として C しか指定されていない FOREIGN KEY 節の一部として *references-clause* が指定された場合と同じになります。

後述の *referential-constraint* の *references-clause* の項を参照してください。

CHECK (*check-condition*)

これは、1 つの列に適用されるチェック制約を定義する簡単な方法です。後述の **CHECK** (*check-condition*) を参照してください。

OPTIONS

ニックネームを作成したときに追加される列オプションを指示します。特定の列オプションを指定する必要のあるラッパーもあります。

ADD

列オプションを追加します。

column-option-name

オプションの名前を指定します。

string-constant

column-option-name の設定を、文字ストリング定数として指定します。

unique-constraint

ユニーク制約または主キー制約を定義します。

CONSTRAINT *constraint-name*

主キー制約、またはユニーク制約の名前を指定します。

UNIQUE (*column-name,...*)

指定した列で構成されるユニーク・キーを定義します。指定する列は NOT NULL として定義されていなければなりません。各 *column-name* (列名) は、ニックネームの列を指定するものでなければなりません。また、同じ列を複数回指定することはできません。

指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。保管される列の長さについては、『CREATE TABLE』の『バイト・カウント』を参照してください。キーの長さの制限については、『SQL と XQuery の制限値』を参照してください。列の長さ属性がページ・サイズに対する索引キーの長さの上限を超えない場合でも、LOB 列、LOB に基づく特殊タイプの列、構造化タイプの列は、ユニーク・キーの一部として使用できません (SQLSTATE 54008)。

ユニーク・キーの列セットは、主キーまたは他のユニーク・キーの列セットと同じにすることはできません (SQLSTATE 01543)。 (LANGLEVEL が SQL92E または MIA の場合は、エラーが戻されます。SQLSTATE 42891)

カタログに記録されているニックネームの記述には、ユニーク・キーとその SPECIFICATION ONLY 指定の索引が含まれます。 SPECIFICATION ONLY 指定の索引は、それぞれの列について昇順に指定された順序で、列に対して自動的に作成されます。 SPECIFICATION ONLY 指定の索引の名前は、ニックネームの作成時にスキーマに存在する既存の索引または SPECIFICATION ONLY 指定の索引と競合しない場合、*constraint-name* (制約名) と同じになります。 SPECIFICATION ONLY 指定の索引名が競合する場合は、名前は SQL の後に文字のタイム・スタンプ (yyymmddhhmmssxxx) が続き、スキーマ名として SYSIBM を伴う名前になります。

PRIMARY KEY (*column-name,...*)

指定された列で構成される主キーを定義します。この節を複数回指定することはできず、指定する列は **NOT NULL** として定義されていなければなりません。各 *column-name* (列名) は、ニックネームの列を指定していなければなりません。また、同じ列を複数回指定することはできません。

指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。保管される列の長さについては、『CREATE TABLE』の『バイト・カウント』を参照してください。キーの長さの制限については、『SQL と XQuery の制限値』を参照してください。列の長さ属性がページ・サイズに対する索引キーの長さの上限を超えない場合でも、LOB 列、LOB に基づく特殊タイプの列、構造化タイプの列は、主キーの一部として使用できません (SQLSTATE 54008)。

主キーの列セットは、ユニーク・キーの列セットと同じであってはなりません (SQLSTATE 01543)。 (LANGLEVEL が SQL92E または MIA の場合は、エラーが戻されます。SQLSTATE 42891)

1 つのニックネームには、主キーを 1 つだけ定義することができます。

カタログに記録されているニックネームの記述には、主キーとその SPECIFICATION ONLY 指定の索引が含まれます。SPECIFICATION ONLY 指定の索引は、それぞれの列について昇順に指定された順序で、列に対して自動的に作成されます。SPECIFICATION ONLY 指定の索引の名前は、ニックネームの作成時にスキーマに存在する既存の索引または SPECIFICATION ONLY 指定の索引と競合しない場合、*constraint-name* (制約名) と同じになります。SPECIFICATION ONLY 指定の索引名が競合する場合は、名前は SQL の後に文字のタイム・スタンプ (*yymmddhhmmssxxx*) が続き、スキーマ名として SYSIBM を伴う名前になります。

referential-constraint

参照制約を定義します。

CONSTRAINT *constraint-name*

参照制約の名前を指定します。

FOREIGN KEY (*column-name,...*)

指定した *constraint-name* (制約名) の参照制約を定義します。

N1 を、ステートメントの対象となるニックネームであると想定します。参照制約の外部キーは、指定された列で構成されます。列名リストの各名前は、N1 の列を指定していなければならず、同じ列を複数回指定することはできません。

指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。保管される列の長さについては、『CREATE TABLE』の『バイト・カウント』を参照してください。キーの長さの制限については、『SQL と XQuery の制限値』を参照してください。外部キーは、255 バイトよりも大きい長さの可変長列で定義できます。LOB 列、LOB に基づく特殊タイプの列、構造化タイプの列は、外部キーの一部として使用できません (SQLSTATE 42962)。外部キーの列の数は、親キーの列の数と同じでなければならず、対応する列のデータ・タイプは互換性があることが必要です (SQLSTATE

42830)。2つの列の記述は、それらの列が互換性のあるデータ・タイプ (両方の列が数字、文字ストリング、GRAPHIC、日時であるか、または同じ特殊タイプ) であれば互換性があります。

references-clause

参照制約の親表または親ニックネーム、および親キーを指定します。

REFERENCES *table-name* または *nickname*

REFERENCE 節に指定される表またはニックネームは、カタログに記述された基本表またはニックネームを識別している必要がありますが、カタログ表を示すものであってはなりません。

参照制約の外部キー、親キー、および親表または親ニックネームが、以前に指定した参照制約の外部キー、親キー、および親表または親ニックネームと同じである場合、参照制約は重複しています。重複した参照制約は無視され、警告が戻されます (SQLSTATE 01543)。

以下の説明では、N2 は指定した親表または親ニックネームを示し、N1 は作成する (または変更する) ニックネームを示します。N1 と N2 は同じニックネームです。

指定された外部キーの列の数は、N2 の親キーと同じ数でなければなりません。また、外部キーの *n* 番目の列の記述は、その親キーの *n* 番目の列の記述と互換性がなければなりません。この規則において、日時の列はストリング列と互換性があるとは見なされません。

FOREIGN KEY 節で指定される参照制約は、N2 が親であり、N1 が従属であるリレーションシップを定義します。

(*column-name*,...)

参照制約の親キーは、指定された列で構成されます。各 *column-name* は、N2 の列を指定する非修飾名でなければなりません。同じ列を複数回指定することはできません。

列名のリストは、主キーまたは N2 に存在するユニーク制約の列セットと一致している (順序は任意) 必要があります (SQLSTATE 42890)。列名のリストの指定がない場合、N2 に主キーがある必要があります (SQLSTATE 42888)。列名リストを省略すると、指定されているとおりの順序でその主キーの列が暗黙に指定されます。

constraint-attributes

参照整合性またはチェック制約に関連付けられた属性を定義します。

NOT ENFORCED

この制約は、挿入、更新、削除などの通常の操作中にデータベース・マネージャーによって課せられません。

ENABLE QUERY OPTIMIZATION

制約は真であると想定され、適切な状況下では照会最適化に使用することができます。

DISABLE QUERY OPTIMIZATION

制約を照会の最適化に使用できません。

CREATE NICKNAME

check-constraint

チェック制約を定義します。 *check-constraint* (チェック制約) は、偽以外に評価されなければならない、または列間の機能従属関係を定義する *search-condition* (検索条件) です。

CONSTRAINT *constraint-name*

チェック制約の名前を指定します。

CHECK (*check-condition*)

チェック制約を定義します。 *check-condition* (チェック条件) はニックネームのすべての行について、真または不明でなければなりません。

search-condition

search-condition には、以下の制限があります。

- 列参照は作成するニックネームの列に対する参照でなければなりません。
- *search-condition* に TYPE 述部を入れることはできません。
- これには、以下のどれも入れることはできません (SQLSTATE 42621)。
 - 副照会
 - 間接参照操作または、有効範囲をもつ参照引数がオブジェクト ID (OID) 列以外の列である Deref 関数
 - SCOPE 節をもつ CAST 指定
 - 列関数
 - deterministic 関数でない関数
 - 外部アクションをもつと定義された関数
 - CONTAINS SQL または READS SQL DATA のいずれかによって定義されたユーザー定義関数
 - ホスト変数
 - パラメーター・マーカー
 - 特殊レジスター
 - グローバル変数
 - ID 列以外の生成列の参照

functional-dependency

列間の機能従属関係を定義します。

列の親セットには、DETERMINED BY 節の直前に来る指定された列が含まれます。列の子セットには、DETERMINED BY 節の直後に来る指定された列が含まれます。 *search-condition* の制約事項すべては、親セット列と子セット列に適用され、列のセットには単純な列参照のみが許可されています (SQLSTATE 42621)。機能従属関係に同じ列を複数回指定することはできません (SQLSTATE 42709)。列のデータ・タイプを LOB データ・タイプ、LOB データ・タイプに基づいた特殊タイプ、または構造化タイプにすることはできません (SQLSTATE 42962)。列の子セットの列を NULL 可能列にすることはできません (SQLSTATE 42621)。

column-definition の一部としてチェック制約を指定する場合、その同じ列に対してのみ列参照を行うことができます。ニックネーム定義の一部として指定されたチェック制約には、それ以前に CREATE NICKNAME ステートメントで定義されている列を指定する列参照を含めることができます。チェック制約の矛盾、重複条件、または同等条件については検査されません。したがって、矛盾したチェック制約や冗長なチェック制約が定義可能であるため、実行時にエラーになる可能性があります。

FOR SERVER *server-name*

CREATE SERVER ステートメントを使用して登録されたサーバーを指定します。このサーバーは、ニックネームのデータにアクセスするのに使用されます。

OPTIONS

ニックネームを作成したときに使用可能にされるニックネーム・オプションを指示します。

ADD

ニックネーム・オプションを追加します。

nickname-option-name

オプションの名前を指定します。

string-constant

nickname-option-name の設定を、文字ストリング定数として指定します。

注

- リレーショナル・データ・ソース・オブジェクトの例は、表とビューです。非リレーショナル・データ・ソース・オブジェクトの例は、Documentum オブジェクトまたは登録済み表、テキスト・ファイル (.txt)、BLAST 検索の実行対象にできるオブジェクト、および Microsoft Excel ファイル (.xls) です。
- ニックネームで示されているデータ・ソース・オブジェクトは、*remote-object-name* の最初の修飾子によって示されているデータ・ソースに存在していなければなりません。
- サポートされるデータ・ソース・データ・タイプのリストは、ラッパーごとに異なります。XML、DECFLOAT、構造化タイプ、および REF タイプなどの DB2 データ・タイプに対応するデータ・ソースのデータ・タイプは、どのラッパーによってもサポートされていません。CREATE NICKNAME ステートメントに、サポートされないデータ・タイプの列を持つ *remote-object-name* を指定すると、エラーが戻されます。

LONG VARCHAR および LONG VARGRAPHIC データ・ソースのデータ・タイプは、CLOB および DBCLOB データ・タイプにそれぞれマップされます。

LONG VARCHAR FOR BIT DATA は BLOB にマップされます。

- DB2 索引名に許可されている最大長は 128 バイトです。索引の名前がこれより長いリレーショナル表にニックネームを作成する場合、その名前の全体のカタログが作成されることはありません。DB2 側で 128 バイトに切り捨てます。そのような文字で構成されているストリングが、索引の属するスキーマ内でユニークでない場合、DB2 は最後の文字を 0 に置き換えてユニークなストリングにしようとします。その結果もユニークでない場合は、DB2 は最後の文字を 1 に変えます。DB2 はこのプロセスを 2 から 9 までの数字を使って続けます。必要であれば、名前の 127 番目の文字を 0 から 9 まで、さらに 126 番目の文字を 0

CREATE NICKNAME

から 9 まで、というように、ユニークな名前が生成されるまで繰り返していきます。例えば、データ・ソース表の索引の 130 バイトの名前を AREALLY...REALLYLONGNAME とします。この索引が属するスキーマ内に、AREALLY...REALLYLONGNA および AREALLY...REALLYLONGN0 という名前が既に存在します。新しい名前は 128 バイトを超過してしまうため、DB2 側で AREALLY...REALLYLONGNA に切り捨てます。この名前はすでにスキーマ内に存在しているため、DB2 は切り捨てた名前を AREALLY...REALLYLONGN0 に変更します。この名前も存在しているため、DB2 は切り捨てた名前を AREALLY...REALLYLONGN1 に変えます。スキーマ内にはこの名前は存在しないので、DB2 はこの名前を新しい名前として受け入れます。

- データ・ソース・オブジェクトにニックネームを作成すると、DB2 はニックネーム列の名前をカタログに保管します。データ・ソース・オブジェクトが表またはビューの場合には、DB2 はニックネーム列名を表またはビュー列名と同じにします。この名前の長さが DB2 列名に許可されている最大長を超える場合、DB2 は名前をこの長さに切り捨てます。切り捨てられた名前が、表またはビューにある他の列の名前と同じでユニークでない場合、DB2 は前の段落で説明されている手順に従い、名前をユニークなものに変更します。
- データ・ソース・オブジェクトに索引が定義されている場合、ニックネームが作成されるときに索引ごとの SPECIFICATION ONLY 指定の索引が作成されます。次のような索引に対しては、データ・ソースに SPECIFICATION ONLY 指定の索引が作成されません。
 - 列名が重複している
 - 列の数が 64 より多い
 - 索引キー部分の長さが合計 1024 バイトを超える
- リモート・データ・ソース・オブジェクトの定義が変更される場合 (例えば、列の削除やデータ・タイプの変更)、ニックネームをドロップしてから再作成しなければなりません。そうしないと、ニックネームを SQL ステートメント内で使用するとき、エラーが戻される場合があります。

例

例 1: HEDGES というスキーマにある、DEPARTMENT というビューのニックネームを作成します。このビューは、OS390A という DB2 for z/OS のデータ・ソースに保管されます。

```
CREATE NICKNAME DEPT
FOR OS390A.HEDGES.DEPARTMENT
```

例 2: 例 1 でニックネームを作成したときのビューから、すべてのレコードを選択します。ビューのニックネームによってビューを参照する必要があります。リモート・ビューは、パススルー・セッション時に限ってデータ・ソースで認識されるときの名前を使用して参照できます。

```
SELECT * FROM DEPT (ニックネーム DEPT が作成された後に有効)
```

```
SELECT * FROM OS390A.HEDGES.DEPARTMENT (無効)
```

例 3: salesdata という名前のスキーマにあるリモート表 JAPAN のニックネームを作成します。データ・ソース上のスキーマ名と表名は小文字で保管されるため、リモート・スキーマ名と表名には二重引用符を付けて指定します。

```
CREATE NICKNAME JPSALES
FOR asia."salesdata"."japan"
```

例 4: 表構造のファイル DRUGDATA1.TXT のニックネームを作成します。ステートメントに FILE_PATH、COLUMN DELIMITER、KEY_COLUMN、および VALIDATE_DATA_FILE ニックネーム・オプションを組み込みます。

```
CREATE NICKNAME DRUGDATA1
(Dcode      INTEGER,
DRUG        CHAR(20),
MANUFACTURER CHAR(20))
FOR SERVER biochem_lab
OPTIONS
(FILE_PATH  '/usr/pat/DRUGDATA1.TXT',
COLUMN_DELIMITER ',',
KEY_COLUMN  'DCODE',
SORTED     'Y',
VALIDATE_DATA_FILE 'Y')
```

例 5: 指定したディレクトリー・パス /home/db2user にある複数の XML ファイルに対して親ニックネーム CUSTOMERS を作成します。以下のオプションを組み込みます。

• 列オプション:

- ID という名前の VARCHAR(5) 列に XPATH 列オプションを指定し、列データを抽出する XML ファイルのエレメントまたは属性を指示します。
- NAME という名前の VARCHAR(16) 列に XPATH 列オプションを指定し、列データを抽出する XML ファイルのエレメントまたは属性を指示します。
- ADDRESS という名前の VARCHAR(30) 列に XPATH 列オプションを指定し、列データを抽出する XML ファイルのエレメントまたは属性を指示します。
- CID という名前の VARCHAR(16) 列に PRIMARY_KEY 列オプションを指定し、ニックネームの階層で親ニックネームとなるカスタマー・ニックネームを指示します。

• ニックネーム・オプション:

- DIRECTORY_PATH ニックネーム・オプションは、データを提供する XML ファイルの場所を指示します。
- XPATH ニックネーム・オプションは、XML ファイル内でデータが始まるエレメントを指示します。
- STREAMING ニックネーム・オプションは、XML ソース・データがエレメントで区切られており、エレメントごとに処理されることを指示します。この例では、エレメントはカスタマー・レコードです。

```
CREATE NICKNAME customers
(id      VARCHAR(5)  OPTIONS(XPATH './@id'),
name    VARCHAR(16) OPTIONS(XPATH './name'),
address VARCHAR(30) OPTIONS(XPATH './address/@street'),
cid     VARCHAR(16) OPTIONS(PRIMARY_KEY 'YES'))
FOR SERVER xml_server
OPTIONS
(DIRECTORY_PATH '/home/db2user',
XPATH '//customer',
STREAMING 'YES')
```

CREATE PROCEDURE

CREATE PROCEDURE ステートメントは、現行サーバーでプロシージャを定義します。

このステートメントを使用して、3 種類のプロシージャを作成できます。それぞれの種類について、個々に説明していきます。

- 外部。このプロシージャの本体はプログラミング言語で書かれています。現行サーバーでさまざまな属性とともに定義されたプロシージャにより、外部実行可能ファイルが参照されます。
- ソース派生。このプロシージャの本体は、ソース・プロシージャの一部であり、ソース派生プロシージャでは、そのソース・プロシージャを参照します。ソース派生プロシージャは、現行サーバーで定義されており、ソース・プロシージャのさまざまな属性を引き継ぎます。ソース・プロシージャがデータ・ソースにあるソース派生プロシージャのことをフェデレーテッド・プロシージャともいいます。
- SQL。このプロシージャの本体は SQL で書かれており、現行サーバーで、プロシージャの様々な属性とともに定義されます。

CREATE PROCEDURE (外部)

CREATE PROCEDURE (外部) ステートメントは、現行サーバーで外部プロシージャを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する CREATE_EXTERNAL_ROUTINE 権限、および以下の少なくとも 1 つ。
 - データベースに対する IMPLICIT_SCHEMA 権限 (プロシージャのスキーマ名が既存のスキーマを指していない場合)
 - スキーマに対する CREATEIN 特権 (プロシージャのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

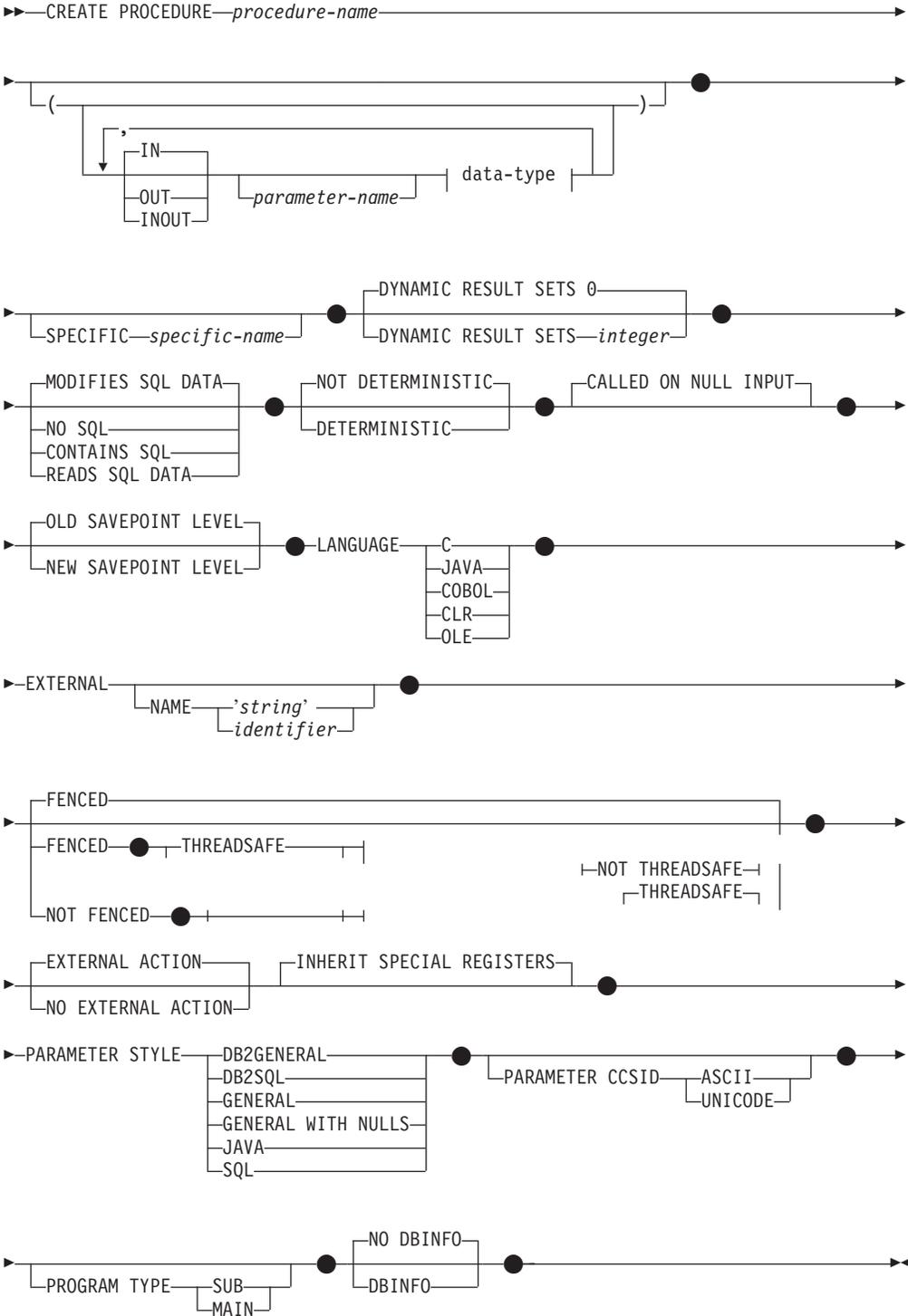
非 fenced のプロシージャを作成するには、ステートメントの許可 ID の特権に、以下の特権の少なくとも 1 つが含まれている必要があります。

- データベースに対する CREATE_NOT_FENCED_ROUTINE 権限
- SYSADM または DBADM 権限

CREATE PROCEDURE (外部)

fenced プロシージャを作成する場合、追加の権限や特権は必要ありません。

構文

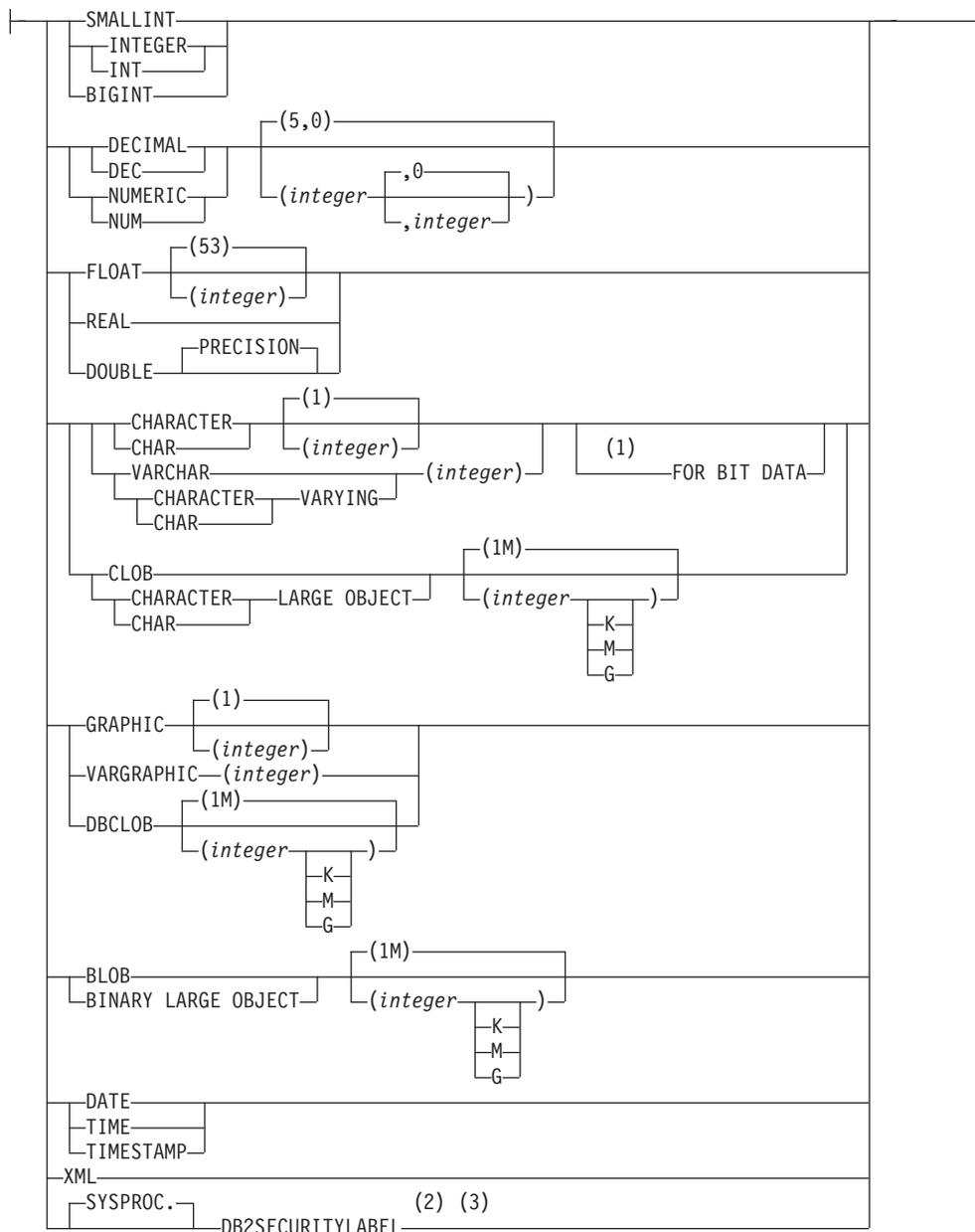


data-type:

CREATE PROCEDURE (外部)



built-in-type:



注:

- 1 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- 2 DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- 3 タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE

42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。

説明

procedure-name

定義するプロシージャの名前を指定します。この名前は、プロシージャを指定する修飾または非修飾の名前です。 *procedure-name* (プロシージャ名) の非修飾形式は SQL ID です (最大長 128)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

暗黙または明示の修飾子を含む名前と、パラメーターの数との組み合わせは、カタログにすでに記述されているプロシージャを指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数との組み合わせは、複数のスキーマ間でユニークである必要はありません。

2 つの部分からなる名前を指定する場合、‘SYS’ で始まる *schema-name* (スキーマ名) は使用できません (SQLSTATE 42939)。

(IN | OUT | INOUT *parameter-name data-type*,...)

プロシージャのパラメーターを指定し、各パラメーターのモード、データ・タイプ、およびオプション名を指定します。このリストには、プロシージャが預期する各パラメーターごとに 1 つの項目を指定する必要があります。

1 つのスキーマに同じ名前の 2 つのプロシージャがある場合、パラメーターの数をまったく同一にすることはできません。シグニチャーが重複していると、SQL エラー (SQLSTATE 42723) を戻します。

例えば、以下のステートメントの場合、

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

2 番目のステートメントは失敗します。その理由は、データ・タイプが異なってもプロシージャのパラメーターの数が同じだからです。

プロシージャによってエラーが戻される場合、OUT パラメーターは未定義で、INOUT パラメーターは未変更です。

IN パラメーターをプロシージャの入力パラメーターとして指定します。プロシージャ内でパラメーターに加えられるすべての変更は、制御が戻されると SQL アプリケーションの呼び出しは行えなくなります。デフォルトは IN です。

OUT

パラメーターをプロシージャの出力パラメーターとして指定します。

INOUT

パラメーターを、プロシージャの入力および出力パラメーターの両方として指定します。

CREATE PROCEDURE (外部)

parameter-name

パラメーターの名前を任意に指定します。パラメーター名は、プロシージャでユニークでなければなりません (SQLSTATE 42734)。

data-type

パラメーターのデータ・タイプを指定します。

- CREATE TABLE ステートメントのデータ・タイプの定義に指定することが可能で、プロシージャの作成に使用されている言語に対応するものがある SQL データ・タイプ指定と省略形を指定できます。
- ユーザー定義データ・タイプはサポートされていません (SQLSTATE 42601)。
- LONG VARCHAR および LONG VARGRAPHIC は、外部プロシージャのパラメーター・タイプとしてサポートされていません。
- XML は、LANGUAGE OLE では無効です。
- プロシージャ内での XML 値の表現は、プロシージャ呼び出しでパラメーターとして渡される XML 値をシリアライズしたバージョンなので、タイプ XML のパラメーターは構文 XML AS CLOB(*n*) を使用して宣言する必要があります。
- CLR は 28 より大きい DECIMAL スケールをサポートしていません (SQLSTATE 42613)。
- 10 進浮動小数点数は、言語 C、Java COBOL、CLR、および OLE ではサポートされていません (SQLSTATE 42613)。
- 配列タイプのパラメーターは、Java プロシージャのみでサポートされます (SQLSTATE 42815)。

SPECIFIC *specific-name*

定義するプロシージャのインスタンスに対するユニーク名を指定します。この特定名は、このプロシージャをドロップする場合、またはこのプロシージャにコメントを付ける場合に使用することができます。これは、プロシージャの呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 128)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別のルーチン・インスタンスを指定するものであってはなりません。そうでない場合、エラー (SQLSTATE 42710) になります。

specific-name は、既存の *procedure-name* と同じでも構いません。

修飾子を指定しない場合、*procedure-name* に使用された修飾子が使用されません。修飾子を指定する場合は、*procedure-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラー (SQLSTATE 42882) になります。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成される固有名は、'SQL' の後に文字タイム・スタンプが続く名前です ('SQLyyymmddhhmmssxxx')。

DYNAMIC RESULT SETS *integer*

プロシージャから戻される結果セットの上限の見積もりを指定します。

NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

プロシージャから SQL ステートメントが発行されるかどうかと、もし発行されればどのタイプかを示します。

NO SQL

プロシージャはどの SQL ステートメントも実行できないことを指示します (SQLSTATE 38001)。

CONTAINS SQL

SQL データの読み取りも変更も行わない SQL ステートメントを、プロシージャで実行できることを指定します (SQLSTATE 38004)。どのプロシージャでもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003)。

READS SQL DATA

SQL データを変更しない SQL ステートメントを、プロシージャで実行できることを指定します (SQLSTATE 38002 または 42985)。どのプロシージャでもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003)。

MODIFIES SQL DATA

このプロシージャは、プロシージャでサポートされていないステートメント以外のすべての SQL ステートメントを実行できることを指定します (SQLSTATE 38003)。

DETERMINISTIC または NOT DETERMINISTIC

この節は、同一の引数値に対してプロシージャが常に同じ結果を戻すか (DETERMINISTIC)、それとも状態値に依存してプロシージャの結果が影響を受けるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC が指定されたプロシージャは、連続で同じ入力を指定して呼び出した場合に常に同じ結果を戻します。

現在、この節はプロシージャの処理に影響を与えません。

CALLED ON NULL INPUT

CALLED ON NULL INPUT は、プロシージャに常に適用されます。これは、任意の引数が NULL かどうかにかかわらず、プロシージャが呼び出されることを意味します。OUT または INOUT パラメーターは、NULL 値を戻す場合も、通常の (NULL 以外の) 値を戻す場合もあります。NULL の引数値の有無のテストはプロシージャで行う必要があります。

OLD SAVEPOINT LEVEL または NEW SAVEPOINT LEVEL

このプロシージャが、セーブポイント名と影響について新しいセーブポイント・レベルを設定するかどうかを指定します。OLD SAVEPOINT LEVEL がデフォルトの動作です。セーブポイント・レベルについて詳しくは、SAVEPOINT ステートメントの説明にある『規則』のセクションを参照してください。

LANGUAGE

この節は必須で、プロシージャの本体が準拠している言語インターフェース規則を指定するのに使用されます。

C データベース・マネージャーは、プロシージャを C プロシージャであ

CREATE PROCEDURE (外部)

るかのように呼び出します。プロシージャは、標準 ANSI C プロトタイプで定義されている C 言語の呼び出し規則およびリンケージ規則に準拠していなければなりません。

JAVA

データベース・マネージャは、Java クラス内のメソッドとしてプロシージャを呼び出します。

COBOL

データベース・マネージャは、プロシージャを COBOL プロシージャであるかのように呼び出します。

CLR

データベース・マネージャは、.NET クラス内のメソッドとしてプロシージャを呼び出します。現在、LANGUAGE CLR は、Windows オペレーティング・システム上で実行するプロシージャでのみサポートされます。NOT FENCED は CLR ルーチンに指定できません (SQLSTATE 42601)。

OLE

データベース・マネージャは、OLE 自動化オブジェクトによって公開されたメソッドであるものとしてプロシージャを呼び出します。ストアード・プロシージャは、OLE 自動化データ・タイプと呼び出しメカニズムに準拠している必要があります。さらに OLE 自動化オブジェクトは、処理中サーバー (DLL) としてインプリメントされる必要もあります。これらの制約事項については、「*OLE Automation Programmer's Reference*」で説明されています。

LANGUAGE OLE は、Windows オペレーティング・システム用の DB2 で保管されたプロシージャに対してのみサポートされます。LANGUAGE OLE を指定したプロシージャには、THREADSAFE は指定できません (SQLSTATE 42613)。

EXTERNAL

この節は、この CREATE PROCEDURE ステートメントを使用して登録する新しいプロシージャが、外部プログラミング言語で作成されたコードに基づいており、文書化されたリンケージの規則とインターフェースに従っていることを示します。

NAME 節の指定がない場合、『NAME *procedure-name*』が想定されます。

NAME 節のフォーマットが正しくない場合、エラーが戻されます (SQLSTATE 42878)。

NAME 'string'

この節は、定義するプロシージャをインプリメントするユーザー作成コードの名前を指定します。

'string' オプションは、最大 254 バイトのストリング定数です。ストリングに使用される形式は、指定した LANGUAGE によって異なります。

- LANGUAGE C の場合:

指定する *string* は、ライブラリー名と作成しているプロシージャを実行するためにデータベース・マネージャが呼び出すそのライブラリー中のプロシージャです。ライブラリー (およびそのライブラリー中のプロシージャ) は、CREATE PROCEDURE ステートメントの実行時に存

在している必要はありません。ただし、プロシージャが呼び出される時点では、該当のライブラリーとそのライブラリー中の該当のプロシージャは存在していなければならない、またデータベース・サーバーのマシンからアクセス可能でなければなりません。

```

▶─'───┬───library_id───┬───'───▶
      │   └───┬───absolute_path_id───┬───┬───!───proc_id───┬───

```

名前は、単一引用符で囲む必要があります。余分なブランクを使用することはできません。

library_id

該当のプロシージャが入っているライブラリーの名前を指定します。データベース・マネージャーは、次のようにしてこのライブラリーを特定します。

- UNIX システムの場合、*library_id* が 'myfunc' と指定されており、データベース・マネージャーが /u/production から実行されていると、データベース・マネージャーは FENCED が指定されていればライブラリー /u/production/sql/lib/function/myproc で、NOT FENCED が指定されていればライブラリー /u/production/sql/lib/function/unfenced/myproc でプロシージャを特定します。
- Windows オペレーティング・システムの場合、データベース・マネージャーは LIBPATH または PATH 環境変数に指定されているディレクトリー・パスから関数を特定します。

これらのディレクトリーのいずれかに存在しているストアード・プロシージャは、登録済み属性を使用しません。

absolute_path_id

プロシージャの絶対パス名を指定します。

例えば、UNIX システムの場合、'/u/jchui/mylib/myproc' を指定すると、データベース・マネージャーは /u/jchui/mylib を調べて myproc プロシージャを探します。

Windows オペレーティング・システムの場合、'd:¥mylib¥myproc.dll' を指定すると、データベース・マネージャーは、d:¥mylib ディレクトリーからファイル myproc.dll をロードします。絶対パス ID がルーチン本体の識別に使用されている場合は、.dll 拡張子を必ず付加してください。

! proc_id

呼び出すプロシージャの入り口点の名前を指定します。感嘆符 (!) は、ライブラリー ID とプロシージャ ID との間の区切り文字です。'!proc8' を指定すると、データベース・マネージャーは、*absolute_path_id* によって指定された場所でライブラリーを探し、そのライブラリー内の入り口点 proc8 を使用します。

ストリングの形式が正しくない場合には、エラーが戻されます (SQLSTATE 42878)。

CREATE PROCEDURE (外部)

各プロシージャの本体は、マウントされてデータベースのすべてのパーティションで使用可能なディレクトリーに入っていないとできません。

- LANGUAGE JAVA の場合:

指定する *string* には、作成中のプロシージャを実行するためにデータベース・マネージャーが呼び出す、任意指定の jar ファイル、クラス ID、およびメソッド ID が含まれています。クラス ID とメソッド ID は、CREATE PROCEDURE ステートメントの実行時には存在している必要はありません。jar_id を指定する場合、ID は、CREATE PROCEDURE ステートメントの実行時に存在していなければなりません。ただし、プロシージャを呼び出す時点では、該当のクラス ID とメソッド ID が存在し、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42884)。

→ ' jar_id : class_id . method_id ' →

名前は、単一引用符で囲む必要があります。余分なブランクを使用することはできません。

jar_id

jar の集合をデータベースヘインストールしたときに、その jar の集合に付けられた jar ID を指定します。これは、単純 ID またはスキーマ修飾 ID のいずれかにすることができます。例えば、'myJar' や 'mySchema.myJar' のようになります。

class_id

Java オブジェクトのクラス ID を指定します。クラスがパッケージの一部である場合、クラス ID の一部に完全なパッケージ接頭部 (例えば、'myPacks.StoredProcs') が含まれている必要があります。Java 仮想マシンは、ディレクトリー './myPacks/StoredProcs/' 中のクラスを探します。Windows オペレーティング・システムでは、Java 仮想マシンはディレクトリー './¥myPacks¥StoredProcs¥' を探索します。

method_id

呼び出す Java クラスのメソッド名を指定します。

- LANGUAGE CLR の場合:

指定された *string* は、作成するプロシージャを実行するためにデータベース・マネージャーが呼び出す .NET アセンブリー (ライブラリーまたは実行可能モジュール)、そのアセンブリー内のクラス、およびそのクラス内のメソッドを表します。モジュール、クラス、およびメソッドは、CREATE PROCEDURE ステートメントの実行時に存在している必要はありません。ただし、プロシージャを呼び出す時点では、モジュール、クラス、およびメソッドは存在していなければならず、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42284)。

'clr' コンパイラー・オプションで管理対象コード拡張を指定してコンパイルされている C++ ルーチンは、'LANGUAGE C' ではなく 'LANGUAGE CLR' としてカタログする必要があります。DB2 は、必要な実行時の決定を行えるようにするために、.NET インフラストラクチャーがプロシージャー内で使用されていることを認識している必要があります。.NET インフラストラクチャーを使用するすべてのプロシージャーは、'LANGUAGE CLR' としてカタログする必要があります。

▶ '—assembly—:—class_id—!—method_id—' ◀

名前は、単一引用符で囲む必要があります。余分なブランクを使用することはできません。

assembly

クラスを含む DLL ファイルまたは他のアセンブリー・ファイルを指定します。ファイル拡張子 (.dll など) まで指定します。絶対パス名を指定しない場合、ファイルは DB2 インストール・パスの関数ディレクトリー (例えば、c:\%sqllib%\function) にあるものとされます。ファイルがインストール関数ディレクトリーのサブディレクトリーにある場合は、絶対パスを指定せずに、ファイル名の前にサブディレクトリーを指定します。例えば、インストール・ディレクトリーが c:\%sqllib

であり、アセンブリー・ファイルが c:\%sqllib%\function\myprocs\mydotnet.dll であるなら、アセンブリーの指定は 'myprocs\mydotnet.dll' とするだけで十分です。このパラメーターの大文字小文字が区別されるかどうかは、ファイル・システムの設定と同じです。

class_id

呼び出すメソッドが属するアセンブリー内のクラスの名前を指定します。クラスがネーム・スペース内にある場合は、クラスだけでなく絶対ネーム・スペースも指定することが必要です。例えば、クラス EmployeeClass がネーム・スペース MyCompany.ProcedureClasses にあるのであれば、MyCompany.ProcedureClasses.EmployeeClass をクラスとして指定します。一部の .NET 言語用のコンパイラーはクラスのネーム・スペースとしてプロジェクト名を追加するため、コマンド行コンパイラーと GUI コンパイラーのどちらを使用するかで動作が異なってくるので注意してください。このパラメーターには、大文字と小文字の区別があります。

method_id

指定したクラス内で呼び出されるメソッドを指定します。このパラメーターには、大文字と小文字の区別があります。

• LANGUAGE OLE の場合:

指定するストリングは、ステートメントが作成しているプロシージャーを実行するためにデータベース・マネージャーが呼び出す OLE のプログラム ID (*progid*) またはクラス ID (*clsid*)、およびメソッド ID (*method_id*) です。プログラム ID またはクラス ID、およびメソッド ID は、CREATE PROCEDURE ステートメントの実行時に存在している必要はありません。ただし、プロシージャーを CALL ステートメントで使用する

CREATE PROCEDURE (外部)

時点で、メソッド ID は存在していなければならない、データベース・サーバーのマシンからアクセス可能でなければなりません。そうでない場合、エラー (SQLSTATE 42724) になります。

```
▶▶ ' {progid} ! {method_id} ' ▶▶  
    {clsid}
```

名前は、単一引用符で囲む必要があります。余分なブランクを使用することはできません。

progid

OLE オブジェクトのプログラム ID を指定します。

progid は、データベース・マネージャーには解釈されず、実行時に OLE に転送されるだけです。指定する OLE オブジェクトは、作成可能である必要があり、実行時バインディング (ディスパッチに基づくバインディングとも呼ばれる) をサポートしている必要があります。規約では、*progid* は次のような形式になります。

```
<program_name>.<component_name>.<version>
```

これは規約でしかなく、厳密な規則ではないので、*progids* をこれとは異なる形式にしてもかまいません。

clsid

作成する OLE オブジェクトのクラス ID を指定します。OLE オブジェクトが *progid* を指定して登録されていない場合に、*progid* を指定する代わりに使用することができます。*clsid* の形式は次のとおりです。

```
{nnnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnnn}
```

ここで 'n' は英数字です。*clsid* は、データベース・マネージャーには解釈されず、実行時に OLE API に転送されるだけです。

method_id

呼び出す OLE オブジェクトのメソッド名を指定します。

NAME identifier

指定する *identifier* は SQL ID です。SQL ID は、ストリングの *library-id* として使用されます。区切られた ID でない場合、ID は大文字に変換されます。ID がスキーマ名で修飾されている場合、スキーマ名の部分は無視されます。この形式の NAME は、LANGUAGE C でのみ使用可能です。

FENCED または NOT FENCED

この節は、プロシージャーをデータベース・マネージャーのオペレーティング環境のプロセスまたはアドレス・スペースで実行しても「安全」か (NOT FENCED)、そうでないか (FENCED) を指定します。

プロシージャーが FENCED として登録されると、データベース・マネージャーは、その内部リソース (データ・バッファーなど) を保護して、そのプロシージャーからアクセスされないようにします。すべてのプロシージャーは、FENCED として実行するか NOT FENCED として実行するかの選択が可能です。一般に、FENCED として実行されるプロシージャーは、NOT FENCED として実行されるものと同じようには実行されません。

注意:

十分に検査されていないプロシージャに **NOT FENCED** を使用すると、DB2 の整合性が損なわれる場合があります。DB2 では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられていますが、**NOT FENCED** プロシージャが使用される場合には、完全な整合性を確保できません。

プロシージャを **NOT FENCED** として登録するには、SYSADM 権限、DBADM 権限、または特殊権限 (**CREATE_NOT_FENCED**) が必要です。LANGUAGE OLE または **NOT THREADSAFE** を指定したプロシージャには、**FENCED** のみを指定できます。

NOT FENCED 節を指定している場合は、LANGUAGE CLR プロシージャを作成できません (SQLSTATE 42601)。

THREADSAFE または NOT THREADSAFE

プロシージャを他のルーチンと同じプロセスで実行しても安全か (**THREADSAFE**)、そうでないか (**NOT THREADSAFE**) を指定します。

プロシージャが OLE 以外の LANGUAGE で定義される場合:

- プロシージャが **THREADSAFE** として定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスにプロシージャを呼び出すことができます。一般に、スレッド・セーフになるには、プロシージャはどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。**FENCED** および **NOT FENCED** プロシージャの両方を **THREADSAFE** にすることができます。
- プロシージャが **NOT THREADSAFE** に定義される場合には、データベース・マネージャーは他のルーチンと同じプロセスにプロシージャを決して呼び出しません。

FENCED プロシージャの場合、LANGUAGE が JAVA または CLR なら **THREADSAFE** がデフォルトです。これ以外のすべての言語の場合は、**NOT THREADSAFE** がデフォルトです。プロシージャが LANGUAGE OLE に定義される場合には、**THREADSAFE** は指定できません (SQLSTATE 42613)。

NOT FENCED プロシージャの場合には、**THREADSAFE** がデフォルトです。**NOT THREADSAFE** を指定することはできません (SQLSTATE 42613)。

EXTERNAL ACTION または NO EXTERNAL ACTION

プロシージャが、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るか (**EXTERNAL ACTION**)、または取らないか (**NO EXTERNAL ACTION**) を指定します。デフォルトは **EXTERNAL ACTION** です。**NO EXTERNAL ACTION** を指定した場合、プロシージャが外部に影響を与えないことを前提とした最適化を、システムは使用できます。

INHERIT SPECIAL REGISTERS

このオプション節は、プロシージャの更新可能な特殊レジスターが、呼び出しステートメントの環境からの初期値を継承するよう指定します。

特殊レジスターに対する変更が、プロシージャの呼び出し元に戻されることはありません。

CREATE PROCEDURE (外部)

更新不能の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、デフォルト値に設定されます。

PARAMETER STYLE

この節は、プロシージャにパラメーターを渡し、プロシージャから値を戻すのに用いる規則を指定するために使用します。

DB2GENERAL

プロシージャは、Java メソッドを使用するために定義された規則に従ったパラメーターの受け渡し規則を使用します。これは、LANGUAGE JAVA を使用する場合にだけ指定する必要があります。

DB2SQL

CALL ステートメントのパラメーターの他に、以下の引数がプロシージャに渡されます。

- CALL ステートメントの各パラメーターの NULL 標識を含むベクトル
- DB2 へ戻される SQLSTATE
- プロシージャの修飾名
- プロシージャの特定名
- DB2 へ戻される SQL 診断ストリング

これは、LANGUAGE C、COBOL、CLR、または OLE を使用する場合にだけ、指定することができます。

GENERAL

プロシージャは、パラメーター受け渡しメカニズムを使用します。ここでは、プロシージャは CALL で指定したパラメーターを受け取ります。パラメーターは言語ごとに直接に渡されることになっているので、SQLDA 構造は使われません。これは、LANGUAGE C、COBOL、または CLR を使用する場合にだけ、指定することができます。

NULL 標識がプログラムに直接渡されることはありません。

GENERAL WITH NULLS

GENERAL で指定した CALL ステートメントのパラメーターの他に、別の引数がプロシージャに渡されます。この別の引数は、CALL ステートメントの各パラメーター用の、NULL 標識のベクトルです。これは、C では短精度整数の配列になります。これは、LANGUAGE C、COBOL、または CLR を使用する場合にだけ、指定することができます。

JAVA

プロシージャは、Java 言語および SQLJ ルーチンの仕様に準拠する規則に従ったパラメーターの受け渡し規則を使用します。IN/OUT および OUT パラメーターは、戻り値を処理するために単一項目配列として渡されます。これは、LANGUAGE JAVA を使用する場合にだけ指定する必要があります。

PARAMETER STYLE JAVA プロシージャでは、DBINFO または PROGRAM TYPE 節はサポートされていません。

SQL

CALL ステートメントのパラメーターの他に、以下の引数がプロシージャに渡されます。

- CALL ステートメントの各パラメーターの NULL 標識
- DB2 へ戻される SQLSTATE
- プロシージャの修飾名
- プロシージャの特定名
- DB2 へ戻される SQL 診断ストリング

これは、LANGUAGE C、COBOL、CLR、または OLE を使用する場合にだけ、指定することができます。

PARAMETER CCSID

プロシージャとやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。プロシージャが呼び出される時のアプリケーション・コード・ページはデータベース・コード・ページです。

UNICODE

ストリング・データが Unicode でエンコードされることを指定します。データベースが Unicode データベースの場合、文字データは UTF-8、GRAPHIC データは UCS-2 になります。データベースが Unicode データベースでない場合は、文字データは UTF-8 になります。いずれの場合も、プロシージャが呼び出される時のアプリケーション・コード・ページは 1208 です。

データベースが Unicode データベースではないのに、PARAMETER CCSID UNICODE を指定したプロシージャを作成すると、そのプロシージャは GRAPHIC タイプ、XML タイプ、またはユーザー定義タイプを取ることができません (SQLSTATE 560C1)。PARAMETER CCSID UNICODE プロシージャは、DB2 バージョン 8.1 以降のクライアントからのみ呼び出すことができます (SQLSTATE 42997)。

データベースが Unicode ではなく、データベース構成に代替照合シーケンスが指定されている場合、PARAMETER CCSID ASCII または PARAMETER CCSID UNICODE を指定したプロシージャを作成できます。プロシージャとやり取りされるすべてのデータは、適切なコード・ページに変換されます。

この節を LANGUAGE OLE、LANGUAGE JAVA、または LANGUAGE CLR とともに指定することはできません (SQLSTATE 42613)。

PROGRAM TYPE

プロシージャでのパラメーターのスタイルが、メインルーチンなのかサブルーチンなのかを指定します。デフォルトは SUB です。

SUB

プロシージャのパラメーターは、別々の引数として渡されます。

CREATE PROCEDURE (外部)

MAIN

プロシージャのパラメーターは、引数カウンター、および引数のベクトルとして渡されます (argc、argv)。呼び出すプロシージャの名前も、"main" となります。このタイプのストアード・プロシージャは、独立した実行可能ファイルではなく、共用ライブラリーと同じ方法で作成する必要があります。PROGRAM TYPE MAIN は、LANGUAGE 節に C、COBOL、または CLR のいずれかが指定されている場合にのみ有効です。

DBINFO または NO DBINFO

DB2 において既知である特定の情報が呼び出されたときに、その情報を追加の呼び出し時引数としてプロシージャに渡すか (DBINFO)、または渡さないか (NO DBINFO) を指定します。NO DBINFO がデフォルト値です。DBINFO は、LANGUAGE OLE ではサポートされません (SQLSTATE 42613)。これは PARAMETER STYLE JAVA、または DB2GENERAL でもサポートされません。

DBINFO を指定すると、以下の情報を含む構造がプロシージャに渡されます。

- データベース名 - 現在接続されているデータベースの名前。
- アプリケーション ID - データベースへの接続ごとに確立された、固有アプリケーション ID。
- アプリケーション許可 ID - アプリケーション実行時の許可 ID。
- コード・ページ - データベースのコード・ページを識別します。
- データベースのバージョン/リリース - プロシージャを呼び出すデータベース・サーバーのバージョン、リリース、および修正レベルを識別します。
- プラットフォーム - サーバーのプラットフォーム・タイプが入ります。

DBINFO 構造はすべての外部ルーチンで共通で、プロシージャに関係ない追加のフィールドを含みます。

注

- まだ存在していないスキーマ名を用いてプロシージャを作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- NOT FENCED として定義される Java ルーチンは、FENCED THREADSAFE として定義されているかのように呼び出されます。
- 動的コンパウンド・ステートメント内から呼び出されるプロシージャは、プロシージャ作成時に OLD SAVEPOINT LEVEL が指定またはデフォルト設定されていたとしても、NEW SAVEPOINT LEVEL を指定して作成されたかのように実行されます。
- PARAMETER STYLE DB2GENERAL 節が指定されている場合、XML パラメーターは、LANGUAGE JAVA 外部プロシージャでのみサポートされます。
- **特権**
 - プロシージャの定義者は、プロシージャに対する WITH GRANT OPTION 付きの EXECUTE 特権と、プロシージャをドロップする権利を常に与えられます。

- プロシージャーを SQL ステートメントで使用する時点で、プロシージャーの定義者はそのプロシージャーによって使用されるすべてのパッケージに対して EXECUTE 特権を持っていないければなりません。
- 互換性
 - DB2 for z/OS との互換性:
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - ASUTIME NO LIMIT
 - COMMIT ON RETURN NO
 - NO COLLID
 - STAY RESIDENT NO
 - Unicode データベースでの CCSID UNICODE
 - PARAMETER CCSID UNICODE が指定されていない場合、非 Unicode データベース内での CCSID ASCII
 - 以前のバージョンの DB2 との互換性:
 - DYNAMIC RESULT SETS の代わりに RESULT SETS を指定できます。
 - CALLED ON NULL INPUT の代わりに NULL CALL を指定できます。
 - DB2GENERAL の代わりに DB2GENRL を指定できます。
 - GENERAL の代わりに SIMPLE CALL を指定できます。
 - GENERAL WITH NULLS の代わりに SIMPLE CALL WITH NULLS を指定できます。
 - PARAMETER STYLE DB2DARI はサポートされています。

例

例 1: Java で書かれたプロシージャーのプロシージャー定義を作成します。このプロシージャーは、パーツ・ナンバーを渡されて、パーツの価格と現在入手可能な数量を戻します。

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                                OUT COST DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
EXTERNAL NAME 'parts.onhand'
LANGUAGE JAVA PARAMETER STYLE JAVA
```

例 2: C で書かれたプロシージャーのプロシージャー定義を作成します。このプロシージャーは、部品番号を渡されて、部品を構成するパーツの数とパーツの合計価格、およびパーツ番号、数量、各パーツの単価をリストする結果セットを戻します。

```
CREATE PROCEDURE ASSEMBLY_PARTS (IN ASSEMBLY_NUM INTEGER,
                                  OUT NUM_PARTS INTEGER,
                                  OUT COST DOUBLE)
EXTERNAL NAME 'parts!assembly'
DYNAMIC RESULT SETS 1 NOT FENCED
LANGUAGE C PARAMETER STYLE GENERAL
```

CREATE PROCEDURE (ソース派生)

CREATE PROCEDURE (ソース派生) ステートメントは、別のプロシージャ (ソース・プロシージャ) に基づくプロシージャ (ソース派生プロシージャ) を登録します。フェデレーテッド・システムにおいて、フェデレーテッド・プロシージャとは、サポートされているデータ・ソースにソース・プロシージャを持つソース派生プロシージャのことです。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

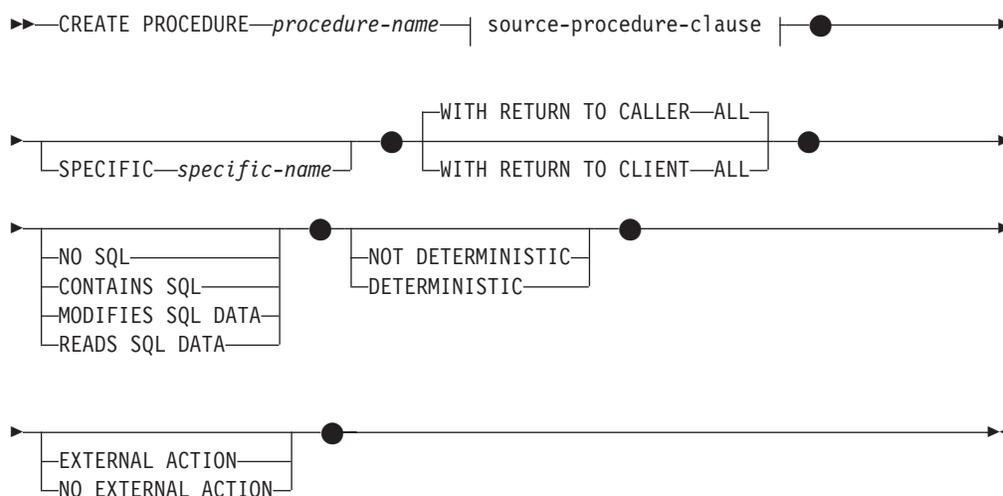
許可

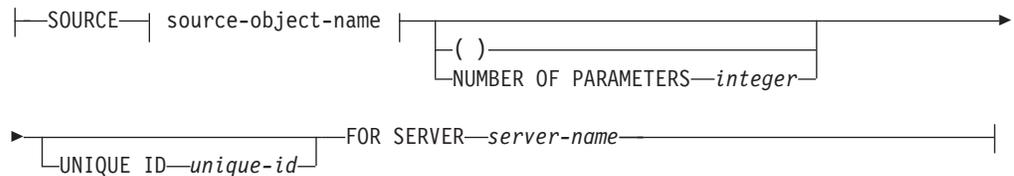
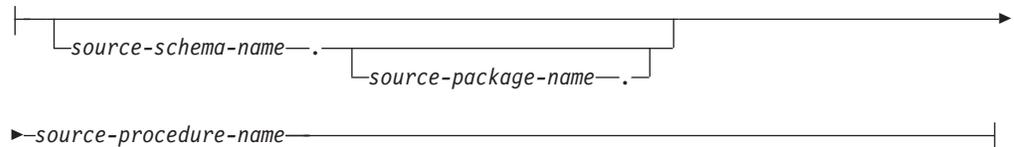
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する IMPLICIT_SCHEMA 権限 (プロシージャのスキーマ名が既存のスキーマを指していない場合)
- スキーマに対する CREATEIN 特権 (プロシージャのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

ユーザー・マッピングが必要なデータ・ソースの場合、ステートメントの許可 ID がデータ・ソースで保持する特権に、リモート・カタログ表からプロシージャの記述を選択できる特権が含まれていなければなりません。

構文



source-procedure-clause:**source-object-name:****説明***procedure-name*

定義するソース派生プロシージャの名前を指定します。この名前は、プロシージャを指定する修飾または非修飾の名前です。 *procedure-name* (プロシージャ名) の非修飾形式は SQL ID です (最大長 128)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスタが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

暗黙または明示の修飾子を含む名前と、パラメーターの数との組み合わせは、カタログにすでに記述されているプロシージャを指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数との組み合わせは、複数のスキーマ間で固有である必要はありません。

2 つの部分からなる名前を指定する場合、'SYS' で始まる *schema-name* (スキーマ名) は使用できません (SQLSTATE 42939)。

フェデレーテッド・システムでは、*procedure-name* はフェデレーテッド・サーバー上のプロシージャの名前です。

SOURCE *source-object-name*

定義するプロシージャで使用されるソース・プロシージャを指定します。フェデレーテッド・システムでは、ソース・プロシージャは、サポートされるデータ・ソースにあるプロシージャです。

source-schema-name

ソース・プロシージャのスキーマ名を示します。スキーマ名を使用してソース・プロシージャを識別する場合は、CREATE PROCEDURE (ソース派生) ステートメントで *source-schema-name* を指定する必要があります。*source-schema-name* に特殊文字や小文字が含まれる場合には、二重引用符で囲まなければなりません。

source-package-name

ソース・プロシージャのパッケージ名を示します。 *source-package-name*

CREATE PROCEDURE (ソース派生)

は、Oracle データ・ソースにのみ適用されます。パッケージ名を使用してソース・プロシージャを識別する場合は、CREATE PROCEDURE (ソース派生) ステートメントで *source-package-name* を指定する必要があります。*source-package-name* に特殊文字や小文字が含まれる場合には、二重引用符で囲まなければなりません。

source-procedure-name

ソース・プロシージャのプロシージャ名を示します。

source-procedure-name に特殊文字や小文字が含まれる場合には、二重引用符で囲まなければなりません。

() パラメーターの数がゼロであることを指定します。

NUMBER OF PARAMETERS *integer*

ソース・プロシージャのパラメーターの数を指定します。 *integer* の最小値は 0、最大値は 32 767 です。

UNIQUE ID *string-constant*

データ・ソースに名前、スキーマ、パラメーター数が同じプロシージャが複数ある場合にソース・プロシージャを一意的に識別するためのものを指定します。 *string-constant* 値 (最大長 128) は、データ・ソースごとに一意的に解釈されます。

FOR SERVER *server-name*

CREATE SERVER ステートメントを使用して登録されたサーバー定義を指定します。

SPECIFIC *specific-name*

定義するソース派生プロシージャのインスタンスに固有の名前を指定します。この特定名は、このソース派生プロシージャをドロップする場合、またはこのソース派生プロシージャにコメントを付ける場合に使用することができます。この名前を使用してソース派生プロシージャを呼び出すことはできません。*specific-name* の非修飾形式は、最大長 18 の SQL ID です。*specific-name* の修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、*specific-name* 値が、アプリケーション・サーバーに存在する別のプロシージャ・インスタンスを示すものであってはなりません。さもないと、エラーが戻されます (SQLSTATE 42710)。

specific-name は、既存の *procedure-name* と同じにすることができます。

修飾子を指定しない場合、*procedure-name* に使用された修飾子が使用されます。修飾子を指定する場合は、*procedure-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42882)。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成される固有名は、'SQL' の後に文字タイム・スタンプが続く名前です ('SQLyyymmddhhmmssxxx')。

WITH RETURN TO CALLER または WITH RETURN TO CLIENT

ソース・プロシージャからの結果セットをどこで扱うかを指示します。呼び出し側またはクライアントに戻される結果セットは 1 つのみです。ソース・プロシージャのコーディングが複数の結果セットを戻すようになっている場合は、

最初の結果セットのみが呼び出し側またはクライアントに戻されます。デフォルトは WITH RETURN TO CALLER です。

WITH RETURN TO CALLER ALL

ソース・プロシージャからの結果セットを呼び出し側に戻すことを指定します。

WITH RETURN TO CLIENT ALL

ソース・プロシージャからの結果セットをクライアント・アプリケーションに直接戻すことを指定します。データ・ソースでの動的結果セットの値が 0 より大きく、ネストされたプロシージャ呼び出しの場合のみ、適用できます。

NO SQL, CONTAINS SQL, MODIFIES SQL DATA, READS SQL DATA

ソース派生プロシージャに含まれる SQL ステートメントのデータ・アクセスのレベルを示します。フェデレーテッド・サーバーにはソース派生プロシージャのソース・プロシージャがないため、データ・ソースでのソース・プロシージャの実行時には、指定したレベルの制約は受けません。ソース派生プロシージャに指定されていることとソース・プロシージャがデータ・ソースで実際に行っていることに矛盾がある場合は、データ不整合が起こる可能性があります。このオプションを明示的に指定しない場合は、ソース・プロシージャの値が使用されます。このオプションがデータ・ソースで使用できない場合、デフォルトは MODIFIES SQL DATA です。このオプションを明示的に指定した場合でも、ソース・プロシージャの値と一致していなければ、エラーが戻されます (SQLSTATE 428GS)。

DETERMINISTIC または NOT DETERMINISTIC

与えられた引数値に対してソース派生プロシージャが常に同じ結果を戻すか (DETERMINISTIC)、それとも示された値に依存してソース派生プロシージャの結果が影響を受けるか (NOT DETERMINISTIC) を指定します。

DETERMINISTIC が指定されたソース派生プロシージャは、同じ入力を指定した連続呼び出しで常に同じ結果を戻します。現在、この節はプロシージャの処理に影響を与えません。このオプションを明示的に指定しない場合は、ソース・プロシージャの値が使用されます。このオプションがデータ・ソースで使用できない場合、デフォルトは NOT DETERMINISTIC です。このオプションを明示的に指定した場合でも、ソース・プロシージャの値と一致していなければ、エラーが戻されます (SQLSTATE 428GS)。

EXTERNAL ACTION または NO EXTERNAL ACTION

ソース派生プロシージャが、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るか (EXTERNAL ACTION)、または取らないか (NO EXTERNAL ACTION) を指定します。NO EXTERNAL ACTION 節が指定されると、フェデレーテッド・データベースは、ソース派生プロシージャは外部に影響を与えないという前提の最適化を使用します。このオプションを明示的に指定しない場合は、ソース・プロシージャの値が使用されます。このオプションがデータ・ソースで使用できない場合、デフォルトは EXTERNAL ACTION です。このオプションを明示的に指定した場合でも、ソース・プロシージャの値と一致していなければ、エラーが戻されます (SQLSTATE 428GS)。

CREATE PROCEDURE (ソース派生)

規則

- *source-object-name* と NUMBER OF PARAMETERS および UNIQUE ID 節でもデータ・ソースにあるプロシージャを識別できない場合は、エラーが戻されます (SQLSTATE 42883)。複数のプロシージャを識別できた場合も、エラーが戻されます (SQLSTATE 42725)。
- UNIQUE ID 節を指定しても、データ・ソースがユニーク ID をサポートしていなければ、エラーが戻されます (SQLSTATE 42883)。

注

- データ・ソース用のフェデレーテッド・プロシージャを登録する前に、フェデレーテッド・サーバーはそのデータ・ソースへアクセスするように構成されなければなりません。この構成には、データ・ソース用ラッパーの登録、データ・ソースのサーバー定義の作成のほかに、フェデレーテッド・サーバーと、ユーザー・マッピングを必要とするデータ・ソースのデータ・ソース・サーバー間の、ユーザー・マッピングの作成が含まれます。
- フェデレーテッド・サーバーで定義される SQL プロシージャや外部プロシージャと違って、フェデレーテッド・プロシージャは呼び出し側の特殊レジスタを継承しません (呼び出し側の *remote-object-name* が DB2 データ・ソース上のプロシージャを指す場合でも)。
- ソース・プロシージャの定義が変更された (たとえば、パラメーターのデータ・タイプが変更された) 場合は、フェデレーテッド・プロシージャをドロップして再作成する必要があります。そうしないと、フェデレーテッド・プロシージャが呼び出されたときにエラーが発生する可能性があります。
- ソース・プロシージャ・パラメーターの長さが 128 文字より長い場合、フェデレーテッド・プロシージャのパラメーター名は 128 バイトに切り捨てられます。
- 互換性
 - ストアード・プロシージャ・ニックネーム作成用の DataJoiner[®] 構文はサポートされていません。バージョン 9 の新しい構文では、パラメーター・タイプ・マッピングがニックネームと同じように扱われ、カタログ参照でリモート・データ・タイプが判別されます。ローカル・パラメーター・タイプは、フォワード・タイプ・マッピングで判別されます。

例

例 1: EMPLOYEE という名前の Oracle プロシージャ用に FEDEMPLOYEE という名前のフェデレーテッド・プロシージャを作成します。フェデレーテッド・サーバー S1 でのリモート・スキーマ名として USER1、リモート・パッケージ名として P1 を使用し、結果セットをクライアントに戻します。

```
CREATE PROCEDURE FEDEMPLOYEE SOURCE USER1.P1.EMPLOYEE
FOR SERVER S1 WITH RETURN TO CLIENT ALL
```

CREATE PROCEDURE (SQL)

CREATE PROCEDURE (SQL) ステートメントは、現行サーバーで SQL プロシージャを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

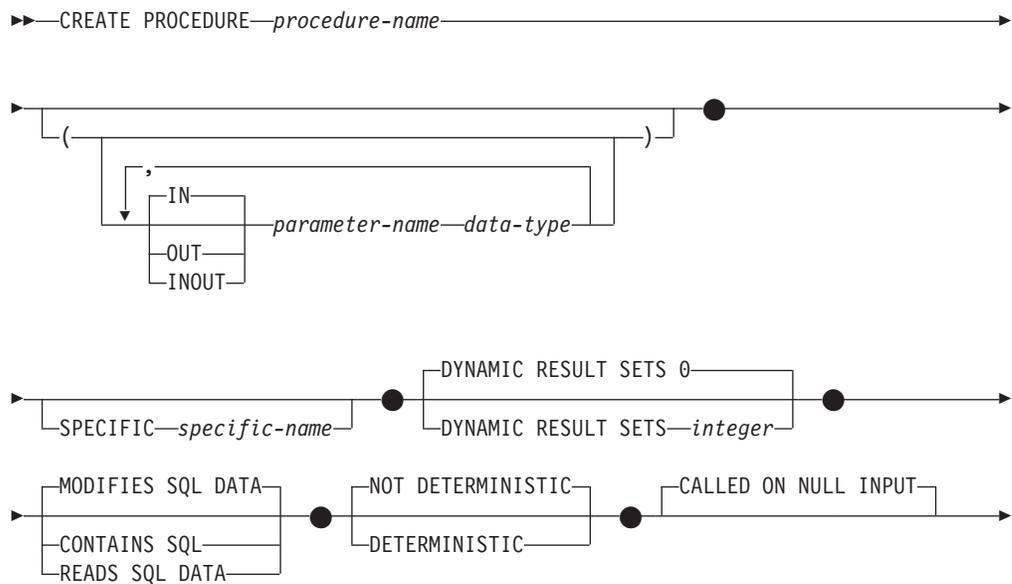
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する BINDADD 特権、および以下のどちらか。
 - データベースに対する IMPLICIT_SCHEMA 権限 (プロシージャの暗黙または明示のスキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (プロシージャのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

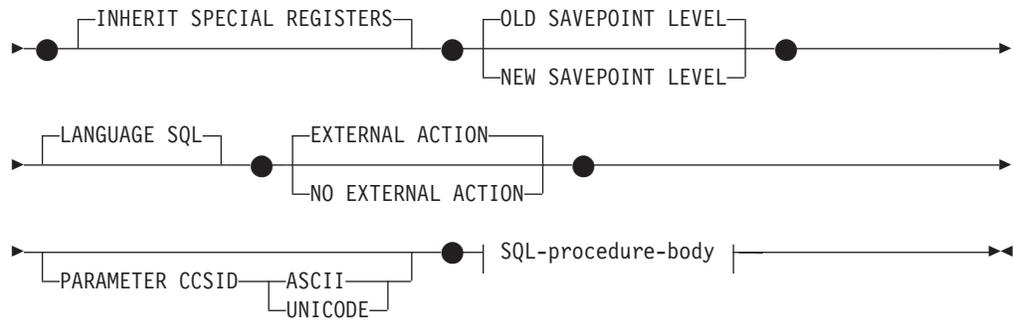
プロシージャの定義者が SYSADM 権限を持つために、プロシージャの作成しかなできない場合、プロシージャ作成のため、その定義者には暗黙的に DBADM 権限が付与されます。

このステートメントの許可 ID に SYSADM 権限または DBADM 権限がない場合には、ステートメントの許可 ID が持つ特権に、プロシージャ本体で指定される SQL ステートメントを呼び出すのに必要なすべての特権が含まれていることも必要です。

構文



CREATE PROCEDURE (SQL)



SQL-procedure-body:

|—SQL-procedure-statement—|

説明

procedure-name

定義するプロシージャの名前を指定します。この名前は、プロシージャを指定する修飾または非修飾の名前です。 *procedure-name* (プロシージャ名) の非修飾形式は SQL ID です (最大長 128)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスタが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

暗黙または明示の修飾子を含む名前と、パラメーターの数との組み合わせは、カタログにすでに記述されているプロシージャを指定するものであってはなりません (SQLSTATE 42723)。非修飾名とパラメーターの数との組み合わせは、そのスキーマ内ではユニークですが、複数のスキーマ間でユニークである必要はありません。

2 つの部分からなる名前を指定する場合、*schema-name* を 'SYS' で始めることはできません。違反すると、エラーが戻されます (SQLSTATE 42939)。

(IN | OUT | INOUT *parameter-name data-type*,...)

プロシージャのパラメーターを指定し、各パラメーターのモード、名前、およびデータ・タイプを指定します。このリストには、プロシージャが预期する各パラメーターごとに 1 つの項目を指定する必要があります。

パラメーターのないプロシージャも登録可能です。この場合、指定するデータ・タイプがない場合でも、括弧はコーディングする必要があります。以下に例を示します。

```
CREATE PROCEDURE SUBWOOFER() ...
```

1 つのスキーマに同じ名前の 2 つのプロシージャがある場合、パラメーターの数をまったく同一にすることはできません。シグニチャーが重複していると、SQL エラー (SQLSTATE 42723) になります。

例えば、以下のステートメントの場合、

```
CREATE PROCEDURE PART (IN NUMBER INT, OUT PART_NAME CHAR(35)) ...
CREATE PROCEDURE PART (IN COST DECIMAL(5,3), OUT COUNT INT) ...
```

2 番目のステートメントは失敗します。その理由は、データ・タイプが異なってもプロシージャのパラメーターの数が同じだからです。

IN | OUT | INOUT

パラメーターのモードを指定します。

プロシージャによってエラーが戻される場合、OUT パラメーターは未定義で、INOUT パラメーターは未変更です。

IN パラメーターをプロシージャの入力パラメーターとして指定します。プロシージャ内でパラメーターに加えられるすべての変更は、制御が戻されると SQL アプリケーションの呼び出しは行えなくなります。デフォルトは IN です。

OUT パラメーターをプロシージャの出力パラメーターとして指定します。

INOUT

パラメーターを、プロシージャの入力および出力パラメーターの両方として指定します。

parameter-name

パラメーターの名前を指定します。パラメーター名は、プロシージャでユニークでなければなりません (SQLSTATE 42734)。

data-type

パラメーターのデータ・タイプを指定します。

- CREATE TABLE ステートメントのデータ・タイプの定義に指定することが可能で、プロシージャの作成に使用されている言語に対応するものがある SQL データ・タイプ指定と省略形を指定できます。
- LONG VARCHAR、LONG VARGRAPHIC、REFERENCE、およびユーザー定義構造化タイプはサポートされません (SQLSTATE 429BB)。

SPECIFIC *specific-name*

定義するプロシージャのインスタンスに対するユニーク名を指定します。この特定名は、このプロシージャをドロップする場合、またはこのプロシージャにコメントを付ける場合に使用することができます。これは、プロシージャの呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 18)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含めて、その名前が、アプリケーション・サーバーに存在する他のプロシージャ・インスタンスを指定するものであってはなりません。そうでない場合、エラー (SQLSTATE 42710) になります。

specific-name は、既存の *procedure-name* と同じにすることができます。

修飾子を指定しない場合、*procedure-name* に使用された修飾子を使用されません。修飾子を指定する場合は、*procedure-name* の明示修飾子または暗黙修飾子と同じでなければなりません。そうでない場合、エラー (SQLSTATE 42882) になります。

CREATE PROCEDURE (SQL)

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成される固有名は、'SQL' の後に文字タイム・スタンプが続く名前です ('SQLyymmddhhmmssxxx')。

DYNAMIC RESULT SETS *integer*

プロシージャから戻される結果セットの上限の見積もりを指定します。

CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

プロシージャに含まれる SQL ステートメントのデータ・アクセスのレベルを示します。

CONTAINS SQL

SQL データの読み取りも変更も行わない SQL ステートメントを、プロシージャで実行できることを指定します (SQLSTATE 38004 または 42985)。プロシージャでサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

READS SQL DATA

SQL データを変更しない SQL ステートメントを、プロシージャで実行できることを指定します (SQLSTATE 38002 または 42985)。プロシージャでサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

MODIFIES SQL DATA

このプロシージャは、プロシージャでサポートされていないステートメント以外のすべての SQL ステートメントを実行できることを指定します (SQLSTATE 38003 または 42985)。

コンパウンド SQL プロシージャ内で BEGIN ATOMIC 節を使用した場合、MODIFIES SQL DATA として定義しない限り、このプロシージャを作成することはできません。

DETERMINISTIC または **NOT DETERMINISTIC**

この節は、同一の引数値に対してプロシージャが常に同じ結果を戻すか (DETERMINISTIC)、それとも状態値に依存してプロシージャの結果が影響を受けるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC が指定されたプロシージャは、連続で同じ入力を指定して呼び出した場合に常に同じ結果を戻します。

現在、この節はプロシージャの処理に影響を与えません。

CALLED ON NULL INPUT

CALLED ON NULL INPUT は、プロシージャに常に適用されます。これは、任意の引数が NULL かどうかにかかわらず、プロシージャが呼び出されることを意味します。OUT または INOUT パラメーターは、NULL 値を戻す場合も、通常の (NULL 以外の) 値を戻す場合もあります。NULL の引数値の有無のテストはプロシージャで行う必要があります。

INHERIT SPECIAL REGISTERS

このオプション節は、プロシージャの更新可能な特殊レジスターが、呼び出しステートメントの環境からの初期値を継承するよう指定します。ネストされたオブジェクト (例えば、トリガーまたはビュー) に呼び出されるルーチンの場合、初期値は (オブジェクト定義から継承するのではなく) ランタイム環境から継承します。

特殊レジスターに対する変更が、プロシーチャーの呼び出し元に戻されることはありません。

更新不能の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、デフォルト値に設定されます。

OLD SAVEPOINT LEVEL または NEW SAVEPOINT LEVEL

このプロシーチャーが、セーブポイント名と影響について新しいセーブポイント・レベルを設定するかどうかを指定します。 **OLD SAVEPOINT LEVEL** がデフォルトの動作です。セーブポイント・レベルについて詳しくは、『SAVEPOINT』の説明にある『規則』の項を参照してください。

LANGUAGE SQL

この節は、プロシーチャー本体が SQL 言語に書き込まれるように指定するのに使用します。

EXTERNAL ACTION または NO EXTERNAL ACTION

プロシーチャーが、データベース・マネージャーによって管理されていないオブジェクトの状態を変更するアクションを取るか (**EXTERNAL ACTION**)、または取らないか (**NO EXTERNAL ACTION**) を指定します。デフォルトは **EXTERNAL ACTION** です。 **NO EXTERNAL ACTION** を指定した場合、プロシーチャーが外部に影響を与えないことを前提とした最適化を、システムは使用できます。

PARAMETER CCSID

プロシーチャーとやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。 **PARAMETER CCSID** 節を指定しない場合のデフォルトは、Unicode データベースでは **PARAMETER CCSID UNICODE**、他のすべてのデータベースでは **PARAMETER CCSID ASCII** になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、**PARAMETER CCSID ASCII** を指定することはできません (SQLSTATE 56031)。

UNICODE

文字データは UTF-8 で記述され、**GRAPHIC** データは UCS-2 で記述されることを指定します。データベースが Unicode データベースでない場合は、**PARAMETER CCSID UNICODE** は指定できません (SQLSTATE 56031)。

SQL-procedure-body

SQL プロシーチャーの本体である SQL ステートメントを指定します。プロシーチャー・コンパウンド・ステートメント内に複数の **SQL-procedure-statement** を指定することができます。『コンパウンド SQL (プロシーチャー)』で、**SQL-procedure-statement** を参照してください。

規則

- 動的コンパウンド・ステートメント内から呼び出されるプロシーチャーは、プロシーチャー作成時に **OLD SAVEPOINT LEVEL** が指定またはデフォルト設定されていたとしても、**NEW SAVEPOINT LEVEL** を指定して作成されたかのように実行されます。

CREATE PROCEDURE (SQL)

注

- まだ存在していないスキーマ名を用いてプロシージャを作成すると、ステートメントの許可 ID に `IMPLICIT_SCHEMA` 権限がある場合に限り、そのスキーマが暗黙に作成されます。そのスキーマの所有者は `SYSIBM` です。スキーマに対する `CREATEIN` 特権が `PUBLIC` に付与されます。

- **特権**

プロシージャの定義者は、プロシージャに対する `WITH GRANT OPTION` 付きの `EXECUTE` 特権と、プロシージャをドロップする権利を常に与えられます。

- **互換性**

- DB2 for z/OS との互換性:
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - `ASUTIME NO LIMIT`
 - `COMMIT ON RETURN NO`
 - `NO COLLID`
 - `STAY RESIDENT NO`
- 以前のバージョンの DB2 との互換性:
 - `DYNAMIC RESULT SETS` の代わりに `RESULT SETS` を指定できます。
 - `CALLED ON NULL INPUT` の代わりに `NULL CALL` を指定できます。

例

例 1: 社員の給与の中央値を戻す SQL プロシージャを作成します。給与の中央値を超える給与を得ている全社員の氏名、肩書き、および給与の入った結果セットを戻します。

```
CREATE PROCEDURE MEDIAN_RESULT_SET (OUT medianSalary DOUBLE)
  RESULT SETS 1
  LANGUAGE SQL
BEGIN
  DECLARE v_numRecords INT DEFAULT 1;
  DECLARE v_counter INT DEFAULT 0;

  DECLARE c1 CURSOR FOR
    SELECT CAST(salary AS DOUBLE)
      FROM staff
      ORDER BY salary;
  DECLARE c2 CURSOR WITH RETURN FOR
    SELECT name, job, CAST(salary AS INTEGER)
      FROM staff
      WHERE salary > medianSalary
      ORDER BY salary;

  DECLARE EXIT HANDLER FOR NOT FOUND
    SET medianSalary = 6666;

  SET medianSalary = 0;
  SELECT COUNT(*) INTO v_numRecords
    FROM STAFF;
  OPEN c1;
  WHILE v_counter < (v_numRecords / 2 + 1)
  DO
    FETCH c1 INTO medianSalary;
    SET v_counter = v_counter + 1;
```

```

END WHILE;
CLOSE c1;
OPEN c2;
END

```

CREATE ROLE

CREATE ROLE ステートメントは、現行のサーバーのロールを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

```

▶▶ CREATE ROLE role-name ◀◀

```

説明

role-name

ロールの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。名前は、現行のサーバー上の既存のロールを識別するものであってはなりません (SQLSTATE 42710)。名前を 'SYS' という文字で始めることはできず、'DBADM'、'NONE'、'NULL'、'PUBLIC'、または 'SECADM' とすることもできません (SQLSTATE 42939)。

例

DOCTOR という名前のロールを作成します。

```

CREATE ROLE DOCTOR

```

CREATE SCHEMA

CREATE SCHEMA ステートメントは、スキーマを定義します。また、オブジェクトを作成して、このステートメントでそのオブジェクトに関する特権を与えることも可能です。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

CREATE SCHEMA

許可

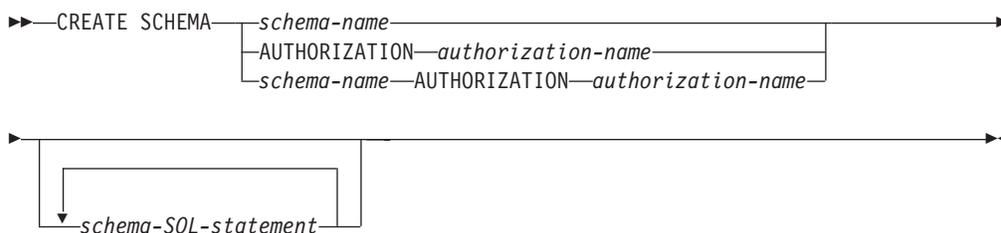
SYSADM 権限または DBADM 権限のある許可 ID は、任意の有効な *schema-name* または *authorization-name* を指定してスキーマを作成できます。

SYSADM 権限または DBADM 権限がない許可 ID は、ステートメントの許可 ID に一致する *schema-name* または *authorization-name* を指定しなければスキーマを作成できません。

ステートメントに *schema-SQL-statement* が含まれている場合、*authorization-name* (指定されていない場合、ステートメントの許可 ID がデフォルト解釈される) が持つ特権には、以下の特権の少なくとも 1 つが含まれている必要があります。

- それぞれの *schema-SQL-statement* を実行するために必要な特権
- SYSADM または DBADM 権限

構文



説明

schema-name

スキーマの名前を指定します。この名前は、カタログですでに記述されているスキーマを指定するものであってはなりません (SQLSTATE 42710)。*'SYS'* で始まる名前は使用できません (SQLSTATE 42939)。スキーマの所有者は、ステートメントを発行した許可 ID です。

AUTHORIZATION *authorization-name*

スキーマの所有者であるユーザーを指定します。*authorization-name* の値は、スキーマの名前の指定にも使用されます。*authorization-name* は、カタログですでに記述されているスキーマを指定するものであってはなりません (SQLSTATE 42710)。

schema-name **AUTHORIZATION** *authorization-name*

schema-name のスキーマを識別します。その所有者が *authorization-name* です。*schema-name* は、カタログですでに記述されているスキーマを識別するものであってはなりません (SQLSTATE 42710)。*schema-name* を指定する際に、*'SYS'* で始めることはできません (SQLSTATE 42939)。

schema-SQL-statement

CREATE SCHEMA ステートメントに組み込むことができる SQL ステートメントは、次のとおりです。

- CREATE TABLE ステートメント (型付き表およびマテリアライズ照会表は除く)
- CREATE VIEW ステートメント (型付きビューは除く)

- CREATE INDEX ステートメント
- COMMENT ステートメント
- GRANT ステートメント

注

- スキーマの所有者は、以下のように決定されます。
 - AUTHORIZATION 節が指定されている場合は、指定された *authorization-name* がスキーマの所有者になります。
 - AUTHORIZATION 節の指定がない場合は、CREATE SCHEMA ステートメントを発行した許可 ID がスキーマの所有者になります。
- スキーマの所有者は、ユーザーであることが想定されます (グループではなく)。
- CREATE SCHEMA ステートメントを使用してスキーマを明示的に作成すると、スキーマの所有者はそのスキーマに関して CREATEIN 特権、DROPIN 特権、および ALTERIN 特権を与えられ、これらの特権を他のユーザーに与えることができます。
- CREATE SCHEMA ステートメントの一部として作成されるオブジェクトの定義者は、スキーマの所有者になります。スキーマの所有者は、CREATE SCHEMA ステートメントの一環として与えられる特権の付与者でもあります。
- CREATE SCHEMA ステートメント中の SQL ステートメント中の非修飾のオブジェクト名は、作成されたスキーマの名前によって暗黙的に修飾されます。
- CREATE ステートメントに、作成するオブジェクトの修飾名が含まれる場合、その修飾名に指定されたスキーマ名は作成されるスキーマの名前と同じでなければなりません (SQLSTATE 42875)。ステートメントで参照されるその他のオブジェクトは、任意の有効なスキーマ名で修飾することができます。
- スキーマ名として SESSION を使用することは推奨されません。宣言済み一時表は SESSION で修飾されていなければならないので、アプリケーションで、持続表と同一の名前を付けた一時表を宣言することがあり得ます。スキーマ名 SESSION の付いた表を参照する SQL ステートメントは、同一名の持続表ではなく宣言済み一時表に解決されてしまいます (ステートメントのコンパイル時に)。組み込み静的および動的な組み込み SQL ステートメントでは、別々の時点で SQL ステートメントのコンパイルが行われるので、結果は、宣言済み一時表がいつ定義されたかによって異なってしまいます。持続表、ビュー、または別名が、SESSION というスキーマ名を使って定義されていなければ、これらの事項にとらわれる必要はありません。

例

例 1: DBADM 権限のあるユーザーが、RICK という名前のスキーマをユーザー RICK を所有者として作成します。

```
CREATE SCHEMA RICK AUTHORIZATION RICK
```

例 2: 部品の在庫表と部品番号の索引があるスキーマを作成します。ユーザー JONES に、表に対する権限を与えます。

```
CREATE SCHEMA INVENTORY
```

```
CREATE TABLE PART (PARTNO SMALLINT NOT NULL,
DESCR VARCHAR(24),
QUANTITY INTEGER)
```

CREATE SCHEMA

```
CREATE INDEX PARTIND ON PART (PARTNO)
```

```
GRANT ALL ON PART TO JONES
```

例 3: 2 つの表がある PERS という名前のスキーマを作成します。それぞれの表には他の表を参照する外部キーがあります。これは、ALTER TABLE ステートメントを使用せずにこのような表のペアを作成する CREATE SCHEMA ステートメントの機能の一例です。

```
CREATE SCHEMA PERS
```

```
CREATE TABLE ORG (DEPTNUMB SMALLINT NOT NULL,  
DEPTNAME VARCHAR(14),  
MANAGER SMALLINT,  
DIVISION VARCHAR(10),  
LOCATION VARCHAR(13),  
CONSTRAINT PKEYDNO  
PRIMARY KEY (DEPTNUMB),  
CONSTRAINT FKEYMGR  
FOREIGN KEY (MANAGER)  
REFERENCES STAFF (ID) )
```

```
CREATE TABLE STAFF (ID SMALLINT NOT NULL,  
NAME VARCHAR(9),  
DEPT SMALLINT,  
JOB VARCHAR(5),  
YEARS SMALLINT,  
SALARY DECIMAL(7,2),  
COMM DECIMAL(7,2),  
CONSTRAINT PKEYID  
PRIMARY KEY (ID),  
CONSTRAINT FKEYDNO  
FOREIGN KEY (DEPT)  
REFERENCES ORG (DEPTNUMB) )
```

CREATE SECURITY LABEL COMPONENT

CREATE SECURITY LABEL COMPONENT ステートメントは、セキュリティー・ポリシーの一環として使用されるコンポーネントを定義します。

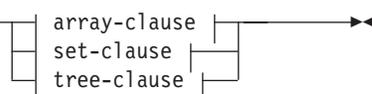
呼び出し

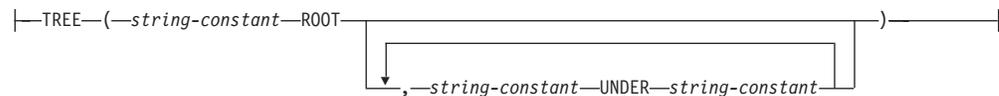
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

```
▶▶ CREATE SECURITY LABEL COMPONENT component-name 
```

array-clause:**set-clause:****tree-clause:****説明***component-name*

セキュリティー・ラベル・コンポーネントに名前を付けます。これは、1 部構成の名前です。名前は、現行のサーバー上の既存のセキュリティー・ラベル・コンポーネントを識別するものであってはなりません (SQLSTATE 42710)。

ARRAY

エレメントを順序付けしたセットを指定します。

string-constant,...

このセキュリティー・ラベル・コンポーネント用の一連の有効値を構成する 1 つ以上のストリング定数値。配列エレメントが出現する順序は重要です。最初のエレメントのほうが、2 番目のエレメントよりもランクは上です。2 番目のエレメントのほうが、3 番目のエレメントよりもランクは上というように、以降同様に続きます。

SET

エレメントの順序付けされていないセットを指定します。

string-constant,...

このセキュリティー・ラベル・コンポーネント用の一連の有効値を構成する 1 つ以上のストリング定数値。エレメントの順序は重要ではありません。

TREE

ノード・エレメントのツリー構造を指定します。

string-constant

このセキュリティー・ラベル・コンポーネント用の一連の有効値を構成する 1 つ以上のストリング定数値。

ROOT

キーワードの後に続く *string-constant* を、ツリーのルート・ノード・エレメントと指定します。

CREATE SECURITY LABEL COMPONENT

UNDER

UNDER キーワードの前の *string-constant* は、**UNDER** キーワードの後に続く *string-constant* の子であると指定します。エレメントを親として使用するには、事前にそのエレメントをルート・エレメントであると定義するか、または別のエレメントの子であると定義する必要があります。そうでないと、エラー (SQLSTATE 42704) が戻されます。

規則

以下の規則は、3 つのタイプのコンポーネントすべて (ARRAY、SET、および TREE) に対して適用されます。

- エレメントの名前中で以下のいずれの文字も使用できません。
 - 左括弧 - (
 - 右括弧 -)
 - コンマ - ,
 - コロン - :
- エレメント名は、32 バイト以下でなければなりません (SQLSTATE 42622)。
- セキュリティー・ラベル・コンポーネントがセットまたはツリーである場合、最大 64 個のエレメントをそのコンポーネントの一部とすることができます。
- **CREATE SECURITY LABEL COMPONENT** ステートメントは、タイプ配列のセキュリティ・ラベル・コンポーネントのために最大 65 535 のエレメントを指定できます。
- エレメント名を同じコンポーネント内で複数回使用することはできません (SQLSTATE 42713)。

例

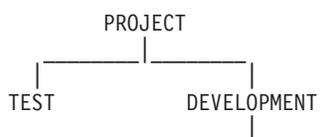
例 1: LEVEL という名前の ARRAY タイプのセキュリティ・ラベル・コンポーネントを作成します。このコンポーネントには、以下の 4 つのエレメントがあります。それらは、ランクの高いものから順に、Top Secret、Secret、Classified、および Unclassified となっています。

```
CREATE SECURITY LABEL COMPONENT LEVEL
  ARRAY ['Top Secret', 'Secret', 'Classified', 'Unclassified']
```

例 2: COMPARTMENTS という名前の SET タイプのセキュリティ・ラベル・コンポーネントを作成します。このコンポーネントには、Research、Analysis、および Collection という 3 つのエレメントがあります。

```
CREATE SECURITY LABEL COMPONENT COMPARTMENTS
  SET {'Collection', 'Research', 'Analysis'}
```

例 3: GROUPS という名前の TREE タイプのセキュリティ・ラベル・コンポーネントを作成します。GROUPS には、PROJECT、TEST、DEVELOPMENT、CURRENT、および FIELD という 5 つのエレメントがあります。以下のダイアグラムは、これらのエレメントの相互関係を示しています。



CREATE SECURITY LABEL

string-constant,...

セキュリティー・コンポーネントの有効エレメントを指定します。有効エレメントとは、セキュリティー・コンポーネントの作成時に指定したものです。このエレメントが無効の場合、エラーが戻されます (SQLSTATE 4274F)。

例

例 1: EMPLOYEESECLABEL というセキュリティー・ラベルを作成します。これは、DATA_ACCESS セキュリティー・ポリシーの一部であり、LEVEL コンポーネントに対してエレメント Top Secret をもち、COMPARTMENTS コンポーネントに対してエレメント Research および Analysis をもちます。

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research', 'Analysis'
```

例 2: LEVEL コンポーネントに対してエレメント Top Secret をもち、COMPARTMENTS コンポーネントに対してエレメント Research をもち EMPLOYEESECLABELREAD というセキュリティー・ラベルを作成します。

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
  COMPONENT LEVEL 'Top Secret',
  COMPONENT COMPARTMENTS 'Research'
```

例 3: COMPARTMENTS コンポーネントに対してエレメント Analysis をもち、LEVEL コンポーネントに対して NULL 値をもつ EMPLOYEESECLABELWRITE というセキュリティー・ラベルを作成します。DATA_ACCESS というセキュリティー・ポリシーが、例 1 および 2 で使用されるセキュリティー・ポリシーと同一であることを前提としています。

```
CREATE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
  COMPONENT COMPARTMENTS 'Analysis'
```

例 4: BEGINNER というセキュリティー・ラベルを作成します。これは、既存の CLASSPOLICY セキュリティー・ポリシーの一部であり、TRUST コンポーネントに対してエレメント Trainee をもち、SECTIONS コンポーネントに対してエレメント Morning をもちます。

```
CREATE SECURITY LABEL CLASSPOLICY.BEGINNER
  COMPONENT TRUST 'Trainee',
  COMPONENT SECTIONS 'Morning'
```

CREATE SECURITY POLICY

CREATE SECURITY POLICY ステートメントは、セキュリティー・ポリシーを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

```

▶▶ CREATE SECURITY POLICY security-policy-name
    COMPONENTS component-name WITH DB2LBACRULES
    [ OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL
    | RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL ]

```

説明

security-policy-name

セキュリティ・ポリシーに名前を付けます。これは、1 部構成の名前です。名前は、現行のサーバー上の既存のセキュリティ・ポリシーを識別するものであってはなりません (SQLSTATE 42710)。

COMPONENTS *component-name*,...

セキュリティ・ラベル・コンポーネントを識別します。名前は、現在のサーバー上にすでに存在するセキュリティ・ラベル・コンポーネントを識別するものでなければなりません (SQLSTATE 42704)。セキュリティ・ポリシーに対して同じセキュリティ・コンポーネントを複数回指定してはなりません (SQLSTATE 42713)。セキュリティ・ポリシーに指定できるセキュリティ・ラベル・コンポーネントの数は 16 個までです (SQLSTATE 54062)。

WITH DB2LBACRULES

このセキュリティ・ポリシーの一部をなすセキュリティ・ラベルどうしを比較するときに、どの規則セットを使用するかを指示します。現在、DB2LBACRULES という 1 つの規則セットしかありません。

OPTIONAL SECURITY LABEL RESTRICTIONS: **OVERVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL or RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL**

このセキュリティ・ポリシーで保護されている表に対して発行された INSERT または UPDATE ステートメント内に明示的に指定されているセキュリティ・ラベルを書き込む許可をユーザーが持たない場合に取りうる措置を指定します。ユーザーのセキュリティ・ラベルおよび免除クレデンシャルによって、明示的に指定されたセキュリティ・ラベルを書き込むユーザー許可が判別されます。デフォルトは **OVERVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL** です。

OVERVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL

挿入または更新の操作での書き込みアクセスに対して、明示的に指定されているセキュリティ・ラベルではなく、ユーザーのセキュリティ・ラベルの値を使用することを指定します。

CREATE SECURITY POLICY

RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL

INSERT または UPDATE ステートメント内に置かれている明示的に指定されたセキュリティー・ラベルの書き込みをユーザーが許可されていない場合、挿入または更新の操作が失敗することを指示します (SQLSTATE 42519)。

注

- **DB2LBACRULES 規則セット:** DB2LBACRULES は、以下の規則を含む事前定義された規則のセットです。DB2LBACREADARRAY、DB2LBACREADSET、DB2LBACREADTREE、DB2LBACWRITEARRAY、DB2LBACWRITASET、DB2LBACWRITETREE。
- グループおよびロールの権限は、セキュリティー・ポリシーが作成されるときにデフォルトでは考慮されません。この動作を変更してそれらが考慮されるようにするには、ALTER SECURITY POLICY ステートメントを使用します。

例

例 1: DB2LBACRULES 規則セットを使用し、LEVEL、および COMPARTMENTS の 2 つのコンポーネントをこの順で持つ DATA_ACCESS という名前のセキュリティー・ポリシーを作成します。両方のコンポーネントがすでに存在していると想定します。

```
CREATE SECURITY POLICY DATA_ACCESS
  COMPONENTS LEVEL, COMPARTMENTS
  WITH DB2LBACRULES
```

例 2: すでに存在すると想定されるコンポーネント MEMBER および BADGE をもつ CONTRIBUTIONS という名前のセキュリティー・ポリシーを作成します。

```
CREATE SECURITY POLICY CONTRIBUTIONS
  COMPONENTS MEMBER, BADGE
  WITH DB2LBACRULES
```

CREATE SEQUENCE

CREATE SEQUENCE ステートメントは、アプリケーション・サーバーでのシーケンスを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

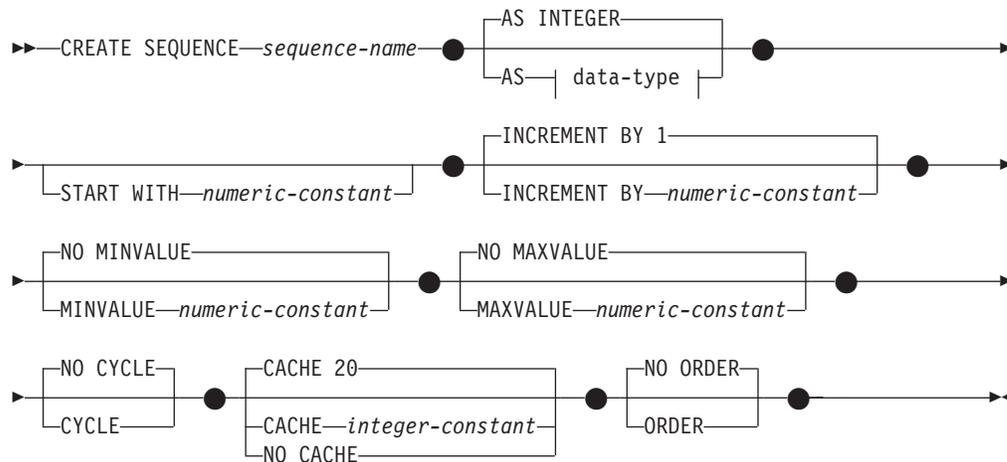
許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

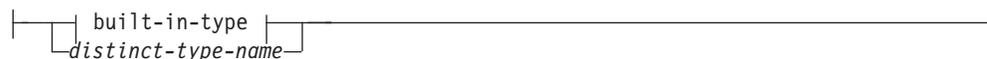
- データベースに対する IMPLICIT_SCHEMA 権限 (シーケンスの暗黙または明示のスキーマ名が存在しない場合)

- スキーマに対する CREATEIN 特権 (シーケンスのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

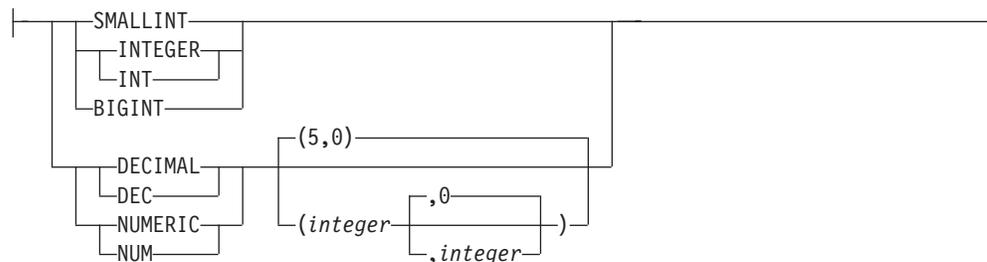
構文



data-type:



built-in-type:



説明

sequence-name

シーケンスを指定します。名前の組み合わせ、また暗黙および明示スキーマ名は、現行のサーバーに存在するシーケンスを識別することはできません (SQLSTATE 42710)。

sequence-name の非修飾フォームは SQL ID です。修飾フォームは、ピリオドと SQL ID が後ろに続く修飾子です。修飾子はスキーマ名です。

シーケンス名がスキーマ名で明示的に修飾されている場合、そのスキーマ名の先頭を 'SYS' にすると、エラーが起こります (SQLSTATE 42939)。

AS data-type

シーケンス値に使用されるデータ・タイプを指定します。データ・タイプは、ゼ

CREATE SEQUENCE

ロの位取りの整数値タイプ (SMALLINT、INTEGER、BIGINT、または DECIMAL) か、ソース・タイプがゼロの位取りの整数値タイプであるユーザー定義の特殊タイプまたは参照タイプにすることができます (SQLSTATE 42815)。デフォルトは INTEGER です。

START WITH *numeric-constant*

シーケンスの最初の値を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、シーケンスに関連するデータ・タイプの列に割り当てられる正または負の値にすることができます (SQLSTATE 42815)。デフォルトは、昇順シーケンスであれば MINVALUE、降順シーケンスであれば MAXVALUE です。

この値は、シーケンスの最大または最小値に達した後、そのシーケンスが循環する値である必要はありません。START WITH 節を使用して、循環に使用される範囲外のシーケンスを開始することができます。循環に使用される範囲は、MINVALUE および MAXVALUE によって定義されています。

INCREMENT BY *numeric-constant*

連続したシーケンス値のインターバルを指定します。この値として、長精度整数定数の値を超えない範囲で (SQLSTATE 42820)、シーケンスに関連したデータ・タイプの列に割り当てることのできる任意の正または負の値を指定できます (SQLSTATE 42815)。ただし、小数点の右側に非ゼロの数字が存在してはなりません (SQLSTATE 428FA)。

この値が負の場合、これは降順シーケンスです。この値が 0 の場合、または正の場合は、昇順になります。デフォルトは 1 です。

MINVALUE または **NO MINVALUE**

降順シーケンスが値の生成を循環または停止する最小値、あるいは最大値に達した後、昇順シーケンスが循環する最小値を指定します。

MINVALUE *numeric-constant*

最小値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、シーケンスに関連するデータ・タイプの列に割り当てられる正または負の値にすることができます (SQLSTATE 42815) が、最大値以下でなければなりません (SQLSTATE 42815)。

NO MINVALUE

昇順シーケンスの場合、値は START WITH 値で、START WITH が指定されない場合には 1 です。降順シーケンスの場合、シーケンスに関連するデータ・タイプの最小値です。これはデフォルトです。

MAXVALUE または **NO MAXVALUE**

昇順シーケンスが値の生成を循環または停止する最大値、あるいは最小値に達した後、降順シーケンスが循環する最大値を指定します。

MAXVALUE *numeric-constant*

最大値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、シーケンスに関連するデータ・タイプの列に割り当てられる正または負の値にすることができます (SQLSTATE 42815) が、最小値以上でなければなりません (SQLSTATE 42815)。

NO MAXVALUE

昇順シーケンスの場合、値はシーケンスに関連するデータ・タイプの最大値です。降順シーケンスの場合、値は **START WITH** 値で、**START WITH** が指定されない場合には -1 です。

CYCLE または NO CYCLE

その最大値または最小値に達した後、シーケンスが値の生成を続行するかどうかを指定します。次の値が境界条件を正確に満たしたとき、またはその値を超えることによって、シーケンスの境界に達します。

CYCLE

最大値または最小値に達した後、このシーケンスについて値の生成を続行することを指定します。このオプションが使用されると、昇順シーケンスが最大値に達した後、その最小値が生成されます。降順シーケンスが最小値に達した後、その最大値が生成されます。シーケンスの最大値および最小値は、循環に使用される範囲を決定します。

CYCLE が有効な場合、重複するシーケンス値が生成される場合があります。

NO CYCLE

シーケンスの最大値または最小値に達した後、そのシーケンスについて値は生成されないことを指定します。これはデフォルトです。

CACHE または NO CACHE

高速アクセスのため、事前割り振り値のいくつかをメモリーに保管するかどうかを指定します。これはパフォーマンスおよびチューニング・オプションです。

CACHE integer-constant

事前割り振りされ、メモリーに保管されるシーケンス値の最大数を指定します。値を事前割り振りしてキャッシュに保管しておくこと、シーケンス値を生成するとき、ログへの同期入出力が少なくなります。

システム障害が起こると、コミットされたステートメントで使用されていないキャッシュ済みシーケンス値はすべて失われます（使用されなくなります）。**CACHE** オプションに指定する値は、システム障害の際に失われても構わないシーケンス値の最大数です。

最小値は 2 です (SQLSTATE 42815)。デフォルト値は **CACHE 20** です。

NO CACHE

シーケンスの値が事前割り振りされないよう指定します。システム障害、シャットダウン、またはデータベース非活動化の際、値が失われることはありません。このオプションが指定されると、シーケンスの値はキャッシュに保管されません。この場合、シーケンスの新しい値が要求されるたびに、ログに対して同期入出力が行われます。

NO ORDER または ORDER

要求の順序でシーケンス番号が生成されるかどうかを指定します。

ORDER

要求の順序でシーケンス番号が生成されるよう指定します。

NO ORDER

要求の順序でシーケンス番号を生成する必要がないことを指定します。これはデフォルトです。

CREATE SEQUENCE

注

- 定数シーケンス (常に定数値を返す) を定義することも可能です。これは、INCREMENT 値にゼロを指定して START WITH 値には MAXVALUE を超えない値を指定するか、あるいは START WITH、MINVALUE、および MAXVALUE に同じ値を指定することによって実行できます。定数シーケンスの場合には、シーケンスに関する NEXT VALUE が呼び出されるたびに、同じ値が戻ります。定数シーケンスは、数値グローバル変数として使用することができます。ALTER SEQUENCE を使用すると、定数シーケンスのために生成される値を調整することができます。
- ALTER SEQUENCE ステートメントを使用して、シーケンスを手動で循環させることができます。NO CYCLE が暗黙的または明示的に指定されている場合、ALTER SEQUENCE ステートメントでシーケンスを再始動または拡張し、そのシーケンスの最大または最小値に達した後でも値の生成を続行できます。
- CYCLE キーワードを指定して、シーケンスが循環するように明示的に指定できます。シーケンスを定義する際に CYCLE オプションを使用して、生成された値が境界に達するたびに循環するよう指示します。シーケンスが自動的に循環するように定義されると (つまり CYCLE が明示的に指定された場合)、増分値が 1 または -1 以外の場合には、シーケンスに対して生成される最大または最小値は、実際に指定された MAXVALUE または MINVALUE ではない可能性があります。たとえば、START WITH=1、INCREMENT=2、MAXVALUE=10 と定義されたシーケンスは、最大値 9 を生成し、値 10 は生成しないはずですが。シーケンスに CYCLE を定義する際、MINVALUE、MAXVALUE、および START WITH の値への影響を考慮してください。
- シーケンス番号のキャッシュは、シーケンス番号の範囲を高速アクセスのためにメモリーに保管することを意味しています。アプリケーションが、次のシーケンス番号をキャッシュから割り振ることができるシーケンスにアクセスしていると、シーケンス番号の割り振りは素早く行われます。ただし、次のシーケンス番号をキャッシュから割り振ることができないシーケンスにアクセスしている場合、シーケンス番号の割り振りは、永続記憶域への入出力操作を待機しなければならない場合があります。CACHE の値を選択するとき、パフォーマンスとアプリケーション要件の関係を考慮しておく必要があります。
- シーケンスの定義者には、WITH GRANT OPTION 付きの ALTER および USAGE 特権が付与されます。シーケンスの所有者はシーケンスをドロップできます。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - コンマは、複数のシーケンス・オプションを分離するのに使用できます。
 - 以下の構文もサポートされています。
 - NOMINVALUE、NOMAXVALUE、NOCYCLE、NOCACHE、および NOORDER。

例

例 1: 1 で始まり、1 つずつ増分し、循環しない、同時に 24 の値をキャッシュに入れる ORG_SEQ というシーケンスを作成します。

```
CREATE SEQUENCE ORG_SEQ
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

CREATE SERVICE CLASS

CREATE SERVICE CLASS ステートメントは、サービス・クラスを定義します。

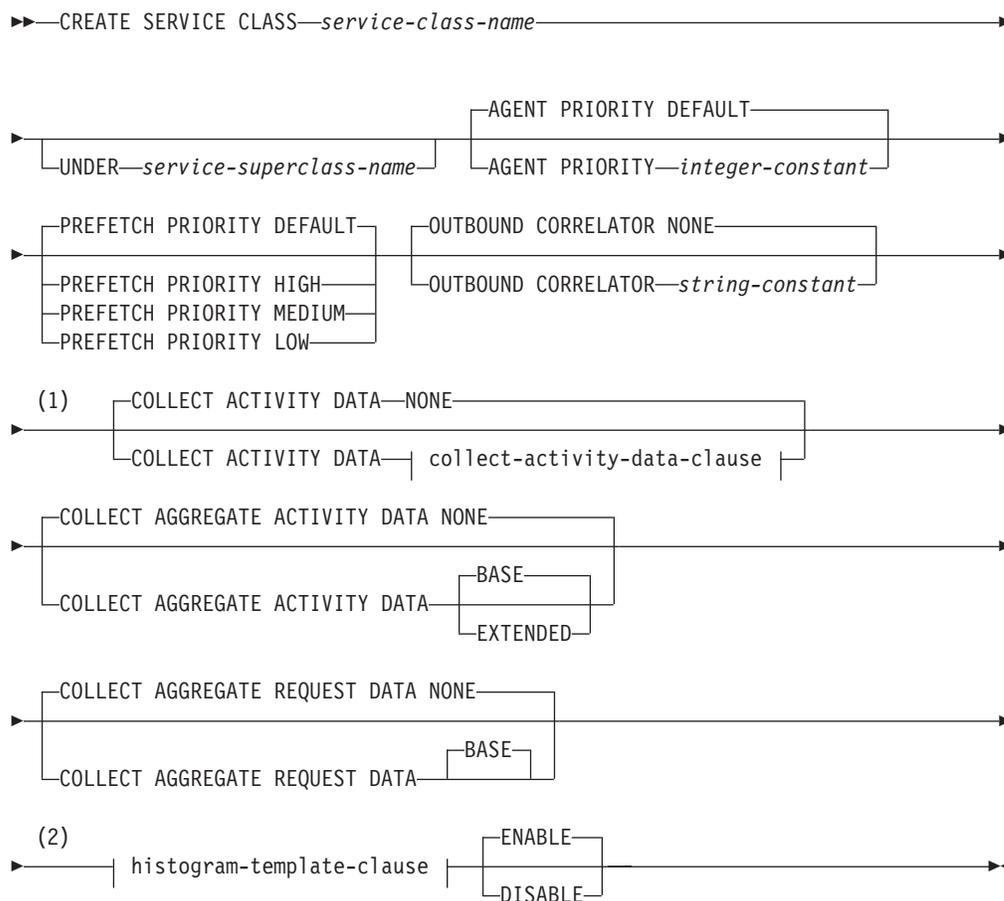
呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

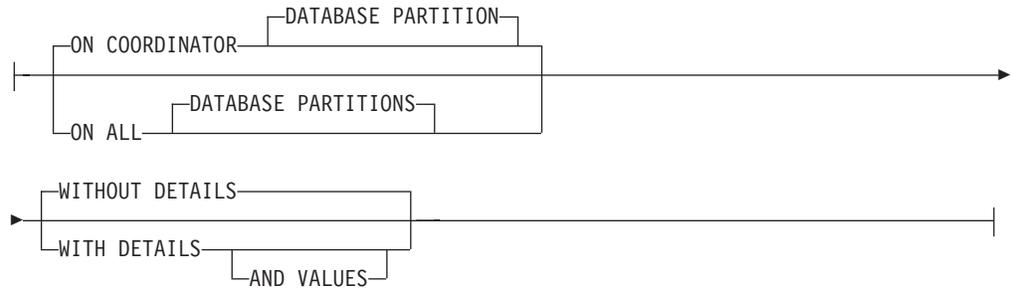
このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文

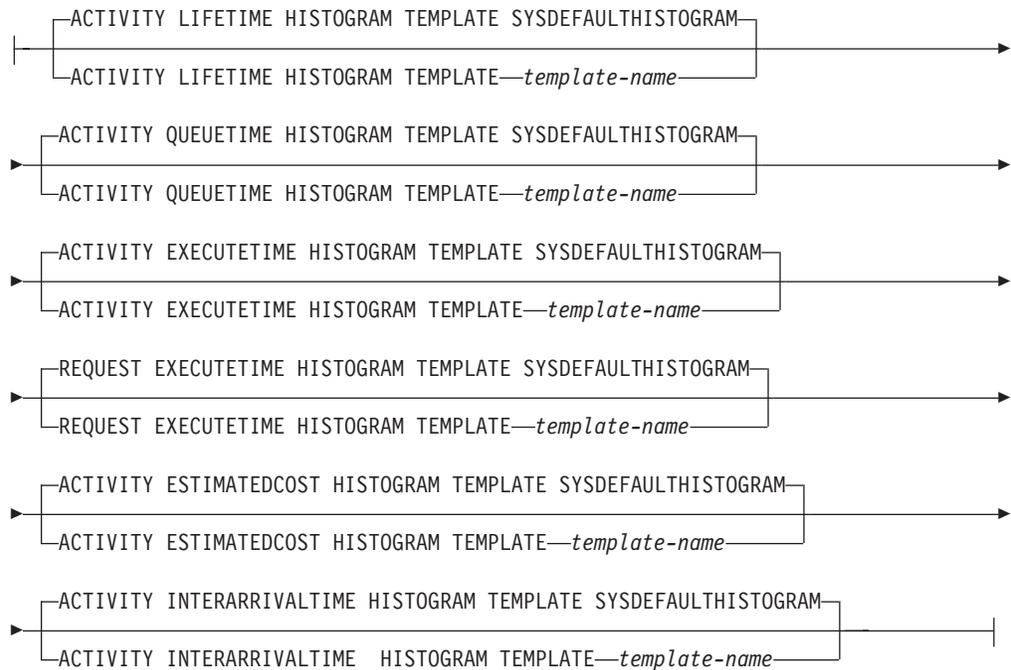


CREATE SERVICE CLASS

collect-activity-data-clause:



histogram-template-clause:



注:

- 1 COLLECT 節は、サービス・サブクラスでのみ有効です。
- 2 HISTOGRAM TEMPLATE 節は、サービス・サブクラスのみで有効です。

説明

service-class-name

サービス・クラスの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。このサービス・クラスがサービス・スーパークラスである場合、*service-class-name* には、既にカタログに存在するサービス・スーパークラスを指定してはなりません (SQLSTATE 42710)。このサービス・クラスがサービス・サブクラスである場合、*service-class-name* には、既にサービス・スーパークラスに存在するサービス・サブクラスを指定してはなりません (SQLSTATE 42710)。このサービス・クラスがサービス・サブク

ラスである場合、*service-class-name* は、そのサービス・スーパークラスと同じであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

UNDER *service-superclass-name*

このサービス・クラスがサービス・スーパークラス *service-superclass-name* のサブクラスであることを指定します。UNDER を指定しなかった場合、サービス・クラスはサービス・スーパークラスになります。*service-superclass-name* は、データベースに存在しているサービス・スーパークラスを指定していなければなりません (SQLSTATE 42704)。サービス・スーパークラスは、デフォルトのサービス・クラスにすることはできません (SQLSTATE 5U029)。

AGENT PRIORITY DEFAULT または **AGENT PRIORITY** *integer-constant*

サービス・クラスで実行されるエージェントの相対的な (差分) オペレーティング・システム優先順位、または DB2 で実行されるスレッドの通常優先順位を指定します。デフォルト値は DEFAULT です。DEFAULT に設定されている場合、特別なアクションは実行されず、サービス・クラスのエージェントは、オペレーティング・システムがすべての DB2 スレッドをスケジュールに入れる場合の通常優先順位に従ってスケジュールされます。このパラメーターが DEFAULT 以外の値に設定されている場合、エージェントは、通常優先順位に、次のアクティビティの開始時点の AGENT PRIORITY を加えた値に等しい優先順位に設定されます。例えば、通常優先順位が 20 で AGENT PRIORITY が -10 に設定されている場合、サービス・クラスのエージェントの優先順位は $20 - 10 = 10$ に設定されます。

UNIX オペレーティング・システムおよび Linux の場合、有効な値は DEFAULT、および -20 から 20 までです (SQLSTATE 42615)。負の値は高い相対優先順位を示します。正の値は低い相対優先順位を示します。

Windows オペレーティング・システムの場合、有効値は DEFAULT および -6 から 6 です (SQLSTATE 42615)。負の値になるほど相対的に低い優先順位を示します。正の値になるほど相対的に高い優先順位を示します。

サービス・サブクラスの AGENT PRIORITY が DEFAULT の場合、その親のスーパークラスの AGENT PRIORITY 値が継承されます。デフォルト・サブクラスの AGENT PRIORITY は変更できません (SQLSTATE 5U032)。

OUTBOUND CORRELATOR が設定されている場合は、AGENT PRIORITY を DEFAULT に設定する必要があります (SQLSTATE 42613)。

注: AIX では、サービス・クラスの中で AGENT PRIORITY を使用してエージェントにさらに高い相対優先順位を設定するには、インスタンス所有者に CAP_NUMA_ATTACH および CAP_PROPAGATE の能力がなければなりません。これらの能力を付与するには、root としてログオンし、以下のコマンドを実行します。

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

PREFETCH PRIORITY

このパラメーターは、サービス・クラス内のエージェントがプリフェッチ要求をサブミットできる優先順位を制御します。有効な値は HIGH、MEDIUM、LOW、または DEFAULT です (SQLSTATE 42615)。HIGH、MEDIUM、および LOW は、優先順位がそれぞれ高、中、および低の優先キューにプリフェッチ要求がサブミットされることを意味します。プリフェッチャーは、優先順位の

CREATE SERVICE CLASS

高いものから低いものへ、順に優先キューを空にします。サービス・クラスのエージェントは、次のアクティビティーの開始時に、**PREFETCH PRIORITY** レベルでプリフェッチ要求をサブミットします。プリフェッチ要求がサブミットされてから **PREFETCH PRIORITY** が変更された場合、要求優先順位は変更されません。デフォルト値は **DEFAULT** です。この設定は、内部的にサービス・スーパークラスの **MEDIUM** にマップされています。あるサービス・サブクラスについて **DEFAULT** が設定された場合、その親のスーパークラスの **PREFETCH PRIORITY** が継承されます。

デフォルト・サブクラスの **PREFETCH PRIORITY** は変更できません (SQLSTATE 5U032)。

OUTBOUND CORRELATOR NONE または **OUTBOUND CORRELATOR**

string-constant

このサービス・クラスのスレッドを外部ワークロード・マネージャー・サービス・クラスに関連付けるかどうかを指定します。

サービス・スーパークラスで **OUTBOUND CORRELATOR** が *string-constant* に設定されていて、サービス・サブクラスで **OUTBOUND CORRELATOR NONE** が設定されている場合、そのサービス・サブクラスは親の **OUTBOUND CORRELATOR** を継承します。AGENT PRIORITY が **DEFAULT** に設定されていない場合は、**OUTBOUND CORRELATOR** を **NONE** に設定する必要があります (SQLSTATE 42613)。デフォルトは **OUTBOUND CORRELATOR NONE** です。

OUTBOUND CORRELATOR NONE

サービス・スーパークラスの場合、このサービス・クラスに関連付けられた外部ワークロード・マネージャー・サービス・クラスがないことを指定します。サービス・サブクラスの場合、外部ワークロード・マネージャー・サービス・クラスとの関連がその親と同じであることを指定します。

OUTBOUND CORRELATOR *string-constant*

このサービス・クラスのスレッドを外部ワークロード・マネージャー・サービス・クラスに関連付けるための相関関係子として使用する *string-constant* を指定します。その外部ワークロード・マネージャーはアクティブでなければなりません (SQLSTATE 5U030)。その外部ワークロード・マネージャーは、*string-constant* の値を認識するように設定されている必要があります。

COLLECT ACTIVITY DATA

このサービス・クラスで実行する各アクティビティーに関する情報を、アクティビティー完了時にアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。デフォルトは **COLLECT ACTIVITY DATA NONE** です。COLLECT ACTIVITY DATA 節は、サービス・サブクラスでのみ有効です。

NONE

このサービス・クラスで実行する各アクティビティーについて、アクティビティー・データを収集しないことを指定します。

ON COORDINATOR DATABASE PARTITION

アクティビティーのコーディネーターのデータベース・パーティションでのみ、アクティビティー・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティーが処理されるすべてのデータベース・パーティションでアクティビティー・データを収集することを指定します。ただし、アクティビティーの詳細と値は、コーディネーターのデータベース・パーティションでのみ収集されます。

WITHOUT DETAILS

このサービス・クラスで実行される各アクティビティーに関するデータを、アクティビティーの実行完了時にアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境のデータを含むアクティビティーについて、これらのデータをアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティーに入力データ値が含まれている場合、それをアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

COLLECT AGGREGATE ACTIVITY DATA

このサービス・クラスについて、集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。COLLECT AGGREGATE ACTIVITY DATA を指定しない場合のデフォルトは、COLLECT AGGREGATE ACTIVITY DATA NONE です。COLLECT AGGREGATE ACTIVITY DATA を指定した場合のデフォルトは、COLLECT AGGREGATE ACTIVITY DATA BASE です。COLLECT AGGREGATE ACTIVITY DATA 節は、サービス・サブクラスでのみ有効です。

BASE

このサービス・クラスについて、基礎集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。基礎集約アクティビティー・データには以下のものが含まれます。

- アクティビティー・コストの最高水準点の見積もり
- 戻り行数の最高水準点
- TEMPORARY 表スペース使用量の最高水準点
- アクティビティー存続時間のヒストグラム
- アクティビティー・キュー時間のヒストグラム
- アクティビティー実行時間のヒストグラム

EXTENDED

このサービス・クラスについて、すべての集約アクティビティー・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。これには、すべての基礎集約アクティビティー・データに加えて、以下のものが含まれます。

CREATE SERVICE CLASS

- アクティビティ・データ操作言語 (DML) の見積コスト・ヒストグラム
- アクティビティ DML の到着間隔時間のヒストグラム

NONE

このサービス・クラスについて集約アクティビティ・データをキャプチャーしないことを指定します。

COLLECT AGGREGATE REQUEST DATA

このサービス・クラスについて、集約要求データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。デフォルトは COLLECT AGGREGATE REQUEST DATA NONE です。COLLECT AGGREGATE REQUEST DATA 節は、サービス・サブクラスでのみ有効です。

BASE

このサービス・クラスについて、基礎集約要求データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。

NONE

このサービス・クラスについて集約要求データをキャプチャーしないことを指定します。

histogram-template-clause

このサービス・クラスで実行されるアクティビティの集約アクティビティ・データを収集する時に使用されるヒストグラム・テンプレートを指定します。HISTOGRAM TEMPLATE 節は、1 つのサービス・サブクラスのみで有効です。

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 アクティビティの所要時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、キューに入っていた時間と実行時間の両方が含まれます。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 アクティビティがキューに入っている時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 アクティビティの実行のための所要時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、アクティビティがキューに入っていた時間は含ま

れません。このヒストグラムにおいて、アクティビティーの実行時間はコーディネーター・データベース・パーティションでのみ収集されます。アイドル時間はこの時間に含まれません。アイドル時間とは、要求が実行されてから同じアクティビティーに属する別の要求が実行されるまでの間、何も作業が実行されていない時間のことです。アイドル時間の一例として、カーソルのオープンが終了してからカーソルからのフェッチが開始するまでの間の時間があります。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE ACTIVITY DATA` 節と、その `BASE` または `EXTENDED` のどちらかのオプションが指定されている場合にのみ収集されます。ネスト・レベル 0 のアクティビティーのみがヒストグラムに含める対象となります。

REQUEST EXECUTETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で、このサービス・クラスで実行される DB2 要求の実行のための所要時間 (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、アクティビティーがキューに入っていた時間は含まれません。このヒストグラムにおいて要求実行時間は、要求が実行されるデータベース・パーティションごとに収集されます。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE REQUEST DATA` 節とその `BASE` オプションが指定されている場合にのみ収集されます。

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*

このサービス・クラスで実行される DML アクティビティーの見積コスト (timeron 単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE ACTIVITY DATA` 節とその `EXTENDED` オプションが指定されている場合にのみ収集されます。ネスト・レベル 0 のアクティビティーのみがヒストグラムに含める対象となります。

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*

1 つの DML アクティビティーの到着から次の DML アクティビティーの到着までの間の時間の長さ (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは `SYSDEFAULTHISTOGRAM` です。この情報は、`COLLECT AGGREGATE ACTIVITY DATA` 節とその `EXTENDED` オプションが指定されている場合にのみ収集されます。

ENABLE または DISABLE

接続とアクティビティーをサービス・クラスにマップできるかどうかを指定します。デフォルトは `ENABLE` です。

ENABLE

接続およびアクティビティーをサービス・クラスにマップできます。

DISABLE

接続およびアクティビティーをサービス・クラスにマップできません。

`DISABLE` (無効) に設定されているサービス・クラスに新しくマップされる接続やアクティビティーは拒否されます (`SQLSTATE 5U028`)。サービス・スーパークラスが無効として設定されている場合は、サービス・サブクラス

CREATE SERVICE CLASS

も無効になります。サービス・スーパークラスが再度有効になった場合、そのサービス・サブクラスはシステム・カタログで定義された状態に戻ります。デフォルト・サービス・クラスを無効にすることはできません (SQLSTATE 5U032)。

規則

- サービス・スーパークラスの下に作成できるサービス・サブクラスの最大数は 61 です (SQLSTATE 5U027)。
- データベースに作成できるサービス・スーパークラスの最大数は 64 です (SQLSTATE 5U027)。
- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U027)。WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- デフォルトのサブクラス SYSDEFAULTSUBCLASS は、すべてのサービス・スーパークラスで自動的に作成されます。
- 全パーティションを通じて、同時に実行できる非コミットの WLM 排他 SQL ステートメントは 1 つのみです。非コミットの WLM 排他 SQL ステートメントが実行されている場合、後続の WLM 排他 SQL ステートメントは、現行の WLM 排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、COMMIT ステートメントが実行されるまで有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

例 1: PETSALLES という名前のサービス・スーパークラスを作成します。PETSALLES のデフォルトのサブクラスは自動的に作成されます。

```
CREATE SERVICE CLASS PETSALLES
```

例 2: サービス・スーパークラス PETSALLES の下に DOGSALES という名前のサービス・サブクラスを作成します。サービス・クラス DOGSALES を使用不可に設定します。

CREATE SERVICE CLASS DOGSALES UNDER PETSALLES DISABLE

例 3 BARNSALES という名前のサービス・スーパークラスを、LOW のプリフェッチャー優先順位で作成します。BARNSALES のデフォルトのサブクラスは自動的に作成されます。BARNSALES サービス・クラスのエージェントによりサブミットされるプリフェッチ要求は、優先順位が低いプリフェッチ・キューに置かれます。

CREATE SERVICE CLASS BARNSALES PREFETCH PRIORITY LOW

CREATE SERVER

CREATE SERVER ステートメントは、データ・ソースをフェデレーテッド・データベースへ定義します。このステートメントでは、SERVER という語と、server- で始まるパラメーター名は、フェデレーテッド・システムでのデータ・ソースを指しています。そのようなシステムでのフェデレーテッド・サーバー、あるいは DRDA アプリケーション・サーバーを指すわけではありません。

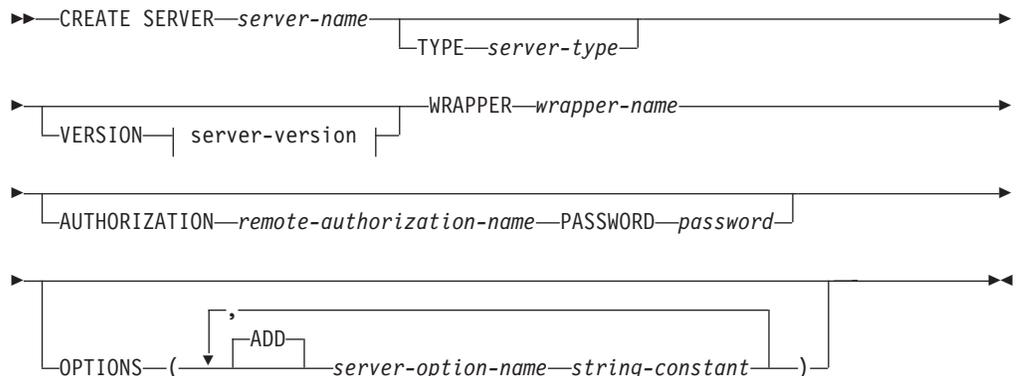
呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

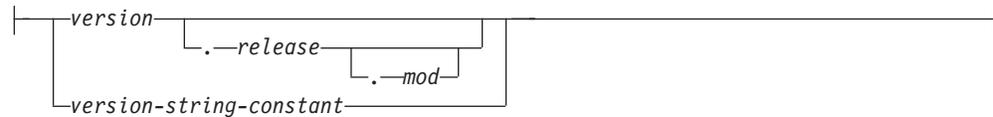
このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文



CREATE SERVER

server-version:



説明

server-name

フェデレーテッド・データベースに対して定義するデータ・ソースを指定します。この名前は、カタログに記述されているデータ・ソースを指すものであってはなりません。*server-name* は、フェデレーテッド・データベース内の表スペースの名前と同じであってはなりません。

リレーショナル・データ・ソースのサーバー定義は、通常リモート・データベースを表します。Oracle など一部のリレーショナル・データベース管理システムでは、各インスタンス内に複数のデータベースを取ることが許可されていません。その代わりに、各インスタンスはフェデレーテッド・システム内のサーバーを表します。

非リレーショナル・データ・ソースの場合、サーバー定義の目的はデータ・ソースごとに異なります。検索タイプやデーモン、Web サイト、または Web サーバーにマップされるサーバー定義もあります。その他の非リレーショナル・データ・ソースの場合、フェデレーテッド・オブジェクトの階層ではデータ・ソース・ファイル (ニックネームで識別される) が特定のサーバー・オブジェクトに関連付けられていることが必要なため、サーバー定義が作成されます。

TYPE *server-type*

server-name に示されるデータ・ソースのタイプを指定します。一部のラッパーには、このパラメーターが必須です。

VERSION

server-name に示されるデータ・ソースのバージョンを指定します。一部のラッパーには、このパラメーターが必須です。

version

バージョン番号を指定します。値は整数でなければなりません。

release

version で示されたバージョンのリリース番号を指定します。値は整数でなければなりません。

mod

release で示されたリリースのモディフィケーション番号を指定します。値は整数でなければなりません。

version-string-constant

バージョンの正式名称を指定します。*version-string-constant* は単一値 (例えば、'8i') にすることができます。あるいは、*version*、*release*、そして該当する場合は *mod* を連結した値にすることができます (例えば、'8.0.3')。

WRAPPER *wrapper-name*

server-name で指定されたサーバー・オブジェクトと対話するために、フェデレーテッド・サーバーが使用するラッパーの名前を指定します。

AUTHORIZATION *remote-authorization-name*

DB2 ファミリーのデータ・ソースにのみ必要です。CREATE SERVER ステートメントの処理時に、必要なアクションをデータ・ソースで実行するときの許可 ID を指定します。この許可 ID は、サーバーに対する後続の接続を確立するときには使用されません。

この ID には、実行するアクションに必要な権限 (BINDADD または同等の権限) がなければなりません。remote-authorization-name が大文字小文字混合または小文字で指定される場合 (かつりモート・データ・ソースに大/小文字の区別のある許可名がある場合) には、remote-authorization-name を二重引用符で囲んでください。

PASSWORD *password*

DB2 ファミリーのデータ・ソースにのみ必要です。remote-authorization-name で表された許可 ID に関連付けられているパスワードを指定します。password が大文字小文字混合または小文字で指定される場合 (かつりモート・データ・ソースに大/小文字の区別のあるパスワードがある場合) には、password を二重引用符で囲んでください。

OPTIONS

サーバー定義を作成したときに使用可能にされるオプションを指示します。サーバー・オプションはサーバー定義を構成するときに使用されます。任意のデータ・ソースのサーバー定義を作成するために使用できるサーバー・オプションもあります。特定のデータ・ソースに固有のサーバー・オプションもあります。

ADD

1 つ以上のサーバー・オプションを使用可能にします。

server-option-name

server-name で示されたデータ・ソースを構成するとき、あるいはそれについての情報を提供するときのいずれかに使うサーバー・オプションを指定します。

string-constant

server-option-name の設定を、文字ストリング定数として指定します。

注

- データ・ソースがパスワードを必要とする場合は、password を指定します。password のいずれかの文字が小文字であれば、password を引用符で囲みます。
- CREATE SERVER ステートメントを使って DB2 ファミリー・インスタンスをデータ・ソースとして定義する場合、DB2 で特定のパッケージをそのインスタンスにバインドする必要があるかもしれません。バインドする必要がある場合、ステートメント内の remote-authorization-name に BIND 権限がなければなりません。バインド操作の完了に要する時間は、データ・ソースの速度とネットワーク接続の速度によって異なります。
- DB2 は指定されたサーバー・バージョンがリモートのサーバー・バージョンと一致するかどうかを検査しません。正しくないサーバー・バージョンを指定すると、DB2 サーバー定義に属するニックネームにアクセスするとき、SQL エラーが生じることがあります。その可能性が最も高いのは、リモートのサーバー・バージョンよりも後のサーバー・バージョンを指定する場合です。その場合、サーバー定義に属するニックネームにアクセスすると、DB2 はリモート・サーバーが認識できない SQL を送信することがあります。

例

例 1: DB2 for z/OS and OS/390® バージョン 7.1 データ・ソースにアクセスするためのサーバー定義を登録します。CRANDALL は、DB2 for z/OS and OS/390 サーバー定義に割り当てられる名前です。DRDA は、このデータ・ソースにアクセスするために使用するラッパーの名前です。さらに、以下のものを指定します。

- GERALD および drowssap は、このステートメントの処理時に、パッケージを CRANDALL でバインドするときの許可 ID とパスワードです。
- CATALOG DATABASE ステートメントに指定された DB2 for z/OS and OS/390 データベースの別名は、CLIENTS390 です。
- CRANDALL にアクセスするときの許可 ID とパスワードは、CRANDALL へ大文字で送信されます。
- CLIENTS390 とフェデレーテッド・データベースは、同じ照合順序を使用します。

```
CREATE SERVER CRANDALL
  TYPE DB2/ZOS
  VERSION 7.1
  WRAPPER DRDA
  AUTHORIZATION "GERALD"
  PASSWORD drowssap
  OPTIONS
    (DBNAME 'CLIENTS390',
     FOLD_ID 'U',
     FOLD_PW 'U',
     COLLATING_SEQUENCE 'Y')
```

例 2: Oracle 9 データ・ソースにアクセスするためのサーバー定義を登録します。CUSTOMERS は、Oracle サーバー定義に割り当てられる名前です。NET8 は、このデータ・ソースにアクセスするために使用するラッパーの名前です。さらに、以下のものを指定します。

- ABC は、Oracle データベース・サーバーが置かれているノードの名前です。
- フェデレーテッド・サーバーの CPU の動作速度が、CUSTOMERS をサポートする CPU の 2 倍であること。
- フェデレーテッド・サーバーの入出力装置のデータ処理速度が、CUSTOMERS の入出力装置の 1.5 倍であること。

```
CREATE SERVER CUSTOMERS
  TYPE ORACLE
  VERSION 9
  WRAPPER NET8
  OPTIONS
    (NODE 'ABC',
     CPU_RATIO '2.0',
     IO_RATIO '1.5')
```

例 3: Excel ラッパーのサーバー定義を登録します。このサーバー定義は、フェデレーテッド・オブジェクトの階層を保持するために必要です。BIOCHEM_LAB は Excel サーバー定義に割り当てられる名前です。EXCEL_2000_WRAPPER は、このデータ・ソースにアクセスするために使用するラッパーの名前です。

```
CREATE SERVER BIOCHEM_DATA
  WRAPPER EXCEL_2000_WRAPPER
```

例 4: BLAST データ・ソースにアクセスするためのサーバー定義を登録します。BLAST_SERVER は BLAST サーバー定義に割り当てられる名前です。このサーバ

一定義がサポートする検索のタイプは、BLASTn 検索タイプです。VERSION は BLAST 検索プログラムのバージョンです。BLAST_WRAPPER は、このデータ・ソースにアクセスするために使用するラッパーの名前です。さらに、以下のものを指定します。

- NODE は、BLAST デーモン・プロセスを実行するサーバーのホスト名です。
- BLAST デーモンが、BLAST ラッパーによってサブミットされたジョブを listen するときに使用するポート番号は、4007 です。

```
CREATE SERVER BLAST_SERVER
  TYPE BLASTn
  VERSION 2.1.2
  WRAPPER BLAST_WRAPPER
  OPTIONS
    (NODE 'big.rs.company.com',
     DAEMON_PORT '4007')
```

CREATE TABLE

CREATE TABLE ステートメントは表を定義します。定義には、その表の名前と、その列の名前および属性を含める必要があります。定義には、主キーやチェック制約など、表の他の属性を含めることができます。

グローバル一時表を宣言するには、DECLARE GLOBAL TEMPORARY TABLE ステートメントを使用します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

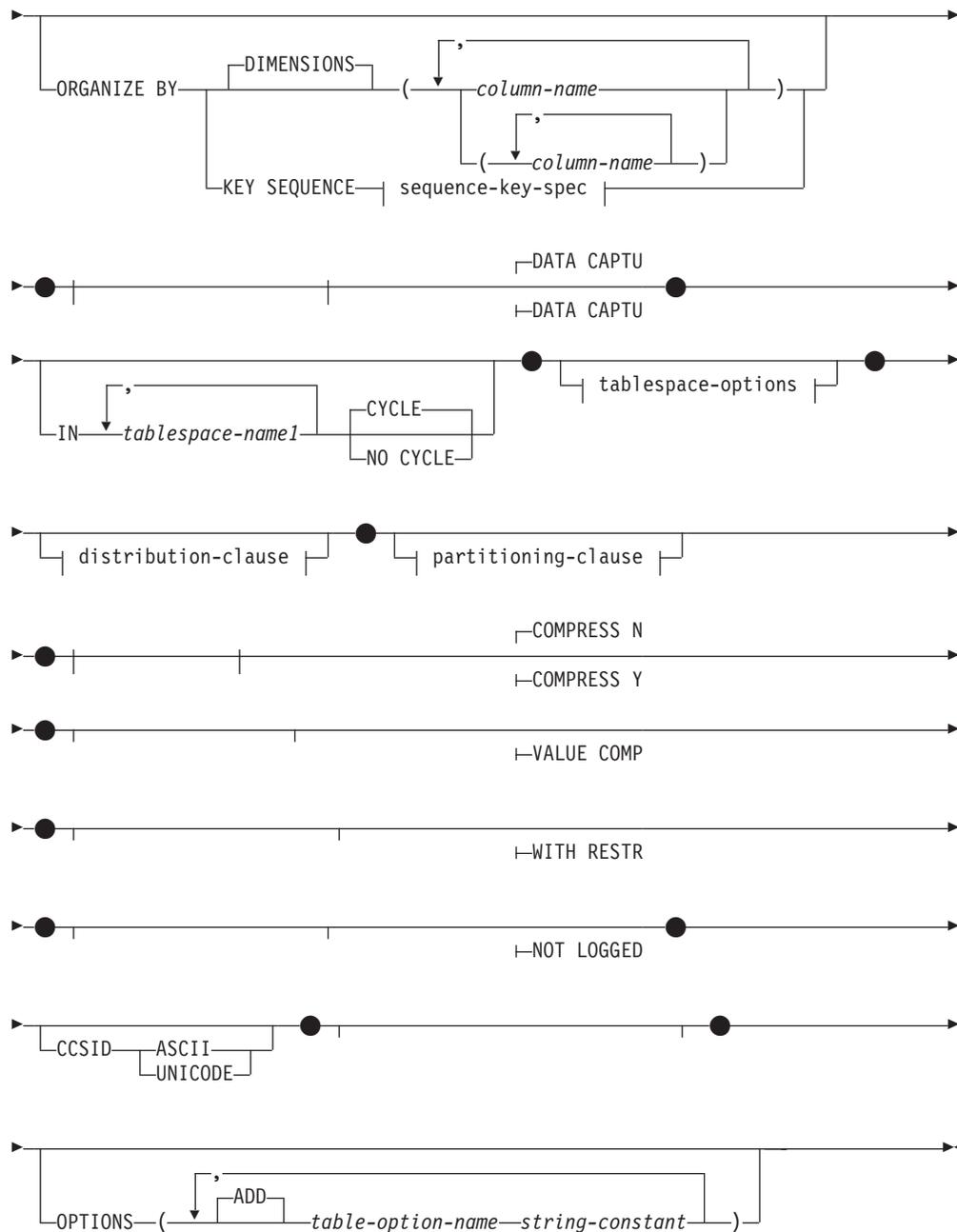
- データベースに対する CREATETAB 権限、および表スペースに対する USE 特権に加えて、以下のいずれか。
 - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙的または明示的スキーマ名が存在しない場合)
 - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

副表を定義する場合には、許可 ID は表階層のルート表の所有者と同じでなければなりません。

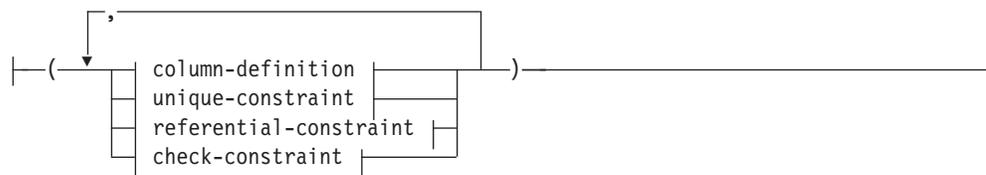
外部キーを定義する場合には、ステートメントの許可 ID が保持する特権として、親表に対する以下のいずれかが必要になります。

- その表に対する REFERENCES 特権
- 指定の親キーのそれぞれの列に対する REFERENCES 特権

CREATE TABLE

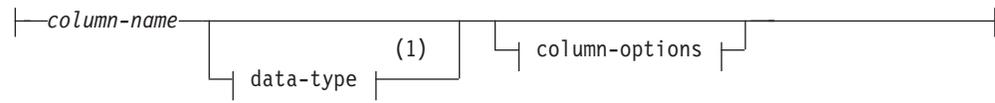


element-list:



CREATE TABLE

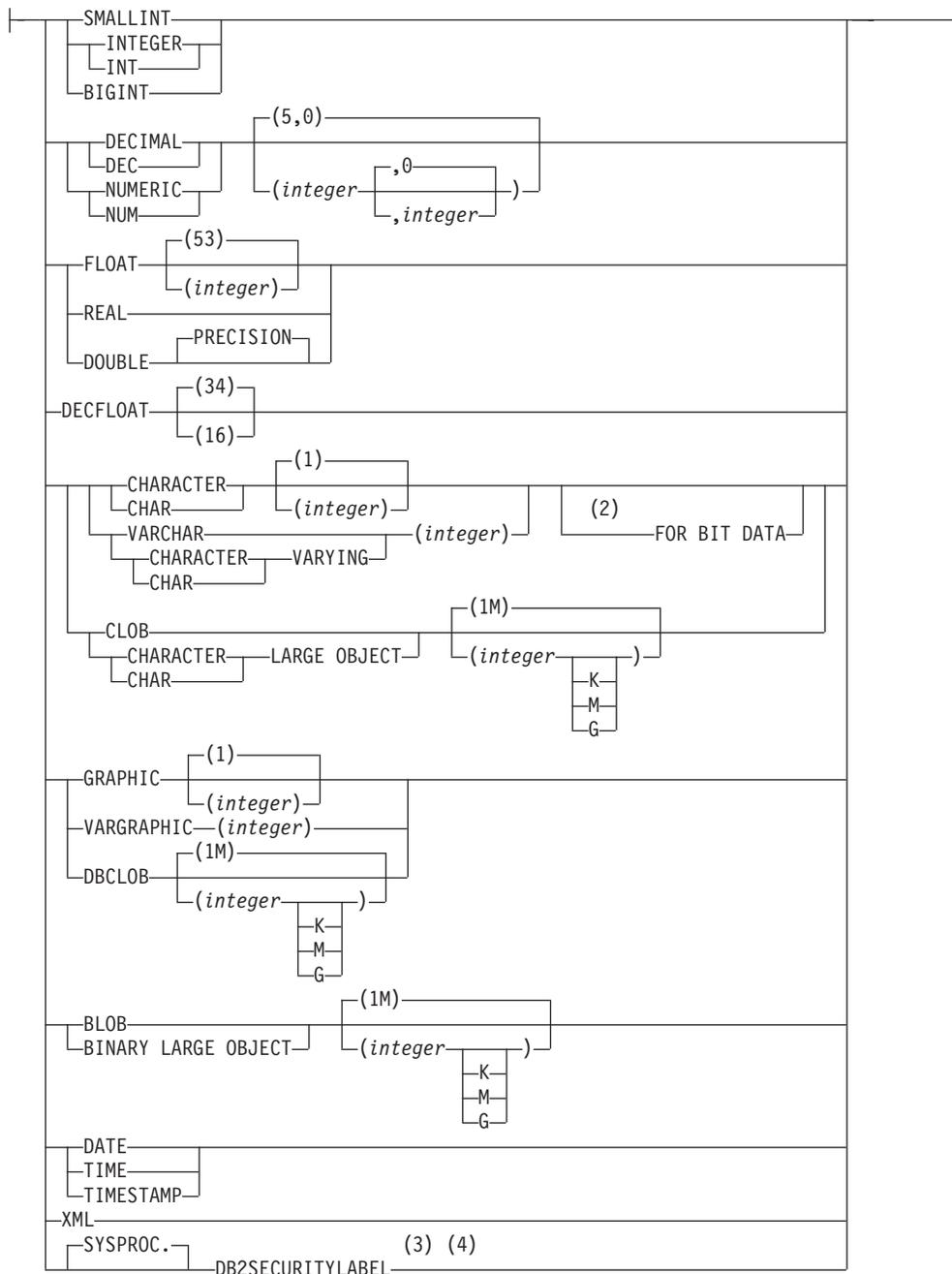
column-definition:



data-type:

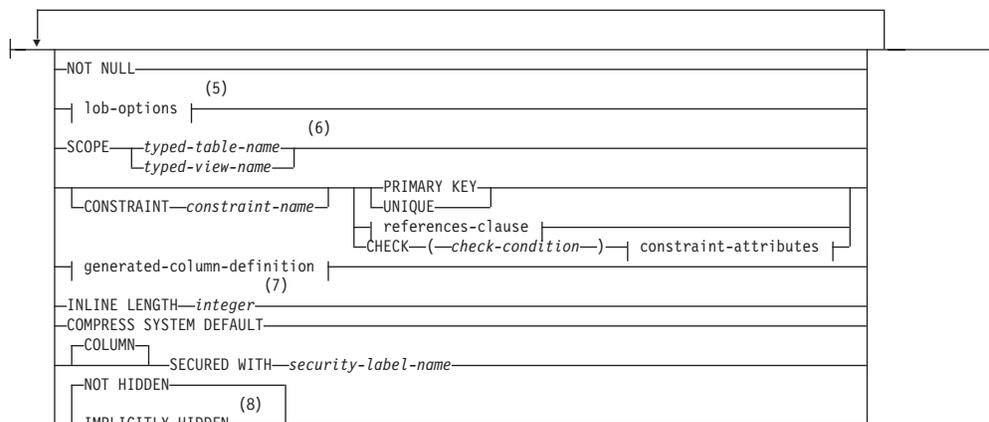


built-in-type:

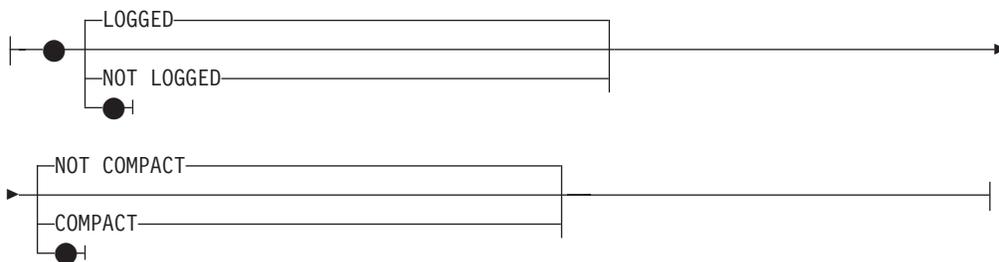


column-options:

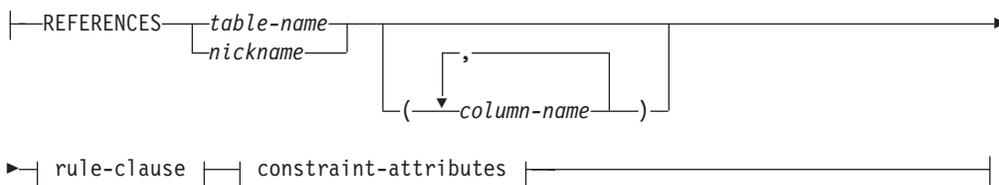
CREATE TABLE



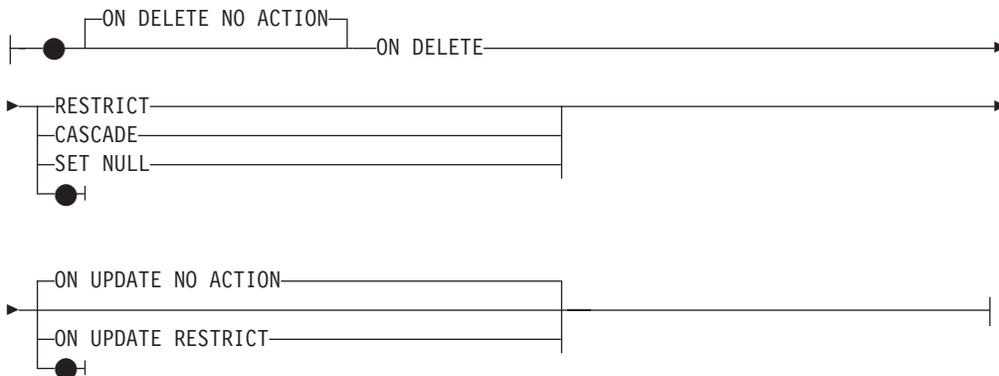
lob-options:



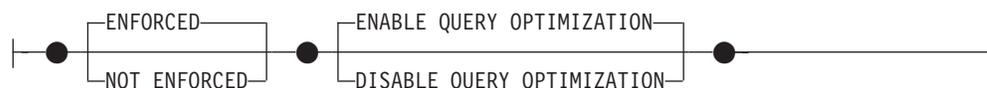
references-clause:



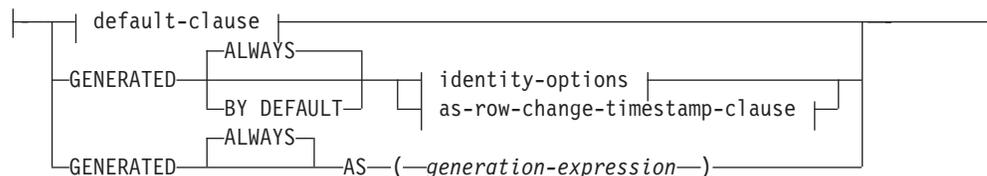
rule-clause:



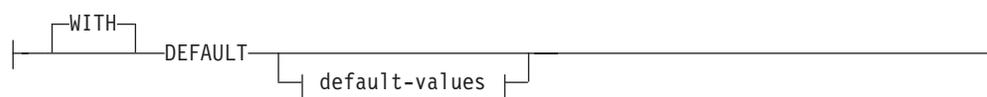
constraint-attributes:



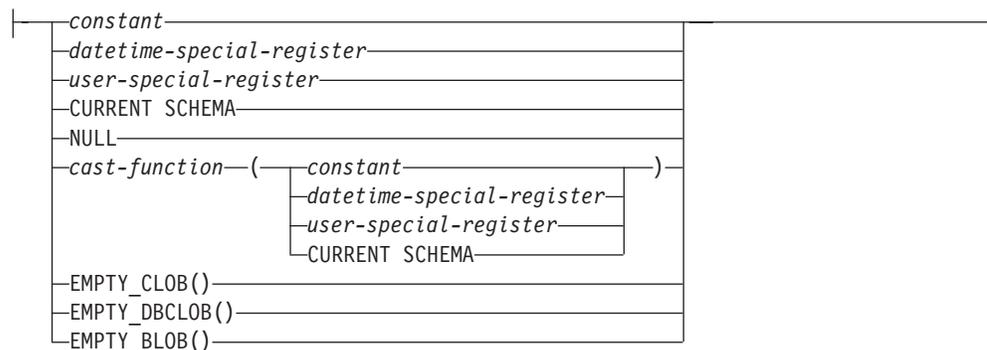
generated-column-definition:



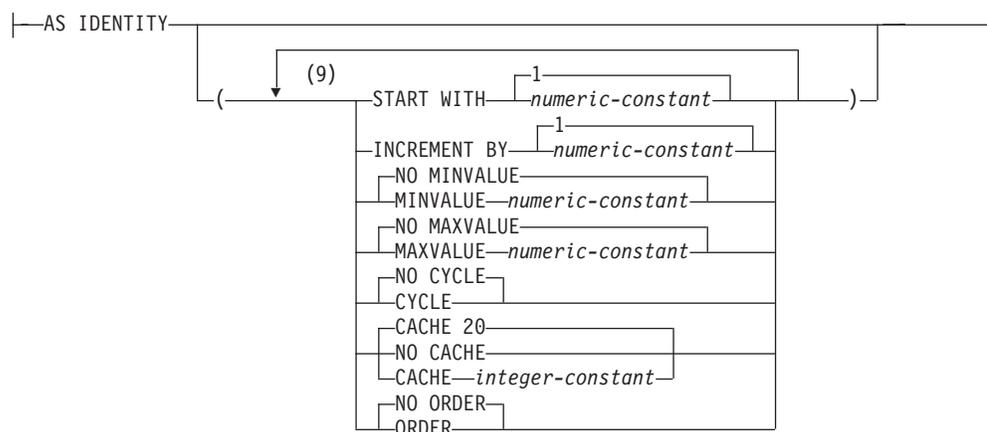
default-clause:



default-values:



identity-options:



CREATE TABLE

as-row-change-timestamp-clause:

(10)
FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

unique-constraint:

CONSTRAINT *constraint-name* UNIQUE
PRIMARY KEY (*column-name*)

referential-constraint:

CONSTRAINT *constraint-name* FOREIGN KEY (*column-name*)
references-clause

check-constraint:

CONSTRAINT *constraint-name* CHECK (*check-condition*)
constraint-attributes

check-condition:

search-condition
functional-dependency

functional-dependency:

column-name DETERMINED BY *column-name*
(*column-name*)

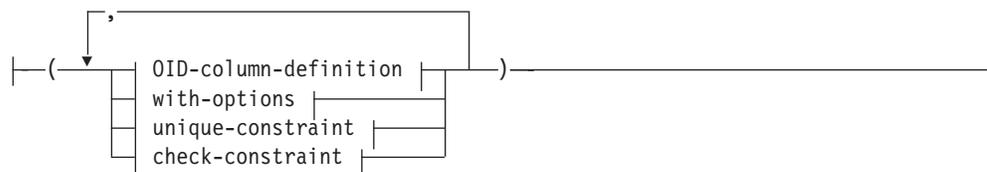
typed-table-options:

HIERARCHY *hierarchy-name*
under-clause typed-element-list

under-clause:

|—UNDER—*supertable-name*—INHERIT SELECT PRIVILEGES—|

typed-element-list:



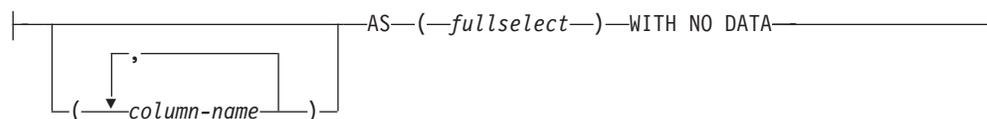
OID-column-definition:

|—REF IS—*OID-column-name*—USER GENERATED—|

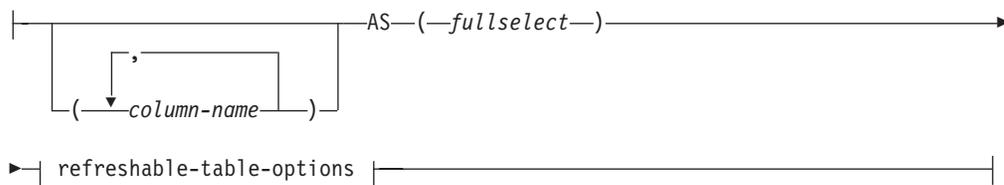
with-options:

|—*column-name*—WITH OPTIONS—| *column-options* |

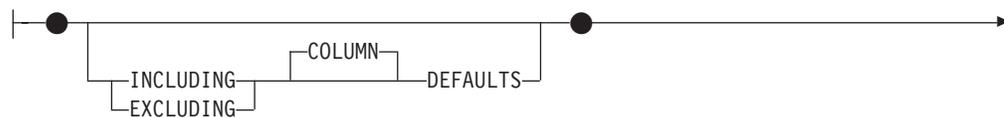
as-result-table:



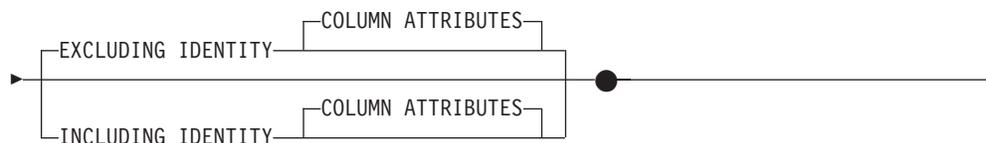
materialized-query-definition:



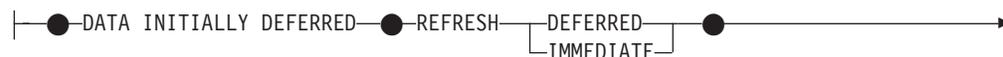
copy-options:



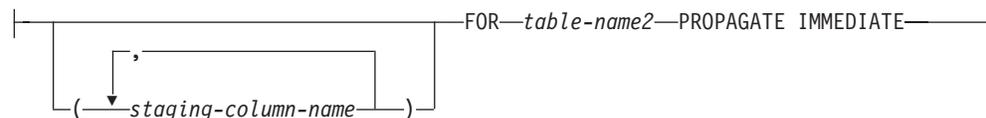
CREATE TABLE



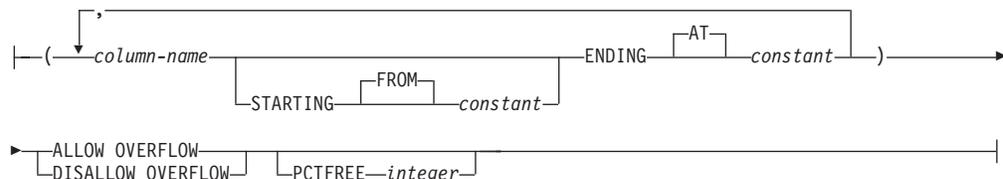
refreshable-table-options:



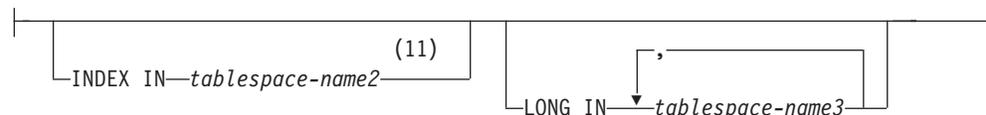
staging-table-definition:



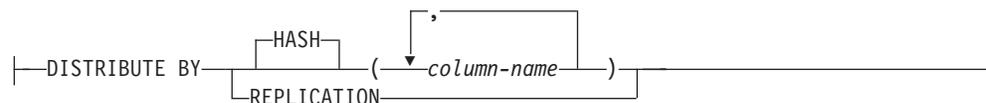
sequence-key-spec:



tablespace-options:



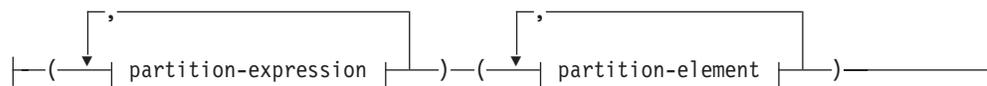
distribution-clause:



partitioning-clause:



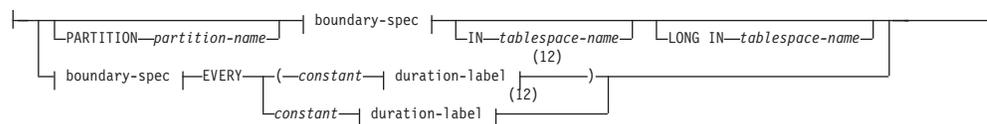
range-partition-spec:



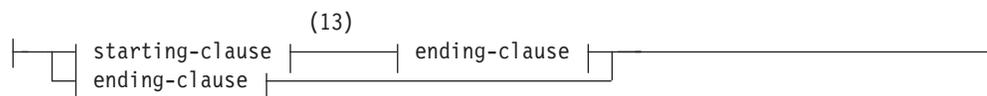
partition-expression:



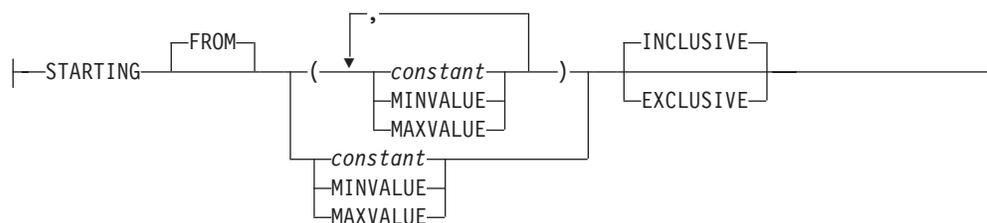
partition-element:



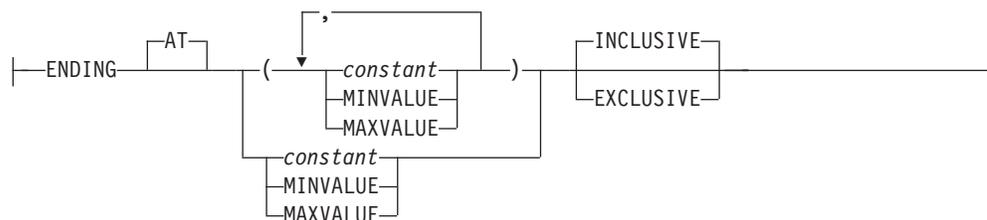
boundary-spec:



starting-clause:



ending-clause:



CREATE TABLE

duration-label:

YEAR
YEARS
MONTH
MONTHS
DAY
DAYS
HOUR
HOURS
MINUTE
MINUTES
SECOND
SECONDS
MICROSECOND
MICROSECONDS

注:

- 1 選択する最初の column-option が、 generation-expression を指定した generated-column-definition の場合、 data-type を省略することができます。これは、 generation-expression の処理結果のデータ・タイプから判別されます。
- 2 FOR BIT DATA 節とその後に続く他の列制約とは、任意の順序で指定できます。
- 3 DB2SECURITYLABEL は、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。
- 4 タイプ DB2SECURITYLABEL の列の場合、NOT NULL WITH DEFAULT は暗黙指定になるので、明示的に指定することはできません (SQLSTATE 42842)。タイプ DB2SECURITYLABEL の列のデフォルト値は、セッション許可 ID の書き込みアクセスのためのセキュリティー・ラベルです。
- 5 lob-options (LOB オプション) 節は、ラージ・オブジェクト・タイプ (BLOB、CLOB、および DBCLOB) と、ラージ・オブジェクト・タイプに基づく特殊タイプに対してのみ適用されます。
- 6 SCOPE 節は REF タイプに対してのみ適用されます。
- 7 INLINE LENGTH は、構造化タイプまたは XML タイプとして定義された列に対してのみ用います。
- 8 IMPLICITLY HIDDEN を指定できるのは、ROW CHANGE TIMESTAMP も指定される場合のみです。
- 9 同じ節を複数回指定することはできません。
- 10 行変更タイム・スタンプ列のデータ・タイプはオプションです。
- 11 どの表スペースに表の索引を組み込むかは、表を作成するときに指定できます。表が範囲パーティション表の場合、索引表スペースは、CREATE INDEX ステートメントの IN 節で指定できます。
- 12 この partition-element の構文は、数値または日時データ・タイプを伴う partition-expression が 1 つだけ存在する場合に有効です。
- 13 最初の partition-element には starting-clause が、最後の partition-element には ending-clause が含まれている必要があります。

説明

システム保守済みマテリアライズ照会表とユーザー保守済みマテリアライズ照会表は、それぞれを個別に識別する必要が生じない限り、どちらもマテリアライズ照会表と呼びます。

table-name

表の名前を指定します。暗黙または明示の修飾子を含む名前は、カタログに記述されている表、ビュー、ニックネーム、または別名を指定するものであってはなりません。スキーマ名は SYSIBM、SYSCAT、SYSFUN、または SYSSTAT であってはなりません (SQLSTATE 42939)。

element-list

表のエレメントを定義します。これには、表の列と制約の定義が含まれます。

column-definition

列の属性を定義します。

column-name

表を構成する列の名前を指定します。名前を修飾したり、表の複数の列に対して同じ名前を使用することはできません (SQLSTATE 42711)。

表には、以下のものを指定できます。

- 4K ページ・サイズの場合、最大 500 列。列のバイト・カウントは 4 005 を超えてはなりません。
- 8K ページ・サイズの場合、最大 1 012 列。列のバイト・カウントは 8 101 を超えてはなりません。
- 16K ページ・サイズの場合、最大 1 012 列。列のバイト・カウントは 16 293 を超えてはなりません。
- 32K ページ・サイズの場合、最大 1 012 列。列のバイト・カウントは 32 677 を超えてはなりません。

詳細については、行サイズを参照してください。

data-type

列のデータ・タイプを指定します。

built-in-type

組み込みタイプとして、以下のいずれかのタイプを使用します。

SMALLINT

短精度整数。

INTEGER または INT

長精度整数。

BIGINT

64 ビット整数。

DECIMAL(precision-integer, scale-integer) または DEC(precision-integer, scale-integer)

10 進数。最初の整数は数値の精度、つまり数字の総桁数です。これは、1 から 31 の範囲で指定できます。2 番目の整数は、数値の位取り、つまり、小数点の右側の桁数です。これは、0 から数値の精度までの範囲で指定できます。

CREATE TABLE

精度と位取りが指定されない場合、5,0 のデフォルト値が使用されます。 **NUMERIC** および **NUM** は、 **DECIMAL** および **DEC** の同義語として使用可能です。

FLOAT(*integer*)

単精度または倍精度の浮動小数点数 (*integer* の値によって異なる)。 *integer* の値は、1 から 53 の範囲の整数でなければなりません。1 から 24 の値は単精度、25 から 53 の値は倍精度を示します。

また、以下を指定することもできます。

REAL 単精度浮動小数点。

DOUBLE

倍精度浮動小数点。

DOUBLE PRECISION

倍精度浮動小数点。

FLOAT

倍精度浮動小数点。

DECFLOAT(*precision-integer*)

10 進浮動小数点数。 *precision-integer* の値は数値の精度です。つまり数字の総桁数で 16 または 34 です。

精度を指定しない場合、デフォルト値 34 が使用されます。

CHARACTER(*integer*) または **CHAR**(*integer*) または **CHARACTER** または **CHAR**

長さ *integer* (整数) の固定長文字ストリング。長さは、1 から 254 の範囲で指定できます。長さの指定がない場合は、1 文字の長さを指定したものと見なされます。

VARCHAR(*integer*)、または **CHARACTER VARYING**(*integer*)、または **CHAR VARYING**(*integer*)

最大長が *integer* の可変長文字ストリング。最大長は、1 から 32 672 の範囲で指定できます。

FOR BIT DATA

列の内容をビット (バイナリー) データとして扱うように指定します。他のシステムとのデータ交換の過程で、コード・ページ変換は行われません。比較は、データベース照合シーケンスに関係なくバイナリーで行われます。

CLOB または **CHARACTER (CHAR) LARGE OBJECT**(*integer* [*K* | *M* | *G*])

文字ラージ・オブジェクト・ストリング (最大長をバイト単位で指定)。

integer *K* | *M* | *G* の意味は、BLOB の場合と同じです。

長さの指定がない場合、長さが 1 048 576 (1 メガバイト) であると想定されます。

1 ギガバイトを超える CLOB ストリングを作成するには、 **NOT LOGGED** オプションを指定する必要があります。

CLOB 列の場合に、FOR BIT DATA 節を指定することはできません。ただし、CLOB 列に CHAR FOR BIT DATA ストリングを割り当てることができ、CLOB ストリングに CHAR FOR BIT DATA ストリングを連結することができます。

GRAPHIC(*integer*)

長さ *integer* (整数) の固定長 GRAPHIC ストリング。長さは、1 から 127 の範囲で指定できます。長さの指定がない場合、長さが 1 であると想定されます。

VARGRAPHIC(*integer*)

最大長が *integer* の可変長 GRAPHIC ストリング。長さは、1 から 16 336 の範囲で指定できます。

DBCLOB(*integer* [*K* | *M* | *G*])

2 バイト文字ラージ・オブジェクト・ストリング (最大長を 2 バイト文字の数で指定)。

integer *K* | *M* | *G* の意味は、BLOB の場合に類似しています。指定する数値が 2 バイト文字 1 個を 1 文字と数えた値であることと、最大サイズが 2 バイト文字 1 073 741 823 個であるという点が違います。

長さの指定がない場合、長さが 1 048 576 (2 バイト文字) であると想定されます。

1 ギガバイトを超える DBCLOB ストリングを作成するには、NOT LOGGED オプションを指定する必要があります。

BLOB または BINARY LARGE OBJECT(*integer* [*K* | *M* | *G*])

バイナリー・ラージ・オブジェクト・ストリング (最大長をバイト単位で指定)。

長さは、1 から 2 147 483 647 バイトの範囲で指定できます。

integer (整数) だけを指定した場合は、それが最大長になります。

integer *K* (大文字または小文字) を指定した場合、最大長は *integer* の 1 024 倍になります。 *integer* の最大値は 2 097 152 です。

integer *M* を指定した場合、最大長は *integer* の 1 048 576 倍になります。 *integer* の最大値は 2 048 です。

integer *G* を指定した場合、最大長は *integer* の 1 073 741 824 倍になります。 *integer* の最大値は 2 です。

計算結果が 2 147 483 648 を超える *K*、*M*、または *G* の倍数を指定すると、使用される実際の値は 2 147 483 647 (2 ギガバイト - 1 バイト) になります。これは LOB 列の最大長です。

長さの指定がない場合、長さが 1 048 576 (1 メガバイト) であると想定されます。

1 ギガバイトを超える BLOB ストリングを作成するには、NOT LOGGED オプションを指定する必要があります。

CREATE TABLE

integer と K、M、または G の間には、任意の数のスペースを使用できます。スペースは必須ではありません。例えば、次の例はすべて有効です。

BLOB(50K) BLOB(50 K) BLOB (50 K)

DATE

日付を示します。

TIME

時刻を示します。

TIMESTAMP

タイム・スタンプを示します。

XML

XML 文書を示します。XML 列には整形 XML 文書だけを挿入できます。データベース・インスタンスに定義されたデータベース・パーティションが 1 つの場合のみ、列のタイプとして XML が可能になります (SQLSTATE 42997)。

XML 列には、以下の制限事項があります。

- 列は、XML データに対する索引以外の索引の一部であってはなりません。したがって、この列を主キーまたはユニーク制約の列として組み込むことはできません (SQLSTATE 42962)。
- 列は参照制約の外部キーであってはなりません (SQLSTATE 42962)。
- この列にデフォルト値 (WITH DEFAULT) を指定することはできません (SQLSTATE 42613)。列が NULL 可能な場合、その列のデフォルトは NULL 値です。
- その列は、分散キーを設定した表では使用できません (SQLSTATE 42997)。
- その列は、範囲クラスター表では使用できません (SQLSTATE 429BG)。
- その列は、範囲パーティション表では使用できません (SQLSTATE 42997)。
- その列は、VALIDATED 述部以外のチェック制約では参照できません (SQLSTATE 42621)。

タイプ XML の列を作成すると、その列に対する XML パス索引が作成されます。タイプ XML の最初の列を作成するときには、表レベルの XML 領域索引も作成されます。これらの索引の名前は、'SQL' の後に文字タイム・スタンプ (yyymmddhhmmssxxx) を付けた形になります。スキーマ名は SYSIBM です。

SYSPROC.DB2SECURITYLABEL

これは、保護対象表の行セキュリティー・ラベル列を定義するために使用しなければならない組み込み特殊タイプです。組み込み特殊タイプ DB2SECURITYLABEL の列の基礎データ・タイプは、VARCHAR(128) FOR BIT DATA です。1 つの表にはタイプ DB2SECURITYLABEL の列を最大で 1 個しか組み込めません (SQLSTATE 428C1)。

distinct-type-name

ユーザー定義タイプの中で特殊タイプであるものを示します。スキーマ名を伴わない特殊タイプ名を指定した場合、その特殊タイプ名は SQL パスのスキーマから探索することによって解決されます (このパスは、静的 SQL の場合は FUNCSPATH プリプロセス・オプションによって、動的 SQL の場合は CURRENT PATH レジスターによって定義されます)。

特殊タイプを使用して列を定義する場合、その列のデータ・タイプはその特殊タイプになります。列の長さや位取りは、それぞれ特殊タイプのソース・タイプの長さや位取りになります。

特殊タイプを使用して定義された列が参照制約の外部キーである場合、主キーの対応する列のデータ・タイプは、同じ特殊タイプでなければなりません。

structured-type-name

ユーザー定義タイプの中で構造化タイプであるものを示します。構造化タイプ名の指定にスキーマ名が含まれていない場合、その構造化タイプ名は SQL パスのスキーマから探索することによって決まります (このパスは、静的 SQL の場合は FUNCSPATH 前処理オプションによって、動的 SQL の場合は CURRENT PATH レジスターによって定義されます)。

構造化タイプを使用して列を定義する場合、その列の静的データ・タイプはその構造化タイプになります。その列には、*structured-type-name* のサブタイプである動的タイプをもつ値を組み込むことができます。

構造化タイプを使用して定義された列を、主キー、ユニーク制約、外部キー、索引キー、または分散キー内で使うことはできません (SQLSTATE 42962)。

列が、構造化タイプを使用して定義されていて、ネストのいずれかのレベルで参照タイプ属性をもっている場合、その参照タイプ属性の有効範囲は解除されます。そのような属性を間接参照操作で使用するには、CAST 指定を使って SCOPE を明示的に指定する必要があります。

REF (*type-name2*)

型付き表への参照。 *type-name2* の指定にスキーマ名が含まれていない場合、そのタイプ名は SQL パス上のスキーマを探索することによって決まります (このパスは、静的 SQL の場合は FUNCSPATH プリプロセス・オプションによって、動的 SQL の場合は CURRENT PATH レジスターによって定義されます)。この列の基礎を成すデータ・タイプは、CREATE TYPE ステートメントの REF USING 節で *type-name2* に対して指定された表示データ・タイプに基づくか、または *type-name2* の入ったデータ・タイプ階層のルート・タイプに基づきます。

column-options

表の列に関連した追加オプションを定義します。

NOT NULL

列に NULL 値が入るのを防止します。

CREATE TABLE

NOT NULL を指定しない場合、列に NULL 値を含めることができます。また、そのデフォルト値は、NULL 値または WITH DEFAULT 節で指定される値のいずれかになります。

NOT HIDDEN または IMPLICITLY HIDDEN

列を隠し列と定義するかどうかを指定します。列を表の暗黙的参照に組み込むかどうか、SQL ステートメントで明示的に参照できるかどうかは隠し属性によって決まります。デフォルトは NOT HIDDEN です。

NOT HIDDEN

列を表の暗黙的参照に組み込むこと、および列を明示的に参照できることを指定します。

IMPLICITLY HIDDEN

名前で明示的に参照されない限り列は SQL ステートメントから不可視であることを指定します。例えば、表に IMPLICITLY HIDDEN 節によって定義された列が組み込まれていると仮定した場合、暗黙的に隠された列は SELECT * の結果に組み込まれません。しかし、暗黙的に隠された列の名前を明示的に参照する SELECT の結果については、結果表にその列が組み込まれます。

IMPLICITLY HIDDEN は、ROW CHANGE TIMESTAMP 列にのみ指定する必要があります (SQLSTATE 42867)。ROW CHANGE TIMESTAMP FOR *table-designator* 式は、IMPLICITLY HIDDEN ROW CHANGE TIMESTAMP 列に解決されます。

IMPLICITLY HIDDEN を表のすべての列に指定することはできません (SQLSTATE 428GU)。

lob-options

LOB データ・タイプのオプションを指定します。

LOGGED

列に対して行われた変更をログに書き込むことを指定します。このような列のデータは、データベース・ユーティリティー (RESTORE DATABASE など) によってリカバリー可能です。LOGGED はデフォルト値です。

1 ギガバイトを超える LOB はログ記録することができません (SQLSTATE 42993)。

NOT LOGGED

列に対して行われた変更をログに書き込まないことを指定します。

NOT LOGGED は、コミットやロールバックの操作には影響しません。つまり、トランザクションがロールバックされても、LOB の値がログ記録されるか否かに関係なくデータベースの整合性は保持されます。ロギングされないため、ロールフォワード操作中、バックアップまたはロード操作の後の LOB データは、ロールフォワード操作中にログ・レコードを再生させることになった LOB 値をゼロで置換したものになります。クラッシュ・リカバリーの過程で、コミットされた変更とロールバックされた変更すべてに、予期された結果が反映されます。

COMPACT

後続の付加操作で使用するためのスペースを LOB ストレージ域の最後に残すことなく、LOB 列の値で消費されるディスク・スペースを最小限にすることを指定します (LOB 値が使用する最後のグループ内の余分なディスク・スペースすべてを解放します)。このようにしてデータを保管した場合、列に対する付加操作 (長さを増加する操作) でパフォーマンスが低下することがあります。

NOT COMPACT

列の LOB 値に対する将来の変更に備えて、いくらかのスペースを挿入するように指定します。これはデフォルトです。

SCOPE

参照タイプ列の有効範囲を指定します。

間接参照演算子の左オペランド、または Deref 関数の引数として使用する列には、すべて有効範囲を指定する必要があります。ターゲット表が定義されるように、後続する ALTER TABLE ステートメントまで参照タイプ列の指定を遅らせることができます (通常は、相互参照表の場合に適用する)。

typed-table-name

型付き表の名前。この表は既に存在しているものか、作成する表と同じ名前のものでなければなりません (SQLSTATE 42704)。

column-name のデータ・タイプは REF(S) でなければなりません。

S は *typed-table-name* のタイプを表します (SQLSTATE 428DM)。

column-name に割り当てられた値が、*typed-table-name* に存在する行を実際に参照しているかどうかを示す検査は行われません。

typed-view-name

型付きビューの名前。このビューは既に存在しているものか、作成するビューと同じ名前のものでなければなりません (SQLSTATE 42704)。*column-name* のデータ・タイプは REF(S) でなければなりません。*S* は *typed-view-name* のタイプを表します (SQLSTATE 428DM)。*column-name* に割り当てられた値が、*typed-view-name* に存在する行を実際に参照しているかどうかを示す検査は行われません。

CONSTRAINT *constraint-name*

制約の名前を指定します。*constraint-name* (制約名) は、同じ CREATE TABLE ステートメントにすでに指定されている制約を指定するものであってはなりません (SQLSTATE 42710)。

この節が省略された場合は、表に定義されている既存の制約の ID の中でユニークな 18 バイトの長さの ID がシステムによって生成されます。(ID は、"SQL" と、タイム・スタンプ機能に基づいて生成される一連の 15 の数字から構成されます。)

PRIMARY KEY 制約またはユニーク制約とともに使用した場合、この *constraint-name* は、制約をサポートするために作成される索引の名前として使用されます。

PRIMARY KEY

これは、1 つの列からなる主キーを定義する簡単な方法です。つまり、

CREATE TABLE

PRIMARY KEY が列 C の定義で指定されている場合、その効果は、PRIMARY KEY(C) 節を独立した節として指定する場合と同じです。

表が副表である場合、主キーはスーパー表から継承されるので (SQLSTATE 429B3)、主キーを指定することはできません。

ROW CHANGE TIMESTAMP 列を主キーの一部として使用することはできません (SQLSTATE 429BV)。

後述の *unique-constraint* の説明の中の PRIMARY KEY を参照してください。

UNIQUE

これは、1 つの列からなるユニーク・キーを定義する簡単な方法です。すなわち、UNIQUE を列 C の定義に指定すると、UNIQUE(C) 節を独立した節として指定した場合と同じ結果になります。

表が副表である場合、ユニーク制約はスーパー表から継承されるので (SQLSTATE 429B3)、ユニーク制約を指定することはできません。

後述の *unique-constraint* の説明の中の UNIQUE を参照してください。

references-clause

これは、1 つの列からなる外部キーを定義する簡単な方法です。つまり、*references-clause* が列 C の定義に指定されている場合、その効果は、列として C しか指定されていない FOREIGN KEY 節の一部として *references-clause* が指定された場合と同じになります。

後述の *referential-constraint* の *references-clause* の項を参照してください。

CHECK (*check-condition*)

これは、1 つの列に適用されるチェック制約を定義する簡単な方法です。後述の CHECK (*check-condition*) を参照してください。

generated-column-definition

列の生成値を指定します。

default-clause

列のデフォルト値を指定します。

WITH

オプション・キーワード。

DEFAULT

INSERT で値が提供されなかった場合、もしくは INSERT や UPDATE で DEFAULT が指定されている場合に、デフォルト値を提供します。DEFAULT キーワードの後にデフォルト値が指定されていない場合、使用されるデフォルト値は列のデータ・タイプによって異なります。『ALTER TABLE』を参照してください。

列を XML として定義する場合、デフォルト値は指定できません (SQLSTATE 42613)。可能なデフォルト値は NULL だけです。

列が型付き表の列に基づいている場合、デフォルト値の定義時には特定のデフォルト値を指定する必要があります。型付き表

のオブジェクト ID の列には、デフォルト値を指定することはできません (SQLSTATE 42997)。

列が特殊タイプを使用して定義される場合、列のデフォルト値は、特殊タイプにキャストされたソース・データ・タイプのデフォルト値になります。

構造化タイプを使用して列を定義する場合は、*default-clause* を指定できません (SQLSTATE 42842)。

column-definition から DEFAULT を省略すると、その列のデフォルト値として NULL 値が使用されます。そのような列を NOT NULL と定義すると、その列には有効なデフォルトはなくなります。

default-values

default-values に指定できるデフォルト値のタイプは、以下のとおりです。

constant

列のデフォルト値として定数を指定します。指定する定数は、次の条件を満たしていなければなりません。

- 割り当ての規則に従って、その列に割り当てることができる値でなければなりません。
- その列が浮動小数点データ・タイプとして定義されている場合を除き、浮動小数点の定数を指定してはなりません。
- 数値定数、または列のデータ・タイプが 10 進浮動小数点の場合には 10 進浮動小数点特殊値でなければなりません。浮動小数点定数はまず DOUBLE として解釈され、次にターゲット列が DECFLOAT である場合は 10 進浮動小数点数に変換されます。DECFLOAT(16) 列では、16 桁を超える精度を持つ 10 進定数は、CURRENT DECFLOAT ROUNDING MODE 特殊レジスターにより指定される丸めモードを使用して丸められます。
- 定数が 10 進定数の場合、その列のデータ・タイプの位取りを超えるゼロ以外の数字を含めてはなりません (例えば、DECIMAL(5,2) の列のデフォルト値として 1.234 を指定することはできません)。
- 指定する定数が 254 バイトを超えてはなりません。この制約には、引用符文字や 16 進定数の X などの接頭部文字も含まれます。さらに、定数が *cast-function* の引数の場合には、完全修飾された関数名から取った文字や括弧も含めて、この制限を超えてはなりません。

datetime-special-register

INSERT、UPDATE、または LOAD の実行時における日時特殊レジスターの値 (CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP) を、その列のデフォルト値として指定します。その列のデータ・タイプは、指定した特殊レジスターに対応するデータ・タイプでなければ

CREATE TABLE

なりません (例えば、CURRENT DATE を指定した場合、データ・タイプは DATE でなければなりません)。

user-special-register

INSERT、UPDATE、または LOAD の実行時におけるユーザー特殊レジスターの値 (CURRENT USER、SESSION_USER、SYSTEM_USER) を、その列のデフォルトとして指定します。その列のデータ・タイプは、ユーザー特殊レジスターの長さ属性よりも短い文字ストリングであってはなりません。なお、SESSION_USER の代わりに USER を、CURRENT USER の代わりに CURRENT_USER を指定することもできます。

CURRENT SCHEMA

INSERT、UPDATE、または LOAD の実行時における CURRENT SCHEMA 特殊レジスターの値を、その列のデフォルト値として指定します。CURRENT SCHEMA を指定した場合、その列のデータ・タイプは、CURRENT SCHEMA 特殊レジスターの長さ属性よりも短い文字ストリングであってはなりません。

NULL

その列のデフォルト値として NULL を指定します。NOT NULL の値が指定された場合は、DEFAULT NULL を同じ列定義に指定できますが、その列をデフォルト値に設定しようとするエラーが生じます。

cast-function

この形式のデフォルト値は、特殊タイプ (distinct type)、BLOB、または日時 (DATE、TIME、または TIMESTAMP) データ・タイプとして定義された列に対してのみ使用することができます。特殊タイプの場合、BLOB や日時タイプに基づく例外があり、関数名が列の特殊タイプの名前に一致していなければなりません。スキーマ名で修飾されている場合には、その特殊タイプのスキーマ名と同じでなければなりません。修飾されていない場合には、関数の解決に用いるスキーマ名は特殊タイプのスキーマ名と同じでなければなりません。日時タイプに基づく特殊タイプで、デフォルト値が定数の場合、必ず関数を使用する必要があります。さらに、その関数名は、暗黙または明示のスキーマ名 SYSIBM を持つ特殊タイプのソース・タイプ名に一致していなければなりません。他の日時列の場合は、対応する日時関数も使用できます。BLOB に基づく BLOB または特殊タイプの場合も、関数を使用する必要があります。その関数名は、暗黙または明示のスキーマ名 SYSIBM を持つ BLOB でなければなりません。

constant

引数として定数を指定します。指定する定数は、特殊タイプのソース・タイプに関する定数の規則 (特殊タイプでない場合は、データ・タイプに関する定数の規則) に

従っていなければなりません。 *cast-function* が BLOB の場合には、定数としてストリング定数を指定する必要があります。

datetime-special-register

CURRENT DATE、CURRENT TIME、または CURRENT TIMESTAMP を指定します。列の特殊タイプのソース・タイプは、指定した特殊レジスターに対応するデータ・タイプでなければなりません。

user-special-register

CURRENT USER、SESSION_USER、または SYSTEM_USER を指定します。列の特殊タイプのソース・タイプのデータ・タイプは、少なくとも 8 バイトの長さのストリング・データ・タイプでなければなりません。 *cast-function* が BLOB の場合には、長さ属性が 8 バイト以上でなければなりません。

CURRENT SCHEMA

CURRENT SCHEMA 特殊レジスターの値を指定します。列の特殊タイプのソース・タイプのデータ・タイプは、CURRENT SCHEMA 特殊レジスターの長さ属性よりも短い文字ストリングであってはなりません。 *cast-function* が BLOB の場合には、長さ属性が 8 バイト以上でなければなりません。

EMPTY_CLOB()、EMPTY_DBCLOB()、または EMPTY_BLOB()

その列のデフォルト値として長さゼロのストリングを指定します。その列は、この関数の結果データ・タイプに対応するデータ・タイプを持っている必要があります。

指定した値が無効な場合、エラーが戻されます (SQLSTATE 42894)。

GENERATED

DB2 が列の値を生成することを指定します。その列が ID 列または ROW CHANGE TIMESTAMP 列と見なされることになる場合には、GENERATED を指定する必要があります。

ALWAYS

行が表に挿入されるときや、*generation-expression* の結果値が変更されるたびに、DB2 が常に列の値を生成することを指定します。この式の結果は、表に保管されます。データ伝搬、またはアンロードおよび再ロード操作を実行しているのでなければ、GENERATED ALWAYS の値をお勧めします。GENERATED ALWAYS は、生成列に必須の値です。

BY DEFAULT

DEFAULT 節を指定して行が挿入または更新されるときに、明示的に値を指定しないかぎり、DB2 が列に値を生成することを

指定します。データ伝搬を使用したり、アンロードおよび再ロードを実行したりするときは、`BY DEFAULT` が推奨される値です。

明示的には要求されませんが、値のユニーク性を確保するために、生成された ID 列のユニークな 1 列の索引を定義してください。

AS IDENTITY

列をこの表の ID 列にすることを指定します。1 つの表には 1 つしか ID 列があってはなりません (SQLSTATE 428C1)。列に関連付けられたデータ・タイプがゼロの位取りの完全な数値タイプになっているか、ソース・タイプのユーザー定義特殊タイプがゼロの位取りの完全な数値タイプになっている場合だけ、`IDENTITY` キーワードが指定可能です (SQLSTATE 42815)。ゼロの位取りの `SMALLINT`、`INTEGER`、`BIGINT`、または `DECIMAL` や、これらのタイプのうちのいずれかに基づいた特殊タイプは、完全な数値タイプと見なされます。これに対して、単精度および倍精度の浮動小数点は、近似数値データ・タイプと見なされます。参照タイプは、完全な数値タイプで表されていても、ID 列と定義することはできません。

ID 列は暗黙で `NOT NULL` になります。ID 列は `DEFAULT` 節を持つことができません (SQLSTATE 42623)。

START WITH *numeric-constant*

ID 列の最初の値を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815)。デフォルトは、昇順シーケンスであれば `MINVALUE`、降順シーケンスであれば `MAXVALUE` です。

INCREMENT BY *numeric-constant*

連続した ID 列値のインターバルを指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815)、長精度整数定数の値を超えることはありません (SQLSTATE 42820)。

この値が負の場合、これは降順シーケンスです。この値が 0 の場合、または正の場合は、昇順シーケンスになります。デフォルトは 1 です。

NO MINVALUE または **MINVALUE**

降順 ID 列が値の生成を循環または停止する最小値、あるいは最大値に達した後、昇順 ID 列が循環する最小値を指定します。

NO MINVALUE

昇順シーケンスの場合、値は `START WITH` 値、または `START WITH` が指定されなかった場合には 1 です。降順シーケンスの場合、列のデータ・タイプの最小値になります。これはデフォルトです。

MINVALUE *numeric-constant*

最小値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815) が、最大値以下でなければなりません (SQLSTATE 42815)。

NO MAXVALUE または **MAXVALUE**

昇順 ID 列が値の生成を循環または停止する最大値、あるいは最小値に達した後、降順 ID 列が循環する最大値を指定します。

NO MAXVALUE

昇順シーケンスの場合、値は列のデータ・タイプの最大値です。降順シーケンスの場合、値は **START WITH** 値、または **START WITH** が指定されなかった場合には -1 です。これはデフォルトです。

MAXVALUE *numeric-constant*

最大値にする数値定数を指定します。この値は、小数点の右側に非ゼロの数字がない (SQLSTATE 428FA) かぎり、この列に割り当てることができる正または負の値にすることができます (SQLSTATE 42815) が、最小値よりも大きいかまたは等しくなければなりません (SQLSTATE 42815)。

NO CYCLE または **CYCLE**

その最大値または最小値が生成された後、この ID 列が値の生成を続行するかどうかを指定します。

NO CYCLE

最大値または最小値に達した後、ID 列について値が生成されないことを指定します。これはデフォルトです。

CYCLE

最大値または最小値に達した後、この列について値の生成が続行されることを指定します。このオプションが使用されると、昇順 ID 列が最大値に達した後は、その最小値が生成されます。降順 ID 列が最小値に達した後は、その最大値が生成されます。ID 列の最大値および最小値は、循環に使用される範囲を決定します。

CYCLE が有効な場合、DB2 が ID 列について重複する値を生成する可能性があります。固有値が望ましい場合、明示的には要求されませんが、値のユニーク性を確保するために、ユニークな 1 列の索引を生成列で定義する必要があります。このような ID 列にユニーク索引が存在し、ユニークではない値が生成されると、エラーが起きます (SQLSTATE 23505)。

NO CACHE または **CACHE**

特定の事前割り振り値を、高速アクセスできるようメモリーに保存するかどうかを指定します。ID 列で新しい値が必要になった場合に、キャッシュの中のものを使用できないときは、新

しいキャッシュ・ブロックの末尾をログ記録する必要があります。ただし、ID 列で新しい値が必要になった場合に、キャッシュの中に未使用の値があるときは、その ID 値を割り振ったほうが、ロギングしなくて済むので高速化されます。これはパフォーマンスおよびチューニング・オプションです。

NO CACHE

ID 列の値を事前割り振りしないことを指定します。

このオプションが指定されると、ID 列の値はキャッシュに保管されません。この場合、新しい ID 値が要求されるたびに、ログに対して同期入出力が行われます。

CACHE integer-constant

事前割り振りされ、メモリーに保管される ID シーケンスの値の数を指定します。ID 列について値が生成される場合、値を事前割り振りしてキャッシュに保管しておくこと、ログへの同期入出力が少なくなります。

ID 列に新しい値が必要でも未使用の値がキャッシュにない場合、値の割り振りによりログへの入出力の待機が呼び出されます。ただし、ID 列に新しい値が必要で、未使用の値がキャッシュにあれば、その ID 値の割り振りが、ログへの入出力なしで素早く行われます。

システム障害に起因するものであっても通常のものであっても、データベースの活動解除が起こると、コミットされたステートメントで使用されていないキャッシュ済みシーケンス値はすべて失われます。つまり使用されなくなります。データベースの活動解除が起きたら失われる可能性のある ID 列値の最大数は、CACHE オプションに指定された値になります。(データベースが ACTIVATE コマンドまたは API を使用して明示的に活動化されない場合には、最終アプリケーションの接続をデータベースから切断すると、暗黙の活動解除が行われます。)

最小値は 2 です (SQLSTATE 42815)。デフォルト値は CACHE 20 です。

NO ORDER または ORDER

要求の順序で ID 値が生成されるかどうかを指定します。

NO ORDER

要求の順序での値を生成する必要がないことを指定します。これはデフォルトです。

ORDER

要求の順序で値が生成されることを指定します。

FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

列が表のタイム・スタンプ列であることを指定します。行が挿入されるたびにその列に値が生成され、列が更新されるたびにその行に値が生成されます。ROW CHANGE TIMESTAMP 列に生成される値は、その行の挿入または更新の時刻に対応するタイム・スタンプ

です。1つのステートメントによって複数の行が挿入または更新される場合、ROW CHANGE TIMESTAMP 列の値は行ごとに異なる可能性があります。

ROW CHANGE TIMESTAMP 列は1つの表内に1つだけ含めることができます (SQLSTATE 428C1)。 *data-type* を指定する場合は、TIMESTAMP でなければなりません (SQLSTATE 42842)。ROW CHANGE TIMESTAMP 列は DEFAULT 節を持つことができません (SQLSTATE 42623)。ROW CHANGE TIMESTAMP 列には NOT NULL を指定する必要があります (SQLSTATE 42831)。

GENERATED ALWAYS AS (*generation-expression*)

列定義が式に基づくことを指定します。(GENERATED ALWAYS 列の式にユーザー定義の外部関数が入っている場合に、その関数の実行可能ファイルを変更する (引数ごとに異なる結果を得るため) と、データの不整合を生じることがあります。これが生じないようにするには、SET INTEGRITY ステートメントを使って、新しい値を強制的に生成させます。) *generation-expression* には、以下のいずれも入れることができません (SQLSTATE 42621)。

- 副照会
- XMLQUERY 式または XMLEXISTS 式
- 列関数
- 間接参照操作または Deref 関数
- 非 deterministic であるユーザー定義関数または組み込み関数
- EXTERNAL ACTION オプションを使用するユーザー定義関数
- NO SQL を指定して定義されていないユーザー定義関数
- ホスト変数またはパラメーター・マーカー
- 特殊レジスター
- グローバル変数
- 列リスト内で後から定義されている列の参照
- 他の生成列の参照
- タイプ XML の列の参照

列のデータ・タイプは *generation-expression* の結果データ・タイプに基づいています。CAST 指定を使って特定のデータ・タイプを強制的に使用し、有効範囲を設けることができます (参照タイプの場合のみ)。 *data-type* を指定すると、適切な割り当て規則に従って、値が列に割り当てられます。NOT NULL 列オプションを使わない限り、生成された列は暗黙で NULL 可能と見なされます。生成済み列のデータ・タイプと *generation-expression* の結果データ・タイプは、等価のものとして定義されている必要があります (『割り当てと比較』を参照してください)。ただし、LOB データ・タイプ、XML、構造化タイプ、およびこれらのいずれかのタイプに基づいた特殊タイプの列と生成式を除きます (SQLSTATE 42962)。

INLINE LENGTH *integer*

このオプションは、構造化タイプまたは XML タイプを使って定義された列に対してだけ有効です (SQLSTATE 42842)。

タイプ XML の列では、*integer* は、基本表の行に格納する XML 文書の内部表記の最大バイト・サイズを指示します。これより大きな内部表記を持つ XML 文書は、基本表の行とは別に、補助ストレージ・オブジェクト内に格納されます。これは自動的に行われます。XML タイプの列には、デフォルト・インライン長はありません。

構造化タイプの列では、*integer* は、行内の残りの値とともにインラインで保管する構造化タイプのインスタンスの最大バイト・サイズを指示します。インラインで保管できない構造化タイプのインスタンスは、LOB 値が処理されるのに似た方法で、基本表の行とは別に保管されます。これは自動的に行われます。構造化タイプ列のデフォルトの **INLINE LENGTH** は、このタイプのインライン長になります (明示的に指定するか、または **CREATE TYPE** ステートメント内のデフォルトとして)。構造化タイプの **INLINE LENGTH** が 292 未満の場合、列の **INLINE LENGTH** には値 292 が使われます。

注: サブタイプのインライン長は、デフォルトのインライン長には数えられませんが、それは、**CREATE TABLE** 時に明示的に **INLINE LENGTH** を指定して、現在および将来のサブタイプに対処できるようにしておかないと、サブタイプのインスタンスはインラインに適合しないことがあることを意味します。

明示的な **INLINE LENGTH** の値は少なくとも 292 でなければならず、32673 を超えてはなりません (SQLSTATE 54010)。

COMPRESS SYSTEM DEFAULT

システム・デフォルト値が、最小限のスペースを使用して保管されるように指定します。 **VALUE COMPRESSION** 節が指定されていない場合には警告が出され (SQLSTATE 01648)、システム・デフォルト値が最小限のスペースを使用して保管されるようにはなりません。

システム・デフォルト値がこのような方法で保管されると、列に対する許可や更新操作の際に余分な検査が行われるために、若干パフォーマンスが低下します。

基本データ・タイプは、DATE、TIME、TIMESTAMP、XML、または構造化データ・タイプであってはなりません (SQLSTATE 42842)。基本データ・タイプが可変長ストリングの場合には、この節は無視されます。表が **VALUE COMPRESSION** に設定されている場合は、長さ 0 のストリング値は自動的に圧縮されます。

COLUMN SECURED WITH *security-label-name*

表に関連するセキュリティー・ポリシーに対応して存在するセキュリティー・ラベルを識別します。名前は非修飾でなければなりません (SQLSTATE 42601)。表にはセキュリティー・ポリシーが関連付けられている必要があります (SQLSTATE 55064)。

unique-constraint

ユニーク制約または主キー制約を定義します。表に分散キーがある場合、ユニーク

ク・キーまたは主キーは分散キーのスーパーセットである必要があります。副表である表では、ユニーク制約または主キー制約を指定することはできません (SQLSTATE 429B3)。主キーまたはユニーク・キーは、ディメンションのサブセットにはなりません (SQLSTATE 429BE)。表がルート表である場合、表とすべての副表に対して制約が適用されます。

CONSTRAINT *constraint-name*

主キー制約、またはユニーク制約の名前を指定します。

UNIQUE (*column-name,...*)

指定した列で構成されるユニーク・キーを定義します。指定する列は NOT NULL として定義されていなければなりません。各 *column-name* (列名) は、表の列を指定するものでなければなりません。また、同じ列を複数回指定することはできません。

指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。列の保管時の長さについては、バイト・カウントを参照してください。キー長の制限については、『SQL の制限』を参照してください。列の長さ属性がページ・サイズに対応する索引キー長制限値の範囲内に収まる場合でも、LOB、XML、これらのタイプのいずれかに基づく特殊タイプ、構造化タイプは、ユニーク・キーの一部として使用できません (SQLSTATE 54008)。

ユニーク・キーの列セットは、主キーまたは他のユニーク・キーの列セットと同じにすることはできません (SQLSTATE 01543)。(LANGLEVEL が SQL92E または MIA の場合は、エラーが戻されます。SQLSTATE 42891)

表が副表である場合、ユニーク制約はスーパー表から継承されるので、(SQLSTATE 429B3)、ユニーク制約を指定することはできません。

カタログに記録されている表の記述には、ユニーク・キーとそのユニーク索引が含まれます。順方向と逆方向のスキャンが可能な双方向のユニーク索引は、それぞれの列について昇順に指定された順序で、列に対して自動的に作成されます。索引の名前は、表の作成時にスキーマに存在する既存の索引と競合しない場合、*constraint-name* (制約名) と同じになります。索引名が競合する場合は、名前は SQL の後に文字のタイム・スタンプ (*yymmddhhmmssxxx*) が続き、スキーマ名として SYSIBM を伴う名前になります。

PRIMARY KEY (*column-name,...*)

指定された列で構成される主キーを定義します。この節を複数回指定することはできず、指定する列は NOT NULL として定義されていなければなりません。各 *column-name* (列名) は、表の列を指定していなければなりません。また、同じ列を複数回指定することはできません。

指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。列の保管時の長さについては、バイト・カウントを参照してください。キー長の制限については、『SQL の制限』を参照してください。列の長さ属性がページ・サイズに対応する索引キー長制限値の範囲内に収まる場合でも、LOB、XML、これらのタイプのいずれかに基づく特殊タイプ、構造化タイプは、主キーの一部として使用できません (SQLSTATE 54008)。

CREATE TABLE

主キーの列セットは、ユニーク・キーの列セットと同じであってはなりません (SQLSTATE 01543)。 (LANGLEVEL が SQL92E または MIA の場合は、エラーが戻されます。SQLSTATE 42891)

1 つの表には、主キーを 1 つだけ定義することができます。

表が副表である場合、主キーはスーパー表から継承されるので (SQLSTATE 429B3)、主キーを指定することはできません。

カタログに記録されている表の記述には、主キーとその主索引が含まれます。順方向と逆方向のスキャンが可能な双方向のユニーク索引は、それぞれの列について昇順に指定された順序で、列に対して自動的に作成されます。索引の名前は、表の作成時にスキーマに存在する既存の索引と競合しない場合、*constraint-name* (制約名) と同じになります。索引名が競合する場合は、名前は SQL の後に文字のタイム・スタンプ (yymmddhhmmssxxx) が続き、スキーマ名として SYSIBM を伴う名前になります。

表に分散キーがある場合、*unique-constraint* (ユニーク制約) の列は分散キー列のスーパーセットである必要があります。列の順序は重要ではありません。

referential-constraint

参照制約を定義します。

CONSTRAINT *constraint-name*

参照制約の名前を指定します。

FOREIGN KEY (*column-name,...*)

指定した *constraint-name* (制約名) の参照制約を定義します。

T1 を、ステートメントの対象となる表であると想定します。参照制約の外部キーは、指定された列で構成されます。列名リストの各名前は、T1 の列を指定していなければならない、同じ列を複数回指定することはできません。

指定する列の数は 64 を超えてはならず、それらの列の保管時の長さの合計は、ページ・サイズに対応する索引キー長制限値を超えてはなりません。列の保管時の長さについては、バイト・カウントを参照してください。キー長の制限については、『SQL の制限』を参照してください。LOB、XML、これらのタイプのうちのいずれかに基づく特殊タイプ、または構造化タイプの列を、外部キーの一部として使用することはできません (SQLSTATE 42962)。外部キーの列の数は、親キーの列の数と同じでなければならず、対応する列のデータ・タイプは互換性があることが必要です (SQLSTATE 42830)。2 つの列の記述は、それらの列が互換性のあるデータ・タイプ (両方の列が数字、文字ストリング、GRAPHIC、日付 / 時間であるか、または同じ特殊タイプ) であれば互換性があります。

references-clause

参照制約の親表または親ニックネーム、および親キーを指定します。

REFERENCES *table-name* または *nickname*

REFERENCE 節に指定される表またはニックネームは、カタログに記述された基本表またはニックネームを識別する必要がありますが、カタログ表を示すものであってはなりません。

参照制約の外部キー、親キー、および親表または親ニックネームが、以前に指定した参照制約の外部キー、親キー、および親表または親ニック

ネームと同じである場合、参照制約は重複しています。重複した参照制約は無視され、警告が戻されます (SQLSTATE 01543)。

以下の説明では、T2 は指定した親表を示し、T1 は作成する (または変更する) 表を示します。(T1 と T2 は同じ表である可能性もあります。)

指定された外部キーの列の数は、T2 の親キーと同じ数でなければなりません。また、外部キーの n 番目の列の記述は、その親キーの n 番目の列の記述と互換性がなければなりません。この規則において、日時の列はストリング列と互換性があるとは見なされません。

(*column-name*,...)

参照制約の親キーは、指定された列で構成されます。各 *column-name* は、T2 の列を指定する非修飾名でなければなりません。同じ列を複数回指定することはできません。

列名のリストは、主キーまたは T2 に存在するユニーク制約の列セットと一致している (順序は任意) 必要があります (SQLSTATE 42890)。列名のリストの指定がない場合、T2 に主キーがある必要があります (SQLSTATE 42888)。列名リストを省略すると、指定されているとおりの順序でその主キーの列が暗黙に指定されます。

FOREIGN KEY 節で指定される参照制約は、T2 が親であり、T1 が従属であるリレーションシップを定義します。

rule-clause

従属表に対するアクションを指定します。

ON DELETE

親表の行が削除された場合、従属表でどのようなアクションを行うかを指定します。次の 4 つのアクションがあります。

- NO ACTION (デフォルト値)
- RESTRICT
- CASCADE
- SET NULL

削除規則は、T2 の行が DELETE または伝搬による削除操作の対象であり、その行の従属行が T1 にある場合に、適用されます。 p は、そのような T2 の行を表すと想定します。

- RESTRICT または NO ACTION を指定すると、エラーになり、行は削除されません。
- CASCADE を指定すると、T1 の p の従属行に削除操作が伝搬します。
- SET NULL が指定された場合、T1 の p のそれぞれの従属行の外部キーの NULL 可能な列が NULL 値に設定されます。

SET NULL は、外部キーの列に NULL 可能な列がない限り指定してはなりません。この節を省略すると、暗黙に ON DELETE NO ACTION が指定されます。

T1 が複数のパスで T2 に連結削除されている場合は、重複する外部キー定義を使用して 2 つの SET NULL 規則を定義することはで

きません。例えば、T1 (i1, i2, i3) という場合があります。Rule1 に外部キー (i1, i2) を使用し、Rule2 に外部キー (i2, i3) を使用するということはできません。

規則の配列順序は次のとおりです。

1. RESTRICT
2. SET NULL OR CASCADE
3. NO ACTION

T1 の任意の行が 2 つの異なる規則によって影響される場合、エラーとなり行は削除されません。

複数の表が関係し、削除規則の 1 つが RESTRICT または SET NULL になっている循環によって表が自身を連結削除するような参照制約は定義できません (SQLSTATE 42915)。

複数のパスによって表が自身や他の表を連結削除する参照制約は、以下の場合を除き、定義できます (SQLSTATE 42915)。

- 表は、CASCADE リレーションシップ (自己参照、または別の表を参照)、および削除規則が RESTRICT または SET NULL の自己参照リレーションシップのいずれの従属表であってもなりません。
- キーとキーの間に同じ列が 1 つでもあると、キーはもう 1 方のキーにオーバーラップします。表が、外部キーにオーバーラップしている複数のリレーションシップを通して別の表に連結削除される場合は、それらのリレーションシップの間で削除規則が一致していなければなりません。また、いずれの削除規則も SET NULL になってはなりません。
- 表と別の表の間に複数のリレーションシップに基づく連結削除が設定されている場合、それらのリレーションシップのうち、少なくとも 1 つに SET NULL の削除規則が指定されていれば、それらのリレーションシップの外部キー定義に分散キー列またはマルチディメンション・クラスタリング (MDC) キー列が含まれてはなりません。
- CASCADE リレーションシップを通して 2 つの表が同じ表に連結削除されている場合、各連結削除パスの最後のリレーションシップの削除規則が RESTRICT または SET NULL であるときは、2 つの表を相互に連結削除することはできません。

T1 の何からの行が別の削除規則の影響を受ける場合、結果は、これらの規則で指定されたすべてのアクションの影響を受けます。すべてのアクションの影響は、T1 の AFTER トリガーと CHECK 制約からも認識されます。例えば、例として、上位の表へのある連結削除パスによって NULL 設定されるターゲットになっていて、同じ上位の表への 2 番目の連結削除パスによって削除されるターゲットになっている行があるとします。この場合、結果として行は削除されます。この派生表では、AFTER DELETE トリガーは活動化されますが、AFTER UPDATE トリガーは活動化されません。

親表または従属表が型付き表階層のメンバーである参照制約に対する上記の規則の適用の場合、それぞれの階層内の任意の表に対して適用されるすべての参照制約が考慮に入られます。

ON UPDATE

親表の行が更新された場合に従属表に対して行うアクションを指定します。この節はオプションです。ON UPDATE NO ACTION はデフォルト値であり、ON UPDATE RESTRICT はそれに代わって指定できる唯一のものです。

NO ACTION と RESTRICT の差異については、『注』のセクションを参照してください。

check-constraint

チェック制約を定義します。*check-constraint* (チェック制約) は、偽以外に評価されなければならない *search-condition* (検索条件)、または列間に定義された機能従属関係です。

CONSTRAINT *constraint-name*

チェック制約の名前を指定します。

CHECK (*check-condition*)

チェック制約を定義します。*search-condition* は、表のすべての行について、真または不明でなければなりません。

search-condition

search-condition には、以下の制限があります。

- 列参照は、作成する表の列に対するものでなければなりません。
- *search-condition* に TYPE 述部を入れることはできません。
- *search-condition* には、以下のいずれも入れることができません (SQLSTATE 42621)。
 - 副照会
 - XMLQUERY 式または XMLEXISTS 式
 - 間接参照操作または、有効範囲をもつ参照引数がオブジェクト ID (OID) 列以外の列である Deref 関数
 - SCOPE 節をもつ CAST 指定
 - 列関数
 - deterministic 関数でない関数
 - 外部アクションをもつと定義された関数
 - CONTAINS SQL または READS SQL DATA のいずれかによって定義されたユーザー定義関数
 - ホスト変数
 - パラメーター・マーカー
 - 特殊レジスター
 - グローバル変数
 - ID 列以外の生成列の参照
 - タイプ XML の列の参照 (VALIDATED 述部の中以外)
 - エラー・トレラントな *nested-table-expression*

functional-dependency

列間の機能従属関係を定義します。

column-name **DETERMINED BY** *column-name* または (*column-name*,...) **DETERMINED BY** (*column-name*,...)

列の親セットには、DETERMINED BY 節の直前に来る指定された列が含まれます。列の子セットには、DETERMINED BY 節の直後に来る指定された列が含まれます。 *search-condition* の制約事項すべては、親セット列と子セット列に適用され、列のセットには単純な列参照のみが許可されています (SQLSTATE 42621)。機能従属関係に同じ列を複数回指定することはできません (SQLSTATE 42709)。列のデータ・タイプを LOB データ・タイプ、LOB データ・タイプに基づく特殊タイプ、XML データ・タイプ、構造化タイプにすることはできません (SQLSTATE 42962)。ROW CHANGE TIMESTAMP 列を主キーの一部として使用することはできません (SQLSTATE 429BV)。列の子セットの列を NULL 可能列にすることはできません (SQLSTATE 42621)。

column-definition の一部としてチェック制約を指定する場合、その同じ列に対してのみ列参照を行うことができます。表定義の一部として指定されたチェック制約には、それ以前に CREATE TABLE ステートメントで定義されている列を指定する列参照を含めることができます。チェック制約の矛盾、重複条件、または同等条件については検査されません。したがって、矛盾したチェック制約や冗長なチェック制約が定義可能であるため、実行時にエラーになる可能性があります。

search-condition として 『IS NOT NULL』 を指定することもできますが、列の NOT NULL 属性を使用することによって直接的に NULL 可能を指定することをお勧めします。例えば、salary が NULL に設定された場合に、CHECK (salary + bonus > 30000) は受け入れられません。これは、CHECK 制約は満たされるか未知かのどちらかでなければならず、この場合 salary は未知であるためです。一方、給与 (salary) が NULL に設定されている場合に、CHECK (salary IS NOT NULL) は偽となり、制約違反と見なされます。

search-condition を伴うチェック制約は、表に対して行の挿入または更新が行われる時点で適用されます。表で定義されるチェック制約は、その表の副表すべてに自動的に適用されます。

挿入、更新、削除、整合性設定などの通常の操作中には、データベース・マネージャーによって機能従属関係が課せられません。機能従属関係は、照会を最適化するために、照会の書き直しの際に使用できるでしょう。機能従属関係の整合性が維持されないと、間違った結果が戻される可能性があります。

constraint-attributes

参照整合性またはチェック制約に関連付けられた属性を定義します。

ENFORCED または **NOT ENFORCED**

挿入、更新、削除などの通常の操作中に、データベース・マネージャーによって制約が課せられるかどうかを指定します。デフォルトは ENFORCED です。

ENFORCED

データベース・マネージャーによって制約が課せられます。機能従属関係に ENFORCED を指定することはできません (SQLSTATE 42621)。ENFORCED は、参照制約がニックネームを参照しているときは指定できません (SQLSTATE 428G7)。

NOT ENFORCED

データベース・マネージャーによって制約が課せられません。この制約に適合することが分かっている表データだけに、制約を指定してください。

ENABLE QUERY OPTIMIZATION または DISABLE QUERY OPTIMIZATION

適切な状況下で、照会の最適化のために、制約または機能従属関係を使用できるかどうかを指定します。デフォルトは ENABLE QUERY OPTIMIZATION です。

ENABLE QUERY OPTIMIZATION

制約が真であると想定され、照会の最適化に使用できます。

DISABLE QUERY OPTIMIZATION

制約を照会の最適化に使用できません。

OF *type-name1*

表の列が *type-name1* で指定される構造化タイプの属性に基づいていることを指定します。*type-name1* の指定にスキーマ名が含まれていない場合、そのタイプ名は SQL パス上のスキーマを探索することによって決まります (このパスは、静的 SQL の場合は FUNCPATH プリプロセス・オプションによって、動的 SQL の場合は CURRENT PATH レジスターによって定義されます)。このタイプ名は、既存のユーザー定義タイプ名である (SQLSTATE 42704) 必要があり、また、少なくとも 1 つの属性があつて (SQLSTATE 42997)、しかもインスタンス化可能な構造化タイプでなければなりません (SQLSTATE 428DP)。

UNDER が指定されていない場合には、オブジェクト ID 列を指定する必要があります (*OID-column-definition* を参照)。このオブジェクト ID 列は、その表の最初の列になります。オブジェクト ID 列の後に、*type-name1* の属性に基づく列が続きます。

HIERARCHY *hierarchy-name*

表階層に関連する階層表を指定します。同時に、これは階層のルート表としても作成されます。型付き表階層に含まれる副表のデータはすべて、この階層表に保管されます。階層表を SQL ステートメントで直接に参照することはできません。*hierarchy-name* は *table-name* になります。暗黙または明示のスキーマ名の入った *hierarchy-name* は、カタログに記述されている表、ニックネーム、ビュー、または別名を指定するものであってはなりません。スキーマ名を指定する場合、作成する表のスキーマ名と同じにする必要があります (SQLSTATE 428DQ)。ルート表の定義時にこの節が省略されると、システムによって名前が生成されます。この名前は、作成する表の名前とその後のユニークな接尾部で構成され、既存の表、ビュー、およびニックネームの ID の中で固有な ID です。

UNDER *supertable-name*

表が *supertable-name* の副表であることを指定します。スーパー表は既存の表で

CREATE TABLE

なければならず (SQLSTATE 42704)、かつ表は *type-name1* のすぐ上のスーパータイプである構造化タイプを使用して定義しなければなりません (SQLSTATE 428DB)。 *table-name* と *supertable-name* のスキーマ名は、同じでなければなりません (SQLSTATE 428DQ)。 *supertable-name* で指定される表には、 *type-name1* で既に定義された既存の副表を含めることはできません (SQLSTATE 42742)。

表の列には、スーパー表のオブジェクト ID 列が含まれています。この列のタイプは、REF(*type-name1*) に変更されており、 *type-name1* の属性に基づく列が続きます (ここでいうタイプには、スーパータイプの属性も含まれていることを念頭に置いてください)。属性名は OID 列名と同じものにするにはできません (SQLSTATE 42711)。

表スペースやデータ・キャプチャーなど、その他の表オプションは、初期状態ではログに記録されません。また、分散キー・オプションは指定できません。これらのオプションはスーパー表から継承されます (SQLSTATE 42613)。

INHERIT SELECT PRIVILEGES

スーパー表に対して SELECT 特権を保持するユーザーやグループはすべて、新しく作成した副表に対しても同様の特権を付与されます。この特権は、副表定義者によって付与されたものと見なされます。

typed-element-list

型付き表の追加エレメントを定義します。これには、列の追加オプション、オブジェクト ID 列 (ルート表のみ) の追加、表の制約事項などが含まれます。

OID-column-definition

型付き表のオブジェクト ID 列を定義します。

REF IS *OID-column-name* USER GENERATED

オブジェクト ID 列 (OID) を表の最初の列として定義することを指定します。表階層のルート表では、OID が必須です (SQLSTATE 428DX)。この表は副表以外の型付き表 (OF 節が必須) でなければなりません (SQLSTATE 42613)。この列の名前は *OID-column-name* という形式で定義されますが、構造化タイプ *type-name1* のどの属性の名前とも同一にすることはできません (SQLSTATE 42711)。さらに、この列はタイプ REF (*type-name1*)、NOT NULL で定義され、システム必須のユニーク索引 (デフォルトの索引名) が生成されます。この列はオブジェクト ID 列 または OID 列 として参照されます。USER GENERATED というキーワードは、行を挿入する際にユーザーが OID 列の初期値を提供しなければならないことを指しています。行を挿入した後は、OID 列を更新することはできません (SQLSTATE 42808)。

with-options

型付き表の列に適用される追加オプションを定義します。

column-name

追加オプションを指定する列の名前を指定します。 *column-name* (列名) は、同じくスーパー表の列ではない表の列の名前に対応していなければなりません (SQLSTATE 428DJ)。列名は、ステートメント内の 1 つの WITH OPTIONS 節に 1 回だけしか指定できません (SQLSTATE 42613)。

タイプ定義 (CREATE TYPE) の一部としてオプションが既に指定されている場合には、ここで指定されているオプションは CREATE TYPE のオプションをオーバーライドします。

WITH OPTIONS *column-options*

指定した列にオプションを定義します。前述の *column-options* を参照してください。表が副表である場合、主キーまたはユニーク制約を指定することはできません (SQLSTATE 429B3)。

LIKE *table-name1* または *view-name* または *nickname*

表の列の名前と記述が、指定された表 (*table-name1*)、ビュー (*view-name*)、またはニックネーム (*nickname*) の列とまったく同じであることを指定します。

LIKE の後に指定する名前は、カタログに存在する表、ビューまたはニックネーム、あるいは宣言済み一時表を識別するものでなければなりません。型付き表または型付きビューを指定することはできません (SQLSTATE 428EC)。

LIKE は、*n* 列を暗黙的な定義で使います。*n* は、指定した表、ビューまたはニックネームにおける列数です (指定した表では暗黙的な隠し列を含む)。既存の表の暗黙的な隠し列に対応する新規表の列も、暗黙的な隠し列と定義されます。

- 表が特定されると、暗黙的な定義には *table-name1* のそれぞれの列の列名、データ・タイプ、隠蔽属性、および NULL 可能特性が入ります。EXCLUDING COLUMN DEFAULTS を指定しないと、列のデフォルト値も入ります。
- ビューが特定されると、暗黙的な定義には *view-name* に指定した全選択のそれぞれの結果列の列名、データ・タイプ、および NULL 可能特性が入ります。
- ニックネームが特定されると、暗黙的な定義には *nickname* のそれぞれの列の列名、データ・タイプ、および NULL 可能特性が入ります。
- LIKE 節に保護対象表を指定すると、新しい表は、その指定した表と同じセキュリティ・ポリシーと保護対象列を継承します。

copy-attributes 節に基づいて、列のデフォルトと ID 列属性を組み込んだり除外したりすることができます。さらにこの暗黙的な定義には、指定した表、ビュー、またはニックネームの他の属性は含まれません。したがって、新しい表にはユニーク制約、外部キー制約、トリガー、または索引はありません。表は IN 節で暗黙的にまたは明示的に指定した表スペースの中に作成されます。また、任意指定の他の節を指定した場合に限り、この表にその任意指定の節が含まれます。

copy-options

これらのオプションは、ソース結果表の定義 (表、ビュー、または全選択) の追加属性をコピーするかどうかを指定します。

INCLUDING COLUMN DEFAULTS

ソース結果表の定義の更新可能な各列の列デフォルトをコピーします。更新可能でない列では、作成される表の対応列にデフォルトが定義されないこととなります。

LIKE *table-name* が指定され、しかも *table-name* が基本表または宣言済み一時表を示す場合、INCLUDING COLUMN DEFAULTS がデフォルトになります。

EXCLUDING COLUMN DEFAULTS

ソース結果表の定義から列デフォルトはコピーされません。

この節がデフォルトです。ただし、LIKE *table-name* が指定され、かつ *table-name* が基本表または宣言済み一時表を示す場合を除きます。

INCLUDING IDENTITY COLUMN ATTRIBUTES

可能であれば、ソース結果表の定義から ID 列属性がコピーされます。ID 列属性をコピーできるのは、表、ビュー、または全選択内の対応する列の要素が、識別特性をもつ基本表列名に直接または間接にマップされる表列の名前またはビュー列の名前である場合です。これら以外の場合はずべて、新規表の列には識別特性は備わりません。以下に例を示します。

- 全選択の選択リストに、ID 列名の複数のインスタンスが入っている場合 (つまり、同一列の複数回の選択の場合)
- 全選択の選択リストに複数の ID 列が含まれている (つまり、結合が関与している) 場合
- ID 列が選択リスト内の式に組み込まれている場合
- 全選択にセット演算 (UNION (合併)、EXCEPT (差)、または INTERSECT (論理積)) が含まれている場合

EXCLUDING IDENTITY COLUMN ATTRIBUTES

ソース結果表の定義から ID 列属性はコピーされません。

*as-result-table**column-name*

表の列の名前を指定します。列名のリストを指定する場合、リスト中の列の名前の数は、*fullselect* の結果表の列の数と同じ数でなければなりません。各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。列名のリストの指定がない場合、表の列は、*fullselect* の結果表の列名を継承します。

全選択の結果表に、無名列の重複列名がある場合には、列名のリストを指定する必要があります (SQLSTATE 42908)。無名列とは、定数、関数、式、またはセット演算から派生した列で、選択リストの AS 節によって名前が指定されていない列を指します。

AS

表の定義に使用され、照会をこの後に指定します。

fullselect

表の基礎となる照会を定義します。作成される列定義は、同じ照会で定義したビューの定義と同じになります。*fullselect* で参照される基本表の暗黙的な隠し列に対応する新規表の列は、新規表では隠し列とは見なされません。

各選択リスト・要素には名前が必要です (式には AS 節を使用します)。 *as-result-table* は、表の属性を定義します。

fullselect に *data-change-table-reference* 節を組み込むことはできません (SQLSTATE 428FL)。

型付き表または型付きビューを参照しない有効な *fullselect* を指定することができます。

WITH NO DATA

照会は、表を定義するときにだけ使われます。照会の結果を使用して表にデータが追加されることはありません。

表の列は、*fullselect* の結果である列の定義に基づいて定義されます。*fullselect* によって FROM 節の 1 つの表が参照される場合、その表の列である選択リスト項目は、参照される表の列名、データ・タイプ、そして NULL 可能特性を使って定義されます。

*materialized-query-definition**column-name*

表の列の名前を指定します。列名のリストを指定する場合、リスト中の列の名前の数は、全選択の結果表の列の数と同じ数でなければなりません。各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。列名のリストの指定がない場合、表の列は、全選択の結果表の列名を継承します。

fullselect の結果表に、無名列の重複列名がある場合には、列名のリストを指定する必要があります (SQLSTATE 42908)。無名列とは、定数、関数、式、またはセット演算から派生した列で、選択リストの AS 節によって名前が指定されていない列を指します。

AS

表の定義に使用され、表に含まれるデータを判別する照会をこの後に指定します。

fullselect

表の基礎となる照会を定義します。作成される列定義は、同じ照会で定義したビューの定義と同じになります。全選択で参照される基本表の暗黙的な隠し列に対応する新規表の列は、新規表では隠し列とは見なされません。

各選択リスト・エレメントには名前が必要です (式には AS 節を使用します)。 *materialized-query-definition* は、マテリアライズ照会表の属性を定義します。選択されたオプションは、次のような全選択の内容も定義します。

全選択に *data-change-table-reference* 節を組み込むことはできません (SQLSTATE 428FL)。

REFRESH DEFERRED または REFRESH IMMEDIATE が指定されていると、全選択で次のものを指定できません (SQLSTATE 428EC)。

- マテリアライズ照会表、宣言済み一時表、または任意の FROM 節での型付き表への参照
- ビューの全選択が、マテリアライズ照会表の全選択に関してリストされたいずれかの制限に違反する場合の、そのビューへの参照
- 参照タイプ (またはそのタイプに基づく特殊タイプ) である式
- 次のいずれかの属性を持つ関数:
 - EXTERNAL ACTION
 - LANGUAGE SQL
 - CONTAINS SQL
 - READS SQL DATA
 - MODIFIES SQL DATA

CREATE TABLE

- 物理的特性に依存する関数 (例えば、DBPARTITIONNUM、HASHEDVALUE、RID_BIT、RID)
- ROW CHANGE 式または行の ROW CHANGE TIMESTAMP 列の参照
- システム・オブジェクトに対する表またはビュー参照 (Explain 表も指定できません)
- 構造化タイプ、LOB タイプ (または LOB タイプに基づく特殊タイプ)、または XML タイプである式
- 保護対象表または保護対象ニックネームの参照

DISTRIBUTE BY REPLICATION を指定する場合、以下の制限が適用されます。

- GROUP BY 節は許可されていません。
- マテリアライズ照会表は単一の表だけを参照できます。すなわち、結合を組み込むことはできません。

REFRESH IMMEDIATE は、以下の場合に指定されます。

- 照会は副選択でなければなりません。ただし、例外として、UNION ALL は GROUP BY の入力表式においてサポートされます。
- 照会は再帰的であってはなりません。
- 照会に以下のものを含めることはできません。
 - ニックネームへの参照
 - deterministic 関数でない関数
 - スカラー全選択
 - 全選択を持つ述部
 - 特殊レジスター
 - グローバル変数
 - SELECT DISTINCT
 - エラー・トレラントな *nested-table-expression*
- FROM 節で複数の表またはビューを参照している場合、明示的な INNER JOIN 構文を使わずに内部結合を 1 つだけ定義できます。
- GROUP BY 節を指定する場合、以下の考慮事項が当てはまります。
 - サポートされている列関数は SUM、COUNT、COUNT_BIG、および GROUPING (DISTINCT は指定しない)。選択リストには COUNT(*) または COUNT_BIG(*) 列が含まれていなければなりません。マテリアライズ照会表の選択リストに SUM(X) (X は NULL 可能な引数) を含める場合は、マテリアライズ照会表の選択リストに COUNT(X) も含める必要があります。これらの列関数は、式の一部とすることはできません。
 - HAVING 節は許可されていません。
 - 複数パーティションのデータベース・パーティション・グループ内の場合、分散キーは GROUP BY 項目のサブセットでなければなりません。

- マテリアライズ照会表には重複した行があってはならず、GROUP BY 節が指定されているかどうかによって、この固有性要件に特有の、以下の制限が適用されます。
 - GROUP BY 節を指定する場合、以下の固有性に関連した制限が当てはまります。
 - すべての GROUP BY 項目が選択リストに含まれていること。
 - GROUP BY に GROUPING SETS、CUBE、または ROLLUP が含まれている場合、選択リスト内の GROUP BY 項目とそれに関連した GROUPING 列関数は、結果セットのユニーク・キーを形成してなければなりません。したがって、以下の制約事項が満たされていなければなりません。
 - どのグループ・セットも反復することはできない。例えば、ROLLUP(X,Y),X は指定できません。これは GROUPING SETS((X,Y),(X),(X)) と同等だからです。
 - X が、GROUPING SETS、CUBE、または ROLLUP 内に出現する NULL 可能な GROUP BY 項目である場合、選択リスト内に GROUPING(X) がなければならない。
 - GROUP BY 節を指定しない場合、以下の固有性に関連した制限が当てはまります。
 - マテリアライズ照会表のユニーク性要件は、それぞれの基礎表に定義されているユニーク・キー制約の 1 つからマテリアライズ・ビュー用のユニーク・キーを導出することによって達成されます。したがって、基礎表には少なくとも 1 つのユニーク・キー制約が定義されている必要があり、それらのキーの列が、マテリアライズ照会表定義の選択リストに現れていなければなりません。
- MAINTAINED BY FEDERATED_TOOL を指定する場合は、ニックネームへの参照だけが FROM 節で許可されます。

REFRESH DEFERRED を指定する場合は、以下のようにします。

- マテリアライズ照会表を作成するときに、その後のステートメントに関連したステージング表を用意するつもりであれば、マテリアライズ照会表の全選択は、REFRESH IMMEDIATE オプションでマテリアライズ照会表を作成するときに使用する全選択の場合と同じ制限事項および規則に従う必要があります。
- 照会が再帰的な場合は、照会処理の最適化のためにマテリアライズ照会表を使用しません。

全選択に GROUP BY 節が含まれるマテリアライズ照会表は、全選択で参照される表からの要約されたデータです。そのようなマテリアライズ照会表は、サマリー表と呼ばれます。サマリー表は、マテリアライズ照会表が特殊化したタイプの表です。

refreshable-table-options

マテリアライズ照会表の属性の再生可能オプションを定義します。

DATA INITIALLY DEFERRED

データは CREATE TABLE ステートメントの一部として表に挿入され

CREATE TABLE

ません。データを表に挿入するには、*table-name* (表名) を指定する REFRESH TABLE ステートメントが使用されます。

REFRESH

表のデータを保守する方法を示します。

DEFERRED

REFRESH TABLE ステートメントを使っていつでも表のデータをリフレッシュできます。表のデータには、REFRESH TABLE ステートメント処理時のスナップショットである照会結果が反映されるにすぎません。この属性を定義したシステム保守済みマテリアライズ照会表には、INSERT、UPDATE、または DELETE ステートメントを使用できません (SQLSTATE 42807)。この属性を定義したユーザー保守のマテリアライズ照会表には、INSERT、UPDATE、または DELETE ステートメントを使用できます。

IMMEDIATE

DELETE、INSERT、または UPDATE の一部として基礎表に加えられた変更は、マテリアライズ照会表にカスケードされます。その場合、表の内容は、どのポイント・イン・タイム指定でも、指定した *subselect* (副選択) を処理する場合と同じ内容になります。この属性を定義したマテリアライズ照会表には、INSERT、UPDATE、または DELETE ステートメントを使用できません (SQLSTATE 42807)。

ENABLE QUERY OPTIMIZATION

適切な状況下では、マテリアライズ照会表を照会最適化に使用することができます。

DISABLE QUERY OPTIMIZATION

マテリアライズ照会表を照会の最適化に使用しません。それでもその表を直接照会することはできます。

MAINTAINED BY

マテリアライズ照会表のデータが、システム、ユーザー、またはレプリケーション・ツールのいずれによって保守されるかを指定します。デフォルトは SYSTEM です。

SYSTEM

マテリアライズ照会表のデータがシステムによって保守されるように指定します。

USER

マテリアライズ照会表のデータがユーザーによって保守されるように指定します。ユーザーは、ユーザー保守済みマテリアライズ照会表に対して、更新、削除、また挿入操作を許可されます。システム保守のマテリアライズ照会表で使用される REFRESH TABLE ステートメントは、ユーザー保守のマテリアライズ照会表に対しては呼び出せません。REFRESH DEFERRED マテリアライズ照会表だけが、MAINTAINED BY USER として定義できます。

FEDERATED_TOOL

マテリアライズ照会表のデータがレプリケーション・ツールによって保守されるように指定します。システム保守のマテリアライズ照

会表で使用される REFRESH TABLE ステートメントは、フェデレーテッド・ツール保守のマテリアライズ照会表に対しては呼び出せません。REFRESH DEFERRED のマテリアライズ照会表だけが、MAINTAINED BY FEDERATED_TOOL として定義されることが可能です。

staging-table-definition

関連付けられたマテリアライズ照会表を通して間接的に、ステージング表によりサポートされる照会を定義します。マテリアライズ照会表の基礎表も、関連付けられたステージング表の基礎表です。ステージング表は、基礎表の内容と同期化するためにマテリアライズ照会表に適用する必要のある変更を収集します。

staging-column-name

ステージング表の列の名前を指定します。列名のリストを指定する場合、ステージング表を定義するマテリアライズ照会表の列の名前の数よりも、2 つ名前の数が多くなければなりません。そのマテリアライズ照会表がマテリアライズ照会表の複製である場合、またはマテリアライズ照会表を定義している照会が GROUP BY 節を含んでいない場合の列名のリストは、ステージング表を定義するマテリアライズ照会表の列の名前の数よりも、3 つ名前の数が多くなければなりません。各 column-name (列名) は、ユニークかつ非修飾でなければなりません。列名のリストの指定がない場合、表の列は、関連付けられたマテリアライズ照会表の列名を継承します。追加の列は GLOBALTRANSID および GLOBALTRANSTIME と名づけられ、3 番目の列が必要な場合には OPERATIONTYPE という名前が付けられます。

表 19. ステージング表に加えられる追加の列

列名	データ・タイプ	列の説明
GLOBALTRANSID	CHAR(8) FOR BIT DATA	伝搬された各行のグローバル・トランザクション ID
GLOBALTRANSTIME	CHAR(13) FOR BIT DATA	トランザクションのタイム・スタンプ
OPERATIONTYPE	INTEGER	伝搬された行に対する操作。挿入、更新、または削除のいずれか。

関連付けられたマテリアライズ照会表の任意の列が、生成された列名と重複する場合には、列名のリストを指定する必要があります (SQLSTATE 42711)。

FOR *table-name2*

ステージング表の定義に使用されるマテリアライズ照会表を指定します。名前 (暗黙的または明示的なスキーマ名を含む) は、REFRESH DEFERRED に定義された現行サーバーに存在するマテリアライズ照会表を指定してなければなりません。関連付けられたマテリアライズ照会表の全選択は、REFRESH IMMEDIATE オプションによってマテリアライズ照会表を作成するのに使用した全選択と同じ制限事項および規則に従う必要があります。

ステージング表の内容が関連付けられたマテリアライズ照会表および基礎ソース表と整合する場合には、マテリアライズ照会表をリフレッシュするために、REFRESH TABLE ステートメントを呼び出して、ステージング表の内容を使用できます。

PROPAGATE IMMEDIATE

削除、挿入、または更新操作の一部として基礎表に加えられた変更は、ステージング表の同じ削除、挿入、または更新操作にカスケードされます。ステージング表が不整合としてマークされていないのであれば、任意のポイント・イン・タイムに、ステージング表の内容は最後にマテリアライズ照会表をリフレッシュしてから基礎表までの差分変更します。

ORGANIZE BY DIMENSIONS (*column-name,...*)

表データをクラスター化するために使用する、各列または列のグループのディメンションを指定します。ディメンション・リストで括弧を使用すると、列のグループは 1 つのディメンションとして扱われよう指定されます。

DIMENSIONS キーワードはオプションです。その定義がこの節を指定する表は、マルチディメンション・クラスタリング (MDC) 表として知られます。

クラスタリング・ブロック索引はそれぞれ指定されたディメンション用に自動的に保持され、節で使用されるすべての列で構成されるブロック索引は、どのクラスタリング・ブロック索引にもすべての列が含まれていない場合には保持されず。ORGANIZE BY 節で使用される列セットは、CREATE INDEX ステートメントの規則に従う必要があります。

ORGANIZE BY 節で指定された各列名は、表に対して定義されなければなりません (SQLSTATE 42703)。ディメンションはディメンション・リストで複数回現れることはありません (SQLSTATE 42709)。ディメンションに ROW CHANGE TIMESTAMP 列を含めることはできません (SQLSTATE 429BV)。表にデータ・タイプ XML の列を組み込むことはできません (SQLSTATE 42601)。

表のページは、表スペースのエクステント・サイズと同じサイズのブロックに配置され、各ブロックの行すべては同じディメンション値の組み合わせを含みます。

表がマルチディメンション・クラスタリング (MDC) 表であると同時にパーティション表でもあることは可能です。そのような表の列は、*range-partition-spec* と MDC キーの両方で使用できます。ただし、表パーティションは、マルチ列であり、マルチディメンションではありません。

ORGANIZE BY KEY SEQUENCE *sequence-key-spec*

表を、指定された範囲のキー・シーケンスの値に基づいて、固定サイズの昇順キー・シーケンスに編成するよう指定します。このような方法で編成された表を、レンジ・クラスター表 といいます。定義された範囲内にある有効なキー値のそれぞれには、物理表上の位置があらかじめ決定されています。レンジ・クラスター表に必要なストレージは、表が作成される時点で使用可能になっていなければならず、また、ストレージには、指定された範囲内にある行の数に行のサイズを乗算しただけのスペースが必要です (スペース所要量の決定に関する詳細は、行サイズおよびバイト・カウントを参照してください)。

column-name

レンジ・クラスター表のシーケンスを決定するユニーク・キーに含まれる、表の列を指定します。列のデータ・タイプは SMALLINT、INTEGER、または BIGINT (SQLSTATE 42611) でなければならず、また列は NOT NULL (SQLSTATE 42831) として定義される必要があります。シーケンス・キーの中で、同じ列を複数回指定することはできません。指定する列の数は 64 を超えてはなりません (SQLSTATE 54008)。

列ごとに昇順で指定されたキー・シーケンスで、列のカatalogにユニーク索引項目が自動的に作成されます。索引の名前は、SQL の後に文字のタイム・スタンプ (yymmddhhmmssxxx) が続き、スキーマ名として SYSIBM がついたものになります。表の編成はこのキーによって配列されているため、実際の索引オブジェクトはストレージに作成されません。同じ列に、レンジ・クラスター表のシーケンス・キーとして主キーまたはユニークな制約が定義されている場合は、この同じ索引項目が制約に使用されます。

キー・シーケンスの指定には、列の制約を反映するチェック制約が存在します。DISALLOW OVERFLOW 節が指定されていると、チェック制約の名前が RCT になり、チェック制約が施行されます。ALLOW OVERFLOW 節が指定されていると、チェック制約の名前が RCT_OFLOW になり、チェック制約は施行されません。

STARTING FROM *constant*

column-name の範囲の下限となる定数を指定します。指定された定数より小さい値は、ALLOW OVERFLOW オプションが指定されている場合以外、許可されません。*column-name* が SMALLINT または INTEGER 列の場合は、定数を INTEGER 定数にする必要があります。*column-name* が BIGINT 列の場合は、定数を INTEGER または BIGINT 定数にする必要があります (SQLSTATE 42821)。開始の定数が指定されない場合、デフォルト値は 1 です。

ENDING AT *constant*

column-name の範囲の上限となる定数を指定します。指定された定数より大きい値は、ALLOW OVERFLOW オプションが指定されている場合以外、許可されません。なお、終了の定数は、開始の定数より大きくなければなりません。*column-name* が SMALLINT または INTEGER 列の場合は、定数を INTEGER 定数にする必要があります。*column-name* が BIGINT 列の場合は、定数を INTEGER または BIGINT 定数にする必要があります (SQLSTATE 42821)。

ALLOW OVERFLOW

レンジ・クラスター表で、定義された範囲外の値を行のキー値として許可することを指定します。レンジ・クラスター表がオーバーフローを許可するように作成される場合、キー値が範囲外にある行にはあらかじめ決定された配列がなく、行は定義された範囲の一番下に置かれます。これらのオーバーフロー行に関する操作は、定義された範囲内にキー値がある行での操作と比べて非効率的です。

DISALLOW OVERFLOW

レンジ・クラスター表で、定義された範囲外の値を行のキー値として許可しないことを指定します (SQLSTATE 23513)。オーバーフロー行を許可しないレンジ・クラスター表では、常に、すべての行が昇順キー・シーケンスで保守されます。

PCTFREE *integer*

各ページに残すフリー・スペースのパーセンテージを指定します。各ページの最初の行は、制約なしに追加されます。ページに行が追加される際には、少なくとも *integer* で指定された分 (%) のフリー・スペースがページに残されます。*integer* の値は 0 から 99 です。システム・カタログ

CREATE TABLE

(SYSCAT.TABLES) の PCTFREE 値 -1 は、デフォルト値として解釈されます。表ページのデフォルト PCTFREE 値は 0 です。

DATA CAPTURE

データベース間のデータのレプリケーションに関する余分な情報を、ログに書き込むかどうかを指定します。この節は、副表を作成する際には指定できません (SQLSTATE 42613)。

表が型付き表である場合、このオプションはサポートされません (SQLSTATE 428DH または 42HDR)。

データ・キャプチャーと行圧縮の間には互換性がありません (SQLSTATE 42997)。

NONE

追加情報をログに記録しないことを指定します。

CHANGES

この表に対する SQL 変更についての追加情報をログに書き込むことを指定します。このオプションは、表を複製する場合で、キャプチャー・プログラムを使用してログからこの表に対する変更内容をキャプチャーする場合に必須です。

表のスキーマ名 (暗黙または明示名) が 18 バイトより長い場合、このオプションはサポートされません (SQLSTATE 42997)。

IN *tablespace-name1*,...

表を作成する表スペースを指定します。表スペースは既存のものでなければなりません。すべてが同じデータベース・パーティション・グループに含まれていることも必要です。また、すべてが REGULAR DMS 表スペースであるか、すべてが LARGE DMS 表スペースであるか、すべてが SMS 表スペースである必要もあります (SQLSTATE 42838)。ステートメントの許可 ID には、それらの表スペースに対する USE 特権が必要です。

表レベルでは、最大で 1 個の IN 節だけが認められています。1 つの表が使用するすべてのデータ表スペースは、同じページ・サイズ、同じエクステント・サイズでなければなりません。すべてが同じプリフェッチ・サイズでない場合は、警告が戻されます。すべての表スペースが AUTOMATIC プリフェッチ・サイズになっていれば、警告は戻されません。

表スペースを 1 つだけ指定した場合、すべての表パーツはその表スペースに保管されます。副表は表階層のルート表から表スペースを継承するので、この節を作成の際に指定することはできません (SQLSTATE 42613)。この節を指定しない場合には、この表の表スペースは次のように決められます。

```
IF 表スペース IBMDEFAULTGROUP (ユーザーがそれに対する
  USE 特権を持つ) が十分なページ・サイズをもって存在する場合
  THEN それを選択します。
ELSE IF 表スペース (ユーザーがそれに対する USE 特権を持つ) が
  十分なページ・サイズをもって存在する場合
  (複数表スペース修飾の場合は下記を参照)
  THEN それを選択します。
ELSE エラーを出します (SQLSTATE 42727)
```

ELSE IF 条件で複数の表スペースが指定されている場合、最小限必要なページ・サイズをもつ表スペースを選択します。複数の表スペースが適格な場合、USE 特権が誰に付与されているかに応じて、以下の優先順位で表スペースを選択します。

1. 許可 ID
2. 許可 ID を保有するグループ
3. PUBLIC

それでも複数の表スペースがそれにあてはまる場合は、最終選択はデータベース・マネージャーによって行われます。

表スペースの判別は、以下の場合に変更されることがあります。

- 表スペースをドロップまたは作成するとき
- USE 特権を付与または取り消すとき

パーティション表では、それぞれのデータ・パーティションを複数の表スペースに配分できます。複数の表スペースを指定する場合、そのすべての表スペースは既存のものでなければならず、すべてが SMS 表スペースであるか、すべてが REGULAR DMS 表スペースであるか、すべてが LARGE DMS 表スペースであることが必要です (SQLSTATE 42838)。ステートメントの許可 ID には、指定したすべての表スペースに対する USE 特権が必要です。

十分な表のページ・サイズは、行のバイト・カウントか列の数のいずれかによって決まります。詳しくは、行サイズを参照してください。

LARGE 表スペースに表を配置すると、以下のようになります。

- REGULAR 表スペースに配置する表よりもサイズを大きくできます。表と表スペースの制限値の詳細については、『SQL の制限値』を参照してください。
- 表のデータ・ページあたり、255 を超える行数をサポートできるので、データ・ページのスペース使用効率が向上します。
- REGULAR 表スペースに配置した表に索引を定義する場合に比べ、索引の 1 つの行項目あたり 2 バイトが追加で必要になります。

CYCLE または NO CYCLE

明示的な表スペースのないデータ・パーティションの数が、指定されたデータ・パーティションの数を超えてもよいかどうかを指定します。

CYCLE

明示的な表スペースのないデータ・パーティションの数が、指定されたデータ・パーティションの数を超えた場合に、データ・パーティションにラウンドロビン方式で表スペースを割り当てることを指定します。

NO CYCLE

明示的な表スペースのないデータ・パーティションの数が、指定されたデータ・パーティションの数を超えてはならないことを指定します (SQLSTATE 428G1)。このオプションを指定すると、データ・パーティションにラウンドロビン方式で表スペースが割り当てられることはありません。

tablespace-options

索引または長形式列の値を保管する表スペースを指定します。表スペースのタイプについては、『CREATE TABLESPACE』を参照してください。

INDEX IN *tablespace-name2*

表の索引を作成する表スペースを指定します。指定する表スペースは既存でなければなりません。表のデータが DMS 表スペースにある場合は DMS 表スペースでなければならず、パーティション表のデータが SMS 表スペースにある場合は SMS 表スペースでなければなりません。ステートメントの許可 ID には、その表スペースに対する USE 特権が必要です。その表スペースは、*tablespace-name1* と同じデータベース・パーティション・グループに含まれている必要があります (SQLSTATE 42838)。

索引を組み込む表スペースの指定は、表の作成時に行うことができます。また、パーティション表の場合は、CREATE INDEX ステートメントの IN 節を指定することによって行うことも可能です。表スペースに対する USE 特権があるかどうかのチェックは、索引の作成時ではなく表の作成時に行われます。

LONG IN *tablespace-name3*

長形式列の値を保管する表スペースを指定します。長形式列には、LOB データ・タイプ、XML タイプ、これらのいずれかをソース・タイプとする特殊タイプ、インラインで保管できない値をもつユーザー定義の構造化タイプで定義された列が含まれます。このオプションは、IN 節で DMS 表スペースを指定した場合にのみ使用できます。

指定した表スペースは存在していなければなりません。これは、データが保管されている同じ表スペースである場合、REGULAR 表スペースとすることもできます。そうでない場合は、ステートメントの許可 ID が USE 特権を保持する LARGE DMS 表スペースでなければなりません。さらに、これは *tablespace-name1* と同じデータベース・パーティション・グループに含まれている必要があります (SQLSTATE 42838)。

長形式、LOB、XML の列を組み込む表スペースの指定は、表の作成時にのみ行えます。USE 特権があるかどうかのチェックは、長形式列または LOB 列を追加するときではなく表の作成時に行われます。

パーティション表で LONG IN 節の使用を制御する規則については、『パーティション表でのラージ・オブジェクトの動作』を参照してください。

distribution-clause

データベース・パーティションの方式、つまり複数のデータベース・パーティションにデータを配分させる方法を指定します。

DISTRIBUTE BY HASH (*column-name,...*)

複数のデータベース・パーティションにデータを分散させる方式として、分散キー という指定の列でデフォルトのハッシュ機能を使用する方式を指定します。*column-name* は、表の列を指定する非修飾名でなければなりません (SQLSTATE 42703)。同じ列を複数回指定することはできません (SQLSTATE 42709)。データ・タイプが BLOB、CLOB、DBCLOB、XML、またはこれらのタイプのいずれかに基づく特殊タイプ、構造化タイプの列

は、分散キーの一部として使用できません (SQLSTATE 42962)。分散キーに ROW CHANGE TIMESTAMP 列を含めることはできません (SQLSTATE 429BV)。副表では、分散キーを指定できません (SQLSTATE 42613)。分散キーは、表階層のルート表から、またはデータ・タイプ XML の列がある表から継承されるからです (SQLSTATE 42997)。この節の指定がなく、この表が複数データベース・パーティションの複数パーティション・データベース・パーティション・グループに存在する場合、その分散キーは次のように定義されます。

- 表が型付き表である場合は、オブジェクト ID 列が分散キーになります。
- 主キーが定義されている場合は、その主キーの最初の列が分散キーになります。
- その他の場合は、分散キーとして有効なデータ・タイプの最初の列が分散キーになります。

分散キーの列は、ユニーク制約を構成する列のサブセットでなければなりません。

デフォルトの分散キーの要件を満たす列が存在しない場合は、分散キーなしで表が作成されます。このような表は、単一パーティションのデータベース・パーティション・グループに対して定義された表スペースでのみ認められています。

単一パーティションのデータベース・パーティション・グループに対して定義された表スペースの表の場合は、分散キーとして有効なデータ・タイプの任意の列の集合を分散キーの定義に使用できます。この節を指定しない場合は、分散キーが作成されません。

分散キーに関連した制約事項については、規則を参照してください。

DISTRIBUTE BY REPLICATION

表が定義される表スペースのデータベース・パーティション・グループの各データベース・パーティションに対して、表に保管されたデータを物理的に複製することを指定します。つまり、それぞれのデータベース・パーティションには、表のデータすべてのコピーが存在することになります。このオプションは、マテリアライズ照会表にのみ指定できます (SQLSTATE 42997)。

partitioning-clause

データベース・パーティション内でデータを分割する方法を指定します。

PARTITION BY RANGE *range-partition-spec*

表の範囲パーティション方式を指定します。

partition-expression

キー・データを指定します。このキー・データに対して、ターゲット・データ・パーティションを決定するための範囲を定義します。

column-name

データ・パーティション・キーの列を指定します。 *column-name* は、表の列を指定する非修飾名でなければなりません (SQLSTATE 42703)。同じ列を複数回指定することはできません (SQLSTATE 42709)。データ・タイプが BLOB、CLOB、DBCLOB、これらのタ

CREATE TABLE

タイプのいずれかに基づく特殊タイプ、構造化タイプの列は、表パーティション・キーの一部として使用できません (SQLSTATE 42962)。

範囲仕様に使用される数値リテラルには、数値リテラル用の規則が適用されます。数値列に対応する範囲で使用されるすべての数値リテラル (10 進浮動小数点特殊値を除く) は、数値定数に指定された規則に従って、整数、浮動小数点数、または 10 進定数として解釈されます。結果として 10 進浮動小数点列では、データ・パーティションの範囲仕様に使用できる最小および最大の数値定数の値は、それぞれ、最小の DOUBLE 値および最大の DOUBLE 値になります。10 進浮動小数点特殊値を範囲仕様で使用することができます。すべての 10 進浮動小数点特殊値は、MINVALUE よりも大きく MAXVALUE よりも小さいものとして解釈されます。

範囲パーティション列に ROW CHANGE TIMESTAMP 列を含めることはできません (SQLSTATE 429BV) 指定する列の数が 16 を超えてはなりません (SQLSTATE 54008)。

NULLS LAST

NULL 値の比較順位を上位に設定することを指定します。

NULLS FIRST

NULL 値の比較順位を下位に設定することを指定します。

partition-element

データ・パーティション・キーの範囲と、その範囲内の表の行を保管する表スペースを指定します。

PARTITION *partition-name*

データ・パーティションの名前を指定します。この名前は、表の他のいずれのデータ・パーティションとも同じではありません (SQLSTATE 42710)。この節を指定しない場合は、「PART」の後に文字形式の整数値が付いた名前になります (このようにして、その表での固有の名前になります)。

boundary-spec

範囲パーティションの境界を指定します。最下位の範囲パーティションには *starting-clause* を組み込み、最上位の範囲パーティションには *ending-clause* を組み込む必要があります (SQLSTATE 56016)。最下位と最上位の間にある範囲パーティションには、*starting-clause* と *ending-clause* のいずれかまたは両方の節を組み込みます。 *ending-clause* だけを指定する場合は、その直前の範囲パーティションにも *ending-clause* を組み込んでおく必要があります (SQLSTATE 56016)。

starting-clause

データ・パーティションの範囲の下限を指定します。開始値は、少なくとも 1 つ指定しなければなりません、開始値の数がデータ・パーティション・キーの列の数を超えてはなりません (SQLSTATE 53038)。指定した値の数が列の数よりも少なければ、残りの値は暗黙的に MINVALUE になります。

STARTING FROM

この後に、*starting-clause* を指定します。

constant

対応する *column-name* のデータ・タイプに割り当てることができるデータ・タイプの定数値を指定します (SQLSTATE 53045)。この値は、表の他の *boundary-spec* の範囲内にあってはなりません (SQLSTATE 56016)。

MINVALUE

対応する *column-name* のデータ・タイプの最小可能値より小さい値を指定します。

MAXVALUE

対応する *column-name* のデータ・タイプの最大可能値より大きい値を指定します。

INCLUSIVE

指定した範囲値をデータ・パーティションに含めることを指定します。

EXCLUSIVE

指定した *constant* 値をデータ・パーティションから除外することを指定します。この指定は、**MINVALUE** または **MAXVALUE** を指定した場合は無視されます。

ending-clause

データ・パーティションの範囲の上限を指定します。開始値は、少なくとも 1 つ指定しなければなりません。開始値の数がデータ・パーティション・キーの列の数を超えてはなりません (SQLSTATE 53038)。指定した値の数が列の数よりも少なければ、残りの値は暗黙的に **MAXVALUE** になります。

ENDING AT

この後に、*ending-clause* を指定します。

constant

対応する *column-name* のデータ・タイプに割り当てることができるデータ・タイプの定数値を指定します (SQLSTATE 53045)。この値は、表の他の *boundary-spec* の範囲内にあってはなりません (SQLSTATE 56016)。

MINVALUE

対応する *column-name* のデータ・タイプの最小可能値より小さい値を指定します。

MAXVALUE

対応する *column-name* のデータ・タイプの最大可能値より大きい値を指定します。

INCLUSIVE

指定した範囲値をデータ・パーティションに含めることを指定します。

EXCLUSIVE

指定した *constant* 値をデータ・パーティションから除外す

ることを指定します。この指定は、MINVALUE または MAXVALUE を指定した場合は無視されます。

IN *tablespace-name*

データ・パーティションが保管される表スペースを指定します。指定する表スペースは、パーティション表の他の表スペースと同じページ・サイズ、同じデータベース・パーティション・グループ、同じスペース管理方式でなければなりません (SQLSTATE 42838)。ステートメントの許可 ID には、その表スペースに対する USE 特権が必要です。この節を指定しない場合は、デフォルトで、表に指定されている表スペースのリストからラウンドロビン方式で表スペースが割り当てられます。

CREATE TABLE ステートメントで LONG IN 節を指定しない場合は、データ・パーティションと同じ表スペースにラージ・オブジェクトが配置されます。

CREATE TABLE ステートメントまたは CREATE INDEX ステートメントで INDEX IN 節を指定しない場合は、表の最初の可視パーティションまたは最初に追加されたパーティションと同じ表スペースに索引が配置されます。

LONG IN *tablespace-name*

長い列データを含むデータ・パーティションが保管される表スペースを指定します。指定する表スペースは、パーティション表の他の表スペースおよびデータ・パーティションと同じページ・サイズ、同じデータベース・パーティション・グループ、同じスペース管理方式でなければなりません (SQLSTATE 42838)。ステートメントの許可 ID には、その表スペースに対する USE 特権が必要です。指定した表スペースのページ・サイズおよびエクステント・サイズは、パーティション表の他のデータ・パーティションのページ・サイズおよびエクステント・サイズと異なる可能性があります。

EVERY (*constant*)

自動生成形式の構文を使用した場合に、各データ・パーティション範囲の幅を指定します。データ・パーティションは、STARTING FROM 値から始まる範囲にこの数の値を組み込む形で作成されます。この形式の構文は、数値または日時の 1 つの列でパーティション化されている表についてのみサポートされています (SQLSTATE 53038)。

パーティション・キー列が数値タイプの場合、最初のパーティションの開始値は、starting-clause に指定されている値になります。最初のパーティションと他のすべてのパーティションの終了値は、パーティションの開始値に、EVERY 節の *constant* として指定されている増分値を加算する形で計算されます。他のすべてのパーティションの開始値は、直前のパーティションの開始値に、EVERY 節の *constant* として指定されている増分値を加算する形で計算されません。

パーティション・キー列が DATE または TIMESTAMP の場合、最初のパーティションの開始値は、starting-clause に指定されている値になります。最初のパーティションと他のすべてのパーティション

の終了値は、パーティションの開始値に、EVERY 節のラベル付き期間として指定されている増分値を加算する形で計算されます。他のすべてのパーティションの開始値は、直前のパーティションの開始値に、EVERY 節のラベル付き期間として指定されている増分値を加算する形で計算されます。

数値列の場合、EVERY の値は正の数値定数でなければならず、日時列の場合、EVERY の値はラベル付き期間でなければなりません (SQLSTATE 53045)。

COMPRESS

表の行にデータ圧縮を適用するかどうかを指定します。

YES

データ行圧縮を使用可能にすることを指定します。表に対する挿入と更新の操作で、圧縮が行われるようになります。表に十分なデータが取り込まれた後で、コンプレッション・ディクショナリーが自動的に作成され、行が圧縮の対象になります。

NO

データ行圧縮を使用不可にすることを指定します。

VALUE COMPRESSION

使用される行形式を判別します。それぞれのデータ・タイプは、使用される行形式に応じた、異なるバイト・カウントを持ちます。詳しくは、バイト・カウントを参照してください。表が型付き表である場合、このオプションは、型付き表階層のルート表に対してのみサポートされます (SQLSTATE 428DR)。

NULL 値は 3 バイトを使用して保管されます。これは、すべてのデータ・タイプの列に対して VALUE COMPRESSION がアクティブでない場合と同じスペース、またはそれより少ないスペースです (CHAR(1) は例外)。列が NULL 可能として定義されているかいないかは、行サイズ計算には影響しません。データ・タイプが VARCHAR、VARGRAPHIC、LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB、BLOB、XML である列の、長さがゼロのデータ値は、2 バイトだけ使用して保管されます。これは、VALUE COMPRESSION がアクティブでない場合に必要とされるストレージを下回ります。COMPRESS SYSTEM DEFAULT オプションを使用して列を定義すると、列のシステム・デフォルト値も 3 バイトの合計ストレージを使用して保管できるようになります。これをサポートする行形式は、各データ・タイプのバイト・カウントを識別し、NULL、長さがゼロの値、またはシステム・デフォルト値の関係した更新を行う際に、データ・フラグメントの原因となる傾向があります。

WITH RESTRICT ON DROP

表をドロップできないこと、また、表を含む表スペースをドロップできないことを指定します。

NOT LOGGED INITIALLY

表を作成する作業単位と同一の作業単位の挿入、削除、更新、索引の作成、索引のドロップ、または表の変更操作によって表に対して行われた変更は、いずれもログに記録されません。このオプションを使用する際の他の考慮事項については、このステートメントの『注』のセクションを参照してください。

カタログの変更と、ストレージに関連する情報は、以後の作業単位で表に対して行われた操作と同様にすべてログ記録されます。

CREATE TABLE

注: 活動化された NOT LOGGED INITIALLY 属性を持つ表に対してログに記録されない活動が生じ、ステートメントに障害が起こる (ロールバックが発生する)、または ROLLBACK TO SAVEPOINT が実行される場合には、その作業単位全体がロールバックされます (SQL1476N)。さらに、NOT LOGGED INITIALLY 属性が活動化されている表は、ロールバックされた後にアクセス不能としてマークされ、ドロップしかできなくなります。したがって、NOT LOGGED INITIALLY 属性が活動化されている作業単位内のエラーは、最小限に抑えるべきです。

CCSID

表に格納されるストリング・データのコード化スキームを指定します。CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは CCSID UNICODE、他のすべてのデータベースでは CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、CCSID ASCII を指定することはできません (SQLSTATE 56031)。

UNICODE

ストリング・データが Unicode でエンコードされることを指定します。データベースが Unicode データベースの場合、文字データは UTF-8、GRAPHIC データは UCS-2 になります。データベースが Unicode データベースでない場合は、文字データは UTF-8 になります。

データベースが Unicode データベースでない場合、CCSID UNICODE を指定して表を作成できますが、以下の規則が適用されます。

- 表を作成するより前に、代替照合シーケンスをデータベース構成に指定する必要があります (SQLSTATE 56031)。CCSID UNICODE 表は、データベース構成に指定されている代替照合シーケンスと照合されます。
- CCSID ASCII を指定して作成された表または表関数と、CCSID UNICODE を指定して作成された表または表関数とを、1 つの SQL ステートメント内で両方とも使用することはできません (SQLSTATE 53090)。このことは、ステートメント内で直接参照されている表および表関数、および間接的に (例えば、参照整合性制約、トリガー、マテリアライズ照会表、およびビューの本体内の表によって) 参照されている表および表関数に適用されます。
- CCSID UNICODE を指定して作成された表は、SQL 関数または SQL メソッド内では参照できません (SQLSTATE 560C0)。
- CCSID UNICODE を指定して作成された表を参照する SQL ステートメントは、SQL 関数または SQL メソッドを呼び出すことができません (SQLSTATE 53090)。
- GRAPHIC タイプ、XML タイプ、およびユーザー定義タイプは CCSID UNICODE 表内では使用できません (SQLSTATE 560C1)。
- 同じ表に CCSID UNICODE 節と DATA CAPTURE CHANGES 節の両方を指定することはできません (SQLSTATE 42613)。
- Explain 表は CCSID UNICODE では作成できません (SQLSTATE 55002)。

- 宣言済みグローバル一時表は CCSID UNICODE では作成できません (SQLSTATE 56031)。
- CCSID UNICODE 表は CREATE SCHEMA ステートメントでは作成できません (SQLSTATE 53090)。
- ロード操作の例外表の CCSID は、この操作のターゲット表と同じでなければなりません (SQLSTATE 428A5)。
- SET INTEGRITY ステートメントの例外表の CCSID は、このステートメントのターゲット表と同じでなければなりません (SQLSTATE 53090)。
- イベント・モニター・データのターゲット表は CCSID UNICODE として宣言されているはなりません (SQLSTATE 55049)。
- CCSID UNICODE 表を参照するステートメントは、DB2 バージョン 8.1 以降のクライアントからのみ呼び出すことができます (SQLSTATE 42997)。
- SQL ステートメントは常にデータベース・コード・ページで解釈されます。特にこのことは、リテラル、16 進数リテラル、および区切り ID 内のすべての文字がデータベース・コード・ページで表記されていないと意味します。そうでないと、文字は置換文字によって置き換えられてしまいます。

呼び出される SQL ステートメント内の表の CCSID に関係なく、アプリケーション内のホスト変数は常にアプリケーション・コードで表記されます。DB2 は、アプリケーション・コード・ページとセクション・コード・ページ間でのデータ変換の必要に応じて、コード・ページ変換を実行します。レジストリー変数 DB2CODEPAGE をクライアント側で設定して、アプリケーション・コード・ページを変更することができます。

SECURITY POLICY

表に関連付けるセキュリティー・ポリシーの名前を指定します。

policy-name

現行のサーバーにすでに存在するセキュリティー・ポリシーを指定します (SQLSTATE 42704)。

OPTIONS (ADD *table-option-name string-constant, ...*)

表オプションは、リモート基本表を識別するために使用します。

table-option-name はオプションの名前です。 *string-constant* は、表オプションの設定を指定します。 *string-constant* は単一引用符で囲む必要があります。

リモート・サーバー (CREATE SERVER ステートメントに指定されたサーバー名) は、OPTIONS 節に指定します。OPTIONS 節を使用して、作成中のリモート基本表のスキーマまたは被修飾名をオーバーライドすることもできます。

スキーマ名を指定することが推奨されています。リモート・スキーマ名が指定されていない場合、表名の修飾子が使用されます。表名に修飾子がない場合、ステートメントの許可 ID が使用されます。

リモート基本表の被修飾名が指定されていない場合、*table-name* が使用されます。

規則

- すべての構造化タイプ列または XML タイプ列のインライン長さも含め、列のバイト・カウントの合計は、表スペースのページ・サイズに基づく行サイズの限界を超えてはなりません (SQLSTATE 54010)。詳しくは、バイト・カウントを参照してください。型付き表の場合、表階層のルート表の列や表階層内の各副表で新たに追加される列すべてに対しては、バイト・カウントが適用されます (追加される副表列は、NULL 不可として定義されたとしても、バイト・カウントの際には NULL 可能と見なされます)。また、各行がどの副表からきたものかを識別するため、4 バイトのオーバーヘッドが追加されます。
- 表内に存在する列の数は、1 012 個を超えてはなりません (SQLSTATE 54011)。型付き表の場合は、表階層内のすべての副表タイプに含まれている属性の合計が 1,010 個を超えてはなりません。
- 型付き表のオブジェクト ID 列は更新できません (SQLSTATE 42808)。
- 表に対して定義されたユニーク・キー制約または主キー制約は、分散キーのスーパーセットでなければなりません (SQLSTATE 42997)。
- 以下の規則は、複数データベース・パーティション・データベースに対してのみ適用されます。
 - LOB、XML、これらのタイプのうちのいずれかに基づく特殊タイプ、構造化タイプの列だけで構成された表は、単一パーティションのデータベース・パーティション・グループで定義されている表スペース内でしか作成できません。
 - 複数パーティションのデータベース・パーティション・グループに対して定義された表スペースの表の分散キー定義は変更できません。
 - 型付き表の分散キー列は OID 列にする必要があります。
 - タイプ XML の列は使用できません。
 - パーティション・ステージング表はサポートされていません。
- レンジ・クラスター表には、次のような制限が適用されます。
 - 複数のデータベース・パーティションがあるデータベースでは、レンジ・クラスター表を指定できません (SQLSTATE 42997)。
 - クラスター索引は作成できません。
 - 列を追加するための表の変更はサポートされていません。
 - 列のデータ・タイプを変更するための表の変更はサポートされていません。
 - PCTFREE を変更するための表の変更はサポートされていません。
 - APPEND ON を設定するための表の変更はサポートされていません。
 - DETAILED 統計は使用できません。
 - 表の移植にロード・ユーティリティを使用することはできません。
 - 列をタイプ XML にすることはできません。
- 表にセキュリティ・ポリシーが関連付けられていて、タイプ DB2SECURITYLABEL の列または SECURED WITH 節で定義されている列が含まれていなければ、表は保護されません。前者の場合は、行レベルの細分度で表が保護されていることを意味し、後者の場合は、列レベルの細分度で表が保護されていることを意味します。
- 表にセキュリティ・ポリシーが関連付けられていなければ、タイプ DB2SECURITYLABEL の列を宣言できません (SQLSTATE 55064)。

- セキュリティー・ポリシーは、型付き表には追加できません (SQLSTATE 428DH)。マテリアライズ照会表やステージング表にも追加できません (SQLSTATE 428FG)。
- エラー・トレラントな *nested-table-expression* は、*materialized-query-definition* の全選択に指定できません (SQLSTATE 428GG)。

注

- まだ存在していないスキーマ名を用いて表を作成すると、ステートメントの許可 ID に `IMPLICIT_SCHEMA` 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は `SYSIBM` です。スキーマに対する `CREATEIN` 特権が `PUBLIC` に付与されます。
- 外部キーが指定されると、
 - 親表の削除を行うパッケージはすべて無効になります。
 - 親キーの少なくとも 1 つの列に対して更新使用の指定があるパッケージは、すべて無効になります。
- 副表を作成すると、表階層内の表のいずれかに従属しているパッケージがすべて無効になります。
- それぞれ 4 000 および 2 000 より大きい数値の `VARCHAR` および `VARGRAPHIC` 列は、`SYSFUN` スキーマの関数での入力パラメーターとして使用しないでください。関数にこの長さを超過する引数値を指定して呼び出すと、エラーが発生します (SQLSTATE 22001)。
- 参照制約の削除規則または更新規則として `NO ACTION` または `RESTRICT` を使用すると、制約がいつ適用されるかが決まります。`RESTRICT` の削除規則または更新規則は、`CASCADE` や `SET NULL` などの変更規則を伴う参照制約を含む他のすべての制約の前に適用されます。`NO ACTION` の削除規則または更新規則は、他の参照制約の後で適用されます。この動作の違いが明白になる例の 1 つとして、互いに関連する複数の表の `UNION ALL` として定義されたビューからの行の削除があります。

```
Table T1 is a parent of table T3; delete rule as noted below.
Table T2 is a parent of table T3; delete rule CASCADE.
```

```
CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2
DELETE FROM V1
```

表 T1 が `RESTRICT` の削除規則を伴う表 T3 の親である場合に、T3 に T1 の親キーの子行があると、制約違反 (SQLSTATE 23001) になります。

表 T1 が表 T3 の親で、T3 の削除規則が `NO ACTION` である場合、T1 からの削除に対して `NO ACTION` 削除規則が適用される前に T2 から行を削除すると、削除規則 `CASCADE` によって、その子行が削除される場合があります。T2 からの削除で、T3 の T1 の親キーの子行すべてが削除されたわけではない場合は、制約違反 (SQLSTATE 23504) になります。

戻される SQLSTATE は、削除規則または更新規則が `RESTRICT` か `NO ACTION` によって異なります。

- 複数パーティションのデータベース・パーティション・グループに対して定義された表スペースの複数の表の場合、分散キーを選択する際に表コロケーションについて考慮する必要があります。以下に考慮事項をリストします。

CREATE TABLE

- コロケーションのためには、各表は同じデータベース・パーティション・グループにある必要があります。表スペースは異なっても構いませんが、同じデータベース・パーティション・グループに定義されている必要があります。
- コロケーションのためには、各表の分散キーの列の数は同じである必要があります。対応するキー列はデータベース・パーティション互換である必要があります。
- 分散キーの選択も、結合のパフォーマンスに影響します。表を他の表と頻繁に結合する場合は、結合する列を両方の表の分散キーにすることを考慮してください。
- 代替のソース (別の表やファイル) からのデータを使用して大きな結果セットを作成する必要があり、表のリカバリーが不要な場合は、`NOT LOGGED INITIALLY` オプションが有用です。このオプションを使用すると、データのロギングにかかるオーバーヘッドが節減されます。このオプションを指定する場合、以下の考慮事項が適用されます。
 - 作業単位がコミットされると、その作業単位の過程で表に対して行われた変更はすべてディスクにフラッシュされます。
 - ロールフォワード・ユーティリティを実行した際に、データベース中の表がロード・ユーティリティによって移植されたか、または `NOT LOGGED INITIALLY` オプションを使用して作成されたことを示すログ記録が見つかり、表は使用不能としてマークされます。その後 `DROP TABLE` ログが見つかり、表はロールフォワード・ユーティリティによってドロップされます。除去しない場合、データベースの回復後に表にアクセスを試みると、エラーが出されます (`SQLSTATE 55019`)。許される唯一の操作は表のドロップです。
 - データベースまたは表スペースのバックアップの一環として、このような表をバックアップすると、表のリカバリーが可能になります。
- `CURRENT REFRESH AGE` を `ANY` にセットし、`CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION` をシステム保守済みマテリアライズ照会表に含まれるようにセットすると、照会の処理を最適化するとき、`ENABLE QUERY OPTIMIZATION` を指定して定義された `REFRESH DEFERRED` システム保守済みマテリアライズ照会表を使用できます。`CURRENT REFRESH AGE` を `ANY` にセットし、`CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION` をユーザー保守済みマテリアライズ照会表に含まれるようにセットすると、照会の処理を最適化するとき、`ENABLE QUERY OPTIMIZATION` を指定して定義された `REFRESH DEFERRED` ユーザー保守済みマテリアライズ照会表を使用できます。`ENABLE QUERY OPTIMIZATION` を指定して定義された `REFRESH IMMEDIATE` マテリアライズ照会表は、必ず最適化の対象として考慮に入れられます。この最適化で `REFRESH DEFERRED` または `REFRESH IMMEDIATE` マテリアライズ照会表を使用できるようにするには、すでに説明された規則以外の特定の規則に全選択を適合させる必要があります。全選択の条件は、以下の規則に従っていなければなりません。
 - `GROUP BY` 節を指定した副選択、または 1 つの表参照を指定した副選択になっている。
 - 選択リストのどこにも `DISTINCT` が含まれていない。
 - 特殊レジスターが含まれていない。
 - グローバル変数が含まれていない。

- deterministic 関数でない関数が含まれていない。
- マテリアライズ照会表の作成時に指定した照会が上記の規則に適合しなければ、警告が戻されます (SQLSTATE 01633)。
- マテリアライズ照会表が REFRESH IMMEDIATE で定義されている場合、またはステージング表が PROPAGATE IMMEDIATE で定義されている場合は、基礎表の挿入、更新、または削除操作を行うことになる変更をしようとする、エラーになる可能性があります。エラーが発生すると、基礎表の挿入、更新、または削除の操作は失敗します。
- ロード操作中や SET INTEGRITY ステートメントの実行中など、制約が大量にチェックされている場合、マテリアライズ照会表やステージング表は使用できません。
- REFRESH IMMEDIATE で定義されたマテリアライズ照会表、または REFRESH DEFERRED に関連ステージング表を指定して定義したマテリアライズ照会表によって参照されている表に対しては、次のような操作を実行することができません。
 - IMPORT REPLACE は使用できません。
 - ALTER TABLE NOT LOGGED INITIALLY WITH EMPTY TABLE は実行できません。
- フェデレーテッド・システムでは、リレーショナル・データ・ソースまたはローカル表のニックネームは、マテリアライズ照会表を作成する基礎表として使用できます。非リレーショナル・データ・ソースのニックネームは、サポートされていません。ニックネームが基礎表のうちの 1 つである場合は、REFRESH DEFERRED オプションを使用する必要があります。パーティション・データベース環境では、ニックネームを参照するシステム保守のマテリアライズ照会表はサポートされていません。
- **透過性 DDL:** フェデレーテッド・システムでは、DB2 SQL を使用してリモート基本表を作成、変更、またはドロップすることができます。この機能は、**透過性 DDL** として知られています。リモート基本表がデータ・ソース上に作成される前に、フェデレーテッド・サーバーはそのデータ・ソースへアクセスするように構成されなければなりません。この構成には、データ・ソースのラッパーの作成、リモート基本表を置くサーバーのサーバー定義の指定、フェデレーテッド・サーバーとデータ・ソース間のユーザー・マッピングの作成が含まれます。

CREATE TABLE ステートメントに含むことができるものに関して、透過性 DDL には以下のようないくつかの制約があります。

- 列および主キーのみがリモート基本表に作成可能です。
- 透過性 DDL によってサポートされる節には、具体的には以下のものがあります。
 - *element-list* 節の中の *column-definition* と *unique-constraint*
 - *column-options* 節の中の NOT NULL と PRIMARY KEY
 - OPTIONS
- リモート・データ・ソースは次のものをサポートする必要があります。
 - DB2 列データ・タイプがマップされるリモート列データ・タイプ
 - CREATE TABLE ステートメントにおける主キー・オプション

CREATE TABLE

データ・ソースの、サポートしていない要求への応答方法によって、エラーが返されるか、または要求が無視される可能性があります。

リモート基本表が透過性 DDL を使用して作成された場合、そのリモート基本表に対してニックネームが自動的に作成されます。

- 親表または従属表が表階層の一部を成すように参照制約を定義することができます。その場合、参照制約は次のような効果を生じます。
 1. INSERT、UPDATE、および DELETE ステートメントの効果は次のとおりです。
 - PT が親表で DT が従属表である参照制約が存在する場合、非 NULL の外部キーをもつ DT の行 (またはその副表のいずれか) ごとに、それに合致する親キーをもつ行が PT (またはその副表のいずれか) 内に必ず存在することが制約によって確実にになります。アクションの開始の仕方に関係なく、この規則は、PT または DT の行に影響を与えるすべてのアクションに対して適用されます。
 2. DROP TABLE ステートメントの効果は次のとおりです。
 - ドロップ済みの表が親表または従属表である参照制約では、制約はドロップされます。
 - ドロップ済みの表のスーパー表が親表である参照制約では、そのドロップ済みの表の行は、スーパー表からの削除を考慮されます。参照制約が検査されて、削除行ごとに削除規則が呼び出されます。
 - ドロップ済みの表のスーパー表が従属表である参照制約の場合、制約は検査されません。従属表から行を削除しても、参照制約違反にはなりません。
- **特権:** 表が作成されると、その表の定義者には CONTROL 特権が付与されます。副表が作成されると、各ユーザーまたはグループが持っているそのすぐ上のスーパー表に対する SELECT 特権が副表に対しても自動的に付与され、その場合は表定義者から特権が付与されたことになります。
- **行サイズ:** 表の行で許可される最大バイト数は、表が作成される表スペースのページ・サイズによって決まります (*tablespace-name*)。次のリストでは、各表スペースのページ・サイズに関連した行サイズの制限と列数の制限を示します。

表 20. 各表スペースのページ・サイズの列数および行サイズの制限

ページ・サイズ	行サイズの制限	列数の制限
4K	4 005	500
8K	8 101	1 012
16K	16 293	1 012
32K	32 677	1 012

表の実際の列数については、次の公式によってさらに制限されます。

$$\text{Total Columns} * 8 + \text{Number of LOB Columns} * 12 \leq \text{Row Size Limit for Page Size}$$

- **バイト・カウント:** 次の表は、列のバイト・カウントを、列のデータ・タイプ別に示したものです。これは、行サイズの計算に使用します。バイト・カウントは VALUE COMPRESSION がアクティブかどうかによって異なります。VALUE

COMPRESSION がアクティブでない場合には、バイト・カウントは列が NULL 可能であるかどうかによっても異なります。

表が構造化タイプに基づいている場合には、副表が定義されているか否かにかかわらず、副表の行を識別するために 4 バイトのオーバーヘッドが確保されます。さらに、追加される副表列は、NULL 不可と定義されたとしたら、バイト・カウント用に NULL 可能なものと見なされる必要があります。

表 21. データ・タイプごとの列のバイト・カウント

データ・タイプ	VALUE COMPRESSION	VALUE COMPRESSION がアクティブでない	
	がアクティブ ¹	列が NULL 可能	列が NULL 可能でない
SMALLINT	4	3	2
INTEGER	6	5	4
BIGINT	10	9	8
REAL	6	5	4
DOUBLE	10	9	8
DECIMAL	(p/2)+3 の整数部分 (p は精度)	(p/2)+2 の整数部分 (p は精度)	(p/2)+1 の整数部分 (p は精度)
DECFLOAT(16)	10	9	8
DECFLOAT(34)	18	17	16
CHAR(n)	n+2	n+1	n
VARCHAR(n)	n+2	n+5 (表内)	n+4 (表内)
LONG VARCHAR ²	22	25	24
GRAPHIC(n)	n*2+2	n*2+1	n*2
VARGRAPHIC(n)	n*2+2	n*2+5 (表内)	n*2+4 (表内)
LONG VARGRAPHIC ²	22	25	24
DATE	6	5	4
TIME	5	4	3
TIMESTAMP	12	11	10
XML (INLINE LENGTH が指定されない場合)	82	85	84
XML (INLINE LENGTH が指定される場合)	INLINE LENGTH +2	INLINE LENGTH +4	INLINE LENGTH +3
LOB ³ の最大長 1024	70	73	72
LOB の最大長 8192	94	97	96
LOB の最大長 65 536	118	121	120
LOB の最大長 524 000	142	145	144
LOB の最大長 4 190 000	166	169	168
LOB の最大長 134 000 000	198	201	200
LOB の最大長 536 000 000	222	225	224
LOB の最大長 1 070 000 000	254	257	256
LOB の最大長 1 470 000 000	278	281	280

CREATE TABLE

表 21. データ・タイプごとの列のバイト・カウント (続き)

データ・タイプ	VALUE COMPRESSION	VALUE COMPRESSION がアクティブでない	
	がアクティブ ¹	列が NULL 可能	列が NULL 可能でない
LOB の最大長	2 147 483	317	316
	647		

¹ その行の VALUE COMPRESSION がアクティブの場合には、行ごとに使用される 2 バイトの追加ストレージがあります。

² データ・タイプ LONG VARCHAR と LONG VARCHARIC は、サポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

³ 各 LOB 値は、その基底レコードに、実際の値の位置へのポインターとなる LOB 記述子を持っています。その記述子のサイズは、列に定義されている最大長によって異なります。

特殊タイプ の場合、バイト・カウントは特殊タイプのソース・タイプの長さに相当します。参照タイプ の場合には、バイト・カウントは、参照タイプの基礎となる組み込みデータ・タイプの長さに相当します。構造化タイプ では、バイト・カウントは `INLINE LENGTH + 4` に相当します。INLINE LENGTH は、`column-options` 節の列に指定された (または暗黙で計算された) 値です。

以下のサンプル表の行サイズは、VALUE COMPRESSION が指定されていないと想定しています。

```
DEPARTMENT 63 (0 + 3 + 33 + 7 + 3 + 17)
ORG         57 (0 + 3 + 19 + 2 + 15 + 18)
```

VALUE COMPRESSION が指定された場合には、行サイズが次のように変更されます。

```
DEPARTMENT 69 (2 + 5 + 31 + 8 + 5 + 18)
ORG         53 (2 + 4 + 16 + 4 + 12 + 15)
```

- ストレージ・バイト・カウント:** 次の表は、列のストレージ・バイト・カウントを、データ値のデータ・タイプ別に示したものです。バイト・カウントは VALUE COMPRESSION がアクティブかどうかによって異なります。VALUE COMPRESSION がアクティブでない場合には、バイト・カウントは列が NULL 可能であるかどうかによっても異なります。表の値は、値を保管するために使用されるストレージ量 (バイト単位) を表します。

表 22. 行形式、データ・タイプ、およびデータ値ベースのストレージ・バイト・カウント

データ値 →	NULL	NULL	長さゼロ	システム・デフォルト ²	他のすべてのデータ値	他のすべてのデータ値	他のすべてのデータ値
VALUE COMPRESSION →	アクティブでない	アクティブ ¹	アクティブ ¹	アクティブ ¹	アクティブでない	アクティブでない	アクティブ ¹
列の NULL 可能性 →	NULL 可能	NULL 可能	n/a	n/a	NULL 可能	NULL 可能でない	n/a
データ・タイプ							
SMALLINT	3	3	-	3	3	2	4
INTEGER	5	3	-	3	5	4	6
BIGINT	9	3	-	3	9	8	10
REAL	5	3	-	3	5	4	6
DOUBLE	9	3	-	3	9	8	10
DECIMAL	(p/2)+2 の整数部分 (p は精度)	3	-	3	(p/2)+2 の整数部分 (p は精度)	(p/2)+1 の整数部分 (p は精度)	(p/2)+3 の整数部分 (p は精度)

表 22. 行形式、データ・タイプ、およびデータ値ベースのストレージ・バイト・カウント (続き)

データ値 →	NULL	NULL	長さゼロ	システム・ デフォルト ²	他のすべての データ値	他のすべての データ値	他のすべての データ値
VALUE COMPRESSION →	アクティブでな い	アクティブ ¹	アクティブ ¹	アクティブ ¹	アクティブでな い	アクティブでな い	アクティブ ¹
列の NULL 可能 性 →	NULL 可能	NULL 可能	n/a	n/a	NULL 可能	NULL 可能でな い	n/a
データ・タイプ							
DECFLOAT(16)	9	3	-	3	9	8	10
DECFLOAT(34)	17	3	-	3	17	16	18
CHAR(<i>n</i>)	<i>n</i> +1	3	-	3	<i>n</i> +1	<i>n</i>	<i>n</i> +2
VARCHAR(<i>n</i>)	5	3	2	2	<i>N</i> +5 (<i>N</i> はデー タ内のバイト数)	<i>N</i> +4 (<i>N</i> はデー タ内のバイト数)	<i>N</i> +2 (<i>N</i> はデー タ内のバイト数)
LONG VARCHAR ³	5	3	2	2	25	24	22
GRAPHIC(<i>n</i>)	<i>n</i> *2+1	3	-	3	<i>n</i> *2+1	<i>n</i> *2	<i>n</i> *2+2
VARGRAPHIC(<i>n</i>)	5	3	2	2	<i>N</i> *2+5 (<i>N</i> はデー タ内のバイト数)	<i>N</i> *2+4 (<i>N</i> はデー タ内のバイト数)	<i>N</i> *2+2 (<i>N</i> はデー タ内のバイト数)
LONG VARGRAPHIC ³	5	3	2	2	25	24	22
DATE	5	3	-	-	5	4	6
TIME	4	3	-	-	4	3	5
TIMESTAMP	11	3	-	-	11	10	12
LOB ² の最大長 1024	5	3	2	2	(60 から 68)+5	(60 から 68)+4	(60 から 68)+2
LOB の最大長 8192	5	3	2	2	(60 から 92)+5	(60 から 92)+4	(60 から 92)+2
LOB の最大長 65 536	5	3	2	2	(60 から 116)+5	(60 から 116)+4	(60 から 116)+2
LOB の最大長 524 000	5	3	2	2	(60 から 140)+5	(60 から 140)+4	(60 から 140)+2
LOB の最大長 4 190 000	5	3	2	2	(60 から 164)+5	(60 から 164)+4	(60 から 164)+2
LOB の最大長 134 000 000	5	3	2	2	(60 から 196)+5	(60 から 196)+4	(60 から 196)+2
LOB の最大長 536 000 000	5	3	2	2	(60 から 220)+5	(60 から 220)+4	(60 から 220)+2
LOB の最大長 1 070 000 000	5	3	2	2	(60 から 252)+5	(60 から 252)+4	(60 から 252)+2
LOB の最大長 1 470 000 000	5	3	2	2	(60 から 276)+5	(60 から 276)+4	(60 から 276)+2
LOB の最大長 2 147 483 647	5	3	2	2	(60 から 312)+5	(60 から 312)+4	(60 から 312)+2
XML	5	3	-	-	85	84	82

¹ その行の VALUE COMPRESSION がアクティブの場合には、行ごとに使用される 2 バイトの追加ストレージがあります。

² 列に対して COMPRESS SYSTEM DEFAULT が指定される場合。

³ データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、サポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

- **ディメンション列:** ディメンション列の各特殊値は、表の別々のブロックに割り当てられるので、“INTEGER(ORDER_DATE)/100”などの式でのクラスタリングはお勧めできません。この場合、表に生成された列が定義可能で、その後この生成された列は ORGANIZE BY DIMENSIONS 節で使用されるかもしれません。式が表の列に関連して単調な場合には、この列の範囲述部を満たすためにディメン

CREATE TABLE

ション索引が DB2 によって使用される可能性があります。例えば、式が単に *column-name + some-positive-constant* の場合には、これは単調な増加です。ユーザ一定義関数、特定の組み込み機能、および 1 つの式で複数の列を使用すると、単調化やその検出を防げます。

非単調な式を持つ、または単調化を識別できない、生成された列に関するディメンションを作成できますが、スライスの範囲照会やこうしたディメンションのセル境界はサポートされません。同等性および IN 述部は、スライスまたはセルによって処理できます。

生成された関数 *fn* に関して以下の事柄が真の場合には、生成された列は単調です。

- 単調な増加。

値 *x1* および *x2* のペアのすべての可能性において、 $x2 > x1$ ならば $fn(x2) > fn(x1)$ となります。以下に例を示します。

```
SALARY - 10000
```

- 単調な減少。

値 *x1* および *x2* のペアのすべての可能性において、 $x2 > x1$ ならば $fn(x2) < fn(x1)$ となります。以下に例を示します。

```
-SALARY
```

- 単調な非減少。

値 *x1* および *x2* のペアのすべての可能性において、 $x2 > x1$ ならば $fn(x2) \geq fn(x1)$ となります。以下に例を示します。

```
SALARY/1000
```

- 単調な非増加。

値 *x1* および *x2* のペアのすべての可能性において、 $x2 > x1$ ならば $fn(x2) \leq fn(x1)$ となります。以下に例を示します。

```
-SALARY/1000
```

式 "PRICE*DISCOUNT" は単調ではありません。表の複数の列が関係するからです。

- **レンジ・クラスター表:** キー・シーケンスによる表の編成は、特定のタイプの表に対して効果があります。表は、有効な値の範囲上で高密度にクラスター化された整数キーを持っている必要があります。この整数キーの列は、NULL 可能であってはならず、キーは論理的に表の主キーでなければなりません。レンジ・クラスター表では、表の編成上、キー値によって指定された行や、キー値の範囲で指定された一定範囲の行に対する直接アクセスを提供する、別個のユニーク索引オブジェクトを必要としません。定義されたキー・シーケンスの範囲内にある行の完全セットに対するすべてのスペースの割り振りは、表の作成時に行われ、レンジ・クラスター表を定義する際に考慮される必要があります。最初の時点で行に削除のマークが付いていたとしても、ストレージ・スペースを他の用途に使用することはできません。キー・シーケンスの範囲全体が、長期間にわたってデータのみを追加するためのものである場合、この表編成は適切な選択ではありません。

- 表に設定できるセキュリティー・ポリシーは、最大で 1 個だけです。
- DB2 は、保護対象表に定義されている参照整合性制約を実施します。この場合の制約違反は、デバッグするのが難しい場合があります。該当するセキュリティー・ラベルや免除証明書がない場合、DB2 では、違反を引き起こした行を確認できないからです。
- 表の列の順序を定義するときは、更新の際のログ対象データ量が最小になるように、頻繁に更新される列は定義の末尾に配置するようにしてください。ROW CHANGE TIMESTAMP 列はこれに該当します。ROW CHANGE TIMESTAMP 列は、行の更新ごとに必ず更新されるようになっています。
- **セキュリティーとレプリケーション:** レプリケーションによって、保護対象表のデータ行がデータベースの外部に複製されることがあります。DB2 では、データベースの外部にあるデータを保護できないので、保護対象表のレプリケーションをセットアップするときには、慎重な作業が必要です。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - CONSTRAINT キーワードは、参照節を定義する *column-definition* から省略できます。
 - *constraint-name* (制約名) を FOREIGN KEY に続けて (CONSTRAINT キーワードなし) 指定することができます。
 - SUMMARY は CREATE の後に任意に指定できます。
 - WITH NO DATA の代わりに DEFINITION ONLY を指定できます。
 - DISTRIBUTE BY 節の代わりに PARTITIONING KEY 節を指定できます。
 - DISTRIBUTE BY REPLICATION の代わりに REPLICATED を指定できます。
 - 以前のバージョンの DB2 との互換性と整合性:
 - *identity-options* 節では、コンマを使って複数のオプションを分離することができます。
 - DB2 for z/OS との互換性:
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - IN database-name.tablespace-name
 - IN DATABASE database-name
 - FOR MIXED DATA
 - FOR SBCS DATA
 - PARTITION の代わりに PART を指定できます。
 - PARTITION *partition-name* の代わりに PARTITION *partition-number* を指定できます。 *partition-number* では、CREATE TABLE ステートメントすでに指定されているパーティションを指定することはできません。 *partition-number* を指定しない場合は、データベース・マネージャーによって固有のパーティション番号が生成されます。
 - ENDING AT の代わりに VALUES を指定できます。
 - 以下の構文もサポートされています。
 - NOMINVALUE、NOMAXVALUE、NOCYCLE、NOCACHE、および NOORDER

CREATE TABLE

例

例 1: DEPARTX 表スペースに表 TDEPT を作成します。DEPTNO、DEPTNAME、MGRNO、および ADMRDEPT は列の名前です。CHAR は、列が文字データを含むことを意味しています。NOT NULL は、列に NULL 値を含めることができないことを示します。VARCHAR は、列のデータが可変長文字データであることを意味します。主キーは、列 DEPTNO で構成されます。

```
CREATE TABLE TDEPT
(DEPTNO CHAR(3) NOT NULL,
DEPTNAME VARCHAR(36) NOT NULL,
MGRNO CHAR(6),
ADMRDEPT CHAR(3) NOT NULL,
PRIMARY KEY(DEPTNO))
IN DEPARTX
```

例 2: SCHED 表スペースに表 PROJ を作成します。PROJNO、PROJNAME、DEPTNO、RESPEMP、PRSTAFF、PRSTDATE、PRENDATE、および MAJPROJ は列の名前です。CHAR は、列が文字データを含むことを意味しています。DECIMAL は、列のデータがパック 10 進数データであることを意味します。5,2 の 5 は 10 進数の桁数、2 は小数点以下の桁数を示します。NOT NULL は、列に NULL 値を含めることができないことを示します。VARCHAR は、列のデータが可変長文字データであることを意味します。DATE は、列のデータが 3 つの部分からなる形式 (年、月、日) の日付情報であることを示しています。

```
CREATE TABLE PROJ
(PROJNO CHAR(6) NOT NULL,
PROJNAME VARCHAR(24) NOT NULL,
DEPTNO CHAR(3) NOT NULL,
RESPEMP CHAR(6) NOT NULL,
PRSTAFF DECIMAL(5,2) ,
PRSTDATE DATE ,
PRENDATE DATE ,
MAJPROJ CHAR(6) NOT NULL)
IN SCHED
```

例 3: 不明の給与はすべて 0 と見なされる EMPLOYEE_SALARY という名前の表を作成します。表スペースが指定されていないので、*IN tablespace-name1* 節について記述された規則に基づいてシステムが選択した表スペースに、表が作成されます。

```
CREATE TABLE EMPLOYEE_SALARY
(DEPTNO CHAR(3) NOT NULL,
DEPTNAME VARCHAR(36) NOT NULL,
EMPNO CHAR(6) NOT NULL,
SALARY DECIMAL(9,2) NOT NULL WITH DEFAULT)
```

例 4: 給与 (SALARY) と距離 (MILES) の合計用の特殊タイプを作成し、デフォルト表スペースに作成される表の列として使用します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが JOHNDOE で、CURRENT PATH がデフォルト値であると想定します ("SYSIBM","SYSFUN","JOHNDOE")。

SALARY の値の指定がない場合には、それを 0 に設定します。また、LIVING_DIST の値の指定がない場合には、それを 1 マイルに設定します。

```
CREATE TYPE JOHNDOE.T_SALARY AS INTEGER WITH COMPARISONS
CREATE TYPE JOHNDOE.MILES AS FLOAT WITH COMPARISONS
```

```
CREATE TABLE EMPLOYEE
  (ID          INTEGER NOT NULL,
   NAME        CHAR (30),
   SALARY      T_SALARY NOT NULL WITH DEFAULT,
   LIVING_DIST MILES   DEFAULT MILES(1) )
```

例 5: 画像 (IMAGE) と音声 (AUDIO) 用の特殊タイプを作成し、表の列として使用します。表スペースが指定されていないので、IN *tablespace-name1* 節について記述された規則に基づいてシステムが選択した表スペースに、表が作成されます。CURRENT PATH はデフォルト値であると想定します。

```
CREATE TYPE IMAGE AS BLOB (10M)
```

```
CREATE TYPE AUDIO AS BLOB (1G)
```

```
CREATE TABLE PERSON
  (SSN        INTEGER NOT NULL,
   NAME       CHAR (30),
   VOICE      AUDIO,
   PHOTO      IMAGE)
```

例 6: HUMRES 表スペースに表 EMPLOYEE を作成します。表には、次のような制約を定義します。

- 部門番号 (DEPT) の値は、10 から 100 の範囲でなければならない。
- 従業員のジョブ (JOB) は、'Sales'、'Mgr'、または 'Clerk' のいずれかでなければならない。
- 1986 年以前からの従業員の給与 (SALARY) はすべて \$40,500 を超えていなければならない。

注: チェック制約に含まれる列が NULL 可能である場合、それらも NULL になる可能性があります。

```
CREATE TABLE EMPLOYEE
  (ID          SMALLINT NOT NULL,
   NAME        VARCHAR(9),
   DEPT        SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
   JOB         CHAR(5) CHECK (JOB IN ('Sales','Mgr','Clerk')),
   HIREDATE    DATE,
   SALARY      DECIMAL(7,2),
   COMM        DECIMAL(7,2),
   PRIMARY KEY (ID),
   CONSTRAINT YEARSAL CHECK (YEAR(HIREDATE) > 1986
    OR SALARY > 40500)
 )
 IN HUMRES
```

例 7: PAYROLL 表スペースに全体が含まれる表を作成します。

```
CREATE TABLE EMPLOYEE .....
 IN PAYROLL
```

例 8: データ部分が ACCOUNTING にあり、索引部分が ACCOUNT_IDX にある表を作成します。

```
CREATE TABLE SALARY.....
 IN ACCOUNTING INDEX IN ACCOUNT_IDX
```

例 9: 表を作成して、SQL の変更内容をデフォルトのフォーマットでログ記録します。

```
CREATE TABLE SALARY1 .....
```

CREATE TABLE

または

```
CREATE TABLE SALARY1 .....  
DATA CAPTURE NONE
```

例 10: 表を作成して、SQL の変更内容を拡張フォーマットでログ記録します。

```
CREATE TABLE SALARY2 .....  
DATA CAPTURE CHANGES
```

例 11: SCHED 表スペースに表 EMP_ACT を作成します。EMPNO、PROJNO、ACTNO、EMPTIME、EMSTDATE、および EMENDATE は列の名前です。表には、次のような制約を定義します。

- すべての行の列セット、EMPNO、PROJNO、および ACTNO の値は、ユニークでなければならない。
- PROJNO の値は、PROJECT 表の PROJNO 列の既存の値と一致していなければならず、プロジェクトが削除される場合、そのプロジェクトを参照する EMP_ACT の行もすべて削除される。

```
CREATE TABLE EMP_ACT  
(EMPNO      CHAR(6) NOT NULL,  
 PROJNO     CHAR(6) NOT NULL,  
 ACTNO      SMALLINT NOT NULL,  
 EMPTIME    DECIMAL(5,2),  
 EMSTDATE   DATE,  
 EMENDATE   DATE,  
 CONSTRAINT EMP_ACT_UNIQ UNIQUE (EMPNO,PROJNO,ACTNO),  
 CONSTRAINT FK_ACT_PROJ FOREIGN KEY (PROJNO)  
                    REFERENCES PROJECT (PROJNO) ON DELETE CASCADE  
)  
IN SCHED
```

ユニーク制約を課すために、EMP_ACT_UNIQ という名前のユニーク索引が同じスキーマ内に自動的に作成されます。

例 12: アイス・ホッケーの栄誉の殿堂に入る、有名なゴールについての情報を保持する表を作成します。この表には、ゴールをきめたホッケー選手の名前、ゴールをきめられたゴールキーパーの名前、日付と場所、ゴールについての説明文などの情報がリストされます。説明列は NULL 可能です。

```
CREATE TABLE HOCKEY_GOALS  
( BY_PLAYER   VARCHAR(30) NOT NULL,  
   BY_TEAM    VARCHAR(30) NOT NULL,  
   AGAINST_PLAYER VARCHAR(30) NOT NULL,  
   AGAINST_TEAM VARCHAR(30) NOT NULL,  
   DATE_OF_GOAL DATE NOT NULL,  
   DESCRIPTION CLOB(5000) )
```

例 13: EMPLOYEE 表に例外表が必要であるとします。これは、以下のステートメントを使用して作成できます。

```
CREATE TABLE EXCEPTION_EMPLOYEE AS  
(SELECT EMPLOYEE.*,  
   CURRENT_TIMESTAMP AS TIMESTAMP,  
   CAST (' AS CLOB(32K)) AS MSG  
FROM EMPLOYEE  
) WITH NO DATA
```

例 14: 次に示されている属性を持つ以下のような表スペースがあるとします。

TBSPACE	PAGESIZE	USER	USERAUTH
DEPT4K	4096	BOBBY	Y
PUBLIC4K	4096	PUBLIC	Y
DEPT8K	8192	BOBBY	Y
DEPT8K	8192	RICK	Y
PUBLIC8K	8192	PUBLIC	Y

- RICK が以下のような表を作成した場合、バイト・カウントは 4005 未満なので、その表は表スペース PUBLIC4K に入れます。しかし BOBBY が同じ表を作成した場合、以下のような明示的な権限付与があって BOBBY は USE 特権を保有しているので、その表は表スペース DEPT4K に入れます。

```
CREATE TABLE DOCUMENTS
(SUMMARY VARCHAR(1000),
REPORT VARCHAR(2000))
```

- BOBBY が以下のような表を作成した場合、バイト・カウントは 4005 を超えるので、その表は表スペース DEPT8K に入れます。また、明示的な権限付与によって BOBBY は USE 特権を保有します。しかし DUNCAN が同じ表を作成すると、それは表スペース PUBLIC8K に入れます。DUNCAN には該当する特権がないからです。

```
CREATE TABLE CURRICULUM
(SUMMARY VARCHAR(1000),
REPORT VARCHAR(2000),
EXERCISES VARCHAR(1500))
```

例 15: 構造化タイプ EMP を使って定義された LEAD 列をもつ表を作成します。LEAD 列に 300 バイトの INLINE LENGTH を指定します。これは、300 バイト以内に収まらない LEAD のインスタンスをすべて、その表以外に保管すること (LOB 値の処理方法と同様に、基本表の行とは別個に) を指示します。

```
CREATE TABLE PROJECTS (PID INTEGER,
LEAD EMP INLINE LENGTH 300,
STARTDATE DATE,
...)
```

例 16: DEPTNO、DEPTNAME、MGRNO、ADMRDEPT、および LOCATION という名前の 5 つの列を持つ表 DEPT を作成します。DB2 によって常に値が生成されるよう、列 DEPT を ID 列として定義することにします。DEPT 列の値は、500 から始まり、1 ずつ増分する必要があります。

```
CREATE TABLE DEPT
(DEPTNO SMALLINT NOT NULL
GENERATED ALWAYS AS IDENTITY
(START WITH 500, INCREMENT BY 1),
DEPTNAME VARCHAR(36) NOT NULL,
MGRNO CHAR(6),
ADMRDEPT SMALLINT NOT NULL,
LOCATION CHAR(30))
```

例 17: YEAR 列で配分され、REGION および YEAR 列にディメンションを持つ SALES 表を作成します。データは、YEAR 列のハッシュ値に従って、データベース・パーティション内に配分されています。各データベース・パーティションで、それらのデータベース・パーティション上の REGION および YEAR 列の値のユニークな組み合わせを基にして、データはエクステントに編成されます。

CREATE TABLE

```
CREATE TABLE SALES
(CUSTOMER VARCHAR(80),
 REGION CHAR(5),
 YEAR INTEGER)
DISTRIBUTE BY HASH (YEAR)
ORGANIZE BY DIMENSIONS (REGION, YEAR)
```

例 18: PURCHASEDATE 列から生成された、PURCHASEYEARMONTH 列を持つ SALES 表を作成します。式を使用して、元の PURCHASEDATE 列に対して単調な列を作成し、それによって、ディメンションとして使用するのに適切です。表は REGION 列で配分されており、各データベース・パーティションで PURCHASEYEARMONTH 列に従って、エクステントに編成されています。つまり、異なる地域は異なるデータベース・パーティションにあり、異なる購入月はこれらのデータベース・パーティション内の異なるセル (またはエクステントのセット) に属します。

```
CREATE TABLE SALES
(CUSTOMER VARCHAR(80),
 REGION CHAR(5),
 PURCHASEDATE DATE,
 PURCHASEYEARMONTH INTEGER
GENERATED ALWAYS AS (INTEGER(PURCHASEDATE)/100))
DISTRIBUTE BY HASH (REGION)
ORGANIZE BY DIMENSIONS (PURCHASEYEARMONTH)
```

例 19 CUSTOMERNUM 列から生成された、CUSTOMERNUMDIM 列を持つ CUSTOMER 表を作成します。式を使用して、元の CUSTOMERNUM 列に対して単調な列を作成し、それによって、ディメンションとして使用するのに適切です。表は CUSTOMERNUMDIM 列に従ってセルに編成され、表内のそれぞれのセルには、50 人の顧客が入っています。ユニーク索引が CUSTOMERNUM に作成された場合、カスタマー番号は、表内のエクステントの特定のセットに、50 の値のセットがあるように、クラスター化されます。

```
CREATE TABLE CUSTOMER
(CUSTOMERNUM INTEGER,
 CUSTOMERNAME VARCHAR(80),
 ADDRESS VARCHAR(200),
 CITY VARCHAR(50),
 COUNTRY VARCHAR(50),
 CODE VARCHAR(15),
 CUSTOMERNUMDIM INTEGER
GENERATED ALWAYS AS (CUSTOMERNUM/50))
ORGANIZE BY DIMENSIONS (CUSTOMERNUMDIM)
```

例 20: ORASERVER という Oracle サーバー上に、EMPLOYEE というリモート基本表を作成します。この新たに作成されたリモート基本表を参照する、EMPLOYEE というニックネームも自動的に作成されます。

```
CREATE TABLE EMPLOYEE
(EMP_NO CHAR(6) NOT NULL,
 FIRST_NAME VARCHAR(12) NOT NULL,
 MID_INT CHAR(1) NOT NULL,
 LAST_NAME VARCHAR(15) NOT NULL,
 HIRE_DATE DATE,
 JOB CHAR(8),
 SALARY DECIMAL(9,2),
 PRIMARY KEY (EMP_NO))
OPTIONS
(REMOTE_SERVER 'ORASERVER',
 REMOTE_SCHEMA 'J15USER1',
 REMOTE_TABNAME 'EMPLOYEE')
```

以下の CREATE TABLE ステートメントは、表名 (または表名と明示的リモート基本表名) を指定して、大文字小文字のいずれか希望する文字にする方法を示したものです。 employee という小文字の ID は、ID が大文字に暗黙的に変換されることを示すために使用されています。

Informix® サーバー上に EMPLOYEE (大文字) というリモート基本表を作成し、その表に EMPLOYEE (大文字) というニックネームを作成します。

```
CREATE TABLE employee
  (EMP_NO CHAR(6) NOT NULL,
  ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER')
```

REMOTE_TABNAME オプションが指定されておらず、かつ *table-name* が引用符で区切られていない場合、通常リモート・データ・ソースでは名前が小文字で保管されるとしても、リモート基本表名は大文字になります。

Informix サーバー上に employee (小文字) というリモート基本表を作成し、その表に EMPLOYEE (大文字) というニックネームを作成します。

```
CREATE TABLE employee
  (EMP_NO CHAR(6) NOT NULL,
  ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER',
  REMOTE_TABNAME 'empToyee')
```

区切り ID をサポートするリモート・データ・ソースで表を作成するときには、REMOTE_TABNAME と、大文字小文字のいずれかで表名を指定した文字ストリング定数を使用します。

Informix サーバー上に employee (小文字) というリモート基本表を作成し、その表に employee (小文字) というニックネームを作成します。

```
CREATE TABLE "employee"
  (EMP_NO CHAR(6) NOT NULL,
  ...)
OPTIONS
  (REMOTE_SERVER 'INFX_SERVER')
```

REMOTE_TABNAME オプションが指定されておらず、かつ *table-name* が引用符で区切られている場合、リモート基本表名は *table-name* と同一になります。

例 21: 生徒 ID を使用して生徒を探すのに使用できるレンジ・クラスター表を作成します。各生徒のレコードには、学校 ID、プログラム ID、生徒番号、生徒 ID、生徒のファーストネーム、生徒のラストネーム、および生徒の成績平均値 (GPA) が含まれます。

```
CREATE TABLE STUDENTS
  (SCHOOL_ID    INTEGER NOT NULL,
  PROGRAM_ID   INTEGER NOT NULL,
  STUDENT_NUM  INTEGER NOT NULL,
  STUDENT_ID   INTEGER NOT NULL,
  FIRST_NAME   CHAR(30),
  LAST_NAME    CHAR(30),
  GPA          DOUBLE)
ORGANIZE BY KEY SEQUENCE
```

CREATE TABLE

```
(STUDENT_ID  
  STARTING FROM 1  
  ENDING AT 1000000)  
DISALLOW OVERFLOW
```

各レコードのサイズは、列の合計に位置合わせとレンジ・クラスター表の行のヘッダーを加算して求めます。この場合は、4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (NULL可能列) + 1 (位置合わせ) + 10 (ヘッダー) で、98 バイトが行のサイズです。ページ・サイズが 4 KB (4096 バイト) なら、ページのオーバーヘッドを差し引いて 4038 バイトが使用可能なので、ページあたりに 41 のレコードを収められるスペースがある計算になります。つまり、100 万人の生徒のレコードを収容するためには、24 391 ページ (100 万をページあたりのレコード数 41 で除算) が必要です。表のオーバーヘッド用に 2 ページを追加するとして、表を作成する際に割り振る 4 KB ページの最終的な数は、24 393 になります。

例 22: 制約名の指定されていない、機能従属関係を持つ DEPARTMENT という表を作成します。

```
CREATE TABLE DEPARTMENT  
(DEPTNO SMALLINT NOT NULL,  
  DEPTNAME VARCHAR(36) NOT NULL,  
  MGRNO CHAR(6),  
  ADMRDEPT SMALLINT NOT NULL,  
  LOCATION CHAR(30),  
  CHECK (DEPTNAME DETERMINED BY DEPTNO) NOT ENFORCED)
```

例 23: 保護対象行をもつ表を作成します。

```
CREATE TABLE TOASTMASTERS  
(PERFORMANCE DB2SECURITYLABEL,  
  POINTS INTEGER,  
  NAME VARCHAR(50))  
SECURITY POLICY CONTRIBUTIONS
```

例 24: 保護対象列をもつ表を作成します。

```
CREATE TABLE TOASTMASTERS  
(PERFORMANCE CHAR(8),  
  POINTS INTEGER COLUMN SECURED WITH CLUBPOSITION,  
  NAME VARCHAR(50))  
SECURITY POLICY CONTRIBUTIONS
```

例 25: 保護対象の行および列をもつ表を作成します。

```
CREATE TABLE TOASTMASTERS  
(PERFORMANCE DB2SECURITYLABEL,  
  POINTS INTEGER COLUMN SECURED WITH CLUBPOSITION,  
  NAME VARCHAR(50))  
SECURITY POLICY CONTRIBUTIONS
```

例 26: パーティション表のラージ・オブジェクトは、デフォルトでデータと同じ表スペースに配置されます。このデフォルトの動作をオーバーライドするには、LONG IN 節を使用して、ラージ・オブジェクトのための表スペースを 1 つ以上指定します。ここでは、DOCUMENTS という名前の表を作成し、その表のラージ・オブジェクト・データを (各データ・パーティションのラウンドロビン方式で) 表スペース TBSP1 と TBSP2 に保管するように設定します。

```
CREATE TABLE DOCUMENTS  
(ID INTEGER,  
  CONTENTS CLOB)
```

```

LONG IN TBSP1, TBSP2
PARTITION BY RANGE (ID)
  (STARTING 1 ENDING 1000
   EVERY 100)

```

あるいは、長形式の構文を使用して、各データ・パーティションの LARGE 表スペースを明示的に指定することも可能です。下の例では、最初のデータ・パーティションの CLOB データを LARGE_TBSP3 に配置し、その他のデータ・パーティションの CLOB データをラウンドロビン方式で LARGE_TBSP1 と LARGE_TBSP2 に分散させます。

```

CREATE TABLE DOCUMENTS
  (ID INTEGER,
   CONTENTS CLOB)
LONG IN LARGE_TBSP1, LARGE_TBSP2
PARTITION BY RANGE (ID)
  (STARTING 1 ENDING 100
   IN TBSP1 LONG IN LARGE_TBSP3,
   STARTING 101 ENDING 1000
   EVERY 100)

```

例 27: 2 つのデータ・パーティションがある ACCESSNUMBERS という名前のパーティション表を作成します。行 (10, NULL) を最初のパーティションに配置し、行 (NULL, 100) を 2 番目の (最後の) データ・パーティションに配置します。

```

CREATE TABLE ACCESSNUMBERS
  (AREA INTEGER,
   EXCHANGE INTEGER)
PARTITION BY RANGE (AREA NULLS LAST, EXCHANGE NULLS FIRST)
  (STARTING (1,1) ENDING (10,100),
   STARTING (11,1) ENDING (MAXVALUE,MAXVALUE))

```

第 2 列の NULL 値は先頭にソートされるので、行 (11, NULL) は最後のデータ・パーティションの下限 (11, 1) よりも下にソートされることになります。つまり、この行を挿入しようとする、エラーが戻されます。行 (12, NULL) は、最後のデータ・パーティションの範囲内に収まります。

例 28: 1 つのデータ・パーティションとパーティション列 PERCENT がある RATIO という名前の表を作成します。

```

CREATE TABLE RATIO
  (PERCENT INTEGER)
PARTITION BY RANGE (PERCENT)
  (STARTING (MINVALUE) ENDING (MAXVALUE))

```

この表定義では、列 PERCENT に任意の整数値を挿入できます。以下の RATIO 表定義の場合は、1 から 100 の間の任意の整数値 (1 と 100 も含む) を列 PERCENT に挿入できます。

```

CREATE TABLE RATIO
  (PERCENT INTEGER)
PARTITION BY RANGE (PERCENT)
  (STARTING 0 EXCLUSIVE ENDING 100 INCLUSIVE)

```

例 29: 2 つの列がある MYDOCS という名前の表を作成します。1 つは ID 列であり、もう 1 つは XML 文書を格納する列です。

```

CREATE TABLE MYDOCS
  (ID INTEGER,
   DOC XML)
IN HLTBSPACE

```

CREATE TABLE

例 30: 4 つの列がある NOTES という名前の表を作成します。そのうちの 1 つは XML の注記を格納する列です。

```
CREATE TABLE NOTES
  (ID          INTEGER,
   DESCRIPTION VARCHAR(255),
   CREATED     TIMESTAMP,
   NOTE        XML)
```

例 31: 各従業員の電話番号と住所を含む表 EMP_INFO を作成します。従業員情報の変更を追跡するために、表に ROW CHANGE TIMESTAMP 列を組み込みます。

```
CREATE TABLE EMP_INFO
  (EMPNO          CHAR(6) NOT NULL,
   EMP_INFOCHANGE NOT NULL GENERATED ALWAYS
   FOR EACH ROW ON UPDATE
   AS ROW CHANGE TIMESTAMP,
   EMP_ADDRESS    VARCHAR(300),
   EMP_PHONENO    CHAR(4),
   PRIMARY KEY (EMPNO) )
```

CREATE TABLESPACE

CREATE TABLESPACE ステートメントは、データベースに新しい表スペースを定義し、その表スペースにコンテナを割り当て、その表スペース定義と属性をカタログに記録します。

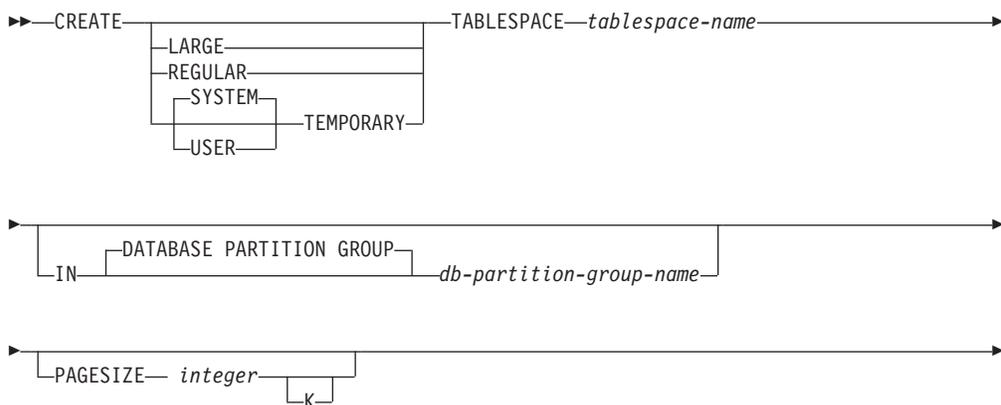
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

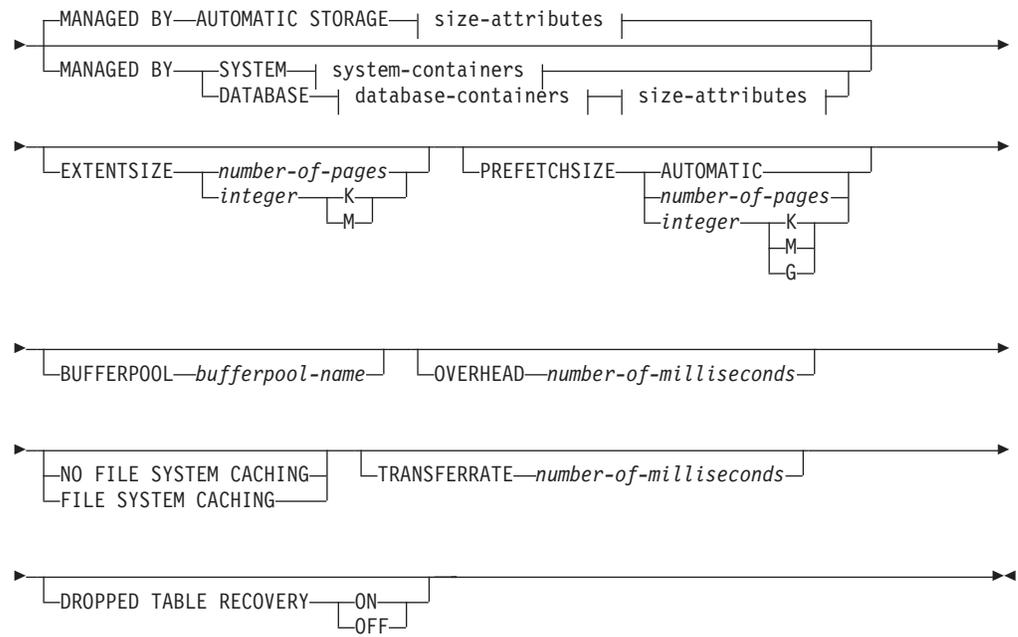
許可

このステートメントの許可 ID が持つ特権には、SYSCTRL または SYSADM 権限が含まれている必要があります。

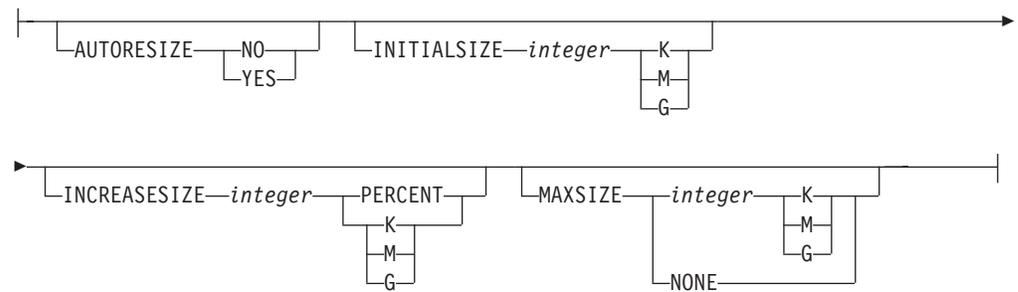
構文



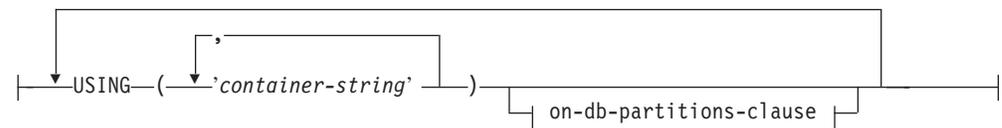
CREATE TABLESPACE



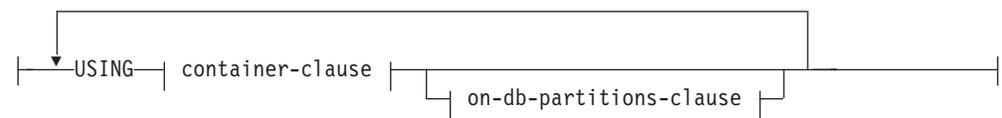
size-attributes:



system-containers:

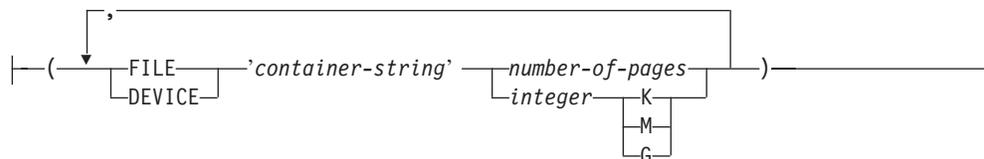


database-containers:

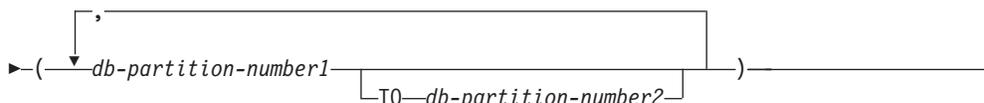


container-clause:

CREATE TABLESPACE



on-db-partitions-clause:



説明

LARGE、REGULAR、SYSTEM TEMPORARY、USER TEMPORARY

作成する表スペースのタイプを指定します。タイプを指定しない場合は、`MANAGED BY` 節によってデフォルトが決定されます。

LARGE

すべての永続データを保管します。このタイプは、データベース管理スペース (DMS) 表スペースでのみ使用できます。また、タイプを指定しない場合の、DMS 表スペースのデフォルト・タイプでもあります。LARGE 表スペースに表を配置すると、以下のようになります。

- REGULAR 表スペースに配置する表よりもサイズを大きくできます。表と表スペースの制限値の詳細については、『SQL の制限値』を参照してください。
- 表のデータ・ページあたり、255 を超える行数をサポートできるので、データ・ページのスペース使用効率が向上します。
- REGULAR 表スペースに配置した表に索引を定義する場合に比べ、索引の 1 つの行項目あたり 2 バイトが追加で必要になります。

REGULAR

すべての永続データを保管します。このタイプは、DMS 表スペースと SMS 表スペースの両方に該当します。SMS 表スペースでは、これが唯一認められているタイプであり、タイプを指定しない場合の、SMS 表スペースのデフォルト・タイプでもあります。

SYSTEM TEMPORARY

一時表 (データベース・マネージャーがソートや結合などの操作を実行するのに使用する作業域) を保管します。データベースには、常に少なくとも 1 つの SYSTEM TEMPORARY 表スペースが必要です。一時表はこの種の表スペースにのみ保管することができるからです。TEMPORARY 表スペースはデータベースの作成時に自動的に作成されます。

USER TEMPORARY

宣言済みグローバル一時表を保管します。データベースの作成時にユーザー

TEMPORARY 表スペースは存在しません。宣言済み一時表を定義できるようにするには、適切な USE 特権を設定した USER TEMPORARY 表スペースを少なくとも 1 つ作成する必要があります。

tablespace-name

表スペースの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *tablespace-name* (表スペース名) は、すでにカタログに存在している表スペースを指定するものであってはなりません (SQLSTATE 42710)。 *tablespace-name* を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

IN DATABASE PARTITION GROUP *db-partition-group-name*

表スペースのデータベース・パーティション・グループを指定します。該当のデータベース・パーティション・グループは存在していなければなりません。SYSTEM TEMPORARY 表スペースの作成の際に指定できるデータベース・パーティション・グループは、IBMTEMPGROUP だけです。DATABASE PARTITION GROUP キーワードはオプションです。

データベース・パーティション・グループを指定しないと、デフォルトのデータベース・パーティション・グループ (IBMDEFAULTGROUP) が、REGULAR、LARGE、および USER TEMPORARY 表スペースに使用されます。SYSTEM TEMPORARY 表スペースには、デフォルト・データベース・パーティション・グループ IBMTEMPGROUP が使われます。

PAGESIZE *integer* [K]

表スペースに使用するページのサイズを定義します。接尾部 K を持たない *integer* の有効値は、4 096、8 192、16 384 または 32 768 です。接尾部 K を持つ *integer* の有効値は、4、8、16、または 32 です。 *integer* と K の間には、任意の数のスペースを使用できます (スペースなしでも可)。ページ・サイズがこれらのいずれの値にも該当しない場合 (SQLSTATE 428DE)、あるいはページ・サイズが表スペースと関連付けられたバッファ・プールのページ・サイズと同じではない場合 (SQLSTATE 428CB) には、エラーが起きます。

デフォルト値は **pagesize** データベース構成パラメーターによって指定されます。これは、データベースの作成時に設定されます。

MANAGED BY AUTOMATIC STORAGE

表スペースを、自動ストレージ表スペースとして指定します。自動ストレージがデータベースに定義されていない場合は、エラーが戻されます (SQLSTATE 55060)。

自動ストレージ表スペースは、システム管理スペース (SMS) 表スペースまたはデータベース管理スペース (DMS) 表スペースのいずれかとして作成されます。DMS を選択し、表スペースのタイプを指定しない場合は、LARGE 表スペースの作成がデフォルトの動作になります。自動ストレージ表スペースでは、データベースに関連したストレージ・パスを基にして、データベース・マネージャーが表スペースに割り当てられるコンテナを判別します。

size-attributes

自動ストレージ表スペース、または自動ストレージ表スペースでない DMS 表スペースの、サイズ属性を指定します。SMS 表スペースは自動サイズ変更可能ではありません。

CREATE TABLESPACE

AUTORESIZE

DMS 表スペースまたは自動ストレージ表スペースの自動サイズ変更機能を使用可能にするかどうかを指定します。自動サイズ変更可能表スペースは、いっぱいになると、自動的にサイズが大きくなります。デフォルトは、DMS 表スペースの場合は NO、自動ストレージ表スペースの場合は YES です。

NO

DMS 表スペースまたは自動ストレージ表スペースの自動サイズ変更機能が、使用不可であることを指定します。

YES

DMS 表スペースまたは自動ストレージ表スペースの自動サイズ変更機能が、使用可能であることを指定します。

INITIALSIZE *integer* K | M | G

自動ストレージ表スペースの、データベース・パーティションあたりの初期サイズを指定します。このオプションは、自動ストレージ表スペースに対してのみ有効です。整数値の後に K (K バイトの場合)、M (M バイトの場合)、または G (G バイトの場合) を指定する必要があります。使用される実際の値は、指定されたものよりも若干小さい場合があることにご注意ください。これは、データベース・マネージャーが表スペース内の各コンテナのサイズの一貫性を維持しようとするためです。また、表スペースの自動サイズ調整を有効にしておけば、新しい表スペースに追加しなければならないメタデータの量に対して初期のサイズが十分でない場合でも、十分なスペースを確保できるまで、DB2 が INCREASESIZE ごとに表スペースを拡張し続けます。表スペースが自動サイズ変更可能であるものの INITIALSIZE 節が指定されていない場合には、データベース・マネージャーが適切な値を判別します。

INCREASESIZE *integer* PERCENT または INCREASESIZE *integer* K | M |

G 自動サイズ変更可能な表スペースがいっぱいになって、スペースの要求がなされたとき、自動的に増加させる量を、データベース・パーティションごとに指定します。整数値の後に以下のものを指定しなければなりません。

- PERCENT。スペースの要求がなされた時点の表スペース・サイズのパーセンテージとして量を指定します。PERCENT を指定する場合、整数値は 0 と 100 の間でなければなりません (SQLSTATE 42615)。
- K (K バイト)、M (M バイト)、または G (G バイト)。バイト単位で量を指定します。

使用される実際の値は、指定されたものよりも若干小さかったり大きかったりすることにご注意ください。これは、データベース・マネージャーが表スペース内の各コンテナの増大の整合性を維持しようとするためです。表スペースが自動サイズ変更可能であるものの INCREASESIZE 節が指定されていない場合には、データベース・マネージャーが適切な値を判別します。

MAXSIZE *integer* K | M | G または MAXSIZE NONE

自動サイズ変更可能な表スペースを自動的に大きくできる、最大サイズを指定します。表スペースが自動サイズ変更可能であるものの MAXSIZE 節が指定されていない場合、デフォルトは NONE です。

integer

DMS 表スペースまたは自動ストレージ表スペースを自動的に大きくで

きる、サイズ上のハード限界を、データベース・パーティションごとに指定します。整数値の後に K (K バイトの場合)、M (M バイトの場合)、または G (G バイトの場合) を指定する必要があります。使用される実際の値は、指定されたものよりも若干小さい場合があることにご注意ください。これは、データベース・マネージャーが表スペース内の各コンテナの増大の整合性を維持しようとするためです。

NONE

表スペースをファイル・システムの容量まで、または表スペースの最大サイズ (『SQL の制限値』で解説) まで増大できるようにすることを指定します。

MANAGED BY SYSTEM

表スペースを、SMS 表スペースとして指定します。表スペースのタイプを指定しない場合は、REGULAR 表スペースの作成がデフォルトの動作になります。

system-containers

SMS 表スペースに対するコンテナを指定します。

USING ('container-string',...)

SMS 表スペースに対して、表スペースに属し、表スペースのデータの保管先となる 1 つまたは複数のコンテナを指定します。コンテナ・ストリング (*container-string*) の長さは、240 バイトを超えてはなりません。

各 *container-string* は、絶対ディレクトリー名または相対ディレクトリー名にすることができます。

ディレクトリー名が絶対でない場合は、データベース・ディレクトリーからの相対ディレクトリーになります。ディレクトリー名は、データベース・ディレクトリーに物理的に関連付けられていないストレージのパス名の別名 (UNIX システムではシンボリック・リンク) にすることもできます。例えば、別個のファイル・システムのシンボリック・リンクとして、`<dbdir>/work/c1` などを指定できます。

ディレクトリー名のコンポーネントのいずれかが存在しない場合は、それがデータベース・マネージャーによって作成されます。表スペースをドロップすると、データベース・マネージャーによって作成されたすべてのコンポーネントが削除されます。 *container-string* で指定されているディレクトリーが存在する場合、そのディレクトリーにはファイルやサブディレクトリーがあってはなりません (SQLSTATE 428B2)。

container-string の形式は、オペレーティング・システムによって異なります。Windows オペレーティング・システムの場合、絶対ディレクトリー・パス名はドライブ名とコロン (:) で始まり、UNIX システムの場合、絶対パス名はスラッシュ (/) で始まります。どのプラットフォームでも、相対パス名の先頭にはオペレーティング・システム固有の文字が付きません。

リモート・リソース (LAN でリダイレクトされたドライブや NFS でマウントされたファイル・システムなど) は、現在、Network Appliance Filers、IBM iSCSI、IBM Network Attached Storage、Network Appliance iSCSI、NEC iStorage S2100、S2200、または S4100、または NEC Storage NS シリーズを Windows DB2 サーバーで使用する場合にのみサポートされます。NEC Storage NS シリーズは、無停電電源装置 (UPS) を使用した場合にのみサポートされています。(スタンバイではなく) 連続 UPS を推奨

CREATE TABLESPACE

します。 AIX 上の NFS でマウントされたファイル・システムは、`-o nointr` オプションを使用して無停電モードでマウントしなければなりません。

on-db-partitions-clause

パーティション・データベースにおいて、コンテナを作成するデータベース・パーティションを指定します。この節を指定しない場合、他のどの *on-db-partitions-clause* にも明示的に指定されていないデータベース・パーティション・グループ内のデータベース・パーティションでコンテナが作成されます。データベース・パーティション・グループ `IBMTEMPGROUP` で定義されている `SYSTEM TEMPORARY` 表スペースについては、*on-db-partitions-clause* を指定しないと、データベースに追加されたすべての新しいデータベース・パーティションでもコンテナが作成されます。

MANAGED BY DATABASE

表スペースを、DMS 表スペースとして指定します。表スペースのタイプを指定しない場合は、LARGE 表スペースの作成がデフォルトの動作になります。

database-containers

DMS 表スペースに対するコンテナを指定します。

USING

container-clause を導きます。

container-clause

DMS 表スペースに対するコンテナを指定します。

(FILE|DEVICE '*container-string*' *number-of-pages*,...)

DMS 表スペースに対して、表スペースに属し、表スペースのデータの保管先となる 1 つまたは複数のコンテナを指定します。コンテナのタイプ (FILE または DEVICE) とそのサイズ (PAGESIZE ページの数) を指定します。このサイズは整数値としても指定でき、その後 K (K バイトの場合)、M (M バイトの場合)、または G (G バイトの場合) を付けます。このように指定した場合、ページ・サイズで割ったバイト数の値よりも小さいか等しい整数で、最大の整数値は、コンテナのページ数を判別するために使用します。FILE コンテナおよび DEVICE コンテナを混合して指定することも可能です。コンテナ・ストリング (*container-string*) の長さは、254 バイトを超えてはなりません。

FILE コンテナの場合、*container-string* は、絶対ファイル名または相対ファイル名でなければなりません。絶対ファイル名以外のファイル名は、データベース・ディレクトリーからの相対パス名になります。ディレクトリー名のコンポーネントのいずれかが存在しない場合は、それがデータベース・マネージャーによって作成されます。ファイルが存在しない場合、データベース・マネージャーによってそのファイルが作成され、指定されたサイズに初期化されます。表スペースをドロップすると、データベース・マネージャーによって作成されたすべてのコンポーネントが削除されます。

注: ファイルが存在する場合は上書きされ、指定したサイズより小さい場合には拡張されます。指定したサイズよりもファイルの方が大きいても、ファイルは切り捨てられません。

DEVICE コンテナの場合、*container-string* は装置名でなければなりません。その装置はすでに存在していなければなりません。

すべてのコンテナは、すべてのデータベースで固有でなければなりません。1つのコンテナは、1つの表スペースにのみ属することができます。コンテナごとに異なるサイズにすることができますが、すべてのコンテナが同じサイズの場合にパフォーマンスは最適になります。*container-string* の正しい形式は、オペレーティング・システムによって異なります。

リモート・リソース (LAN でリダイレクトされたドライブや NFS でマウントされたファイル・システムなど) は、現在、Network Appliance Filers、IBM iSCSI、IBM Network Attached Storage、Network Appliance iSCSI、NEC iStorage S2100、S2200、または S4100、または NEC Storage NS シリーズを Windows DB2 サーバーで使用する場合にのみサポートされます。NEC Storage NS シリーズは、無停電電源装置 (UPS) を使用した場合にのみサポートされています。(スタンバイではなく) 連続 UPS を推奨します。

on-db-partitions-clause

パーティション・データベースにおいて、コンテナを作成するデータベース・パーティションを指定します。この節を指定しない場合、他のどの *on-db-partitions-clause* にも明示的に指定されていないデータベース・パーティション・グループ内のデータベース・パーティションでコンテナが作成されます。データベース・パーティション・グループ IBMTEMPGROUP で定義されている SYSTEM TEMPORARY 表スペースについては、*on-db-partitions-clause* を指定しないと、データベースに追加されたすべての新しいデータベース・パーティションでもコンテナが作成されます。

on-db-partitions-clause

パーティション・データベースにおいて、コンテナを作成するデータベース・パーティションを指定します。

ON DBPARTITIONNUMS

個々のデータベース・パーティションを指定することを示すキーワードです。DBPARTITIONNUM は DBPARTITIONNUMS の同義語です。

db-partition-number1

データベース・パーティション番号を指定します。

TO *db-partition-number2*

データベース・パーティション番号の範囲を指定します。

db-partition-number2 の値は、*db-partition-number1* の値以上でなければなりません (SQLSTATE 428A9)。コンテナは、指定する値の範囲内にある (その値も含む) 各データベース・パーティションで作成されます。指定するデータベース・パーティションは、表スペースのデータベース・パーティション・グループに含まれているものでなければなりません。

番号によって指定するデータベース・パーティションと、データベース・パーティションの範囲によって指定するすべてのデータベース・パーティションは、表スペースのデータベース・パーティション・グループに含まれてい

CREATE TABLESPACE

るものでなければなりません (SQLSTATE 42729)。データベース・パーティション番号を明示的に、または範囲として指定できるのは、このステートメントのただ 1 つの *on-db-partitions-clause* の中だけです (SQLSTATE 42613)。

EXTENTSIZE *number-of-pages*

次のコンテナーに移る前にコンテナーに書き込まれる PAGESIZE ページの数を指定します。このエクステント・サイズ値は、後に K (K バイトの場合)、または M (M バイトの場合) を付けた整数値として指定することもできます。このように指定した場合、バイト数をページ・サイズで割った値よりも小さいか等しい整数で、最大の整数値が、エクステント・サイズの値を判別するために使用します。データが保管されていくにつれて、データベース・マネージャーはコンテナー間を繰り返し循環します。

デフォルト値は **dft_extent_sz** データベース構成パラメーターによって指定されます。有効な範囲は 2 から 256 ページです。

PREFETCHSIZE

照会によって参照される前に、照会に必要なとなるデータを読み取るよう指定し、照会が入出力の実行を待たずに済むようにします。

デフォルト値は **dft_prefetch_sz** データベース構成パラメーターによって指定されます。

AUTOMATIC

表スペースのプリフェッチ・サイズが自動的に更新されるように指定します。プリフェッチ・サイズは、次の公式を使って DB2 により管理されます。

$$\begin{aligned} \text{プリフェッチ・サイズ} = \\ & (\text{コンテナーの数}) * \\ & (\text{コンテナーごとの物理ディスクの数}) * \\ & (\text{エクステント・サイズ}) \end{aligned}$$

コンテナーごとの物理ディスクの数のデフォルトは、**DB2_PARALLEL_IO** レジストリー変数によって指定されているのでない限り、1 です。

表スペース内のコンテナー数が増えるたびに (1 つ以上のコンテナーを追加またはドロップする **ALTER TABLESPACE** ステートメントの正常実行に続いて)、**DB2** はプリフェッチ・サイズを自動的に更新します。プリフェッチ・サイズはデータベースの開始時に更新されます。

number-of-pages

データのプリフェッチの実行中に、表スペースから読み取られる PAGESIZE ページの数を指定します。このプリフェッチ・サイズ値は、後に K (K バイトの場合)、M (M バイトの場合)、または G (G バイトの場合) を付けた整数値としても指定できます。このように指定した場合、バイト数をページ・サイズで割った値を超えない最大の整数が、プリフェッチ・サイズのページ値の数を判別するために使用されます。

BUFFERPOOL *bufferpool-name*

この表スペースの表に対して使用するバッファー・プールの名前を指定します。バッファー・プールは存在する必要があります (SQLSTATE 42704)。これを指定しない場合、デフォルトのバッファー・プール

(IBMDEFAULTBP) が使用されます。バッファーク・プールのページ・サイズは、表スペースに指定された (またはデフォルト指定された) ページ・サイズと一致していなければなりません (SQLSTATE 428CB)。バッファーク・プールに対して、この表スペースのデータベース・パーティション・グループを定義する必要があります (SQLSTATE 42735)。

OVERHEAD *number-of-milliseconds*

入出力コントローラーのオーバーヘッド/ディスク・シーク/待ち時間を指定します。この値は、照会の最適化の過程で入出力コストを判別するのに使用されます。*number-of-milliseconds* の値は、数値リテラル (整数、10 進数、浮動小数点数) です。この値がすべてのコンテナで同一でない場合は、表スペースに属するすべてのコンテナの平均値になります。

バージョン 9 以降で作成されたデータベースの場合、入出力コントローラーのオーバーヘッド/ディスク・シーク/待ち時間のデフォルトは 7.5 ミリ秒です。以前のバージョンの DB2 からバージョン 9 以降にマイグレーションしたデータベースの場合、デフォルトは 12.67 ミリ秒です。

FILE SYSTEM CACHING または **NO FILE SYSTEM CACHING**

入出力操作をファイル・システム・レベルでキャッシュに入れるかどうかを指定します。どちらのオプションも指定されない場合、デフォルトは次のようになります。

- **FILE SYSTEM CACHING:** JFS on AIX、Linux System z[®]、Solaris 上のすべての非 VxFS ファイル・システム、HPUX、すべてのプラットフォーム上の SMS TEMPORARY 表スペース・ファイル、すべての LOB およびラージ・データの場合。
- **NO FILE SYSTEM CACHING:** その他のすべてのプラットフォームおよびファイル・システム・タイプの場合。

FILE SYSTEM CACHING

ターゲット表スペース内のすべての入出力操作が、ファイル・システム・レベルでキャッシュに入れられるように指定します。

NO FILE SYSTEM CACHING

すべての入出力操作がファイル・システム・レベル・キャッシュをバイパスするように指定します。

TRANSFERRATE *number-of-milliseconds*

1 ページをメモリーに読み込む時間を指定します。この値は、照会の最適化の過程で入出力コストを判別するのに使用されます。*number-of-milliseconds* の値は、数値リテラル (整数、10 進数、浮動小数点数) です。この値がすべてのコンテナで同一でない場合は、表スペースに属するすべてのコンテナの平均値になります。

バージョン 9 以降で作成されたデータベースの場合、1 ページをメモリーに読み込む時間のデフォルトは 0.06 ミリ秒です。以前のバージョンの DB2 からバージョン 9 以降にマイグレーションしたデータベースの場合、デフォルトは 0.18 ミリ秒です。

DROPPED TABLE RECOVERY

指定された表スペースからドロップされた表を、ROLLFORWARD DATABASE コマンドの RECOVER DROPPED TABLE オプションによっ

CREATE TABLESPACE

てリカバリーできるようにするかどうかを指定します。この節は、REGULAR 表スペースまたは LARGE 表スペースにのみ指定できます (SQLSTATE 42613)。

ON

ドロップされた表が回復可能であることを指定します。バージョン 8 以降は、これがデフォルトです。

OFF

ドロップされた表が回復不能であることを指定します。バージョン 7 では、これがデフォルトです。

規則

- 自動ストレージがデータベースに定義されていない場合は、エラーが戻されます (SQLSTATE 55060)。
- INITIALSIZE 節を、MANAGED BY SYSTEM または MANAGED BY DATABASE 節とともに指定することはできません (SQLSTATE 42601)。
- AUTORESIZE、INCREASESIZE、または MAXSIZE 節を、MANAGED BY SYSTEM 節とともに指定することはできません (SQLSTATE 42601)。
- AUTORESIZE、INITIALSIZE、INCREASESIZE、または MAXSIZE 節は、TEMPORARY 自動ストレージ表スペースを作成するために指定することができません (SQLSTATE 42601)。
- INCREASESIZE または MAXSIZE 節は、表スペースが自動サイズ変更不可である場合には指定できません (SQLSTATE 42601)。
- AUTORESIZE は、ロー・デバイス・コンテナを使用するよう定義された DMS 表スペースに対しては、使用可能にできません (SQLSTATE 42601)。
- 表スペースの初期サイズは、5 つのエクステントを保持するのに十分な大きさでなければなりません (SQLSTATE 57011)。
- 表スペースの最大サイズは、その初期サイズよりも大きくなければなりません (SQLSTATE 560B0)。
- コンテナ操作 (ADD、EXTEND、RESIZE、DROP、または BEGIN NEW STRIPE SET) は、自動ストレージ表スペースに対しては実行できません。なぜならそのような表スペースのスペース管理は、データベース・マネージャーが制御しているからです (SQLSTATE 42858)。
- 各コンテナ定義には、53 バイトに加えて、コンテナ名を保管するのに必要なバイト数が必要です。表スペースのすべてのコンテナ名を結合した長さは、20480 バイトを超えることはできません (SQLSTATE 54034)。
- パーティション・データベースで、複数のデータベース・パーティションが同じ物理ノードに存在する場合、複数のデータベース・パーティションに同じ装置またはパスを指定することはできません (SQLSTATE 42730)。この環境では、それぞれのデータベース・パーティションごとに固有の *container-string* を指定するか、相対パス名を使用してください。

注

- 表スペースをデータベース管理表スペース (DMS) にするか、システム管理表スペース (SMS) にするかは、トレードオフの関係にある基本的な選択です (それぞれの特徴をふまえた上で、どちらが要件に適切かを検討してください)。

- データベースに複数の TEMPORARY 表スペースが存在する場合は、使用率のバランスを調整するために、複数の表スペースが「ラウンドロビン」式で使用されます。
- SMS コンテナまたは DMS コンテナの作成時に、コンテナ・ストリング構文にデータベース・パーティション式を指定できます。データベース・パーティション式は一般に、パーティション・データベース・システムで複数の論理データベース・パーティションを使用する場合に指定します。この指定により、コンテナ名がデータベース・パーティション・サーバー間で固有のものになります。この式を指定すると、データベース・パーティション番号がコンテナ名の一部になります。追加の引数を指定すれば、引数の結果もコンテナ名の一部になります。

データベース・パーティション式を示すには、引数 『 \$N』 ([ブランク]\$N) を使用します。データベース・パーティション式はコンテナ名内で自由に使用できます。また、複数のデータベース・パーティションを指定できます。データベース・パーティション式を終了するにはスペース文字を使用します。スペースの後ろにあるものは、データベース・パーティション式が評価された後でコンテナ名に付加されます。コンテナ名の中で、データベース・パーティション式の後ろにスペースがない場合は、ストリングの残りは式の一部であると見なされます。引数は、以下のいずれかの形式でのみ使用できます。

表 23. コンテナを作成するための引数： 演算子は左から右へと評価されます。この例では、データベース・パーティション番号は 5 であるとします。

構文	例	値
[blank]\$N	" \$N"	5
[blank]\$N+[number]	" \$N+1011"	1016
[blank]\$N%[number]	" \$N%3" ^a	2
[blank]\$N+[number]%[number]	" \$N+12%13"	4
[blank]\$N%[number]+[number]	" \$N%3+20"	22

^a % は係数演算子を表します。

以下に例を示します。

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
(device '/dev/rcont $N' 20000)
```

On a two database partition system, the following containers would be created:

```
/dev/rcont0 - on DATABASE PARTITION 0
/dev/rcont1 - on DATABASE PARTITION 1
```

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
(file '/DB2/containers/TS2/container $N+100' 10000)
```

On a four database partition system, the following containers would be created:

```
/DB2/containers/TS2/container100 - on DATABASE PARTITION 0
/DB2/containers/TS2/container101 - on DATABASE PARTITION 1
/DB2/containers/TS2/container102 - on DATABASE PARTITION 2
/DB2/containers/TS2/container103 - on DATABASE PARTITION 3
```

CREATE TABLESPACE

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
('/TS3/cont $N%2','/TS3/cont $N%2+2')
```

On a two database partition system, the following containers would be created:

```
/TS3/cont0 - On DATABASE PARTITION 0
/TS3/cont2 - On DATABASE PARTITION 0
/TS3/cont1 - On DATABASE PARTITION 1
/TS3/cont3 - On DATABASE PARTITION 1
```

If database partition = 5, the containers:

```
'/dbdir/node $N /cont1'
'/ $N+1000 /file1'
' $N%10 /container'
'/dir/ $N2000 /dmscont'
```

are created as:

```
'/dbdir/node5/cont1'
'/1005/file1'
'5/container'
'/dir/2000/dmscont'
```

- 自動ストレージ表スペースは、SMS 表スペースまたは DMS 表スペースのいずれかとして作成されます。DMS は、REGULAR 表スペースおよび LARGE 表スペースの場合に選択され、SMS は TEMPORARY 表スペースの場合に選択されます。この動作は将来のリリースで変更される可能性があるため、この動作に依存することはできないことに注意してください。DMS を選択し、表スペースのタイプを指定しない場合は、LARGE 表スペースの作成がデフォルトの動作になります。
- 自動ストレージ表スペースの作成には、コンテナ定義は含まれません。データベース・マネージャーはコンテナのロケーションおよびサイズを、データベースに関連したストレージ・パスを基にして、自動的に判別します (適用できる場合)。データベース・マネージャーは、LARGE 表スペースおよび REGULAR 表スペースを、最大サイズに達しないかぎり、必要に応じて拡張しようとします。これには既存のコンテナの拡張、または新規ストライプ・セットへのコンテナの追加が含まれます。データベースをアクティブにするたびに、データベース・マネージャーは、異常な状態にない TEMPORARY 表スペース用のコンテナの数およびロケーションを自動的に再構成します。
- LARGE 自動ストレージ表スペースまたは REGULAR 自動ストレージ表スペースは、表スペースが使用している既存のストレージ・パスの 1 つのスペースがなくなるまでは、新規ストレージ・パスを使用しません (ALTER DATABASE ステートメントの説明を参照してください)。TEMPORARY 自動ストレージ表スペースは、データベースが非アクティブにされた後、再度アクティブにされた場合にのみ、新規ストレージ・パスを使用できます。
- **互換性**
 - 以前のバージョンの DB2 との互換性:
 - DBPARTITIONNUM の代わりに NODE を指定できます。
 - DBPARTITIONNUMS の代わりに NODES を指定できます。
 - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できません。
 - LARGE の代わりに LONG を指定できます。

例

例 1: UNIX システムで、それぞれ 10 000 個の 4K ページを持った 3 つの装置を使用して、LARGE DMS 表スペースを作成します。それらの入出力特性も指定します。

```
CREATE TABLESPACE PAYROLL
  MANAGED BY DATABASE
  USING (DEVICE '/dev/rhdisk6' 10000,
        DEVICE '/dev/rhdisk7' 10000,
        DEVICE '/dev/rhdisk8' 10000)
  OVERHEAD 12.67
  TRANSFERRATE 0.18
```

例 2: Windows で、3 つの別個のドライブの 3 つのディレクトリーを使用し、エクステント・サイズを 64 ページ、プリフェッチ・サイズを 32 ページに指定して、REGULAR SMS 表スペースを作成します。

```
CREATE TABLESPACE ACCOUNTING
  MANAGED BY SYSTEM
  USING ('d:%acc_tbsp', 'e:%acc_tbsp', 'f:%acc_tbsp')
  EXTENTSIZE 64
  PREFETCHSIZE 32
```

例 3: UNIX システムで、それぞれ 50 000 ページの 2 つのファイル、および 256 ページのエクステント・サイズを使用して、SYSTEM TEMPORARY DMS 表スペースを作成します。

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
  MANAGED BY DATABASE
  USING (FILE 'dbtmp/tempspace2.f1' 50000,
        FILE 'dbtmp/tempspace2.f2' 50000)
  EXTENTSIZE 256
```

例 4: UNIX システムで、データベース・パーティション・グループ ODDNODEGROUP (データベース・パーティション 1、3、5) に LARGE DMS 表スペースを作成します。各データベース・パーティションで、装置 /dev/rhdisk0 の 10 000 個の 4K ページを使用します。また、それぞれのデータベース・パーティションに、40 000 個の 4K ページがあるデータベース・パーティション固有の装置を指定します。

```
CREATE TABLESPACE PLANS
  MANAGED BY DATABASE
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
  ON DBPARTITIONNUM (1)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
  ON DBPARTITIONNUM (3)
  USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
  ON DBPARTITIONNUM (5)
```

例 5: DATATS という名の LARGE 自動ストレージ表スペースを作成し、表スペースのサイズと拡張に関するすべての決定をシステムに任せます。

```
CREATE TABLESPACE DATATS
```

または

```
CREATE TABLESPACE DATATS
  MANAGED BY AUTOMATIC STORAGE
```

CREATE TABLESPACE

例 6: TEMPDATA という名の SYSTEM TEMPORARY 自動ストレージ表スペースを作成します。

```
CREATE TEMPORARY TABLESPACE TEMPDATA
```

または

```
CREATE TEMPORARY TABLESPACE TEMPDATA  
MANAGED BY AUTOMATIC STORAGE
```

例 7: 初期サイズが 100 M バイト、最大サイズが 1 G バイトの、USERSPACE3 という名の LARGE 自動ストレージ表スペースを作成します。

```
CREATE TABLESPACE USERSPACE3  
INITIALSIZE 100 M  
MAXSIZE 1 G
```

例 8: 拡張率が 10 パーセント (つまり、自動的にサイズ変更されるたびに、合計サイズが 10 パーセントずつ増加する) で、最大サイズが 512 M バイトの、LARGEDATA という名の LARGE 自動ストレージ表スペースを作成します。INITIALSIZE 節を指定する代わりに、表スペースの適切な初期サイズをデータベース・マネージャーに決定させます。

```
CREATE LARGE TABLESPACE LARGEDATA  
INCREASESIZE 10 PERCENT  
MAXSIZE 512 M
```

例 9: 2 つのファイル・コンテナを持ち (各コンテナのサイズは 1 M バイト)、拡張率が 2 M バイト、最大サイズが 100 M バイトの、USERSPACE4 という名の LARGE DMS 表スペースを作成します。

```
CREATE TABLESPACE USERSPACE4  
MANAGED BY DATABASE USING (FILE '/db2/file1' 1 M, FILE '/db2/file2' 1 M)  
AUTORESIZE YES  
INCREASESIZE 2 M  
MAXSIZE 100 M
```

例 10: Windows オペレーティング・システムで、ロー・デバイスを使用して、LARGE DMS 表スペースを作成します。

- 物理ドライブ全体を指定する場合は、`%%.¥physical-drive` という形式を使用します。

```
CREATE TABLESPACE TS1  
MANAGED BY DATABASE USING (DEVICE '%%.¥PhysicalDrive5' 10000,  
DEVICE '%%.¥PhysicalDrive6' 10000)
```

- ドライブ名を使用して論理パーティションを指定する場合は、以下のようにします。

```
CREATE TABLESPACE TS2  
MANAGED BY DATABASE USING (DEVICE '%%.¥G:' 10000,  
DEVICE '%%.¥H:' 10000)
```

- ボリュームのグローバル・ユニーク ID (GUID) を使用して論理パーティションを指定する場合は、db2listvolumes ユーティリティを使用して、各ローカル・パーティションのボリュームの GUID を取得してから、対象の論理パーティションの GUID を表スペース・コンテナ節にコピーします。

```
CREATE TABLESPACE TS3  
MANAGED BY DATABASE USING (  
DEVICE '%%?¥Volume{2ca6a0c1-8542-11d8-9734-00096b5322d2}%' 20000M)
```

マシンで使用できるドライブ名の数よりも多くのパーティションがある場合は、ドライブ名形式よりもボリュームの GUID を使用するほうが便利です。

- ジャンクション・ポイント (またはボリュームのマウント・ポイント) を使用して論理パーティションを指定する場合は、NTFS 形式の別のボリュームにロー・パーティションをジャンクション・ポイントとしてマウントしてから、その NTFS ボリュームのジャンクション・ポイントのパスをコンテナ・パスとして指定します。以下に例を示します。

```
CREATE TABLESPACE TS4
MANAGED BY DATABASE USING (DEVICE 'C:¥JUNCTION¥DISK_1' 10000,
DEVICE 'C:¥JUNCTION¥DISK_2' 10000)
```

DB2 はまず、パーティションに対する照会によって、そのパーティションにファイル・システムがあるかどうかを確認します。ある場合は、そのパーティションをロー・デバイスとしては扱いません。DB2 は、そのパーティションで通常のファイル・システム入出力操作を実行します。

CREATE THRESHOLD

CREATE THRESHOLD ステートメントはしきい値を定義します。

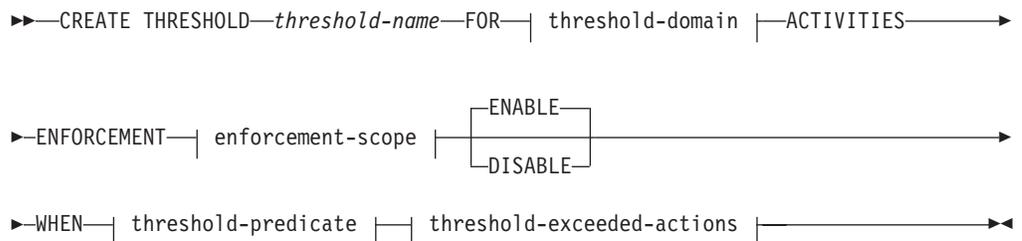
呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

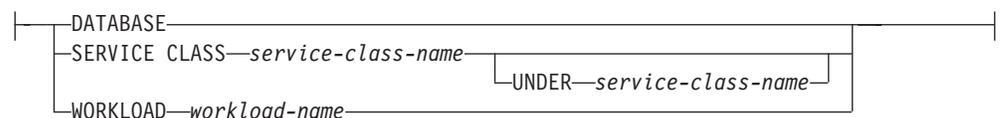
許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文



threshold-domain:

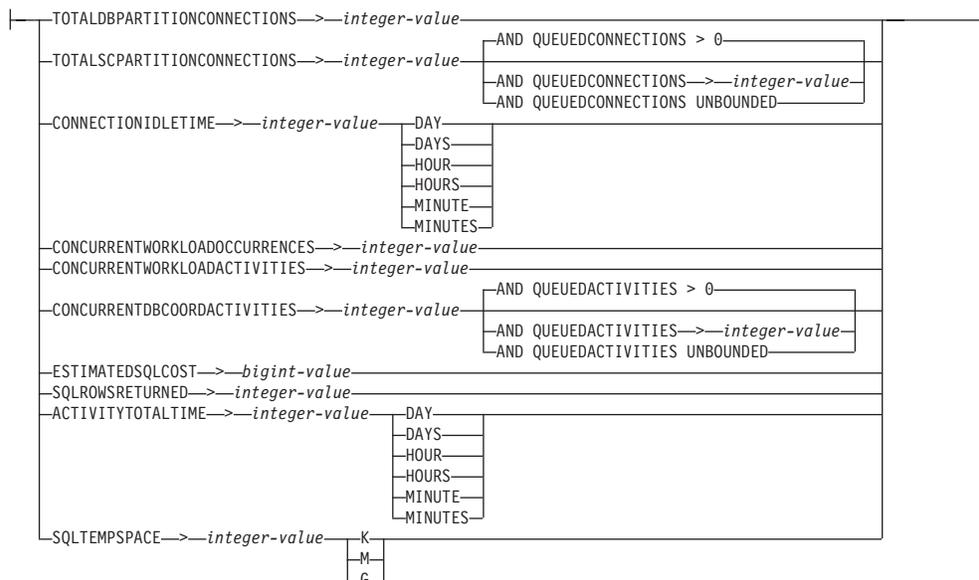


CREATE THRESHOLD

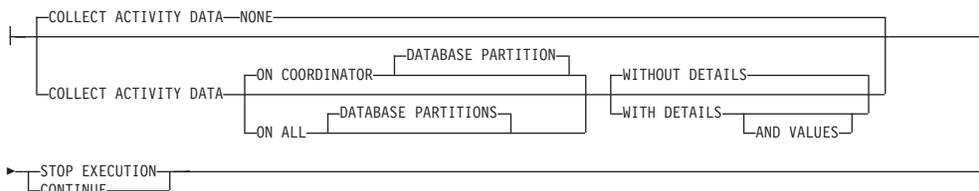
enforcement-scope:



threshold-predicate:



threshold-exceeded-actions:



説明

threshold-name

しきい値に名前を付けます。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *threshold-name* は、現行のサーバー上に既存のしきい値を識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

FOR *threshold-domain* ACTIVITIES

しきい値の定義ドメインを指定します。

DATABASE

このしきい値はデータベース内のすべてのアクティビティに適用されます。

SERVICE CLASS *service-class-name*

このしきい値は、サービス・クラス *service-class-name* 内で実行中のアクティビティに適用されます。UNDER が指定されない場合、*service-class-name* には既存のサービス・スーパークラスを指定する必要があります (SQLSTATE 42704)。UNDER が指定される場合は、*service-class-name* には UNDER キーワードの後に指定されるサービス・スーパークラスの既存のサービス・サブクラスを指定する必要があります (SQLSTATE 42704)。

UNDER *service-class-name*

サービス・スーパークラスを指定します。*service-class-name* には既存のサービス・スーパークラスを指定する必要があります (SQLSTATE 42704)。

WORKLOAD *workload-name*

このしきい値は指定されたワークロードに適用されます。*workload-name* には既存のワークロードを指定する必要があります (SQLSTATE 42704)。

ENFORCEMENT *enforcement-scope*

しきい値の強制の有効範囲。

DATABASE

しきい値は、定義ドメイン内のすべてのデータベース・パーティション、つまりデータベースのすべてのデータベース・パーティションとサービス・クラスのすべてのデータベース・パーティションで強制されます。

DATABASE PARTITION

しきい値は、データベース・パーティション単位で強制されます。しきい値を強制するための、全データベース・パーティション間での調整は行われません。

WORKLOAD OCCURRENCE

しきい値は、ワークロード・オカレンス内でのみ強制されます。同一データベース・パーティション上に 2 つのワークロード・オカレンスが並行して実行されている場合、このしきい値の実行カウントはそれぞれのワークロード・オカレンスごとにカウントされます。

ENABLE または **DISABLE**

データベース・マネージャーでしきい値を使用可能にするかどうかを指定します。

ENABLE

データベース・アクティビティの実行を制限するために、データベース・マネージャーでしきい値を使用します。

DISABLE

データベース・アクティビティの実行を制限するためにデータベース・マネージャーでしきい値を使用しません。

WHEN *threshold-predicate*

しきい値の条件を指定します。

TOTALDBPARTITIONCONNECTIONS > *integer-value*

この条件は、データベース・パーティション上で並行して実行できるコードイネーター接続の数の上限を定義します。この値には、ゼロを含む任意の正

CREATE THRESHOLD

整数を指定できます (SQLSTATE 42820)。値ゼロは、新しいコーディネーター接続を接続させないことを意味します。現在実行中の接続やキューに入っている接続は続行されます。この条件の定義ドメインは DATABASE でなければならず、強制の有効範囲は DATABASE PARTITION にする必要があります (SQLSTATE 5U037)。

TOTALSCPARTITIONCONNECTIONS > *integer-value*

この条件は、特定のサービス・スーパークラスのデータベース・パーティション上で並行して実行できるコーディネーター接続の数の上限を定義します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、新しい接続をサービス・クラスに結合しないことを意味します。現在実行中の接続やキューに入っている接続は続行されます。この条件の定義ドメインは SERVICE SUPERCLASS でなければならず、強制の有効範囲は DATABASE PARTITION にする必要があります (SQLSTATE 5U037)。

AND QUEUEDCONNECTIONS > *integer-value* または **AND QUEUEDCONNECTIONS UNBOUNDED**

コーディネーター接続が最大数を越えた場合のキュー・サイズを指定します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、コーディネーター接続をキューに入れないことを意味します。UNBOUNDED を指定した場合は、指定されたコーディネーター接続の最大数を越えた接続がすべてキューに入れられ、*threshold-exceeded-actions* は実行されません。デフォルト値はゼロです。

CONNECTIONIDLETIME > *integer-value* (DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)

この条件は、データベース・マネージャーが接続をアイドル状態のままにしておく時間の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。有効な期間キーワードを使用して、*integer-value* に適切な時間の単位を指定してください。この条件の定義ドメインは DATABASE または SERVICE SUPERCLASS でなければならず、強制の有効範囲は DATABASE にする必要があります (SQLSTATE 5U037)。この条件は、コーディネーターのデータベース・パーティションで論理的に強制されます。

STOP EXECUTION アクションを CONNECTIONIDLETIME しきい値とともに指定した場合、しきい値を超過すると、アプリケーションの接続はドロップされます。それ以降にアプリケーションがデータ・サーバーへのアクセスを試行しても、SQLSTATE 5U026 を受け取ることはありません。

このしきい値の最小の細分は 5 分であるため、指定された値は、最も近いゼロ以外の 5 分の倍数に切り上げられます。切り上げの結果がオーバーフローまたはアンダーフローになる場合は、整数の範囲の中で最も近い値が選択されます。

このしきい値の最大値は、2,147,400 秒です。2,147,400 秒よりも大きい秒に相当する値が指定された場合は、この秒数に設定されます。

DB2CHECKCLIENTINTERVAL レジストリー変数を変更してチェックの頻度を下げると、このしきい値の細分が影響を受けることがあります。

CONCURRENTWORKLOADOCCURRENCES > *integer-value*

この条件は、各データベース・パーティションでの並行するワークロード・オカレンスの数の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。この条件の定義ドメインは WORKLOAD でなければならず、強制の有効範囲は DATABASE PARTITION にする必要があります (SQLSTATE 5U037)。

CONCURRENTWORKLOADACTIVITIES > *integer-value*

この条件は、各データベース・パーティションのワークロードで並行して実行されるコーディネーター・アクティビティーとネストされたアクティビティーの数の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。この条件の定義ドメインは WORKLOAD でなければならず、この条件の強制の有効範囲は WORKLOAD OCCURRENCE にする必要があります (SQLSTATE 5U037)。

ネストされるアクティビティーは、それぞれ以下の条件を満たしている必要があります。

- 認識されているコーディネーター・アクティビティーである必要があります。認識されているアクティビティーのタイプに含まれないネストされたコーディネーター・アクティビティーは、カウントされません。同じように、リモート・ノード要求などのネストされたサブエージェント・アクティビティーもカウントされません。
- SQL ステートメントを発行するユーザー作成のプロシージャなどの、ユーザー・ロジックから直接呼び出される必要があります。

したがって、DB2 ユーティリティーや SYSIBM、SYSFUN、または SYSPROC スキーマのルーチンの呼び出しによって自動的に開始された、ネストされたコーディネーター・アクティビティーは、このしきい値で指定された上限にカウントされません。

制約の設定やマテリアライズ照会表のリフレッシュによって開始されるアクティビティーなどの内部 SQL アクティビティーも、データベース・マネージャーによって開始されるもので、ユーザー・ロジックによって直接呼び出されるものではないため、このしきい値にはカウントされません。

CONCURRENTDBCOORDACTIVITIES > *integer-value*

この条件は、指定されたドメイン内のすべてのデータベース・パーティションで並行して実行できる、認識されているデータベース・コーディネーター・アクティビティーの数の上限を定義します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、新しいデータベース・コーディネーター・アクティビティーを実行しないことを意味します。現在実行中またはキューに入っているデータベース・コーディネーター・アクティビティーは続行されます。この条件の定義ドメインは DATABASE、作業アクション (作業アクション定義ドメインのしきい値は CREATE WORK ACTION SET または ALTER WORK ACTION SET ステートメントを使用して作成される)、SERVICE SUPERCLASS、または SERVICE SUBCLASS でなければならず、強制の有効範囲は DATABASE にする必要があります (SQLSTATE 5U037)。

CREATE THRESHOLD

重要: CONCURRENTDBCOORDACTIVITIES しきい値が原因で、外部からの操作を必要とするキュー・ベースの競合が発生することがあります。詳しくは、『CONCURRENTDBCOORDACTIVITIES しきい値』を参照してください。

AND QUEUEDACTIVITIES > integer-value または AND QUEUEDACTIVITIES UNBOUNDED

データベース・コーディネーター・アクティビティーが最大数を越えた場合のキュー・サイズを指定します。この値には、ゼロを含む任意の正整数を指定できます (SQLSTATE 42820)。値ゼロは、データベース・コーディネーター・アクティビティーをキューに入れないことを意味します。UNBOUNDED を指定した場合、指定されたデータベース・コーディネーター・アクティビティーの最大数を越えたデータベース・コーディネーター・アクティビティーがすべてキューに入れられ、*threshold-exceeded-actions* は実行されません。デフォルト値はゼロです。

ESTIMATEDSQLCOST > bigint-value

この条件は、アクティビティーのオプティマイザー割り当てコスト (timeron 単位) の上限を定義します。この値には、ゼロ以外の任意の 64 ビット正整数を指定できます (SQLSTATE 42820)。この条件の定義ドメインは DATABASE、作業アクション (作業アクション定義ドメインのしきい値は CREATE WORK ACTION SET または ALTER WORK ACTION SET ステートメントを使用して作成される)、SERVICE SUPERCLASS、または SERVICE SUBCLASS でなければならず、強制の有効範囲は DATABASE にする必要があります (SQLSTATE 5U037)。この条件は、コーディネーターのデータベース・パーティションで強制されます。この条件では、以下のアクティビティーが追跡されます。

- データ操作言語 (DML) タイプのコーディネーター・アクティビティー。
- ユーザー・ロジックから呼び出されるネストされた DML アクティビティー。したがって、データベース・マネージャー (ユーティリティー、プロシージャー、内部 SQL など) によって開始される DML アクティビティーは、この条件では追跡されません (ただし、コストが親の見積もりに含まれている場合は、これらのアクティビティーは間接的に追跡されます)。

SQLROWSRETURNED > integer-value

この条件は、アプリケーション・サーバーからクライアント・アプリケーションに戻される行の数の上限を定義します。この値には、ゼロ以外の任意の整数を指定できます (SQLSTATE 42820)。この条件の定義ドメインは DATABASE、作業アクション (作業アクション定義ドメインのしきい値は CREATE WORK ACTION SET または ALTER WORK ACTION SET ステートメントを使用して作成される)、SERVICE SUPERCLASS、または SERVICE SUBCLASS でなければならず、強制の有効範囲は DATABASE にする必要があります (SQLSTATE 5U037)。この条件は、コーディネーターのデータベース・パーティションで強制されます。この条件では、以下のアクティビティーが追跡されます。

- DML タイプのコーディネーター・アクティビティー。

- ユーザー・ロジックから派生するネストされた DML アクティビティ。ユーティリティ、プロシージャー、または内部 SQL によってデータベース・マネージャーから開始されたアクティビティは、この条件による影響を受けません。

プロシージャー内から戻される結果セットは、個々のアクティビティとして別個に扱われます。プロシージャーそのものから戻される行の集約はありません。

ACTIVITYTOTALTIME > *integer-value* (DAY | DAYS | HOUR | HOURS | MINUTE | MINUTES)

この条件は、アクティビティがキューに入れられている時間を含んだ、データベース・マネージャーがアクティビティの実行のために許可する時間の上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。有効な期間キーワードを使用して、*integer-value* に適切な時間の単位を指定してください。この条件の定義ドメインは DATABASE、作業アクション (作業アクション定義ドメインのしきい値は CREATE WORK ACTION SET または ALTER WORK ACTION SET ステートメントを使用して作成される)、SERVICE SUPERCLASS、または SERVICE SUBCLASS でなければならず、強制的有効範囲は DATABASE にする必要があります (SQLSTATE 5U037)。この条件は、コーディネーターのデータベース・パーティションで論理的に強制されます。

このしきい値の最小の細分は 5 分であるため、指定された値は、最も近いゼロ以外の 5 分の倍数に切り上げられます。切り上げの結果がオーバーフローまたはアンダーフローになる場合は、整数の範囲の中で最も近い値が選択されます。

このしきい値の最大値は、2,147,400 秒です。2,147,400 秒よりも大きい秒に相当する値が指定された場合は、この秒数に設定されます。

DB2CHECKCLIENTINTERVAL レジストリー変数を変更してチェックの頻度を下げると、このしきい値の細分が影響を受けることがあります。

SQLTEMPSPACE > *integer-value* (K | M | G)

この条件は、すべてのデータベース・パーティションでの SYSTEM TEMPORARY 表スペースのサイズの上限を定義します。この値には、ゼロ以外の正整数を指定できます (SQLSTATE 42820)。

integer-value K (大文字または小文字のどちらでも可) を指定した場合、最大サイズは *integer-value* の 1024 倍です。 *integer-value M* を指定した場合は、最大サイズは *integer-value* の 1 048 576 倍です。 *integer-value G* を指定した場合は、最大サイズは *integer-value* の 1 073 741 824 倍です。

この条件の定義ドメインは DATABASE、作業アクション (作業アクション定義ドメインのしきい値は CREATE WORK ACTION SET または ALTER WORK ACTION SET ステートメントを使用して作成される)、SERVICE SUPERCLASS、または SERVICE SUBCLASS でなければならず、強制的有効範囲は DATABASE PARTITION にする必要があります (SQLSTATE 5U037)。この条件では、以下のアクティビティが追跡されます。

- DML タイプのコーディネーター・アクティビティと、それに対応するサブエージェント作業 (サブセクション実行)。

CREATE THRESHOLD

- ユーザー・ロジックから派生するネストされた DML アクティビティと、それに対応するサブエージェント作業 (サブセッション実行)。ユーティリティ、プロシージャ、または内部 SQL によってデータベース・マネージャーから開始されたアクティビティは、この条件による影響を受けません。

threshold-exceeded-actions

条件を超過したときに実行するアクションを指定します。条件を超過するたびに、しきい値違反イベント・モニター (アクティブになっている場合) にイベントが記録されます。

STOP EXECUTION

アクティビティの実行を停止し、エラーを戻します (SQLSTATE 5U026)。

CONTINUE

アクティビティの実行を停止しません。

COLLECT ACTIVITY DATA

しきい値を超過した各アクティビティに関するデータを、アクティビティが完了したときに、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。デフォルトは COLLECT ACTIVITY DATA NONE です。COLLECT ACTIVITY DATA が指定されている場合、デフォルトは WITHOUT DETAILS です。

NONE

しきい値を超過する各アクティビティについて、アクティビティ・データを収集しないことを指定します。

ON COORDINATOR DATABASE PARTITION

アクティビティのコーディネーターのデータベース・パーティションでのみアクティビティ・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティが処理されるすべてのデータベース・パーティションでアクティビティ・データを収集することを指定します。予測しきい値の場合、しきい値を超過した場合の CONTINUE アクションも指定した場合のみ、アクティビティ情報がすべてのパーティションで収集されます。反応しきい値の場合、ON ALL DATABASE PARTITIONS 節を指定しても効果はなく、アクティビティ情報は常にコーディネーター・パーティションでのみ収集されます。予測しきい値および反応しきい値の両方について、アクティビティの詳細または値は、コーディネーター・パーティションでのみ収集されます。

WITHOUT DETAILS

しきい値を超過する各アクティビティに関するデータを、アクティビティが完了したときに、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境の情報はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティについて、これらの情報をアクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティーに入力データ値が含まれている場合、それをアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

注

- **しきい値超過アクション CONTINUE とイベント・モニター・データ:** イベント・モニター・データは、しきい値条件を超過したときにパーティションにつき一度だけ収集されます。しきい値超過アクションが CONTINUE の場合、アクティビティーは引き続き実行され、影響するパーティションでそのしきい値のイベント・モニター・データはそれ以上収集されません。例えば、10 分の時間しきい値にアクション CONTINUE が指定されているとします。アクティビティーが上限の 10 分を超過すると、影響するパーティションで、そのしきい値のイベント・モニター・データが収集されます。
- **サービス・クラスの静止:** 通常は静止できないサービス・クラス (例えば、デフォルト・ユーザー・クラスやデフォルト・システム・クラス) の静止をシミュレートするには、TOTALSPARTITIONCONNECTIONS しきい値条件を使用できます。SYSDEFAULTADMWORKLOAD で実行される DBADM 権限を持つユーザーにはしきい値が適用されない一方で、静止サービス・クラスはどのユーザーにも使用できないため、これは便利です。したがって、デフォルトのサービス・クラスは直接静止させることはできませんが、唯一、しきい値を使用して、DBADM 権限を持つユーザーが SYSDEFAULTADMWORKLOAD を使用してデータベースに接続する際にそれらを結合できるようにすることによってそれが可能になります。

例

例 1: データベース内のすべてのアクティビティーに対して TEMPORARY 表スペースの最大使用量を 50M (データベース・パーティションごとに) に強制するしきい値を作成します。このしきい値に違反するアクティビティーはすべて停止させます。

```
CREATE THRESHOLD DBMAX50MEGTEMPSPACE
FOR DATABASE ACTIVITIES
ENFORCEMENT DATABASE PARTITION
WHEN SQLTEMPSPACE > 50 M
STOP EXECUTION
```

例 2: データベース内のすべてのアクティビティーのデフォルトの実行時間に最大 1 時間という制限を設ける、2 つ目のしきい値を作成します。このしきい値に違反するアクティビティーはすべて停止させます。

```
CREATE THRESHOLD DBMAX1HOURLRUNTIME
FOR DATABASE ACTIVITIES
ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 1 HOUR
STOP EXECUTION
```

例 3: 平均より多くの TEMPORARY スペースを使用し、実行時間が 1 時間を超える照会をホストするために BIGQUERIES という名前のサービス・スーパークラスが作成されたとします。このサービス・クラス内に定義されるしきい値は、データベース・レベルで先に設定した値をオーバーライドします。このスーパークラス内

CREATE THRESHOLD

部のしきい値に違反するアクティビティーを引き続き実行させ、かつ詳しい分析のために詳細情報を収集する方法に注目してください。

```
CREATE THRESHOLD BIGQUERIESMAX500MEGTEMPSPACE
FOR SERVICE CLASS BIGQUERIES ACTIVITIES
ENFORCEMENT DATABASE PARTITION
WHEN SQLTEMPSPACE > 500 M
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
CONTINUE
```

```
CREATE THRESHOLD BIGQUERIESLONGRUNNINGTIME
FOR SERVICE CLASS BIGQUERIES ACTIVITIES
ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 10 HOURS
COLLECT ACTIVITY DATA WITH DETAILS AND VALUES
CONTINUE
```

例 4: PAYROLL という名前のワークロードが存在するとします。このワークロード内のアクティビティーの最大数が 10 以下にするしきい値を作成します。

```
CREATE THRESHOLD MAXACTIVITIESINPAYROLL
FOR WORKLOAD PAYROLL ACTIVITIES
ENFORCEMENT WORKLOAD OCCURRENCE
WHEN CONCURRENTWORKLOADACTIVITIES > 10
STOP EXECUTION
```

例 5: サービス・クラス BIGQUERIES 内の並行するアクティビティーの最大数が 2 になるように強制するしきい値を作成します。

```
CREATE THRESHOLD MAXBIGQUERIESCONCURRENCY
FOR SERVICE CLASS BIGQUERIES ACTIVITIES
ENFORCEMENT DATABASE
WHEN CONCURRENTDBCOORDACTIVITIES > 2
STOP EXECUTION
```

CREATE TRANSFORM

CREATE TRANSFORM ステートメントは、グループ名で識別されるトランスフォーマーメーション関数を定義します。この関数は、ホスト言語プログラムおよび外部関数を相手に構造化タイプ値を交換するために使います。

呼び出し

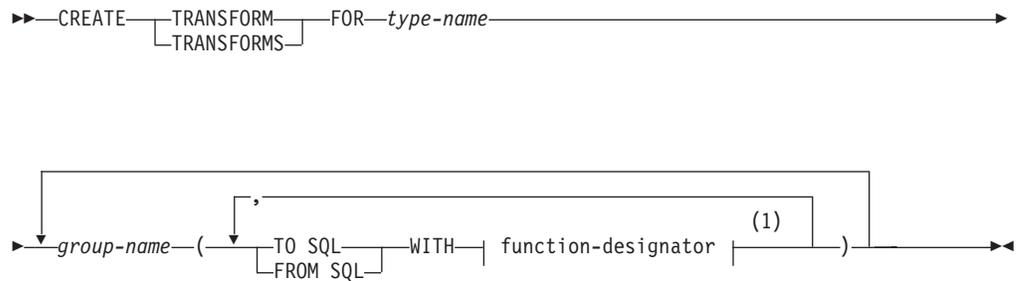
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- *type-name* で指定されたタイプの所有者と、指定された関数ごとの EXECUTE 特権
- SYSADM または DBADM 権限

構文



注:

- 1 同じ節を複数回指定することはできません。

説明

TRANSFORM または TRANSFORMS

1 つ以上のトランスフォーム・グループを定義することを指示します。いずれかのバージョンのキーワードを指定することができます。

FOR type-name

トランスフォーム・グループを定義する対象となるユーザー定義構造化タイプの名前を指定します。

動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のない *type-name* の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のない *type-name* の修飾子が暗黙指定されます。この *type-name* は既存のユーザー定義タイプであり (SQLSTATE 42704)、しかも構造化タイプでなければなりません (SQLSTATE 42809)。構造化タイプまたはこれと同じタイプ階層内の他の構造化タイプが、特定のグループ名を使ってトランスフォームをすでに定義済みであるとはなりません (SQLSTATE 42739)。

group-name

トランスフォーム・グループに名前を付けます。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *group-name* は、指定された *type-name* のカタログにすでに存在するトランスフォーム・グループを指定するものであってはなりません (SQLSTATE 42739)。 *group-name* を文字 'SYS' で始めることはできません (SQLSTATE 42939)。どのグループに対しても、FROM SQL と TO SQL 関数のそれぞれの指名を最大で 1 つずつ指定することができます (SQLSTATE 42628)。

TO SQL

SQL ユーザー定義構造化タイプ・フォーマットに値をトランスフォームするのに使用する特定の関数を定義します。この関数では、すべてのパラメーターを組み込みデータ・タイプとして使用しなければならず、戻りタイプは *type-name* です。

FROM SQL

SQL ユーザー定義構造化タイプを表す組み込みデータ・タイプ値に値をトランスフォームするのに使用する特定の関数を定義します。この関数では、データ・

CREATE TRANSFORM

タイプ *type-name* の 1 つのパラメーターを使う必要があり、1 つの組み込みデータ・タイプ (または一連の組み込みデータ・タイプ) を戻します。

WITH *function-designator*

トランスフォーム関数を一意的に識別します。

FROM SQL を指定する場合、*function-designator* に、次のような要件を満たす関数を指定しなければなりません。

- タイプ *type-name* の 1 つのパラメーターがある
- 戻りタイプは、組み込みタイプであるか、またはすべてが組み込みタイプの列をもつ行である
- シグニチャーは、LANGUAGE SQL を指定しているか、または LANGUAGE SQL をもつ別の FROM SQL トランスフォーム関数の使用を指定している

TO SQL を指定する場合、*function-designator* に、次のような要件を満たす関数を指定しなければなりません。

- すべてのパラメーターに組み込みタイプがある
- 戻りタイプは *type-name* である
- シグニチャーは、LANGUAGE SQL を指定しているか、または LANGUAGE SQL をもつ別の TO SQL トランスフォーム関数の使用を指定している

function-designator に、これらの要件 (FROM SQL トランスフォーム関数として使用する場合の要件、または TO SQL トランスフォーム関数として使用する場合の要件) を満たさない関数を指定すると、エラー (SQLSTATE 428DC) になります。

function-designator によって、メソッドをトランスフォームとして指定することはできません (FUNCTION ACCESS を使用して指定した場合でも)。トランスフォームとして機能できるのは、CREATE FUNCTION ステートメントによって定義された関数のみです (SQLSTATE 42704 または 42883)。

詳しくは、16 ページの『関数、メソッド、およびプロシージャの指定子』を参照してください。

規則

- FROM SQL 関数から戻された 1 つ以上の組み込みタイプが、TO SQL 関数のパラメーターである 1 つ以上の組み込み関数と直接対応しているべきです。これが、この 2 つの関数の相反する関係の論理的結末です。FROM トランスフォームと TO トランスフォームの間のこの関連が保持されない場合は、エラー (SQLSTATE -3) になります。

注

- 静的 SQL の場合は TRANSFORM GROUP プリコンパイル・オプションまたは BIND オプションを使って、動的 SQL の場合は SET CURRENT DEFAULT TRANSFORM GROUP ステートメントを使って、アプリケーション・プログラム内でトランスフォーム・グループを指定しない場合に、そのアプリケーション・プログラムが、*type-name* で指定されたユーザー定義の構造化タイプに基づいているホスト変数を検索または送信しようとする、トランスフォーム・グループ 'DB2_PROGRAM' 内のトランスフォーム関数が使われます (定義されている場合)。データ・タイプ *type-name* の値を検索すると、FROM SQL トランスフォームが呼び出され、構造化タイプは、トランスフォーム関数から戻された組み込

みデータ・タイプにトランスフォームされます。同様に、データ・タイプ *type-name* の値に割り当てられることになるホスト変数を送信すると、TO SQL トランスフォームが呼び出され、組み込みデータ・タイプ値は構造化タイプ値にトランスフォームされます。ユーザー定義のトランスフォーム・グループを指定しない場合や、'DB2_PROGRAM' グループ (特定の構造化タイプのもの) が定義されていない場合、エラーが生じます (SQLSTATE 42741)。

- 構造化タイプ・ホスト変数を表す組み込みデータ・タイプは、次のように割り当てる必要があります。
 - [割当元]プリコンパイル時に (検索割り当て規則を使用し) TRANSFORM GROUP オプションで定義した構造化タイプ用の FROM SQL トランスフォーム関数結果
 - [割当先] プリコンパイル時に (ストレージ割り当て規則を使用し) TRANSFORM GROUP オプションで定義された構造化タイプ用の TO SQL トランスフォーム関数のパラメーター

ホスト変数が、該当するトランスフォーム関数での規定のタイプと互換性のある割り当てでない場合、エラーが起きます (組み込みの場合は SQLSTATE 42821、バインドアウトの場合は SQLSTATE 42806)。ストリングの割り当てが原因のエラーの詳細は、『ストリング割り当て』を参照してください。

- パラメーターまたは戻りタイプとしてデータ・タイプ *type-name* を使って、SQL で作成されていないユーザー定義関数を呼び出す場合、そのたびに 'DB2_FUNCTION' という名前のデフォルト・トランスフォーム・グループ内で指定されているトランスフォーム関数が使用されます。これが行われるのは、関数に TRANSFORM GROUP 節を指定しない場合です。データ・タイプ *type-name* の引数を使って関数を呼び出す場合、FROM SQL トランスフォームが実行され、構造化タイプは、トランスフォーム関数から戻された組み込みデータ・タイプにトランスフォームされます。同様に、関数の戻りデータ・タイプがデータ・タイプ *type-name* である場合、TO SQL トランスフォームが呼び出され、外部関数プログラムから戻された組み込みデータ・タイプ値は、構造化タイプ値にトランスフォームされます。
- 構造化タイプの中に、やはり構造化タイプである属性が入っている場合、それに関連したトランスフォーム関数は、すべてのネストされた構造化タイプを繰り返し展開 (またはアセンブル) する必要があります。つまり、トランスフォーム関数の結果やパラメーターは、サブジェクト構造化タイプ (ネストされたすべての構造化タイプも含む) のすべての基本属性を表す一連の組み込みタイプだけで構成されることを意味します。ネストされた構造化タイプを処理するために、トランスフォーム関数が「カスケード化」されることはありません。
- このステートメントに指定する関数は、このステートメントの実行時に上記の概説どおりの規則に従って解決されます。これらの関数は、この後の SQL ステートメント内で (暗黙的に) 使用された場合、これ以外の解決プロセスをたどりません。このステートメントで定義されたトランスフォーム関数は、このステートメントで解決されるとおりに記録されます。
- 特定のタイプの属性またはサブタイプを作成またはドロップしたときは、ユーザー定義構造化タイプのトランスフォーム関数も変更する必要があります。
- ある特定のトランスフォーム・グループについて、FROM SQL トランスフォームと TO SQL トランスフォームを指定できるのは、同じ *group-name* 節、別の *group-name* 節、または別の CREATE TRANSFORM ステートメントのいずれか

CREATE TRANSFORM

においてです。既存のグループ定義をあらかじめドロップしておかないと、FROM SQL トランスフォームまたは TO SQL トランスフォームの指定を再定義できないことが唯一の制約事項です。それによって、たとえば、特定のグループの FROM SQL トランスフォームを先に定義しておいてから、後で同じグループ用の対応する TO SQL トランスフォームを定義することができます。

例

例 1: ユーザー定義構造化タイプの多角形を、C 用にカスタマイズしたトランスフォーム関数と Java 用に特殊化したトランスフォーム関数に関連付ける 2 つのトランスフォーム・グループを作成します。

```
CREATE TRANSFORM FOR POLYGON
  mystruct1 (FROM SQL WITH FUNCTION myxform_sqlstruct,
            TO SQL WITH FUNCTION myxform_structsql)
  myjava1   (FROM SQL WITH FUNCTION myxform_sqljava,
            TO SQL WITH FUNCTION myxform_javasql)
```

CREATE TRIGGER

CREATE TRIGGER ステートメントは、データベースにトリガーを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- BEFORE または AFTER トリガーを定義する表に対する ALTER 特権
- INSTEAD OF トリガーを定義するビューに対する CONTROL 特権
- INSTEAD OF トリガーを定義するビューの所有者
- トリガーを定義する表またはビューのスキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

および以下のいずれか

- データベースに対する IMPLICIT_SCHEMA 権限 (トリガーの暗黙または明示のスキーマ名が存在しない場合)
- スキーマに対する CREATEIN 特権 (トリガーのスキーマ名が既存のスキーマを指している場合)

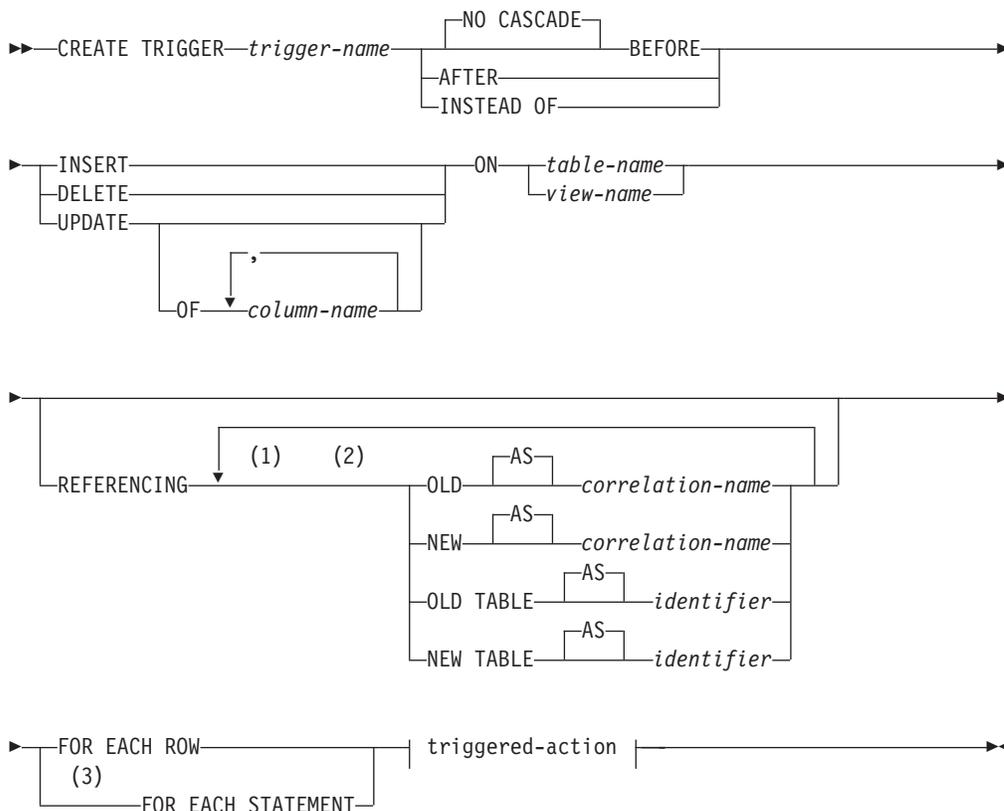
このステートメントの許可 ID に SYSADM 権限または DBADM 権限がない場合には、トリガーが存在する限り、ステートメントの許可 ID が持つ特権 (グループ特権は除外) に、以下のすべてが含まれている必要があります。

- 遷移変数または遷移表を指定する場合は、トリガーを定義する表に対する SELECT 特権

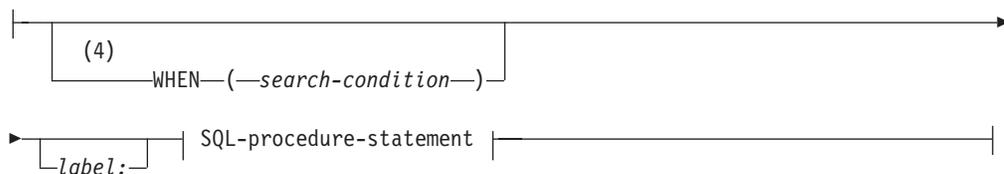
- トリガー・アクション条件で参照される表またはビューに対する SELECT 特権
- 指定したトリガー SQL ステートメントを呼び出すために必要な特権

SYSADM 権限を所持しているために、トリガーの定義者がトリガーの作成しか行えないような場合、その定義者には、トリガーを作成できるようにする目的で明示的な DBADM 権限が付与されます。

構文

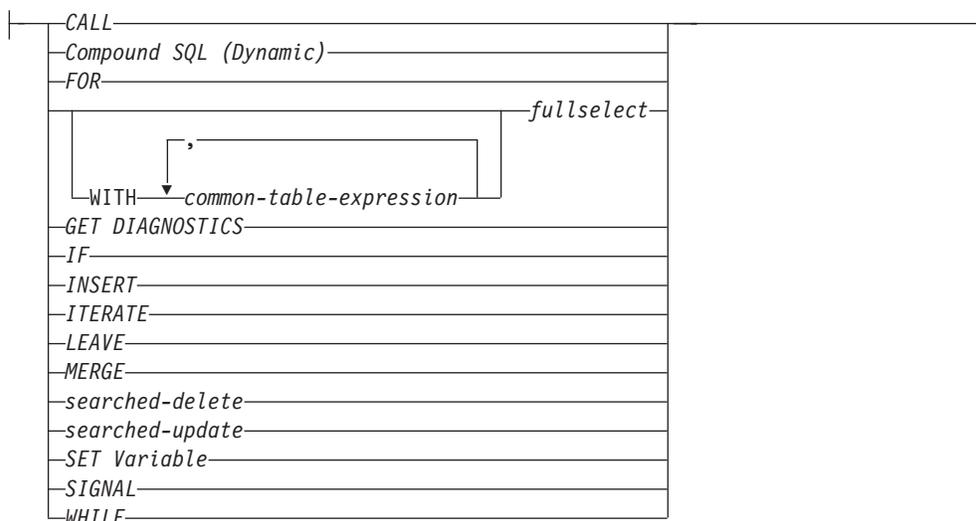


triggered-action:



SQL-procedure-statement:

CREATE TRIGGER



注:

- 1 OLD および NEW は、それぞれ一度だけ指定できます。
- 2 OLD TABLE および NEW TABLE はそれぞれ一度だけ、AFTER トリガーまたは INSTEAD OF トリガーにのみ指定できます。
- 3 FOR EACH STATEMENT を BEFORE トリガーまたは INSTEAD OF トリガーに指定することはできません。
- 4 WHEN 条件を INSTEAD OF トリガーに指定することはできません。

説明

trigger-name

トリガーの名前を指定します。暗黙のスキーマ名または明示スキーマ名を含む名前、カタログにすでに記述されているトリガーを指定するものではありません (SQLSTATE 42710)。2つの部分からなる名前を指定する場合、'SYS' で始まるスキーマ名は使用できません (SQLSTATE 42939)。

NO CASCADE BEFORE

対象となる表の実際の更新によって生じる変更がデータベースに適用される前に、関連するトリガー・アクションを適用することを指定します。また、このトリガーのトリガー・アクションが、他のトリガーを活動化することがないことも指定します。

AFTER

対象となる表の実際の更新によって生じる変更がデータベースに適用された後で、関連するトリガー・アクションを適用することを指定します。

INSTEAD OF

関連したトリガー・アクションが、対象となるビューに対するアクションを置換することを指定します。INSTEAD OF トリガーは、対象となる特定のビューに対する各操作ごとに1つだけ許可されます (SQLSTATE 428FP)。

INSERT

対象となる表または対象となるビューに INSERT 操作が適用される場合には必ず、このトリガーに関連するトリガー・アクションを実行することを指定します。

DELETE

対象となる表または対象となるビューに DELETE 操作が適用される場合には必ず、このトリガーに関連するトリガー・アクションを実行することを指定します。

UPDATE

指定した列または暗黙に指定される列に従って、対象となる表または対象となるビューに UPDATE 操作が適用される場合には必ず、このトリガーに関連するトリガー・アクションを実行することを指定します。

オプションの *column-name* のリストの指定がない場合、暗黙に表のすべての列が指定されます。したがって、*column-name* リストを省略すると、表の列のいずれかの更新によってトリガーが活動化されることが暗に指定されます。

OF *column-name*,...

指定する各 *column-name* (列名) は、基本表の列でなければなりません (SQLSTATE 42703)。トリガーが BEFORE トリガーである場合、指定される *column-name* は、ID 列以外の生成列にすることはできません (SQLSTATE 42989)。 *column-name* リストに 1 つの *column-name* を複数回指定することはできません (SQLSTATE 42711)。 トリガーは、*column-name* リストに指定した列が更新される場合にのみ活動化されることになります。この節は、INSTEAD OF トリガーには指定できません (SQLSTATE 42613)。

ON*table-name*

BEFORE トリガーまたは AFTER トリガー定義の対象となる表を指定します。その名前は、基本表、または基本表の名前がわかる別名を指定しなければなりません (SQLSTATE 42704 または 42809)。この名前に、カタログ表 (SQLSTATE 42832)、マテリアライズ照会表 (SQLSTATE 42997)、宣言済み一時表 (SQLSTATE 42995)、あるいはニックネーム (SQLSTATE 42809) を指定することはできません。

view-name

INSTEAD OF トリガー定義の対象となるビューを指定します。その名前では、非型付きビュー、またはタイプ XML の列を含まない非型付きビューに解決できる別名を指定しなければなりません (SQLSTATE 42704 または 42809)。指定する名前は、カタログ・ビューであってはなりません (SQLSTATE 42832)。指定する名前は、WITH CHECK OPTION (対称ビュー) を使用して定義されるビュー、または対称ビューが直接的または間接的に定義されたビューであってはなりません (SQLSTATE 428FQ)。

REFERENCING

遷移変数 の相関名と遷移表 の表名を指定します。相関名には、トリガーとなる SQL 操作によって影響を受ける一連の行の中の特定の行を指定します。表名には、影響を受ける行の集合全体を指定します。トリガーとなる SQL 操作によって影響を受ける各行をトリガー・アクションで使用するには、次のようにして指定される *correlation-name* (相関名) によって列を修飾します。

OLD AS *correlation-name*

トリガーとなる SQL 操作の前の時点での行の状態を指定する相関名を指定します。

CREATE TRIGGER

NEW AS *correlation-name*

トリガーとなる SQL 操作および、すでに実行された BEFORE トリガーの SET ステートメントによって、変更された時の行の状態を指定する相関名を指定します。

NEW AS 相関名のタイプは、XML が可能です。

トリガーとなる SQL 操作によって影響を受ける行全体の集合をトリガー・アクションで使用するには、次のように指定される一時表名を使用します。

OLD TABLE AS *identifier*

影響を受ける行の集合の、トリガーとなる SQL 操作の前のものを識別する一時表名を指定します。

NEW TABLE AS *identifier*

トリガーとなる SQL 操作および、すでに実行された BEFORE トリガーの SET ステートメントによって、変更された行の状態を識別する一時表名を指定します。

REFERENCING 節には、次の規則が適用されます。

- OLD および NEW の相関名と、OLD TABLE および NEW TABLE の名前は、いずれも同じであってはなりません (SQLSTATE 42712)。
- 1 つのトリガーには、*correlation-name* として、1 つの OLD と 1 つの NEW だけしか指定できません (SQLSTATE 42613)。
- 1 つのトリガーには、*identifier* として 1 つの OLD TABLE と 1 つの NEW TABLE しか指定できません (SQLSTATE 42613)。
- OLD *correlation-name* と OLD TABLE *identifier* は、トリガー・イベントが DELETE 操作または UPDATE 操作のいずれかである場合にしか使用できません (SQLSTATE 42898)。操作が DELETE 操作の場合、OLD *correlation-name* は、削除された行の値を取り込みます。操作が UPDATE 操作の場合、その UPDATE 操作の前の時点での行の値を捕らえるものとなります。同じことが OLD TABLE *identifier* とそれによって影響を受ける行の集合にも適用されます。
- NEW *correlation-name* と NEW TABLE *identifier* は、トリガー・イベントが INSERT 操作または UPDATE 操作のいずれかである場合にしか使用できません (SQLSTATE 42898)。どちらの操作でも、NEW の値は、元の操作によって提供されたが、その時点までに実行された BEFORE トリガーによってさらに変更された、行の新しい状態を捕らえます。同じことが NEW TABLE *identifier* とそれによって影響を受ける行の集合にも適用されます。
- OLD TABLE または NEW TABLE の ID は、BEFORE トリガーには定義できません (SQLSTATE 42898)。
- NEW の遷移変数は、AFTER トリガーには定義できません (SQLSTATE 42987)。
- OLD または NEW の相関名は、FOR EACH STATEMENT トリガーには定義できません (SQLSTATE 42899)。
- 遷移表は変更できません (SQLSTATE 42807)。

- トリガー・アクションの中での遷移表の列と遷移変数への参照の合計回数が、表内の列数の限界を超えてはなりません。また、その長さの合計が、表の中の行の最大長を超えてはなりません (SQLSTATE 54040)。
- 各 *correlation-name* と各 *identifier* の有効範囲は、トリガー定義全体です。

FOR EACH ROW

トリガーとなる SQL 操作によって影響を受ける対象となる表またはビューの各行ごとに、トリガー・アクションが一度ずつ適用されるよう指定します。

FOR EACH STATEMENT

トリガー・アクションが、ステートメント全体で一度だけ適用されることを指定します。このタイプのトリガー精度は、BEFORE トリガーまたは INSTEAD OF トリガーには指定できません (SQLSTATE 42613)。指定すると、トリガーとなる UPDATE または DELETE ステートメントによって影響を受ける行がない場合でも、UPDATE トリガーまたは DELETE トリガーが活動化されることになってしまいます。

triggered-action

トリガーを活動化するときに行われるアクションを指定します。トリガー・アクションは *SQL-procedure-statement*、および *SQL-procedure-statement* 実行のオプション条件から構成されています。

WHEN

(*search-condition*)

真、偽、または不明である条件を指定します。 *search-condition* によって、あるトリガー処置を実行すべきかどうかを決定する機能が与えられます。関連するアクションは、指定された検索条件が真である場合のみ実行されます。WHEN 節が省略されると、関連する *SQL-procedure-statement* が常に実行されます。

WHEN 節を INSTEAD OF トリガーに指定することはできません (SQLSTATE 42613)。

XML データ・タイプを使用する遷移変数の参照は、VALIDATED 述部でのみ使用できます。

label:

SQL プロシージャ・ステートメントのラベルを指定します。ラベルは、リスト内でネストされたコンパウンド・ステートメントを含め、SQL プロシージャ・ステートメントのリスト内でユニークでなければなりません。ネストされていないコンパウンド・ステートメントは、同じラベルを使用できることに注意してください。SQL プロシージャ・ステートメントのリストは、おそらく SQL 制御ステートメントの中にあります。

FOR ステートメント、WHILE ステートメント、および動的コンパウンド・ステートメントだけにラベルを組み込むことができます。

SQL-procedure-statement

トリガー・アクションの一部にする SQL ステートメントを指定します。コンパウンド SQL 内のニックネームに対する検索更新、検索削除、挿入、またはマージ操作はサポートされません。

XML タイプの列に対する BEFORE トリガーのトリガー・アクションで、SET ステートメントを介して XMLVALIDATE 関数を呼び出す、XML タ

CREATE TRIGGER

IPの値を変更せずにそのままにしておく、または SET ステートメントを使用して XML タイプの値に NULL を割り当てることができます。

SQL-procedure-statement には、サポートされていないステートメントを入れることはできません (SQLSTATE 42987)。

SQL-procedure-statement は、未定義の遷移変数 (SQLSTATE 42703)、フェデレーテッド・オブジェクト (SQLSTATE 42997)、または宣言済み一時表 (SQLSTATE 42995) を参照できません。

BEFORE トリガー内の *SQL-procedure-statement* には、以下の制限があります。

- MODIFIES SQL DATA で定義されたプロシージャを呼び出す CALL ステートメント、または MERGE ステートメントにすることができません (SQLSTATE 42987)。
- REFRESH IMMEDIATE で定義されたマテリアライズ照会表を参照できません (SQLSTATE 42997)。
- NEW 遷移変数内の ID 列以外の生成された列を参照できません (SQLSTATE 42989)。

注

- すでに行が含まれている表にトリガーを追加しても、トリガー・アクションは活動化されません。そのため、トリガーが表内のデータに制約を適用するように設計されている場合、既存の行についてはそれらの制約が満たされない可能性があります。
- 2 つのトリガーのイベントが同時に発生する場合 (例えばイベント、活動化のタイミング、および対象の表が同じである場合)、最初に作成されたトリガーが最初に実行されます。
- トリガーの定義後に対象の表に列が追加された場合、次の規則が適用されます。
 - トリガーが、明示的な列リストなしで指定された UPDATE トリガーである場合、新しい列への更新によってトリガーが活動化されます。
 - その列は、それ以前に定義されたトリガーのトリガー・アクションからは見えません。
 - OLD TABLE および NEW TABLE の各遷移表に、この列は含まれません。したがって、遷移表に対して "SELECT *" を実行しても、追加列は含められません。
- トリガー・アクションで参照される表に 1 つの列を追加した場合、新しい列はそのトリガー・アクションからは見えません。
- *SQL-procedure-statement* に指定されている全選択の結果は、トリガーの内部または外部では使用不可です。
- トリガー・コンパウンド・ステートメント内で呼び出されるプロシージャは、COMMIT または ROLLBACK ステートメントを発行できません (SQLSTATE 42985)。
- 検索 UPDATE ステートメント、検索 DELETE ステートメント、または INSERT ステートメント内のニックネームへの参照を含むプロシージャはサポートされていません (SQLSTATE 25000)。
- **表アクセスの制限:**

プロシージャが READS SQL DATA または MODIFIES SQL DATA として定義されている場合は、プロシージャ内のステートメントは、このプロシージャを呼び出したコンパウンド・ステートメントによって変更される表にアクセスすることはできません (SQLSTATE 57053)。プロシージャが MODIFIES SQL DATA として定義されている場合は、プロシージャ内のステートメントは、このプロシージャを呼び出したコンパウンド・ステートメントによって読み取られるまたは変更される表を変更できません (SQLSTATE 57053)。

- カスケードされた参照制約のサイクルに関係のある表に対して定義された BEFORE DELETE トリガーは、そのトリガーが定義されている表への参照や、参照整合性制約のサイクルの評価中にカスケードされて変更された他の表への参照には含めないでください。そのようなトリガーを含めると、結果がデータによってまちまちになり、一貫性のない状態が生じてしまう可能性があります。

このようなトリガーを最も簡明な形式にする方法は、自己参照の参照制約のある表に対する BEFORE DELETE トリガーや CASCADE の削除規則に、*triggered-action* に関係のある表への参照を含めないようにすることです。

- トリガーを作成すると、特定のパッケージは無効として扱われるようになります。
 - 明示的な列リストなしの UPDATE トリガーを作成した場合、ターゲット表またはビューに対して更新操作を使用するパッケージは無効になります。
 - 列リストを指定した UPDATE トリガーを作成した場合、ターゲット表に対して更新操作を使用するパッケージは、そのパッケージにおいて、CREATE TRIGGER ステートメントの *column-name* リストの中の少なくとも 1 つの列に対しても更新を使用する場合にのみ無効になります。
 - INSERT トリガーを作成した場合、ターゲット表またはビューに対して挿入操作を使用するパッケージは無効になります。
 - 削除トリガーを作成した場合、ターゲット表またはビューに対して削除操作を使用するパッケージは無効になります。
- パッケージは、アプリケーション・プログラムが明示的にバインドまたは再バインドされるまで、あるいはアプリケーション・プログラムが実行されてデータベース・マネージャーが自動的にそれを再バインドするまで、無効のままになります。
- **作動不能トリガー:** 作動不能トリガーは、使用可能でなくなったために活動化されないトリガーです。以下の場合、トリガーは操作不能になります。
 - トリガーを実行するため、そのトリガーの作成者が持っていなければならない特権が取り消された。
 - トリガーされたアクションが依存する、表、ビュー、または別名といったオブジェクトがドロップされた。
 - トリガーが定義されたビューが操作不能になった。
 - トリガーのサブジェクト表である別名がドロップされた。

実際、操作不能トリガーは、DROP または REVOKE ステートメントのカスケード規則の結果、トリガー定義がドロップされたトリガーです。例えば、ビューがドロップされると、そのビューを使用して *SQL-procedure-statement* が定義されているトリガーが操作不能になります。

CREATE TRIGGER

トリガーが操作不能になると、そのトリガーを活動化していた操作を実行するステートメントを持つパッケージはすべて無効とマークされます。パッケージが(明示的または暗黙的に)再バインドされると、操作不能トリガーは完全に無視されます。同様に、トリガーを活動化していた操作を実行する動的 SQL ステートメントを含むアプリケーションも、作動不能トリガーを完全に無視します。

トリガー名は、DROP TRIGGER および COMMENT ON TRIGGER の各ステートメントにも指定できます。

操作不能トリガーは、その定義テキストを使用して CREATE TRIGGER ステートメントを出すことによって再作成できます。このトリガー定義テキストは、SYSCAT.TRIGGERS カタログ・ビューの TEXT 列に保管されています。操作不能トリガーを再作成するため、そのトリガーを明示的にドロップする必要はありません。操作不能トリガーと同じ *trigger-name* で CREATE TRIGGER ステートメントを出すと、警告とともに、その操作不能トリガーは置換されます (SQLSTATE 01595)。

作動不能トリガーであることは、SYSCAT.TRIGGERS カタログ・ビューの VALID 列が X であることによって示されます。

- **トリガー実行中のエラー:** トリガー SQL ステートメントの実行時に発生したエラーは、エラーが重大であると見なされた場合以外は SQLSTATE 09000 を使用して戻されます。重大エラーであれば、重大エラー SQLSTATE が返されます。重大エラーでない場合、SQLCA の SQLERRMC フィールドには、トリガー名、SQLCODE、SQLSTATE、および障害のあるトークンから入るだけの数のトークンが組み込まれます。

SQL-procedure-statement には SIGNAL SQLSTATE ステートメントまたは RAISE_ERROR 関数が組み込まれていることがあります。どちらの場合も、返される SQLSTATE は、SIGNAL SQLSTATE ステートメントまたは RAISE_ERROR 条件に指定されているものです。

- まだ存在していないスキーマ名を用いてトリガーを作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- データベース・マネージャーが ID 列用に生成する値は、どの BEFORE トリガーの実行よりも前に生成されます。したがって、生成される ID 値は BEFORE トリガーにとって可視の値です。
- ROW CHANGE TIMESTAMP 列用にデータベース・マネージャーが生成する値は、どの BEFORE トリガーの実行よりも後に生成されます。したがって、ROW CHANGE TIMESTAMP 値は、BEFORE トリガーにとって可視でない値です。
- 生成された式列用にデータベース・マネージャーが生成する値は、どの BEFORE トリガーの実行よりも後に生成されます。したがって、その式で生成される値は、BEFORE トリガーにとって可視でない値です。
- **読み取り専用ビュー:** ビューに INSTEAD OF トリガーを追加すると、そのビューの読み取り専用特性が影響を受けます。読み取り専用ビューが INSTEAD OF トリガーと従属関係にある場合、INSTEAD OF トリガーに定義される操作のタイプは、ビューが削除、挿入、または更新が可能かどうかを定義します。

- **遷移変数値および INSTEAD OF トリガー:** INSTEAD OF INSERT トリガーにおいて可視の新規遷移変数または新規遷移表列の初期値は、以下のように設定されます。
 - 挿入操作で列に値が明示的に指定される場合、対応する新規遷移変数は、明示的に指定されたその値となります。
 - 挿入操作で列に値が明示的に指定されない場合、または DEFAULT 節が指定される場合は、対応する新規遷移変数は、以下のようになります。
 - ビュー列が更新可能な場合 (INSTEAD OF トリガーなしで)、基礎表列のデフォルト値
 - それ以外の場合、NULL 値
- INSTEAD OF UPDATE トリガーにおいて可視の新規遷移変数の初期値は、以下のように設定されます。
- 更新操作で列に値が明示的に指定される場合、対応する新規遷移変数は、明示的に指定されたその値となります。
 - 更新操作で列に DEFAULT 節が明示的に指定される場合、対応する新規遷移変数は、以下のようになります。
 - ビュー列が更新可能な場合 (INSTEAD OF トリガーなしで)、基礎表列のデフォルト値
 - それ以外の場合、NULL 値
 - その他の場合、対応する新規遷移変数は、その行の列における既存の値となります。
- **トリガーおよび型付き表:** 表階層のどのレベルの型付き表にも、BEFORE または AFTER トリガーを付加することができます。SQL ステートメントが複数のトリガーを活動化する場合、それらのトリガーは、それぞれ型付き表階層の別々の表に付加されていても、作成順に実行されます。

トリガーが活動化されたとき、その遷移変数 (OLD、NEW、OLD TABLE、NEW TABLE) 内に副表の行が入っていることがあります。ただし、付加先の表で定義されている列しか入っていません。

INSERT、UPDATE、および DELETE ステートメントの効果は次のとおりです。

- **行トリガー:** SQL ステートメントを使って表行の INSERT、UPDATE、または DELETE を行うと、このステートメントは、行の入った最も限定的な表とその表のすべてのスーパー表に付加されている行トリガーを活動化します。SQL ステートメントがどのように表にアクセスするかに関係なく、常にこのような規則になります。例えば、UPDATE EMP コマンドを実行すると、更新済みの行の一部が、副表 MGR に入ることがあります。EMP 行の場合、EMP とそのスーパー表に付加されている行トリガーが活動化されます。MGR 行の場合、MGR とそのスーパー表に付加されている行トリガーが活動化されます。
- **ステートメント・トリガー:** INSERT、UPDATE、または DELETE ステートメントは、このステートメントによって影響を受ける可能性のある表 (およびそのスーパー表) に付加されているステートメント・トリガーを活動化します。そのような表内の実際の行が影響を受けたかどうかに関係なく、常にこのような規則になります。例えば、INSERT INTO EMP コマンドで、EMP とそのスーパー表のステートメント・トリガーを活動化します。別の例として、副表行が更新も削除もされていない場合でも、UPDATE EMP または DELETE

CREATE TRIGGER

EMP コマンドで、EMP とそのスーパー表と副表のステートメント・トリガーが活動化されます。同様に、UPDATE ONLY (EMP) または DELETE ONLY (EMP) コマンドは、EMP とそのスーパー表のステートメント・トリガーを活動化しますが、副表のステートメント・トリガーは活動化しません。

DROP TABLE ステートメントの効果: DROP TABLE ステートメントは、ドロップしようとしている表に付加されているどのトリガーも活動化しません。ただし、ドロップされる表が副表である場合、そのドロップされる表の行すべては、スーパー表から削除されるものと見なされます。したがって、表 T の場合は次のようになります。

- 行トリガー: DROP TABLE T は、T の行ごとに、T のすべてのスーパー表に付加されている行タイプの削除トリガーを活動化します。
- ステートメント・トリガー: DROP TABLE T は、T に行が入っているかどうかに関係なく、T のすべてのスーパー表に付加されているステートメント・タイプの削除トリガーを活動化します。

ビューでのアクション: ビューでのアクションによってどのトリガーが活動化されるかを予測するには、ビュー定義を使ってそのアクションを、基本表でのアクションに変換します。以下に例を示します。

1. SQL ステートメントで UPDATE V1 を実行します。V1 は、サブビュー V2 をもつ型付きビューです。V1 は基礎表 T1 をもち、V2 は基礎表 T2 をもっているとします。ステートメントは、T1、T2、およびそれらの副表内の行に影響を与える可能性があるため、T1 と T2 およびそのすべての副表とスーパー表のステートメント・トリガーが活動化されます。
 2. SQL ステートメントで UPDATE V1 を実行します。V1 は、サブビュー V2 をもつ型付きビューです。V1 は SELECT ... FROM ONLY(T1) と定義されていて、V2 は SELECT ... FROM ONLY(T2) と定義されていると仮定します。ステートメントは、T1 と T2 の副表内の行には影響を与えないため、T1 と T2 およびそれぞれのスーパー表のステートメント・トリガーは活動化されますが、これらの表の副表のものは活動化されません。
 3. SQL ステートメントで UPDATE ONLY(V1) を実行します。V1 は、SELECT ... FROM T1 と定義された型付きビューです。ステートメントは、T1 とその副表に影響を与える可能性があります。したがって、T1 とそのすべての副表とスーパー表のステートメント・トリガーが活動化されます。
 4. SQL ステートメントで UPDATE ONLY(V1) を実行します。V1 は、SELECT ... FROM ONLY(T1) と定義された型付きビューです。この場合、V1 がサブビューをもち T1 が副表をもっている場合、T1 だけがステートメントから影響を受けることができます。したがって、T1 とそのスーパー表のステートメント・トリガーが活動化されます。
- **MERGE ステートメントおよびトリガー:** MERGE ステートメントは、更新、削除、および挿入操作を実行できます。MERGE ステートメントの適用可能な UPDATE、DELETE、または INSERT トリガーは、更新、削除、または挿入操作の実行時に活動化されます。
 - **互換性**
 - 以前のバージョンの DB2 との互換性:

- OLD TABLE の代わりに OLD_TABLE、NEW TABLE の代わりに NEW_TABLE をそれぞれ指定できます。
- MODE DB2SQL は、FOR EACH ROW または FOR EACH STATEMENT の後に指定できます。

例

例 1: 会社が管理する従業員の数の自動追跡を実行する 2 つのトリガーを作成します。このトリガーは、次の表に作用します。

- EMPLOYEE 表 (列は ID、NAME、ADDRESS、および POSITION)
- COMPANY_STATS 表 (列は NBEMP、NBPRODUCT、および REVENUE)

最初のトリガーは、新しい従業員を採用するたびに (つまり EMPLOYEE 表に新しい行が挿入されるたびに)、従業員数に 1 を加算します。

```
CREATE TRIGGER NEW_HIRED
AFTER INSERT ON EMPLOYEE
FOR EACH ROW
UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

2 番目のトリガーは、従業員が会社を退職するたびに (つまり EMPLOYEE 表から行が削除されるたびに)、従業員数から 1 を減算します。

```
CREATE TRIGGER FORMER_EMP
AFTER DELETE ON EMPLOYEE
FOR EACH ROW
UPDATE COMPANY_STATS SET NBEMP = NBEMP - 1
```

例 2: 部品のレコードが更新されると、以下の検査と (必要ならば) アクションを実行するトリガーを作成します。

- 手持ち数量 (ON_HAND) が最大在庫量 (MAX_STOCKED) の 10% 未満になった場合、その部品の品目数として最大在庫量から手持ち数量を引いた数を指定した出荷依頼書を発行します。

このトリガーは、PARTNO、DESCRIPTION、ON_HAND、MAX_STOCKED、および PRICE の列を含む PARTS 表に作用します。

ISSUE_SHIP_REQUEST は、追加部品の注文書を、発注先に送るユーザー定義関数です。

```
CREATE TRIGGER REORDER
AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.ON_HAND < 0.10 * N.MAX_STOCKED)
BEGIN ATOMIC
VALUES(ISSUE_SHIP_REQUEST(N.MAX_STOCKED - N.ON_HAND, N.PARTNO));
END
```

例 3: 更新の結果、現行の給与の 10 % を超える昇給になった場合にエラーを生じさせるトリガーを作成します。

```
CREATE TRIGGER RAISE_LIMIT
AFTER UPDATE OF SALARY ON EMPLOYEE
REFERENCING NEW AS N OLD AS O
FOR EACH ROW
WHEN (N.SALARY > 1.1 * O.SALARY)
SIGNAL SQLSTATE '75000' SET MESSAGE_TEXT='Salary increase>10%'
```

CREATE TRIGGER

例 4: 株価の変更を記録し追跡するアプリケーションについて考えます。データベースには、CURRENTQUOTE および QUOTEHISTORY という 2 つの表が含まれています。

Tables: CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
QUOTEHISTORY (SYMBOL, QUOTE, QUOTE_TIMESTAMP)

CURRENTQUOTE の QUOTE (相場) 列が更新されると、新しい相場とタイム・スタンプを QUOTEHISTORY 表にコピーするようにします。CURRENTQUOTE の STATUS (状況) 列も、次のような株の状況が反映されるように更新します。

1. 値上がり
2. 今年の新高値
3. 値下がり
4. 今年の新安値
5. 変わらず

これを実現する CREATE TRIGGER ステートメントは、次のようになります。

- 状況を設定するトリガーの定義

```
CREATE TRIGGER STOCK_STATUS
NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
FOR EACH ROW
BEGIN ATOMIC
  SET NEWQUOTE.STATUS =
  CASE
    WHEN NEWQUOTE.QUOTE >
      (SELECT MAX(QUOTE) FROM QUOTEHISTORY
       WHERE SYMBOL = NEWQUOTE.SYMBOL
        AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
    THEN 'High'
    WHEN NEWQUOTE.QUOTE <
      (SELECT MIN(QUOTE) FROM QUOTEHISTORY
       WHERE SYMBOL = NEWQUOTE.SYMBOL
        AND YEAR(QUOTE_TIMESTAMP) = YEAR(CURRENT DATE) )
    THEN 'Low'
    WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
    THEN 'Rising'
    WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
    THEN 'Dropping'
    WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
    THEN 'Steady'
  END;
END;
```

- 変更内容を QUOTEHISTORY 表に記録するトリガーの定義

```
CREATE TRIGGER RECORD_HISTORY
AFTER UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE
FOR EACH ROW
BEGIN ATOMIC
  INSERT INTO QUOTEHISTORY
  VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT_TIMESTAMP);
END
```

例 5: org 表の従業員レコードのロケーション・フィールドの変更をオーバーライドするトリガーを作成します。小さな会社を買収したときに取得した新しい従業員レコードを処理して、従業員に割り振られているターゲット・ロケーションが「Toronto」の場合に、新しいターゲット・ロケーションを「Los Angeles」にする、

といった場合に、このトリガーは便利です。この BEFORE トリガーによって、アプリケーションがそのフィールドにどんな値を割り振るにしても、最終結果値を「Los Angeles」に設定できます。

```
CREATE TRIGGER LOCATION_TRIGGER
NO CASCADE
BEFORE UPDATE ON ORG
REFERENCING
  OLD AS PRE
  NEW AS POST
FOR EACH ROW
WHEN (POST.LOCATION = 'Toronto')
  SET POST.LOCATION = 'Los Angeles';
END
```

例 6 新製品の記述を含んだ XML 文書を、SAMPLE データベースの PRODUCT 表に挿入する前に、自動的に妥当性検査するための BEFORE トリガーを作成します。

```
CREATE TRIGGER NEWPROD NO CASCADE BEFORE INSERT ON PRODUCT
REFERENCING NEW AS N
FOR EACH ROW
BEGIN ATOMIC
  SET (N.DESRIPTION) = XMLVALIDATE(N.DESRIPTION
  ACCORDING TO XMLSCHEMA ID product);
END
```

CREATE TRUSTED CONTEXT

CREATE TRUSTED CONTEXT ステートメントは、現行サーバーでトラステッド・コンテキストを定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

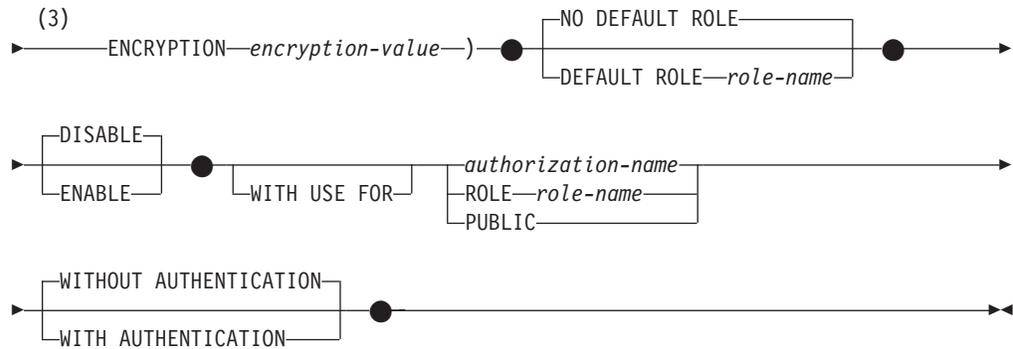
許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

構文

```
►► CREATE TRUSTED CONTEXT context-name BASED UPON CONNECTION USING address-value
► SYSTEM AUTHID authorization-name ATTRIBUTES ( address-value )
► ADDRESS address-value [ WITH ENCRYPTION encryption-value ]
```

CREATE TRUSTED CONTEXT



注:

- 1 ATTRIBUTES、DEFAULT ROLE、ENABLE、および WITH USE 節はそれぞれ 1 度しか指定できません (SQLSTATE 42614)。
- 2 属性名およびそれに対応する値はそれぞれ固有でなければなりません (SQLSTATE 4274D)。
- 3 ENCRYPTION を複数回指定することはできませんが (SQLSTATE 42614)、WITH ENCRYPTION は指定されている ADDRESS ごとに指定できます。

説明

context-name

トラステッド・コンテキストの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。名前は、現行のサーバーに既に存在するトラステッド・コンテキストを識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

BASED UPON CONNECTION USING SYSTEM AUTHID *authorization-name*

コンテキストがシステム許可 ID *authorization-name* によって確立される接続であることを指定します。これを既存のトラステッド・コンテキストと関連付けてはなりません (SQLSTATE 428GL)。これにステートメントの許可 ID を指定することはできません (SQLSTATE 42502)。

ATTRIBUTES (...)

トラステッド・コンテキストが定義される 1 つ以上の接続トラスト属性のリストを指定します。

ADDRESS *address-value*

クライアントがデータベース・サーバーと通信するために使用する実際の通信アドレスを指定します。サポートされるプロトコルは TCP/IP のみです。ADDRESS 属性は複数回指定できますが、*address-value* の対はそれぞれ属性のセットで固有でなければなりません (SQLSTATE 4274D)。

トラステッド接続を確立するときにトラステッド・コンテキストの ADDRESS 属性に対して複数の値が定義されている場合、候補となる接続によって使用されるアドレスがトラステッド・コンテキストの ADDRESS 属性の定義値のいずれかと一致していると、その接続はこの属性と一致しているとみなされます。

address-value

ADDRESS トラスト属性と関連付けられる値を含むストリング定数を指定します。 *address-value* は、IPv4 アドレス、IPv6 アドレス、またはセキュア・ドメイン名でなければなりません。

- IPv4 アドレスの先頭にスペースが含まれてはなりません。このアドレスは小数点付き 10 進数アドレスとして表されます。例えば IPv4 アドレスは 9.112.46.111 のようになります。値 'localhost' またはそれに相当する表現 '127.0.0.1' は、一致という結果になりません。代わりにホストの実 IPv4 アドレスを指定する必要があります。
- IPv6 アドレスの先頭にスペースが含まれてはなりません。このアドレスはコロン区切りの 16 進アドレスとして表されます。例えば IPv6 アドレスは 2001:0DB8:0000:0000:0008:0800:200C:417A のようになります。IPv4 がマップされた IPv6 アドレス (例えば ::ffff:192.0.2.128) は、一致という結果になりません。同じように、'localhost' またはその IPv6 短表現 '::1' も一致という結果になりません。
- ドメイン・ネームはドメイン・ネーム・サーバーで IP アドレスに変換されます。結果として生成される IPv4 または IPv6 アドレスはこのサーバーで決定されます。例えばドメイン・ネームは corona.torolab.ibm.com のようになります。ドメイン・ネームが IP アドレスに変換されたとき、この変換の結果が 1 つ以上の IP アドレスのセットになる場合があります。その場合、接続開始時の IP アドレスがドメイン名変換後の IP アドレスのいずれかと一致すると、着信接続はトラステッド・コンテキスト・オブジェクトの ADDRESS 属性と一致しているとみなされます。トラステッド・コンテキスト・オブジェクトを作成するとき、特に動的ホスト構成プロトコル (DHCP) 環境では、静的 IP アドレスの代わりにドメイン・ネーム値を ADDRESS 属性に提供することをお勧めします。DHCP ではデバイスがネットワークと接続するたびに IP アドレスが変わります。そのため、トラステッド・コンテキスト・オブジェクトの ADDRESS 属性に静的 IP アドレスを提供すると、デバイスによっては意図せずにトラステッド接続を取得してしまう場合があります。トラステッド・コンテキスト・オブジェクトの ADDRESS 属性にドメイン・ネームを指定すると、DHCP 環境におけるこの問題を回避できます。

WITH ENCRYPTION *encryption-value*

この特定の *address-value* に関するデータ・ストリームまたはネットワーク暗号化の最小暗号化レベルを指定します。この *encryption-value* は、この特定の *address-value* に関するグローバル ENCRYPTION 属性の設定をオーバーライドします。

encryption-value

この特定の *address-value* に関する ENCRYPTION トラスト属性と関連付けられる値を含むストリング定数を指定します。*encryption-value* は以下のいずれかでなければなりません (SQLSTATE 42615)。

- NONE。特定レベルの暗号化は不要です。

CREATE TRUSTED CONTEXT

- **LOW**。最小の低レベルの暗号化が必要です。着信接続がこの特定アドレスの暗号化設定と一致する場合、データベース・マネージャーの認証タイプは `DATA_ENCRYPT` でなければなりません。
- **HIGH**。着信接続がこの特定アドレスの暗号化設定と一致する場合、DB2 クライアントと DB2 サーバーの間のデータ通信に関して Secure Sockets Layer (SSL) 暗号化を使用する必要があります。

ENCRYPTION *encryption-value*

データ・ストリームまたはネットワーク暗号化の最小暗号化レベルを指定します。デフォルトは `NONE` です。

encryption-value

この特定の *address-value* に関する ENCRYPTION トラスト属性と関連付けられる値を含むストリング定数を指定します。 *encryption-value* は以下のいずれかでなければなりません (SQLSTATE 42615)。

- **NONE**。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、特定レベルの暗号化は不要です。
- **LOW**。最小の低レベルの暗号化が必要です。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、データベース・マネージャーの認証タイプは `DATA_ENCRYPT` でなければなりません。
- **HIGH**。着信接続がこのトラステッド・コンテキスト・オブジェクトの ENCRYPTION 属性と一致する場合、DB2 クライアントと DB2 サーバーの間のデータ通信に関して Secure Sockets Layer (SSL) 暗号化を使用する必要があります。

以下の表は、既存の接続で使用される暗号化に応じたトラステッド・コンテキストの使用可能性について要約しています。接続でトラステッド・コンテキストを使用できない場合は警告が戻され (SQLSTATE 01679)、SQLCA の SQLWARN8 フィールドが 'Y' に設定されます。これは接続が通常の (非トラステッド) 接続であることを示します。

表 24. 暗号化とトラステッド・コンテキスト

既存の接続で使用される暗号化	トラステッド・コンテキストの ENCRYPTION 値	接続でトラステッド・コンテキストを使用できるかどうか
暗号化なし	'NONE'	はい
暗号化なし	'LOW'	いいえ
暗号化なし	'HIGH'	いいえ
低レベルの暗号化 (DATA_ENCRYPT)	'NONE'	はい
低レベルの暗号化 (DATA_ENCRYPT)	'LOW'	はい
低レベルの暗号化 (DATA_ENCRYPT)	'HIGH'	いいえ

表 24. 暗号化とトラステッド・コンテキスト (続き)

既存の接続で使用される暗号化	トラステッド・コンテキストの ENCRYPTION 値	接続でトラステッド・コンテキストを使用できるかどうか
高レベルの暗号化 (SSL)	'NONE'	はい
高レベルの暗号化 (SSL)	'LOW'	はい
高レベルの暗号化 (SSL)	'HIGH'	はい

NO DEFAULT ROLE または DEFAULT ROLE *role-name*

デフォルトのロールをこのトラステッド・コンテキストに基づくトラステッド接続と関連付けるかどうかを指定します。デフォルトは NO DEFAULT ROLE です。

NO DEFAULT ROLE

トラステッド・コンテキストがデフォルトのロールを持たないことを指定します。

DEFAULT ROLE *role-name*

role-name がトラステッド・コンテキストのデフォルトのロールであることを指定します。 *role-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。トラステッド・コンテキストの定義の一部としてユーザー固有のロールがユーザーに定義されていない場合、このトラステッド・コンテキストに基づいて、トラステッド接続の中でこのロールがそのユーザーに使用されます。

DISABLE または ENABLE

トラステッド・コンテキストを使用可能の状態で作成するか、または使用不可の状態で作成するかを指定します。デフォルトは DISABLE です。

DISABLE

トラステッド・コンテキストを使用不可の状態で作成することを指定します。トラステッド接続を確立するとき、使用不可のトラステッド・コンテキストは考慮されません。

ENABLE

トラステッド・コンテキストを使用可能の状態で作成することを指定します。

WITH USE FOR

このトラステッド・コンテキストに基づくトラステッド接続を使用できるユーザーを指定します。

authorization-name

指定された *authorization-name* でトラステッド接続を使用できることを指定します。 WITH USE FOR 節内に *authorization-name* を複数回指定することはできません (SQLSTATE 428GM)。これにステートメントの許可 ID を指定することもできません (SQLSTATE 42502)。トラステッド・コンテキストの定義で PUBLIC とユーザーのリストの両方からのアクセスが許可されている場合、ユーザーの指定が PUBLIC の指定をオーバーライドします。例えば、トラステッド・コンテキストの定義で PUBLIC WITH AUTHENTICATION と JOE WITHOUT AUTHENTICATION 両方のアクセスが許可されているとします。トラステッド・コンテキストが JOE によ

CREATE TRUSTED CONTEXT

て使用される場合、認証は不要です。しかし、トラステッド・コンテキストが GEORGE によって使用される場合は認証が必要になります。

ROLE *role-name*

トラステッド接続がトラステッド・コンテキストを使用しているときに使用されるユーザーのロールが *role-name* であることを指定します。*role-name* は現行のサーバーに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。ユーザーに対して明示的に指定されたロールは、トラステッド・コンテキストと関連付けられているすべてのデフォルトのロールをオーバーライドします。

PUBLIC

すべてのユーザーがこのトラステッド・コンテキストに基づくトラステッド接続を使用できることを指定します。PUBLIC を複数回指定することはできません (SQLSTATE 428GM)。そのようなトラステッド接続を使用するユーザーはすべて、関連したトラステッド・コンテキストについてデフォルトのロールと関連付けられている特権を使用します。トラステッド・コンテキストでデフォルトのロールが定義されない場合、このトラステッド・コンテキストに基づくトラステッド接続を使用するユーザーにロールが関連付けられません。

WITHOUT AUTHENTICATION または **WITH AUTHENTICATION**

トラステッド接続でユーザーを切り替えるときにユーザーの認証が必要かどうかを指定します。デフォルトは WITHOUT AUTHENTICATION です。

WITHOUT AUTHENTICATION

トラステッド接続で現行のユーザーをこのユーザーに切り替えるときに認証を必要としないことを指定します。

WITH AUTHENTICATION

トラステッド接続で現行のユーザーをこのユーザーに切り替えるときに認証を必要とすることを指定します。

規則

- トラステッド・コンテキスト排他 SQL ステートメントの後は、COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。トラステッド・コンテキスト排他 SQL ステートメントは次のとおりです。
 - CREATE TRUSTED CONTEXT、ALTER TRUSTED CONTEXT、または DROP (TRUSTED CONTEXT)
- グローバル・トランザクション内でトラステッド・コンテキスト排他 SQL ステートメントを発行することはできません。例えば、フェデレーテッド・トランザクションにおける 2 フェーズ・コミットの一部として開始されるグローバル・トランザクションまたは XA トランザクションなどの場合です (SQLSTATE 51041)。

注

- トラステッド・コンテキスト定義の一部として IP アドレスを提供するとき、そのアドレスの形式はネットワークで有効なものでなければなりません。例えば、ネットワークが IPv4 であるのに IPv6 形式のアドレスを提供しても一致には至りません。混合環境では、IPv4 と IPv6 両方のアドレス表現を指定するとよいで

しょう。さらに望ましいのは、セキュア・ドメイン・ネーム (例えば `corona.torolab.ibm.com`) を指定することです。その場合、アドレス・フォーマットの詳細が非表示になるからです。

- **トラステッド・コンテキストの定義にロールを指定する:** トラステッド・コンテキストの定義では、特定の許可 ID のロールと、トラステッド・コンテキストの定義で特定のロールが指定されていない許可 ID で使用されるデフォルトのロールを指定することができます。このロールはトラステッド・コンテキストに基づくトラステッド接続では使用できますが、その接続外では使用できません。
- トラステッド接続を使用してデータ操作言語 (DML) SQL ステートメントを発行すると、関連トラステッド・コンテキストの定義内の許可 ID で有効なコンテキストが割り当てられたロールが保持する特権は、ステートメントの許可 ID によって直接保持される、またはステートメントの許可 ID が保持するその他のロールによって間接的に保持されるその他の特権に加えて考慮されます。
- データ定義言語 (DDL) SQL ステートメントでは、関連トラステッド・コンテキストの定義内の許可 ID で有効なコンテキストが割り当てられたロールが保持する特権は考慮されません。例えば、オブジェクトを作成する場合は、ステートメントの許可 ID はコンテキストに割り当てられたロールが保持する特権を組み込まずに作成することができなければなりません。
- 同じマシン上で既存のアプリケーションと同じクレデンシャルを使用して DB2 を認証する新規アプリケーションをインストールし、それがトラステッド・コンテキストを利用する場合、新規アプリケーションも同じトラステッド・コンテキスト・オブジェクトを利用する可能性があります (例えばトラステッド・コンテキストのロールを継承するなど)。これはセキュリティー管理者の意図ではない可能性があります。セキュリティー管理者は DB2 監査機能をオンにして、トラステッド・コンテキスト・オブジェクトを利用しているアプリケーションを見つけ出すこともできます。
- データベース・パーティション全体を通じて、同時に実行できる非コミットのトラステッド・コンテキスト排他 SQL ステートメントは 1 つのみです。非コミットのトラステッド・コンテキスト排他 SQL ステートメントが実行されている場合、後続のトラステッド・コンテキスト排他 SQL ステートメントは、現行のトラステッド・コンテキスト排他 SQL ステートメントがコミットまたはロールバックされるまで待機します。
- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

例 1: このトラステッド・コンテキストに基づくトラステッド接続の現行ユーザーを 2 つの異なるユーザー ID に切り替えられるようにトラステッド・コンテキストを作成します。接続の現行ユーザーがユーザー ID JOE に切り替えられる場合には認証を必要としません。しかし、接続の現行ユーザーがユーザー ID BOB に切り替えられる場合には認証が必要になります。トラステッド・コンテキストは `context-role` というデフォルトのロールを持ちます。これは、このトラステッド・コンテキストの領域内で作業しているユーザーがロール `context-role` と関連付けられている特権を継承することを暗黙に示します。

```
CREATE TRUSTED CONTEXT APPSERVER
  BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
  DEFAULT ROLE CONTEXT_ROLE
```

CREATE TRUSTED CONTEXT

```
ENABLE
ATTRIBUTES (ADDRESS '9.26.113.204')
WITH USE FOR JOE WITHOUT AUTHENTICATION
BOB WITH AUTHENTICATION
```

例 2: このトラステッド・コンテキストに基づくトラステッド接続の現行ユーザーを、認証なしで任意のユーザー ID に切り替えられるようにトラステッド・コンテキストを作成します。

```
CREATE TRUSTED CONTEXT SECUREROLE
BASED UPON CONNECTION USING SYSTEM AUTHID PBIRD
ENABLE
ATTRIBUTES (ADDRESS '9.26.113.204')
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

例 3: このトラステッド・コンテキストに基づくトラステッド接続の現行ユーザーを、認証なしで任意のユーザー ID に切り替えられるようにトラステッド・コンテキストを作成します。このトラステッド・コンテキストと例 2 で作成したトラステッド・コンテキストの違いは、このトラステッド・コンテキストには ENCRYPTION という追加属性があるという点です。トラステッド・コンテキスト SECUREROLEENCRYPT の ENCRYPTION 属性設定で、このトラステッド・コンテキスト属性と一致するためには接続に使用される暗号化設定が少なくとも「低レベルの暗号化」(586 ページの表 24 を参照) でなければならないことが明示されています。

```
CREATE TRUSTED CONTEXT SECUREROLEENCRYPT
BASED UPON CONNECTION USING SYSTEM AUTHID SHARPER
ENABLE
ATTRIBUTES (ADDRESS '9.26.113.204'
ENCRYPTION 'LOW')
WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
```

例 4: アドレス 9.26.146.201 および 9.26.146.203 からユーザー WRJAIBI によって確立される接続は暗号化が使用されていないときにトラステッド接続となり、アドレス 9.26.146.202 からユーザー WRJAIBI によって確立される接続ではトラステッド接続となるために LOW レベルの暗号化を必要とするようなトラステッド・コンテキストを作成します。

```
CREATE TRUSTED CONTEXT WALIDLOCSENSITIVE
BASED UPON CONNECTION USING SYSTEM AUTHID WRJAIBI
ENABLE
ATTRIBUTES (ADDRESS '9.26.146.201',
ADDRESS '9.26.146.202' WITH ENCRYPTION 'LOW',
ADDRESS '9.26.146.203'
ENCRYPTION 'NONE')
```

CREATE TYPE (配列)

CREATE TYPE (配列) ステートメントは配列タイプを定義します。配列タイプは、組み込みデータ・タイプのいずれかに基づいています。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- データベースに対する IMPLICIT_SCHEMA 権限 (配列タイプのスキーマ名が既存のスキーマを指していない場合)
- スキーマに対する CREATE_IN 特権 (配列タイプのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

構文

```

▶▶ CREATE TYPE type-name AS data-type [ARRAY [ integer-constant ] ]
▶]

```

説明

type-name

タイプの名前を指定します。名前 (暗黙または明示の修飾子を含む) は、現行サーバーに既に存在するその他のタイプ (組み込みタイプまたはユーザー定義タイプ) と同じであってはなりません。非修飾名は、組み込みデータ・タイプ、または BOOLEAN、BINARY、または VARBINARY と同一の名前であってはなりません (SQLSTATE 42918)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*type-name* として使用することはできません (SQLSTATE 42939)。これらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

2 つの部分からなる *type-name* を指定する場合、文字 'SYS' で始まるスキーマ名は使用してはなりません (SQLSTATE 42939)。

data-type

配列エレメントのデータ・タイプを指定します。これは、『CREATE TABLE』で記述されているデータ・タイプの 1 つで、LONG VARCHAR、LONG VARGRAPHIC、REFERENCE、SYSPROC.DB2SECURITYLABEL、XML、またはユーザー定義タイプ以外のものです (SQLSTATE 429C2)。

ARRAY [*integer-constant*]

タイプが *integer-constant* の最大カーディナリティーを持つ配列であることを指定します。値はゼロより大きく、最も大きい正整数値より小さい正数でなければなりません (SQLSTATE 42820)。デフォルトは最も大きい正整数値 (2 147 483 647) です。

特定のシステムにおける配列のカーディナリティーは、DB2 アプリケーションが使用可能な合計メモリー量により制限されます。そのため、大きなカーディナリティーの配列を作成することはできませんが、すべてのエレメントを使用できるとは限りません。

CREATE TYPE (配列)

例

例 1: エレメント数が最大 5 でデータ・タイプが DECIMAL(10,0) である PHONENUMBERS という名前の配列タイプを作成します。

```
CREATE TYPE PHONENUMBERS AS DECIMAL(10,0)
ARRAY[5]
```

例 2: スキーマ GENERIC にデフォルトの数のエレメントを持つ、NUMBERS という名前の配列タイプを作成します。

```
CREATE TYPE GENERIC.NUMBERS AS BIGINT
ARRAY[]
```

CREATE TYPE (特殊)

CREATE TYPE (特殊) ステートメントは、特殊タイプを定義します。特殊タイプは、常に組み込みデータ・タイプのいずれかに基づいています。このステートメントの正常な実行により、該当の特殊タイプとそのソース・タイプとの間をキャストする関数も生成され、また必要に応じてその特殊タイプで使用する比較演算子 (=、<>、<、<=、>、および >=) に対するサポートが生成されます。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID には、以下の特権が少なくとも 1 つ含まれている必要があります。

- データベースに対する IMPLICIT_SCHEMA 権限 (特殊タイプのスキーマ名が既存のスキーマを指していない場合)
- スキーマに対する CREATEIN 特権 (特殊タイプのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

構文

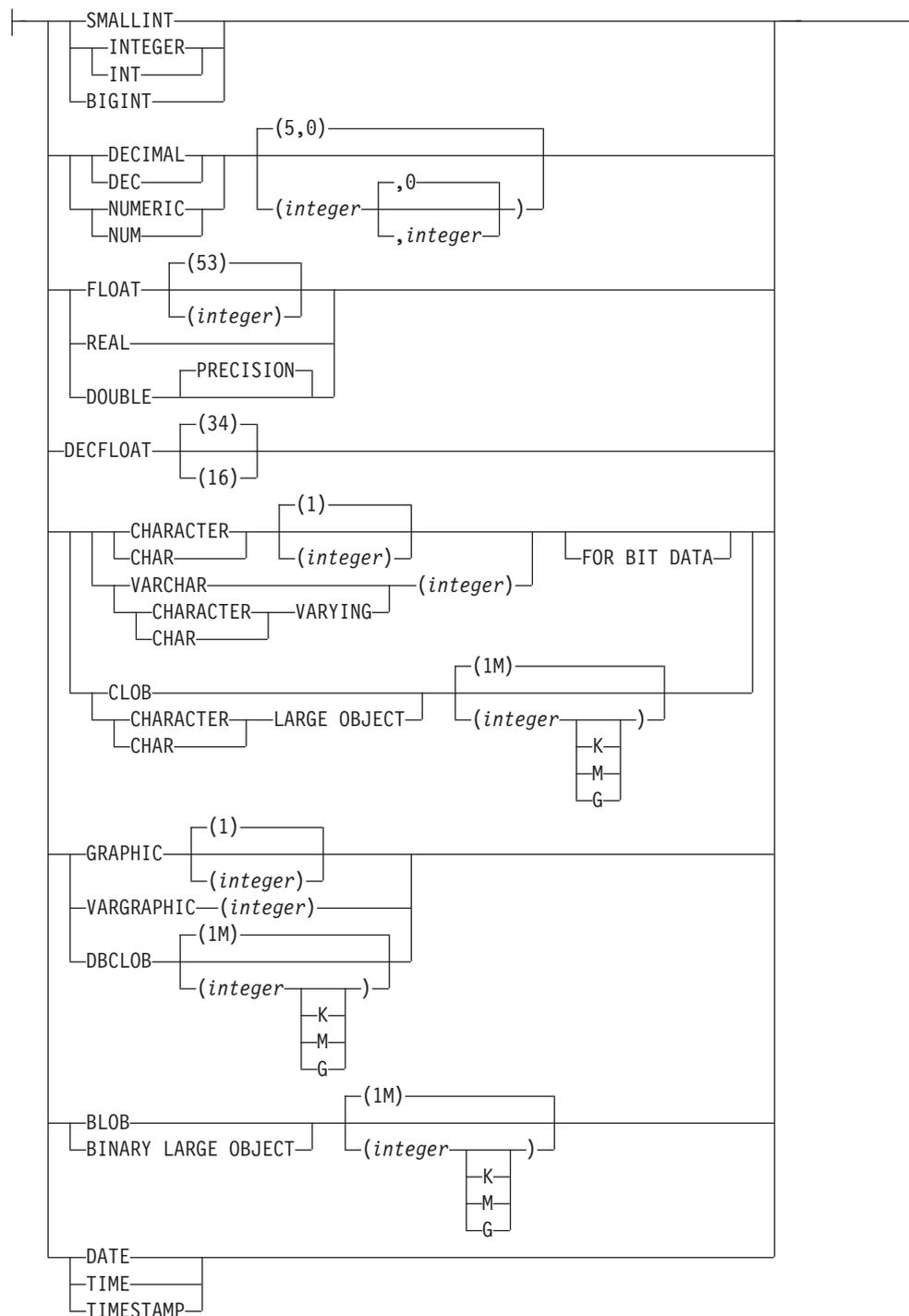
```
▶▶ CREATE TYPE distinct-type-name AS _____▶▶
```

```
▶▶ | source-data-type | _____ (1) WITH COMPARISONS _____▶▶
```

source-data-type:

```
▶▶ | built-in-type | _____▶▶
```

built-in-type:



注:

- 1 すべてのソース・データ・タイプに必須。ただし、比較がサポートされていない LOB は除きます。

CREATE TYPE (特殊)

説明

distinct-type-name

特殊タイプの名前を指定します。名前 (暗黙または明示の修飾子を含む) は、現行サーバーに既に存在するその他のタイプ (組み込みタイプまたはユーザー定義タイプ) と同じであってはなりません。非修飾名は、組み込みデータ・タイプ、または BOOLEAN、BINARY、または VARBINARY と同一の名前であってはなりません (SQLSTATE 42918)。非修飾名は、ARRAY、INTERVAL、または ROWID にするべきでもありません。

動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*distinct-type-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

2 つの部分からなる *distinct-type-name* を指定する場合、文字 'SYS' で始まるスキーマ名は使用してはなりません (SQLSTATE 42939)。

source-data-type

特殊タイプの内部表示のベースとして使用されるデータ・タイプを指定します。データ・タイプは、組み込みデータ・タイプでなければなりません。組み込みデータ・タイプの詳細については、『CREATE TABLE』を参照してください。ソース・データ・タイプをタイプ XML または ARRAY にすることはできません (SQLSTATE 42601)。プラットフォーム間でアプリケーションの移植性を確保するために、以下のデータ・タイプ名を使用することをお勧めします。

- FLOAT ではなく DOUBLE または REAL
- NUMERIC ではなく DECIMAL
- LONG VARCHAR ではなく VARCHAR、BLOB、または CLOB
- LONG VARGRAPHIC ではなく VARGRAPHIC または DBCLOB

WITH COMPARISONS

特殊タイプの 2 つのインスタンスを比較するシステム生成の比較演算子を作成することを指定します。ソース・データ・タイプが BLOB、CLOB、または DBCLOB の場合、これらのキーワードは指定できません。指定した場合には、警告 (SQLSTATE 01596) が出され、比較演算子は生成されません。それ以外のソース・データ・タイプの場合、WITH COMPARISONS キーワードは必須です。

注

• 特権

ユーザー定義タイプの定義者は、特殊タイプに関して自動的に生成されるすべての関数で、EXECUTE 特権 WITH GRANT OPTION を必ず与えられます。

CREATE TYPE (特殊) ステートメントの間に自動的に生成されるすべての関数での EXECUTE 特権は、PUBLIC に与えられます。

- まだ存在していないスキーマ名を用いて特殊タイプを作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- ソース・タイプとの間のキャストに必要な次の関数が生成されます。
 - 特殊タイプをソース・タイプに変換する関数
 - ソース・タイプを特殊タイプに変換する関数
 - ソース・タイプが SMALLINT の場合、INTEGER から特殊タイプに変換する関数
 - ソース・タイプが CHAR の場合、VARCHAR から特殊タイプに変換する関数
 - ソース・タイプが GRAPHIC の場合、VARGRAPHIC から特殊タイプに変換する関数

一般に、これらの関数の形式は次のようになります。

```
CREATE FUNCTION source-type-name (distinct-type-name)
  RETURNS source-type-name ...
```

```
CREATE FUNCTION distinct-type-name (source-type-name)
  RETURNS distinct-type-name ...
```

ソース・タイプがパラメーター化タイプである場合、特殊タイプをソース・タイプに変換する関数の関数名は、パラメーターなしのソース・タイプの名前になります(詳細については、596 ページの表 25 を参照)。この関数の戻り値のタイプには、CREATE TYPE (特殊) ステートメントに指定されたパラメーターが含まれます。ソース・タイプを特殊タイプに変換するための関数の入力パラメーターは、そのパラメーターを含むソース・タイプになります。例えば、次のようにします。

```
CREATE TYPE T_SHOESIZE AS CHAR(2)
  WITH COMPARISONS
```

```
CREATE TYPE T_MILES AS DOUBLE
  WITH COMPARISONS
```

上記の指定により、次の関数が生成されます。

```
FUNCTION CHAR (T_SHOESIZE) RETURNS CHAR (2)
```

```
FUNCTION T_SHOESIZE (CHAR (2))
  RETURNS T_SHOESIZE
```

```
FUNCTION DOUBLE (T_MILES) RETURNS DOUBLE
```

```
FUNCTION T_MILES (DOUBLE) RETURNS T_MILES
```

生成された cast 関数のスキーマは、特殊タイプのスキーマと同じです。この名前と同じ名前でもシグニチャーも同じ他の関数が、データベースにすでに存在してはなりません (SQLSTATE 42710)。

次の表は、事前定義されているすべてのデータ・タイプについて、特殊タイプをソース・タイプに変換する関数、およびソース・タイプを特殊タイプに変換する関数の名前を示しています。

CREATE TYPE (特殊)

表 25. 特殊タイプに対する CAST 関数

ソース・タイプ名	関数名	パラメーター	戻りタイプ
CHAR	<i>distinct-type-name</i>	CHAR (<i>n</i>)	<i>distinct-type-name</i>
	CHAR	<i>distinct-type-name</i>	CHAR (<i>n</i>)
VARCHAR	<i>distinct-type-name</i>	VARCHAR (<i>n</i>)	<i>distinct-type-name</i>
	VARCHAR	<i>distinct-type-name</i>	VARCHAR (<i>n</i>)
CLOB	<i>distinct-type-name</i>	CLOB (<i>n</i>)	<i>distinct-type-name</i>
	CLOB	<i>distinct-type-name</i>	CLOB (<i>n</i>)
BLOB	<i>distinct-type-name</i>	BLOB (<i>n</i>)	<i>distinct-type-name</i>
	BLOB	<i>distinct-type-name</i>	BLOB (<i>n</i>)
GRAPHIC	<i>distinct-type-name</i>	GRAPHIC (<i>n</i>)	<i>distinct-type-name</i>
	GRAPHIC	<i>distinct-type-name</i>	GRAPHIC (<i>n</i>)
	<i>distinct-type-name</i>	VARGRAPHIC (<i>n</i>)	<i>distinct-type-name</i>
VARGRAPHIC	<i>distinct-type-name</i>	VARGRAPHIC (<i>n</i>)	<i>distinct-type-name</i>
	VARGRAPHIC	<i>distinct-type-name</i>	VARGRAPHIC (<i>n</i>)
DBCLOB	<i>distinct-type-name</i>	DBCLOB (<i>n</i>)	<i>distinct-type-name</i>
	DBCLOB	<i>distinct-type-name</i>	DBCLOB (<i>n</i>)
SMALLINT	<i>distinct-type-name</i>	SMALLINT	<i>distinct-type-name</i>
	<i>distinct-type-name</i>	INTEGER	<i>distinct-type-name</i>
	SMALLINT	<i>distinct-type-name</i>	SMALLINT
INTEGER	<i>distinct-type-name</i>	INTEGER	<i>distinct-type-name</i>
	INTEGER	<i>distinct-type-name</i>	INTEGER
BIGINT	<i>distinct-type-name</i>	BIGINT	<i>distinct-type-name</i>
	BIGINT	<i>distinct-type-name</i>	BIGINT
DECIMAL	<i>distinct-type-name</i>	DECIMAL (<i>p,s</i>)	<i>distinct-type-name</i>
	DECIMAL	<i>distinct-type-name</i>	DECIMAL (<i>p,s</i>)
NUMERIC	<i>distinct-type-name</i>	DECIMAL (<i>p,s</i>)	<i>distinct-type-name</i>
	DECIMAL	<i>distinct-type-name</i>	DECIMAL (<i>p,s</i>)
REAL	<i>distinct-type-name</i>	REAL	<i>distinct-type-name</i>
	<i>distinct-type-name</i>	DOUBLE	<i>distinct-type-name</i>
	REAL	<i>distinct-type-name</i>	REAL
FLOAT(<i>n</i>) ただし <i>n</i> ≤ 24	<i>distinct-type-name</i>	REAL	<i>distinct-type-name</i>
	<i>distinct-type-name</i>	DOUBLE	<i>distinct-type-name</i>
	REAL	<i>distinct-type-name</i>	REAL
FLOAT(<i>n</i>) ただし <i>n</i> > 24	<i>distinct-type-name</i>	DOUBLE	<i>distinct-type-name</i>
	DOUBLE	<i>distinct-type-name</i>	DOUBLE
FLOAT	<i>distinct-type-name</i>	DOUBLE	<i>distinct-type-name</i>
	DOUBLE	<i>distinct-type-name</i>	DOUBLE
DOUBLE	<i>distinct-type-name</i>	DOUBLE	<i>distinct-type-name</i>
	DOUBLE	<i>distinct-type-name</i>	DOUBLE

表 25. 特殊タイプに対する CAST 関数 (続き)

ソース・タイプ名	関数名	パラメーター	戻りタイプ
DOUBLE PRECISION	<i>distinct-type-name</i>	DOUBLE	<i>distinct-type-name</i>
	DOUBLE	<i>distinct-type-name</i>	DOUBLE
DECFLOAT	<i>distinct-type-name</i>	DECFLOAT(<i>n</i>)	<i>distinct-type-name</i>
	DECFLOAT	<i>distinct-type-name</i>	DECFLOAT(<i>n</i>)
DATE	<i>distinct-type-name</i>	DATE	<i>distinct-type-name</i>
	DATE	<i>distinct-type-name</i>	DATE
TIME	<i>distinct-type-name</i>	TIME	<i>distinct-type-name</i>
	TIME	<i>distinct-type-name</i>	TIME
TIMESTAMP	<i>distinct-type-name</i>	TIMESTAMP	<i>distinct-type-name</i>
	TIMESTAMP	<i>distinct-type-name</i>	TIMESTAMP

注: NUMERIC および FLOAT は、移植可能アプリケーションのユーザー定義タイプを作成する場合にはお勧めできません。代わりに DECIMAL および DOUBLE を使用してください。

上記の表には、特殊タイプが定義されている場合に自動的に生成される関数だけを示しています。したがって、CREATE FUNCTION ステートメントを使用して、特殊タイプに対応するユーザー定義関数を登録し、それらのユーザー定義関数を適切な組み込み関数に基づくものにしてからでなければ、どの組み込み関数 (AVG、MAX、LENGTH など) も、特殊タイプに対してサポートされません。特に、組み込み列関数に基づくユーザー定義関数を登録することが可能である点に注意してください。

WITH COMPARISONS 節を使用して特殊タイプが作成された場合、システム生成の比較演算子が作成されます。これらの比較演算子の作成により、SYSCAT.ROUTINES カタログ・ビューに新しい関数としての項目が生成されます。

これらの演算子や cast 関数を SQL ステートメントで正しく使用するには、SQL パスに特殊タイプのスキーマ名が含まれていなければなりません。または FUNCPATH BIND オプションを参照してください。

• 互換性

- 以前のバージョンの DB2 との互換性:
 - CREATE TYPE の代わりに CREATE DISTINCT TYPE を指定できます。
 - LONG VARCHAR と LONG VARGRAPHIC データ・タイプ、および cast 関数は、サポートされていますが推奨されておらず、将来のリリースでは除去される可能性があります。WITH COMPARISONS 節は、引き続き LONG VARCHAR および LONG VARGRAPHIC データ・タイプをサポートしません。

例

例 1: INTEGER データ・タイプに基づく、SHOESIZE という名前の特殊タイプを作成します。

```
CREATE TYPE SHOESIZE AS INTEGER WITH COMPARISONS
```

CREATE TYPE (特殊)

またこの結果、比較演算子 (=、<>、<、<=、>、>=)、INTEGER を戻す cast 関数 INTEGER(SHOESIZE)、および SHOESIZE を戻す cast 関数 SHOESIZE(INTEGER) が作成されます。

例 2: DOUBLE データ・タイプに基づく、MILES という名前の特殊タイプを作成します。

```
CREATE TYPE MILES AS DOUBLE WITH COMPARISONS
```

またこの結果、比較演算子 (=、<>、<、=、>、>=)、DOUBLE を戻す cast 関数 DOUBLE(MILES)、および MILES を戻す cast 関数 MILES(DOUBLE) が作成されます。

CREATE TYPE (構造化)

CREATE TYPE ステートメントは、ユーザー定義の構造化タイプを定義します。ユーザー定義構造化タイプには、属性を含めないこともできますし、複数の属性を含めることもできます。構造化タイプには、スーパータイプからの属性を継承するサブタイプを指定することができます。ステートメントの実行が正常に完了すると、属性値の検索と更新のためのメソッドが生成されます。また、このステートメントの実行が正常に完了すると、列内で使用する構造化タイプのインスタンスを作成する関数と、該当の参照タイプとその表示タイプとをキャストする関数、およびその参照タイプ上の比較演算子 (=、<>、<、<=、>、および >=) をサポートする関数も生成されます。

また、CREATE TYPE ステートメントは、ユーザー定義構造化タイプと一緒に使用されるユーザー定義メソッドの任意のメソッド仕様も定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

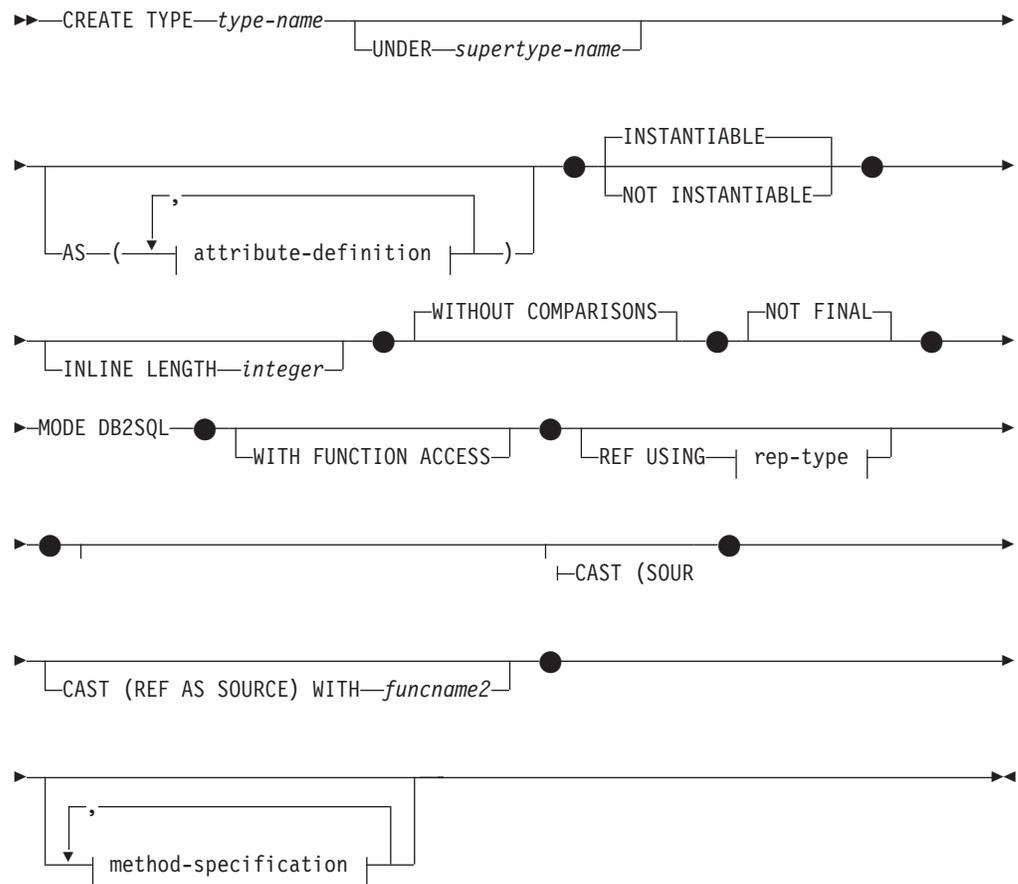
許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

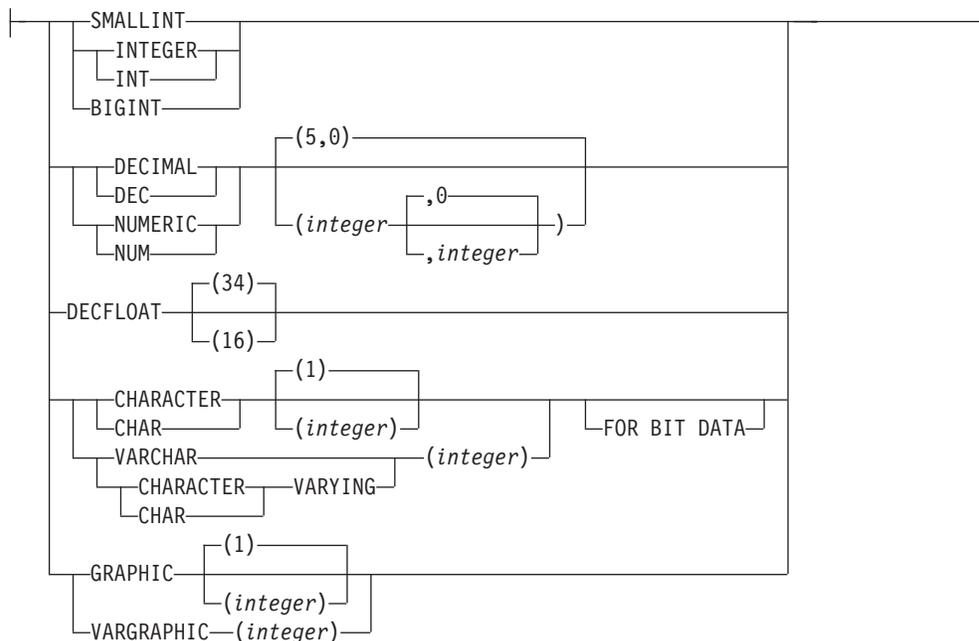
- データベースに対する IMPLICIT_SCHEMA 権限 (このタイプのスキーマ名が既存のスキーマを指していない場合)
- スキーマに対する CREATEIN 特権 (タイプのスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

UNDER が指定されていて、このステートメントの許可 ID がタイプ階層のルート・タイプの所有者と同じではない場合には、SYSADM または DBADM 権限が必要です。

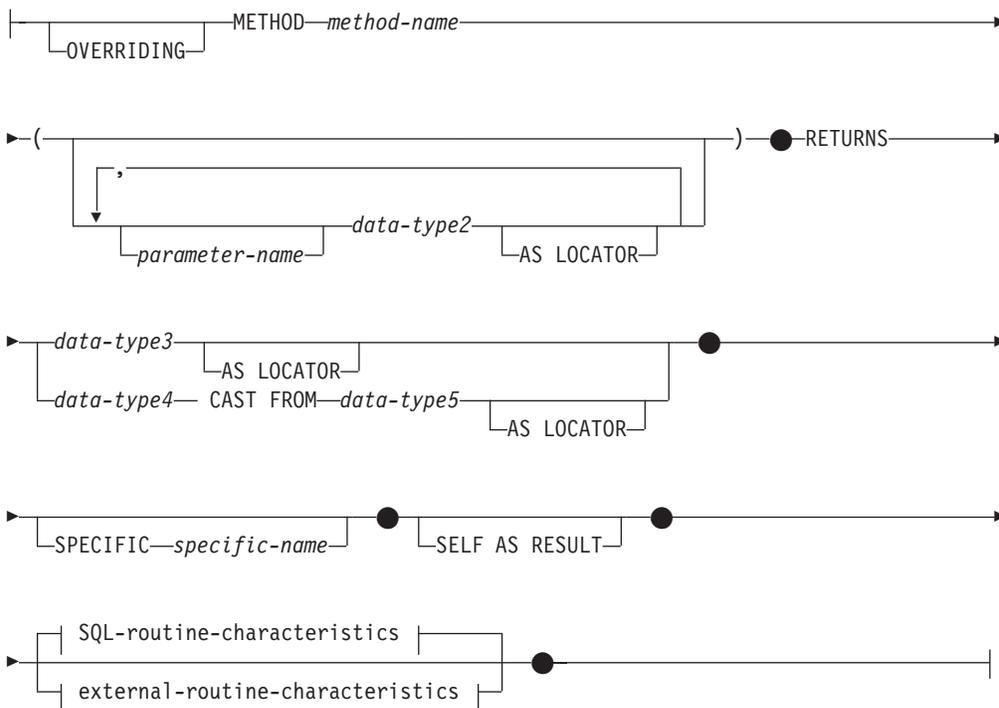
構文

**attribute-definition:****rep-type:**

CREATE TYPE (構造化)



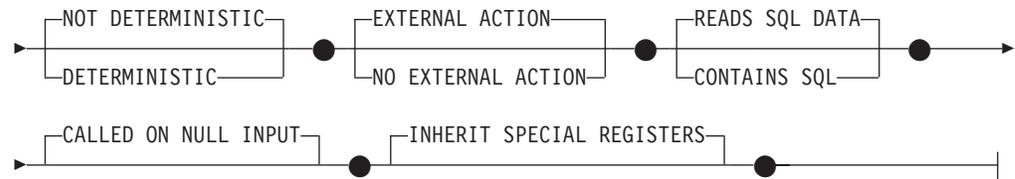
method-specification:



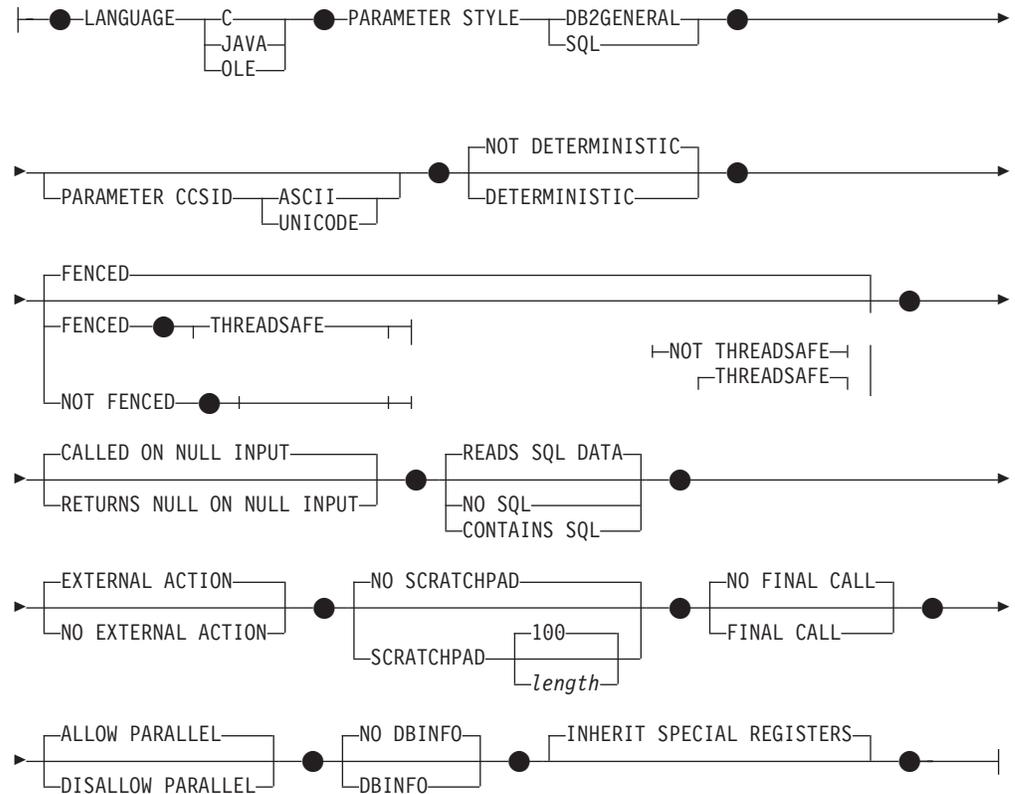
SQL-routine-characteristics:



CREATE TYPE (構造化)



external-routine-characteristics:



説明

type-name

タイプの名前を指定します。この名前 (暗黙または明示の修飾子を含む) は、現行サーバーに既に存在する他のタイプ (組み込みタイプ、構造化タイプ、特殊タイプを含む) と同じであってはなりません。非修飾名は、組み込みデータ・タイプ、BINARY、VARBINARY、BOOLEAN と同一の名前であってはなりません (SQLSTATE 42918)。非修飾名は、ARRAY、INTERVAL、または ROWID にするべきでもありません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*type-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、

CREATE TYPE (構造化)

SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

2 つの部分からなる *type-name* を指定する場合、文字 'SYS' で始まるスキーマ名は使用してはなりません (SQLSTATE 42939)。

UNDER *supertype-name*

この構造化タイプが指定した *supertype-name* のサブタイプであることを指定します。 *supertype-name* は既存の構造化タイプを指定する必要があります (SQLSTATE 42704)。 *supertype-name* がスキーマ名なしで指定される場合、SQL パス上でスキーマを検索することにより、タイプは解決されます。構造化タイプには、スーパータイプの属性すべてと、それに続く *attribute-definition* の追加属性が含まれます。

attribute-definition

構造化タイプの属性を定義します。

attribute-name

属性の名前です。この構造化タイプのその他の属性またはスーパータイプと同じ *attribute-name* を付けることはできません (SQLSTATE 42711)。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*attribute-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

data-type

属性のデータ・タイプです。これは、『CREATE TABLE』でリストされているデータ・タイプの 1 つで、XML 以外のものです (SQLSTATE 42601)。このデータ・タイプには既存のデータ・タイプを指定する必要があります (SQLSTATE 42704)。 *data-type* がスキーマ名なしで指定される場合、SQL パス上でスキーマを検索することにより、タイプは解決されます。『CREATE TABLE』に種々のデータ・タイプの説明が記載されています。属性データ・タイプが参照タイプである場合、参照するターゲット・タイプはこのステートメントに既に存在する構造化タイプであるか、またはこのステートメントで作成されたものでなければなりません (SQLSTATE 42704)。

実行時に、該当タイプのインスタンスが、同一タイプまたはそのサブタイプの別のインスタンスを直接または間接に取り込むことを許容するタイプ定義を防止するため、その属性のいずれかが、自身を直接または間接に使用する仕方でタイプを定義することはできません (SQLSTATE 428EP)。

lob-options

LOB タイプと関連したオプション (あるいは LOB に基づく特殊タイプ) を指定します。 *lob-options* の詳細については、『CREATE TABLE』を参照してください。

INSTANTIABLE または NOT INSTANTIABLE

構造化タイプのインスタンスを作成できるかどうかを指定します。インスタンス化不能な構造化タイプとは、以下のような意味です。

- インスタンス化不能タイプには、コンストラクター関数が生成されない

- インスタンス化不能タイプは、表またはビューのタイプとして使用することができない (SQLSTATE 428DP)
- インスタンス化不能タイプは、列のタイプとして使用することができる (その列には、NULL 値またはインスタンス化可能なサブタイプのインスタンスだけを挿入することができる)

インスタンス化不能タイプのインスタンスを作成するには、インスタンス化可能サブタイプを作成する必要があります。NOT INSTANTIABLE を指定すると、新規のタイプのインスタンスを作成できなくなります。

INLINE LENGTH *integer*

このオプションは、表の行内の残りの値とインラインで保管する構造化タイプ列のインスタンスの最大サイズ (バイト数) を指示します。指定したインライン長よりも長い構造化タイプまたはそのサブタイプのインスタンスは、LOB 値が処理されるのと同様の方法で、基本表の行とは別に保管されます。

指定した **INLINE LENGTH** が、新たに作成したタイプのコンストラクター関数の結果サイズよりも小さく (32 バイトに、属性ごとに 10 バイトを加算したもの)、しかも 292 バイトより小さいと、エラーが生じます (SQLSTATE 429B2)。属性数には、タイプのスーパータイプから継承されたすべての属性が含まれることに注意してください。

タイプの **INLINE LENGTH** は、指定値またはデフォルト値のどちらであっても、構造化タイプを使用する列のデフォルトのインライン長になります。このデフォルトは、CREATE TABLE 時にオーバーライドすることができます。

型付き表のタイプとして構造化タイプを使用すると、**INLINE LENGTH** には何の意味もなくなります。

構造化タイプのデフォルトの **INLINE LENGTH** はシステムによって計算されます。この後に示す公式では、以下のような用語を使います。

短い属性 (*short attribute*)

SMALLINT、INTEGER、BIGINT、REAL、DOUBLE、FLOAT、DATE、または TIME のデータ・タイプのいずれかをもつ属性を指します。さらに、これらのタイプに基づいた特殊タイプまたは参照タイプも含まれます。

短くない属性 (*non-short attribute*)

残りのデータ・タイプのいずれか、またはこれらのデータ・タイプに基づく特殊タイプの属性を指します。

システムは、次のようにデフォルトのインライン長を計算します。

1. 以下のような公式を使って、短くない属性の追加スペース所要量を割り出します。

$$\text{space_for_non_short_attributes} = \text{SUM}(\text{attributelength} + n)$$

n は以下のように定義されます。

- ネストされた構造化タイプの属性には 0 バイト
- 非 LOB 属性には 2 バイト
- LOB 属性には 9 バイト

CREATE TYPE (構造化)

attributelength は、表 26 に示すとおり、属性に指定されているデータ・タイプに基づく値です。

2. 以下のような公式を使って、デフォルトの合計インライン長を計算します。

$$\text{default_length}(\text{structured_type}) = (\text{number_of_attributes} * 10) + 32 + \text{space_for_non-short_attributes}$$

number_of_attributes は、スーパータイプから継承される属性も含めた構造化タイプの合計属性数です。ただし、*number_of_attributes* には、*structured_type* の任意のサブタイプに定義されているどの属性も含まれません。

表 26. 属性データ・タイプのバイト・カウント

属性データ・タイプ	バイト・カウント
DECIMAL	$(p/2)+1$ の整数部分 (p は精度)
DECFLOAT(n)	n が 16 の場合、バイト・カウントは 8 です。 n が 34 の場合、バイト・カウントは 16 です。
CHAR(n)	n
VARCHAR(n)	n
GRAPHIC(n)	$n * 2$
VARGRAPHIC(n)	$n * 2$
TIMESTAMP	10
LOB タイプ	各 LOB 属性は、構造化タイプ・インスタンス内に、実際の値の位置へのポインターとなる LOB 記述子を持っています。その記述子のサイズは、その LOB 属性に定義されている最大長によって異なります (表 27を参照してください)。
特殊タイプ	特殊タイプのソース・タイプの長さ
参照タイプ	参照タイプの基礎となる組み込みデータ・タイプの長さ
構造化タイプ	$\text{inline_length}(\text{attribute_type})$

表 27. LOB の最大長の関数としての LOB 記述子のサイズ

LOB の最大長	LOB 記述子のサイズ
1024	72
8192	96
65 536	120
524 000	144
4 190 000	168
134 000 000	200
536 000 000	224
1 070 000 000	256
1 470 000 000	280
2 147 483 647	316

WITHOUT COMPARISONS

構造化タイプのインスタンスで比較関数がサポートされていないことを示します。

NOT FINAL

この構造化タイプをスーパータイプとして使用できることを示します。

MODE DB2SQL

この節は必須であり、このタイプでコンストラクター関数を直接呼び出すために使用します。

WITH FUNCTION ACCESS

将来作成されるメソッドを含め、該当タイプとそのサブタイプのすべてのメソッドに対して、関数表記を使ってアクセスできることを指示します。この節を指定できるのは、UNDER 節が指定されていない構造化タイプの階層のルート・タイプだけです (SQLSTATE 42613)。この節は、メソッドを呼び出す表記よりもこの形式の表記のほうが望ましいアプリケーションで、関数表記を使用できるようにするために提供されています。

REF USING *rep-type*

この構造化タイプの参照タイプの表示 (基礎データ・タイプ) として使用される組み込みデータ・タイプとそのサブタイプをすべて定義します。この節を指定できるのは、UNDER 節が指定されていない構造化タイプの階層のルート・タイプだけです (SQLSTATE 42613)。*rep-type* は、REAL、FLOAT、DECFLOAT、BLOB、CLOB、DBCLOB、配列型、または構造化タイプであってはならず、32 672 バイト以下の長さでなければなりません (SQLSTATE 42613)。

構造化タイプの階層のルート・タイプにこの節を指定しない場合、REF USING VARCHAR(16) FOR BIT DATA が想定されます。

CAST (SOURCE AS REF) WITH *funcname1*

システムにより生成される関数で、データ・タイプ *rep-type* の値を、この構造化タイプの参照タイプにキャストする関数の名前を定義します。*funcname1* の一部としてスキーマ名を指定することはできません (SQLSTATE 42601)。cast 関数は、構造化タイプと同じスキーマ内で生成されます。この節を指定しない場合、*funcname1* のデフォルト値は *type-name* (構造化タイプの名前) になります。*funcname1(rep-type)* に一致する関数シグニチャーが、同じスキーマ内に存在してはなりません (SQLSTATE 42710)。

CAST (REF AS SOURCE) WITH *funcname2*

システムにより生成される関数で、この構造化タイプの参照タイプ値を、データ・タイプ *rep-type* にキャストする関数の名前を定義します。*funcname2* の一部としてスキーマ名を指定することはできません (SQLSTATE 42601)。cast 関数は、構造化タイプと同じスキーマ内で生成されます。この節を指定しない場合、*funcname2* のデフォルト値は *rep-type* (表示タイプの名前) になります。

method-specification

このタイプのメソッドを定義します。メソッドは、CREATE METHOD ステートメントで本体を与えられて初めて、実際に使用できるようになります (SQLSTATE 42884)。

OVERRIDING

定義するメソッドが、定義するタイプのスーパータイプのメソッドをオーバーライドすることを指定します。オーバーライドすることによって、サブタ

CREATE TYPE (構造化)

イブのメソッドを再インプリメントできるようになるので、より具体的な機能が提供されます。オーバーライドは、以下のタイプのメソッドではサポートされません。

- 表および行メソッド
- PARAMETER STYLE JAVA を使用して宣言される外部メソッド
- 索引拡張で述部として使用できるメソッド
- システムによって生成される mutator メソッドまたは observer メソッド

このようなメソッドをオーバーライド使用とすると、エラーになります (SQLSTATE 42745)。

メソッドを有効なオーバーライド・メソッドにする場合は、定義するタイプの適切なスーパータイプの 1 つに元のメソッドが 1 つ存在していなければならず、オーバーライド・メソッドと元のメソッドの間に以下の関係が存在している必要があります。

- 定義するメソッドと元のメソッドのメソッド名が同じである。
- 定義するメソッドと元のメソッドのパラメーターの数が同じである。
- 定義するメソッドの各パラメーターのデータ・タイプと、元のメソッドの対応するパラメーターのデータ・タイプが同一である。この要件では、暗黙の SELF パラメーターは考慮されません。

このような元のメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 428FV)。

オーバーライド・メソッドは、以下の属性を元のメソッドから継承します。

- 言語
- 決定性の指示
- 外部アクションの指示
- 引数に NULL 値がある場合にメソッドを呼び出すかどうかの指示
- 結果のキャスト (元のメソッドで指定されている場合)
- SELF AS RESULT の指示
- SQL データ・アクセスまたは CONTAINS SQL の指示
- 外部メソッドの場合は、以下のとおりです。
 - パラメーターのスタイル
 - パラメーターと結果のロケーターの指示 (元のメソッドで指定されている場合)
 - FENCED、SCRATCHPAD、FINAL CALL、ALLOW PARALLEL、および DBINFO の指示
 - INHERIT SPECIAL REGISTER および THREADSAFE の指示

method-name

定義しようとするメソッドを指定します。これは、修飾されていない SQL ID でなければなりません (SQLSTATE 42601)。メソッド名は、CREATE TYPE に使用されるスキーマで暗黙的に修飾されます。

述部のキーワードとして使用されるいくつかの名前は、システム使用に予約されており、*method-name* として使用することはできません (SQLSTATE 42939)。それらの名前は、SOME、ANY、ALL、NOT、AND、OR、

BETWEEN、NULL、LIKE、EXISTS、IN、UNIQUE、OVERLAPS、SIMILAR、MATCH および比較演算子です。

一般に、メソッドのシグニチャーがそれぞれ異なっている場合は、同じ名前を複数のメソッドに使用することができます。

parameter-name

パラメーター名を指定します。その名前は SELF であってはなりません。これは、メソッドの暗黙のサブジェクト・パラメーターの名前です (SQLSTATE 42734)。メソッドが SQL メソッドである場合、そのすべてのパラメーターに名前が付いていなければなりません (SQLSTATE 42629)。宣言するメソッドが別のメソッドをオーバーライドする場合は、パラメーター名は、オーバーライドされるメソッドの対応するパラメーターの名前と正確に一致している必要があります。そうでないと、エラーが戻されます (SQLSTATE 428FV)。

data-type2

各パラメーターのデータ・タイプを指定します。メソッドが受け取るはずの各パラメーターごとに 1 つの項目をこのリストに指定する必要があります。暗黙の SELF パラメーターを含め、90 を超える数のパラメーターを使うことはできません。この限界を超えると、エラーになります (SQLSTATE 54023)。

CREATE TABLE ステートメントに列タイプとして指定でき、しかもメソッドの作成に使用されている言語に対応するタイプが存在する SQL データ・タイプと省略形を指定できます。SQL データ・タイプとホスト言語データ・タイプの対応関係の詳細については、以下の関連トピックのリストから該当する言語に関係するトピックを参照してください。

注: 該当する SQL データ・タイプが構造化タイプである場合、ホスト言語データ・タイプに対するデフォルト・マッピングはありません。構造化タイプとホスト言語データ・タイプとをマッピングするには、ユーザー定義のトランスフォーム関数を使用する必要があります。

DECIMAL (または NUMERIC)、および 10 進浮動小数点数は、LANGUAGE C と OLE では無効です (SQLSTATE 42815)。

XML データ・タイプは使用できません (SQLSTATE 42815)。

REF を指定することができますが、これには定義された有効範囲はありません。メソッドの本体で、まず参照タイプをキャストして有効範囲をもたせて初めて、パス式内でその参照タイプを使用できるようになります。同様に、メソッドから戻された参照も、まずキャストして有効範囲をもたせて初めて、パス式内で使用できるようになります。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。これは、実際の値の代わりに LOB ロケーターをメソッドに渡すことを指定します。これにより、メソッドに渡すバイト数を大幅に削減することができ、パフォーマンスも向上します。メソッドにとって実際に必要になる値が数バイトだけである場合は特にそうです。

CREATE TYPE (構造化)

LOB または LOB に基づく特殊タイプ以外のタイプに対して AS LOCATOR を指定すると、エラーが発生します (SQLSTATE 42601)。

メソッドが FENCED の場合や、LANGUAGE が SQL の場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

宣言するメソッドが別のメソッドをオーバーライドする場合は、パラメーターの AS LOCATOR 指示は、オーバーライドされるメソッドの対応するパラメーターの AS LOCATOR 指示と正確に一致している必要があります (SQLSTATE 428FV)。

宣言するメソッドが別のメソッドをオーバーライドする場合は、各パラメーターの FOR BIT DATA 指示は、オーバーライドされるメソッドの対応するパラメーターの FOR BIT DATA 指示と正確に一致している必要があります。 (SQLSTATE 428FV)。

RETURNS

これは必須の節であり、メソッドの結果を指定します。

data-type3

メソッドの結果のデータ・タイプを指定します。この場合、上記のメソッドのパラメーター *data-type2* の項で説明したのと全く同じ考慮事項が当てはまります。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。これは、実際の値の代わりに LOB ロケーターがメソッドから渡されることを示します。

LOB または LOB に基づく特殊タイプ以外のタイプに対して AS LOCATOR を指定すると、エラーが発生します (SQLSTATE 42601)。

メソッドが FENCED の場合や、LANGUAGE が SQL の場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

定義するメソッドが別のメソッドをオーバーライドする場合は、この節を指定できません (SQLSTATE 428FV)。

メソッドが別のメソッドをオーバーライドする場合、*data-type3* は、データ・タイプが構造化タイプであれば、オーバーライドされるメソッドの結果のデータ・タイプのサブタイプでなければなりません。そうでない場合は、両方のデータ・タイプは同じでなければなりません (SQLSTATE 428FV)。

data-type4 **CAST FROM** *data-type5*

メソッドの結果のデータ・タイプを指定します。

この節は、メソッド・コードから戻されたデータ・タイプとは異なるデータ・タイプを、呼び出しステートメントに戻すのに使用されます。

data-type5 は、*data-type4* パラメーターにキャスト可能でなければなりません。キャスト可能でないと、エラーが戻されます (SQLSTATE 42880)。

data-type4 の長さ、精度または位取りは、*data-type5* から推断することができるので、*data-type4* に指定されるパラメーター化タイプの長さ、精度、または位取りを指定する必要はありません (指定は可能です)。代わりに、VARCHAR() のような空の括弧を使用できます。パラメーター値が異なるデ

ータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

特殊タイプは、*data-type5* に指定するタイプとしては無効です (SQLSTATE 42815)。XML は、*data-type4* または *data-type5* に指定するタイプとしては無効です (SQLSTATE 42815)。

キャスト操作は実行時検査の対象にもなり、その結果、変換エラーが戻される可能性があります。

AS LOCATOR

LOB タイプまたは LOB タイプに基づく特殊タイプの場合、AS LOCATOR 節を追加することができます。これは、実際の値の代わりに LOB ロケーターがメソッドから渡されることを示します。

LOB または LOB に基づく特殊タイプ以外のタイプに対して AS LOCATOR を指定すると、エラーが発生します (SQLSTATE 42601)。

メソッドが FENCED の場合や、LANGUAGE が SQL の場合、AS LOCATOR 節は指定できません (SQLSTATE 42613)。

定義するメソッドが別のメソッドをオーバーライドする場合は、この節を指定できません (SQLSTATE 428FV)。

定義するメソッドが別のメソッドをオーバーライドする場合は、FOR BIT DATA 節を指定できません (SQLSTATE 428FV)。

SPECIFIC *specific-name*

定義するメソッドのインスタンスに対するユニーク名を指定します。この名前は、メソッドの本体の作成やメソッドのドロップのときに使用することができます。これは、メソッドの呼び出しには使用できません。 *specific-name* の非修飾形式は SQL ID です (最大長 18)。修飾形式は、*schema-name* の後にピリオドと SQL ID が続きます。暗黙または明示の修飾子も含め、その名前が、アプリケーション・サーバーに存在する別の個別メソッド名を指定するものであってはなりません。そうでない場合は、エラーになります (SQLSTATE 42710)。

specific-name は、既存の *method-name* と同じでも構いません。

修飾子を指定しない場合、*type-name* に使用された修飾子が使用されます。修飾子を指定する場合は、*type-name* の明示または暗黙の修飾子と同じでなければなりません。そうでない場合は、エラーになります (SQLSTATE 42882)。

specific-name の指定がない場合、固有名がデータベース・マネージャーによって生成されます。生成されるユニーク名は、SQL の後に文字のタイム・スタンプが続く名前です (SQLyymmddhhmmssxxx)。

SELF AS RESULT

このメソッドがタイプ保存メソッドであることを指定します。その意味は次のとおりです。

- 宣言された戻りタイプは、宣言されたサブジェクト・タイプと同じでなければなりません (SQLSTATE 428EQ)。
- SQL ステートメントがコンパイルされ、タイプ保存メソッドに解決されると、そのメソッド結果の静的タイプは、サブジェクト引数の静的タイプと同じになります。

CREATE TYPE (構造化)

- メソッドをインプリメントする場合、結果の動的タイプが、サブジェクト引数の静的タイプと同じになる (SQLSTATE 2200G) ようにし、そしてその結果 NULL にならない (SQLSTATE 22004) ようにする必要があります。

定義するメソッドが別のメソッドをオーバーライドする場合は、この節を指定できません (SQLSTATE 428FV)。

SQL-routine-characteristics

CREATE METHOD を使ってこのタイプに定義されるメソッド本体の特性を指定します。

LANGUAGE SQL

この節を使って、単一の RETURN ステートメントを使って SQL でメソッドを作成することを指示します。メソッド本体は、CREATE METHOD ステートメントを使って指定します。

PARAMETER CCSID

SQL メソッドとやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。

UNICODE

文字データは UTF-8 で記述され、GRAPHIC データは UCS-2 で記述されることを指定します。データベースが Unicode データベースでない場合は、PARAMETER CCSID UNICODE は指定できません (SQLSTATE 56031)。

NOT DETERMINISTIC または DETERMINISTIC

この節はオプションですが、特定の引数の値に対してメソッドが常に同じ結果を返すか (DETERMINISTIC)、それとも状態値に応じてメソッドの結果が異なるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC メソッドは、同じ入力を使用して正しく呼び出した場合に常に同じ結果を返します。NOT DETERMINISTIC を指定すると、同じ入力によって常に同じ結果が生じる利点に基づく最適化ができなくなります。メソッド本体が特殊レジスターにアクセスしたり、別の非 deterministic ルーチン呼び出ししたりする場合、明示的または暗黙的に NOT DETERMINISTIC を指定しなければなりません (SQLSTATE 428C2)。

EXTERNAL ACTION または NO EXTERNAL ACTION

この節はオプションであり、データベース・マネージャーによって管理されていないオブジェクトの状態を変更する処置をメソッドが行うか否かを指定します。EXTERNAL ACTION を指定すると、外部からメソッドへの影響がないことを前提とした最適化ができなくなります。(例えば、メッセージの送信、警報音による通知、ファイルへのレコードの書き込みなど。)

READS SQL DATA または CONTAINS SQL

どのタイプの SQL ステートメントを実行できるかを指示します。サポートされている SQL ステートメントは RETURN ステートメントであるので、式が副照会であるかどうかで区別を行います。

READS SQL DATA

SQL データを変更しない SQL ステートメントを、メソッドで実行できることを指定します (SQLSTATE 42985)。SQL ステートメント内でニックネームを参照することはできません (SQLSTATE 42997)。

CONTAINS SQL

SQL データの読み取りも変更も行わない SQL ステートメントを、メソッドで実行できることを指定します (SQLSTATE 42985)。

CALLED ON NULL INPUT

このオプション節は、引数が NULL 値か否かに関係なくユーザー定義メソッドを呼び出すことを指定します。これは、NULL 値を戻す場合も、通常の (NULL 以外の) 値を戻す場合もあります。ただし、NULL の引数値の有無のテストはメソッドが行う必要があります。

定義するメソッドが別のメソッドをオーバーライドする場合は、この節を指定できません (SQLSTATE 428FV)。

NULL CALL は、CALLED ON NULL INPUT の同義語として使うことができます。

INHERIT SPECIAL REGISTERS

このオプションの節は、メソッド内の更新可能特殊レジスターが、初期値を呼び出し側ステートメントの環境から継承することを指定します。カーソルの選択ステートメントから呼び出されるメソッドの場合は、初期値はカーソルがオープンされた環境から継承されます。ネストされたオブジェクトで呼び出されるルーチン (トリガーまたはビューなど) の場合は、初期値はランタイム環境から継承されます (オブジェクト定義からは継承されません)。特殊レジスターに対する変更が、関数の呼び出し側に戻されることはありません。

更新不能の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、デフォルト値に設定されます。

external-routine-characteristics**LANGUAGE**

この節は必須で、ユーザー定義メソッドの本体が準拠している言語インターフェース規則を指定するのに使用します。

C これは、データベース・マネージャーが、ユーザー定義メソッドを C の関数であるかのように呼び出すことを意味します。ユーザー定義メソッドは、標準 ANSI C プロトタイプで定義されている C 言語の呼び出しおよびリンケージの規則に準拠していなければなりません。

JAVA

データベース・マネージャーは、Java クラスのメソッドとしてユーザー定義メソッドを呼び出します。

CREATE TYPE (構造化)

OLE

データベース・マネージャは、OLE 自動化オブジェクトによって公開されたメソッドとして、ユーザー定義メソッドを呼び出します。メソッドは、「*OLE Automation Programmer's Reference*」に説明されている OLE 自動化データ・タイプと呼び出しメカニズムに準拠している必要があります。

LANGUAGE OLE は、Windows 32 ビット・オペレーティング・システムで保管されたユーザー定義メソッドに対してのみサポートされます。THREADSAFE は、LANGUAGE OLE で定義されたメソッドに指定することはできません (SQLSTATE 42613)。

PARAMETER STYLE

この節は、メソッドに対してパラメーターを渡し、そこから値を戻すのに用いる規則を指定するのに使用されます。

DB2GENERAL

Java クラスのメソッドとして定義された外部メソッドとの間で、パラメーターを渡し、値を戻す場合に用いる規則を指定します。これは、LANGUAGE JAVA を使用する場合にだけ指定する必要があります。

DB2GENERAL の同義語として値 DB2GENRL が使用可能です。

SQL

C 言語の呼び出しとリンケージの規則、または OLE 自動化オブジェクトによって公開されたメソッドに準拠する規則を、この外部メソッドとの間でパラメーターを渡し、値を戻す場合の規則として指定します。これは、LANGUAGE C または LANGUAGE OLE を使用する場合に指定する必要があります。

PARAMETER CCSID

外部メソッドとやり取りされるすべてのストリング・データに使用されるコード化スキームを指定します。PARAMETER CCSID 節を指定しない場合のデフォルトは、Unicode データベースでは PARAMETER CCSID UNICODE、他のすべてのデータベースでは PARAMETER CCSID ASCII になります。

ASCII

ストリング・データがデータベース・コード・ページでエンコードされることを指定します。データベースが Unicode データベースの場合は、PARAMETER CCSID ASCII を指定することはできません (SQLSTATE 56031)。

UNICODE

文字データは UTF-8 で記述され、GRAPHIC データは UCS-2 で記述されることを指定します。データベースが Unicode データベースでない場合は、PARAMETER CCSID UNICODE は指定できません (SQLSTATE 56031)。

この節を LANGUAGE OLE とともに指定することはできません (SQLSTATE 42613)。

DETERMINISTIC または NOT DETERMINISTIC

この節はオプションですが、特定の引数の値に対してメソッドが常に同じ結

果を戻すか (DETERMINISTIC)、それとも状態値に応じてメソッドの結果が異なるか (NOT DETERMINISTIC) を指定します。つまり DETERMINISTIC メソッドは、同じ入力を使用して正しく呼び出した場合に常に同じ結果を戻します。NOT DETERMINISTIC を指定すると、同じ入力によって常に同じ結果が生じる利点に基づく最適化ができなくなります。非 deterministic タイプの例としては、結果タイプに影響を与える方法で特殊レジスター、グローバル変数、または非 deterministic 関数を参照するような場合です。

FENCED または NOT FENCED

この節は、データベース・マネージャーの操作環境のプロセスまたはアドレス・スペースでメソッドを実行しても「安全」か (NOT FENCED)、そうでないか (FENCED) を指定します。

メソッドが FENCED として登録されると、データベース・マネージャーは、その内部リソース (データ・バッファなど) を保護して、そのメソッドからアクセスされないようにします。多くのメソッドは、FENCED または NOT FENCED のどちらかで実行するように選択することができます。一般に、FENCED として実行されるメソッドは、NOT FENCED として実行されるものと同じようには実行されません。

注意:

十分にチェックされていないメソッドに NOT FENCED を使用すると、DB2 の整合性に危険を招く場合があります。DB2 では、発生する可能性のある一般的な不注意による障害の多くに対して、いくつかの予防措置がとられています。NOT FENCED ユーザー定義メソッドが使用される場合には、完全な整合性を確保できません。

LANGUAGE OLE または NOT THREADSAFE を指定したメソッドには、FENCED のみを指定できます (SQLSTATE 42613)。

メソッドが FENCED で NO SQL オプションが指定されている場合、AS LOCATOR 節を指定できません (SQLSTATE 42613)。

メソッドを NOT FENCED として登録するには、SYSADM 権限、DBADM 権限、または特殊権限 (CREATE_NOT_FENCED_ROUTINE) が必要です。

THREADSAFE または NOT THREADSAFE

メソッドを他のルーチンと同じプロセスで実行しても「安全」か (THREADSAFE)、そうでないか (NOT THREADSAFE) を指定します。

メソッドが OLE 以外の LANGUAGE で定義される場合:

- メソッドが THREADSAFE に定義されている場合には、データベース・マネージャーは他のルーチンと同じプロセスでメソッドを呼び出すことができます。一般に、スレッド・セーフにするには、メソッドはどのグローバルあるいは静的データ域をも使用してはなりません。多くのプログラミング解説書には、スレッド・セーフ・ルーチンの作成に関する説明が含まれています。FENCED および NOT FENCED メソッドの両方が THREADSAFE になることが可能です。
- メソッドが NOT THREADSAFE として定義される場合には、データベース・マネージャーは他のルーチンと同じプロセスにメソッドを決して呼び出しません。

CREATE TYPE (構造化)

FENCED メソッドについては、LANGUAGE が JAVA の場合、THREADSAFE がデフォルトです。これ以外のすべての言語の場合は、NOT THREADSAFE がデフォルトです。メソッドが LANGUAGE OLE とともに定義される場合には、THREADSAFE は指定されません (SQLSTATE 42613)。

NOT FENCED メソッドについては、THREADSAFE がデフォルトです。NOT THREADSAFE を指定することはできません (SQLSTATE 42613)。

RETURNS NULL ON NULL INPUT または CALLED ON NULL INPUT

このオプション節を使用すると、非サブジェクト引数のいずれかが NULL 値の場合に、外部メソッドを呼び出さないようにすることができます。

RETURNS NULL ON NULL INPUT が指定されており、実行時にメソッドの引数のいずれかが NULL 値の場合、このメソッドは呼び出されず、結果は NULL 値になります。

CALLED ON NULL INPUT を指定すると、NULL 値の引数の数に関係なくメソッドが呼び出されます。これは、NULL 値を戻す場合も、通常の (NULL 以外の) 値を戻す場合もあります。ただし、NULL の引数値の有無のテストはメソッドが行う必要があります。

値 NULL CALL は、後方互換性またはファミリーの互換性のために、CALLED ON NULL INPUT の同義語として使うことができます。同様に、NOT NULL CALL は、RETURNS NULL ON NULL INPUT の同義語として使用できます。

以下の 2 つのケースでは、この指定が無視されます。

- 対象となる引数が NULL の場合。この場合、メソッドは実行されずに結果は NULL になります。
- パラメーターがないものとしてメソッドを定義した場合。この場合、この NULL 引数条件が成立することはありません。

NO SQL、CONTAINS SQL、READS SQL DATA

メソッドが SQL ステートメントを発行するかどうか、および（発行する場合の）タイプを示します。

NO SQL

メソッドはどの SQL ステートメントも実行できないことを指示します (SQLSTATE 38001)。

CONTAINS SQL

SQL データの読み取りも変更も行わない SQL ステートメントを、メソッドで実行できることを指定します (SQLSTATE 38004 または 42985)。どのメソッドでもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

READS SQL DATA

SQL データを変更しない SQL ステートメントを、メソッドで実行できることを指定します (SQLSTATE 38002 または 42985)。どのメソッドでもサポートされていないステートメントは、これとは異なるエラーを戻します (SQLSTATE 38003 または 42985)。

EXTERNAL ACTION または NO EXTERNAL ACTION

この節はオプションであり、データベース・マネージャーによって管理され

ていないオブジェクトの状態を変更する処置をメソッドが行うか否かを指定します。EXTERNAL ACTION を指定すると、外部からメソッドへの影響がないことを前提とした最適化ができなくなります。

NO SCRATCHPAD または SCRATCHPAD length

この節はオプションであり、この外部メソッドに対してスクラッチパッドを用意するか否かを指定するのに使用できます。メソッドを再入可能にすることを強くお勧めします。再入可能にすると、スクラッチパッドが、呼び出しのたびにメソッドに「状態を保管」させる手段になります。

SCRATCHPAD を指定すると、ユーザー定義メソッドの最初の呼び出し時に、その外部メソッドによって使用されるスクラッチパッドにメモリーが割り振られます。このスクラッチパッドには、次の特性があります。

- length を指定して、スクラッチパッドのバイト単位のサイズを設定します。これは 1 から 32,767 でなければなりません (SQLSTATE 42820)。デフォルト値は 100 です。
- すべて X'00' に初期化されます。
- その有効範囲は、該当の SQL ステートメントです。SQL ステートメントでの外部メソッドに対する参照ごとに 1 つのスクラッチパッドがあります。

したがって、次のステートメントのメソッド X が SCRATCHPAD キーワードを指定して定義されると、3 つのスクラッチパッドが割り当てられます。

```
SELECT A, X..(A) FROM TABLEB
WHERE X..(A) > 103 OR X..(A) < 19
```

ALLOW PARALLEL が指定されているか、またはデフォルト値として使用された場合、その有効範囲は上記とは異なります。メソッドが複数のデータベース・パーティションで実行される場合、メソッドが処理されるそれぞれのデータベース・パーティションにおいて、SQL ステートメントでのメソッドへのそれぞれの参照ごとにスクラッチパッドが割り当てられます。同様に、パーティション内並列処理をオンにして照会が実行される場合、3 より多くのスクラッチパッドが割り当てられることがあります。

スクラッチパッドは持続します。その内容は、外部メソッドの呼び出しごとに保存されます。外部メソッドのある呼び出しによってスクラッチパッドに加えられた変更はいずれも、次の呼び出し時に存続しています。データベース・マネージャーは、各 SQL ステートメントの実行開始時に、スクラッチパッドを初期設定します。各副照会の実行開始時には、データベース・マネージャーによってスクラッチパッドがリセットされます。FINAL CALL オプションが指定されている場合、システムは、スクラッチパッドのリセットに先立って、最終呼び出しを行います。

スクラッチパッドは、外部メソッドが獲得できるシステム・リソース (メモリーなど) の中央点として使用することもできます。メソッドは、最初の呼び出しでメモリーを獲得し、そのアドレスをスクラッチパッドに保管して、後の呼び出しでそれを参照することができます。

CREATE TYPE (構造化)

このようにシステム・リソースが獲得される場合、FINAL CALL キーワードも指定する必要があります。そうすると、ステートメントの最後で特殊な呼び出しが行われ、外部メソッドは獲得したシステム・リソースをすべて解放することができます。

SCRATCHPAD を指定すると、ユーザー定義メソッドを呼び出すたびに、スクラッチパッドをアドレッシングする外部メソッドに追加の引数が渡されません。

NO SCRATCHPAD を指定すると、外部メソッドに対してスクラッチパッドは割り振られず、渡されません。

NO FINAL CALL または FINAL CALL

この節はオプションであり、外部メソッドに対する最終呼び出しが行われるか否かを指定します。このような最終呼び出しの目的は、外部メソッドが獲得したシステム・リソースすべてを解放できるようにすることです。外部メソッドがメモリーなどのシステム・リソースを獲得し、それをスクラッチパッドに固定するような状況では、これを SCRATCHPAD キーワードと共に使用すると便利です。

FINAL CALL を指定すると、実行時に、呼び出しのタイプを指定する外部メソッドに追加の引数が渡されます。呼び出しのタイプは次のとおりです。

- 通常呼び出し。SQL 引数が渡され、結果が戻されることが予想されません。
- 最初の呼び出し。この SQL ステートメントのメソッドに対する参照に対応する外部メソッドの最初の呼び出しです。最初の呼び出しは通常呼び出しです。
- 最終呼び出し。外部メソッドがリソースを解放できるようにするそのメソッドに対する最終呼び出しです。最終呼び出しは、通常呼び出しではありません。この最終呼び出しは、以下の時点で行われます。
 - ステートメント終了時。これは、カーソル指向型のステートメントでカーソルがクローズされた場合、あるいはステートメントが実行を終了した場合に発生します。
 - トランザクション終了時。これは、通常のステートメント終了が発生しなかった場合に発生します。例えば、何らかの理由で、アプリケーションのロジックが、カーソルをクローズしないようになっている場合があります。

WITH HOLD として定義されたカーソルがオープンされている間に、コミット操作が発生すると、それ以降のカーソルのクローズ時、またはアプリケーションの終了時に最終呼び出しが行われます。

NO FINAL CALL を指定すると、「呼び出しタイプ」の引数は外部メソッドに渡されず、最終呼び出しは行われません。

ALLOW PARALLEL または DISALLOW PARALLEL

この節はオプションで、メソッドへの 1 つの参照で、メソッドの呼び出しを並列化できるか否かを指定します。一般には、ほとんどのスカラー・メソッドの呼び出しは並列化可能ですが、並列化できないメソッド (1 つのスク

ラッチパッドのコピーに依存するメソッドなど) もあります。メソッドに対して ALLOW PARALLEL または DISALLOW PARALLEL を指定すると、DB2 はその指定を受け入れます。

メソッドにどちらのキーワードが当てはまるかを判別するには、以下の点を検討する必要があります。

- メソッドのすべての呼び出しが、互いに完全に独立していますか? YES の場合には、ALLOW PARALLEL を指定します。
- メソッドを呼び出すごとに、次の呼び出しに関係する値を提供するスクラッチパッドが更新されますか? (例えば、カウンターの増分によって。) YES の場合には、DISALLOW PARALLEL を指定するか、またはデフォルトを受け入れます。
- 1 つのデータベース・パーティションでのみ起こる必要のある外部アクションがメソッドによって実行されますか? YES の場合には、DISALLOW PARALLEL を指定するか、またはデフォルトを受け入れます。
- コストのかかる初期化処理の実行回数を最小にするためだけに、スクラッチパッドを使用していますか? YES の場合には、ALLOW PARALLEL を指定します。

いずれの場合も、すべての外部メソッドの本体は、すべてのデータベース・パーティションで使用可能なディレクトリーにある必要があります。

構文図は、デフォルト値が ALLOW PARALLEL であることを示しています。しかし、ステートメントで以下のオプションの 1 つ又は複数指定されている場合は、デフォルトは DISALLOW PARALLEL です。

- NOT DETERMINISTIC
- EXTERNAL ACTION
- SCRATCHPAD
- FINAL CALL

NO DBINFO または DBINFO

この節はオプションで、DB2 において既知である特定の情報を追加の呼び出し時に引数としてメソッドに渡すか (DBINFO)、または渡さないか (NO DBINFO) を指定します。NO DBINFO がデフォルト値です。DBINFO は、LANGUAGE OLE ではサポートされません (SQLSTATE 42613)。定義するメソッドが別のメソッドをオーバーライドする場合は、この節を指定できません (SQLSTATE 428FV)。

DBINFO を指定すると、以下の情報をもつ構造がメソッドに渡されます。

- データベース名 - 現在接続されているデータベースの名前。
- アプリケーション ID - データベースへの接続ごとに確立された、固有アプリケーション ID。
- アプリケーション許可 ID - アプリケーション実行時の許可 ID。このメソッドとアプリケーションとの中間でネストされているメソッドは無関係です。
- コード・ページ - データベースのコード・ページを識別します。
- スキーマ名 - 表名とまったく同じ条件のもとでは、スキーマの名前が入ります。その他の場合はブランクです。

CREATE TYPE (構造化)

- 表名 - メソッド参照が UPDATE ステートメントの SET 節の右側にある場合、または INSERT ステートメントの VALUES リストの項目である場合のいずれかに限り、更新または挿入される表の非修飾名が入ります。その他の場合はブランクです。
- 列名 - 表名とまったく同じ条件で、更新または挿入される列の名前が入ります。その他の場合はブランクです。
- データベースのバージョン/リリース - メソッドを呼び出すデータベース・サーバーのバージョン、リリース、および修正レベルを識別します。
- プラットフォーム - サーバーのプラットフォーム・タイプが入ります。
- 表メソッドの結果の列番号 - メソッドには当てはまりません。

INHERIT SPECIAL REGISTERS

このオプションの節は、メソッド内の特殊レジスターが、初期値を呼び出し側ステートメントから継承することを指定します。カーソルの場合は、初期値はカーソルがオープンされる時に継承されます。

特殊レジスターに対する変更が、メソッドの呼び出し元に戻されることはありません。

一部の特殊レジスター (日時特殊レジスターなど) は、現在実行中のステートメントのプロパティを反映するので、呼び出し元からの継承は行われません。

注

- まだ存在していないスキーマ名を用いて構造化タイプを作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
- 属性なしで定義された構造化サブタイプは、属性をすべてスーパータイプから継承するサブタイプを定義します。 UNDER 節も他のどの属性も指定しない場合、タイプは、属性なしの、タイプ階層のルート・タイプになります。
- タイプ階層に新たにサブタイプを追加すると、パッケージが無効になることがあります。パッケージは、その新しいタイプのスーパータイプに依存していると、無効になることがあります。このような従属関係は、TYPE 述部または TREAT 指定を使用した結果として生じます。
- 構造化タイプは、属性の数が 4082 個以下でなければなりません (SQLSTATE 54050)。
- 関数と同じシグニチャーをもつメソッド (関数の最初のパラメーター・タイプと、メソッドのサブジェクト・タイプの比較による) を指定することはできません。
- 元のメソッドは、別のメソッドをオーバーライドしたり、元のメソッドによってオーバーライドされたりしてはなりません (SQLSTATE 42745)。さらに、関数とメソッドは、オーバーライド関係にあってはなりません。つまり、関数は、サブジェクト S を第 1 パラメーターとしてもつメソッドであると見なされる場合、S のスーパータイプの別のメソッドをオーバーライドしてはならず、S のサブタイプの別のメソッドによってオーバーライドされてはならないという意味です (SQLSTATE 42745)。

- ある構造化タイプを作成すると、そのタイプで使用される一連の関数とメソッドが自動的に生成されます。これらの関数とメソッドはすべて、構造化タイプと同じスキーマ内で生成されます。生成された関数またはメソッドのシグニチャーが、このスキーマに存在する関数のシグニチャーと競合またはそれをオーバーライドする場合、このステートメントは失敗します (SQLSTATE 42710)。構造化タイプをドロップしないで、生成された関数またはメソッドをドロップすることはできません (SQLSTATE 42917)。次のような関数とメソッドが生成されます。

- 関数

- 参照比較

REF(*type-name*) という参照タイプでは、=、<>、<、<=、>、>= という名前の 6 つの比較関数が生成されます。これらの関数はそれぞれ

REF(*type-name*) というタイプのパラメーターを 2 つ受け取ってから、真、偽、または不明という値を戻します。REF(*type-name*) の比較演算子は、REF(*type-name*) の基礎データ・タイプと同じ動作をするように定義されます。(タイプ階層に含まれる参照表示タイプはすべて同一のもので、これにより、(S と T が共通のスーパータイプを持っている場合は) REF(S) と REF(T) の比較が可能になります。表の OID 列は、表階層内では固有にすることができるため、(それぞれが別の行を参照していても) 1 つの表階層の REF(T) 値を別の表階層の REF(T) 値と「等しく」することができます。)

参照タイプの有効範囲は比較の対象にはなりません。

- cast 関数

生成された参照タイプである REF(*type-name*) とこの参照タイプの基礎データ・タイプとの間をキャストするために 2 つの cast 関数が生成されます。

- 基礎タイプから参照タイプへとキャストする関数の名前は、暗黙的または明示的な *funcname1* です。

この関数の形式は以下のとおりです。

```
CREATE FUNCTION funcname1 (rep-type)
  RETURNS REF(type-name) ...
```

- 参照タイプから基礎タイプ (参照タイプの) へとキャストする関数の名前は、暗黙的または明示的な *funcname2* です。

この関数の形式は以下のとおりです。

```
CREATE FUNCTION funcname2 ( REF(type-name) )
  RETURNS rep-type ...
```

ある種の *rep-type* には、定数からのキャストを操作する *funcname1* を使って生成された追加の cast 関数があります。

- *rep-type* が SMALLINT の場合、追加で生成された cast 関数の形式は以下のとおりです。

```
CREATE FUNCTION funcname1 (INTEGER)
  RETURNS REF(type-name)
```

- *rep-type* が CHAR(n) の場合、追加で生成された cast 関数の形式は以下のとおりです。

CREATE TYPE (構造化)

```
CREATE FUNCTION funcname1 ( VARCHAR(n))  
  RETURNS REF(type-name)
```

- *rep-type* が GRAPHIC(*n*) の場合、追加で生成された cast 関数の形式は以下のとおりです。

```
CREATE FUNCTION funcname1 (VARGRAPHIC(n))  
  RETURNS REF(type-name)
```

それらの演算子や cast 関数を SQL ステートメントで正しく使用するには、SQL パスに構造化タイプのスキーマ名が組み込まれていなければなりません。

- コンストラクター関数

コンストラクター関数は、そのタイプの新しいインスタンスを構成可能にするために生成されます。この新しいインスタンスでは、スーパータイプから継承する属性も含め、そのタイプのどの属性も NULL になります。

生成されるコンストラクター関数の形式は、以下のとおりです。

```
CREATE FUNCTION type-name (  
  RETURNS type-name  
  ...
```

NOT INSTANTIABLE を指定すると、コンストラクター関数は生成されません。

- メソッド

- observer メソッド

構造化タイプの各属性ごとに observer メソッドが定義されます。observer メソッドは、各属性ごとに属性タイプを戻します。対象が NULL の場合、observer メソッドは、属性タイプの NULL 値を戻します。

例えば、C1..STREET、C1..CITY、C1..COUNTRY、および C1..CODE を使って、構造化タイプ ADDRESS のインスタンスの属性を監視することができます。

生成される observer メソッドのメソッド・シグニチャーは、次のようなステートメントが実行された場合に似ています。

```
CREATE TYPE type-name  
  ...  
  METHOD attribute-name()  
  RETURNS attribute-type
```

type-name は、構造化タイプ名です。

- mutator メソッド

構造化タイプの各属性ごとに、タイプ保存の mutator メソッドが定義されます。構造化タイプのインスタンス内の属性を変更するには、mutator メソッドを使用します。mutator メソッドは、各属性ごとに、サブジェクトのコピーの指定属性に引数を割り当てることで変更されたそのコピーを戻します。

例えば、`C1..CODE('M3C1H7')` を使って、構造化タイプ `ADDRESS` のインスタンスを変異することができます。サブジェクトが `NULL` の場合、`mutator` メソッドはエラーを生じます (SQLSTATE 2202D)。

生成される `mutator` メソッドのメソッド・シングニチャーは、次のようなステートメントが実行された場合に似ています。

```
CREATE TYPE type-name
...
METHOD attribute-name (attribute-type)
  RETURNS type-name
```

属性のデータ・タイプが `SMALLINT`、`REAL`、`CHAR`、または `GRAPHIC` である場合、定数を使用する変異をサポートするため、次のような追加の `mutator` メソッドが生成されます。

- `attribute-type` が `SMALLINT` の場合、追加の `mutator` はタイプ `INTEGER` の引数をサポートします。
 - `attribute-type` が `REAL` の場合、追加の `mutator` はタイプ `DOUBLE` の引数をサポートします。
 - `attribute-type` が `CHAR` の場合、追加の `mutator` はタイプ `VARCHAR` の引数をサポートします。
 - `attribute-type` が `GRAPHIC` の場合、追加の `mutator` はタイプ `VARGRAPHIC` の引数をサポートします。
- 列タイプとして構造化タイプを使用する場合、そのタイプのインスタンスの長さは、実行時に 1 GB を超えないようにしなければなりません (SQLSTATE 54049)。
- 既存の構造化タイプの新しいサブタイプを作成する (列タイプとして使用するため) 場合、それに関連した既存の構造化タイプのサポートとしてすでに作成されているすべてのトランスフォーム関数を再検査し、必要があれば更新してください。その新しいタイプが、特定のタイプとして同じ階層内にあっても、あるいはネストされたタイプの階層内にあっても、そのタイプに関連した既存のトランスフォーム関数を変更して、新規のサブタイプによって導入される新しい属性の一部または全部を組み込む必要があると考えられます。概して、それは、UDF とクライアント・アプリケーションから構造化タイプにアクセスさせるための特定のタイプ (またはタイプ階層) に関連した一連のトランスフォーム関数セットであるため、特定の複合階層内のすべての属性 (つまり、すべてのサブタイプとそのネストされた構造化タイプの推移的閉包を含む) をサポートするように、トランスフォーム関数を作成しなければなりません。

既存のタイプの新しいサブタイプを作成すると、作成されたタイプのスーパータイプで定義されるメソッドで、しかもオーバーライドが可能なメソッドに從属するすべてのパッケージは無効になります。

• 表アクセスの制限

メソッドが `READS SQL DATA` として定義されている場合は、メソッド内のステートメントは、このメソッドを呼び出したステートメントによって変更される表にアクセスすることはできません (SQLSTATE 57053)。例えば、メソッド `BONUS()` が `READS SQL DATA` として定義されているとします。ステートメン

CREATE TYPE (構造化)

ト UPDATE DEPTINFO SET SALARY = SALARY + EMP..BONUS() が呼び出されると、BONUS メソッド内の SQL ステートメントは、EMPLOYEE 表を読み取ることができません。

• 特権

- ユーザー定義タイプの定義者は、構造化タイプ用に自動的に生成されるすべてのメソッドおよび関数に対する EXECUTE 特権 WITH GRANT OPTION を常に受け取ります。EXECUTE 特権は、CREATE METHOD ステートメントを使用してメソッド本体が定義されない限り、CREATE TYPE ステートメントで明示的に指定されるメソッドに対しては付与されません。ユーザー定義タイプの定義者には、ALTER TYPE ステートメントを使用してメソッド指定をドロップする権利があります。CREATE TYPE (特殊) ステートメントの間に自動的に生成されるすべての関数での EXECUTE 特権は、PUBLIC に与えられます。
- SQL ステートメントで外部メソッドを使用する場合は、メソッドの定義者は、メソッドが使用するどのパッケージに対しても EXECUTE 特権を持っている必要があります。

- パーティション・データベース環境では、外部ユーザー定義関数またはメソッドでの SQL の使用はサポートされていません (SQLSTATE 42997)。
- 索引拡張を定義するには、NO SQL として定義されたルーチンしか使用できません (SQLSTATE 428F8)。
- NOT FENCED として定義される Java ルーチンは、FENCED THREADSAFE として定義されているかのように呼び出されます。

• 互換性

- DB2 for z/OS との互換性:
 - 以下の構文が許容されます。
 - DETERMINISTIC の代わりに NOT VARIANT を指定できます。
 - NOT DETERMINISTIC の代わりに VARIANT を指定できます。
 - CALLED ON NULL INPUT の代わりに NULL CALL を指定できます。
 - RETURNS NULL ON NULL INPUT の代わりに NOT NULL CALL を指定できます。
 - 以下の構文は、外部メソッドのデフォルトの振る舞いとして受け入れられません。
 - ASUTIME NO LIMIT
 - NO COLLID
 - PROGRAM TYPE SUB
 - STAY RESIDENT NO
 - Unicode データベースでの CCSID UNICODE
 - PARAMETER CCSID UNICODE が指定されていない場合、非 Unicode データベース内での CCSID ASCII
 - 以下の構文は、SQL メソッドのデフォルトの振る舞いとして受け入れられます。
 - Unicode データベースでの CCSID UNICODE
 - 非 Unicode データベースでの CCSID ASCII

- 以前のバージョンの DB2 との互換性:
 - PARAMETER STYLE SQL の代わりに PARAMETER STYLE DB2SQL を指定できます。

例

例 1: 部門のタイプを作成します。

```
CREATE TYPE DEPT AS
  (DEPT_NAME  VARCHAR(20),
   MAX_EMPS  INT)
  REF USING INT
  MODE DB2SQL
```

例 2: 従業員タイプおよびマネージャー・サブタイプから構成されるタイプ階層を作成します。

```
CREATE TYPE EMP AS
  (NAME       VARCHAR(32),
   SERIALNUM  INT,
   DEPT       REF(DEPT),
   SALARY     DECIMAL(10,2))
  MODE DB2SQL

CREATE TYPE MGR UNDER EMP AS
  (BONUS     DECIMAL(10,2))
  MODE DB2SQL
```

例 3: アドレスのタイプ階層を作成します。アドレスは、列のタイプとして使用するためのものです。インライン長は指定されていないので、DB2 がデフォルト長を計算します。該当のアドレスが、特定の入力アドレスにどのくらい近いかを計算する外部メソッドを、アドレス・タイプ定義内にカプセル化します。CREATE METHOD ステートメントを使ってメソッド本体を作成します。

```
CREATE TYPE address_t AS
  (STREET    VARCHAR(30),
   NUMBER    CHAR(15),
   CITY      VARCHAR(30),
   STATE     VARCHAR(10))
  NOT FINAL
  MODE DB2SQL
  METHOD SAMEZIP (addr address_t)
  RETURNS INTEGER
  LANGUAGE SQL
  DETERMINISTIC
  CONTAINS SQL
  NO EXTERNAL ACTION,

  METHOD DISTANCE (address_t)
  RETURNS FLOAT
  LANGUAGE C
  DETERMINISTIC
  PARAMETER STYLE SQL
  NO SQL
  NO EXTERNAL ACTION

CREATE TYPE germany_addr_t UNDER address_t AS
  (FAMILY_NAME VARCHAR(30))
  NOT FINAL
  MODE DB2SQL
```

CREATE TYPE (構造化)

```
CREATE TYPE us_addr_t UNDER address_t AS
(ZIP VARCHAR(10))
NOT FINAL
MODE DB2SQL
```

例 4: ネストされた構造化タイプ属性をもつタイプを作成します。

```
CREATE TYPE PROJECT AS
(PROJ_NAME VARCHAR(20),
PROJ_ID INTEGER,
PROJ_MGR MGR,
PROJ_LEAD EMP,
LOCATION ADDR_T,
AVAIL_DATE DATE)
MODE DB2SQL
```

CREATE TYPE MAPPING

CREATE TYPE MAPPING ステートメントは、以下のデータ・タイプ間のマッピングを定義します。

- フェデレーテッド・データベースに定義される予定の、データ・ソース表またはビューの列のデータ・タイプ。
- フェデレーテッド・データベースに定義済みの、対応するデータ・タイプ。

マッピングによって、フェデレーテッド・データベース・データ・タイプを以下に含まれているデータ・タイプに関連付けることができます。

- 指定したデータ・ソース
- データ・ソースの範囲。たとえば、特定のタイプおよびバージョンのすべてのデータ・ソース

データ・タイプのマッピングは、既存のデータ・タイプでは不十分な場合にのみ作成する必要があります。

ニックネームの作成時または表の作成時 (透過性 DDL) に複数のタイプ・マッピングが適用できる場合、最新のマッピングが適用されます。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

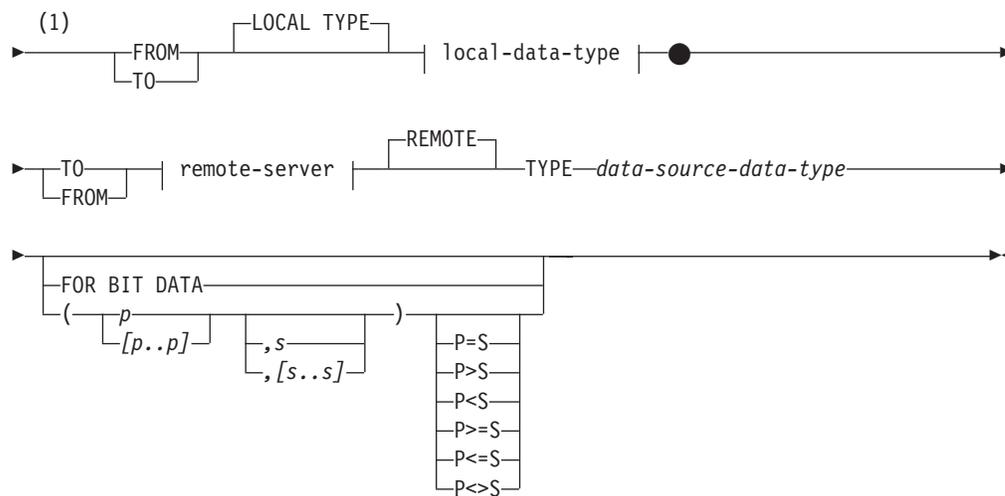
許可

このステートメントの許可 ID が持つ特権には、 SYSADM または DBADM 権限が含まれている必要があります。

構文

```
→ CREATE TYPE MAPPING type-mapping-name →
```

CREATE TYPE MAPPING

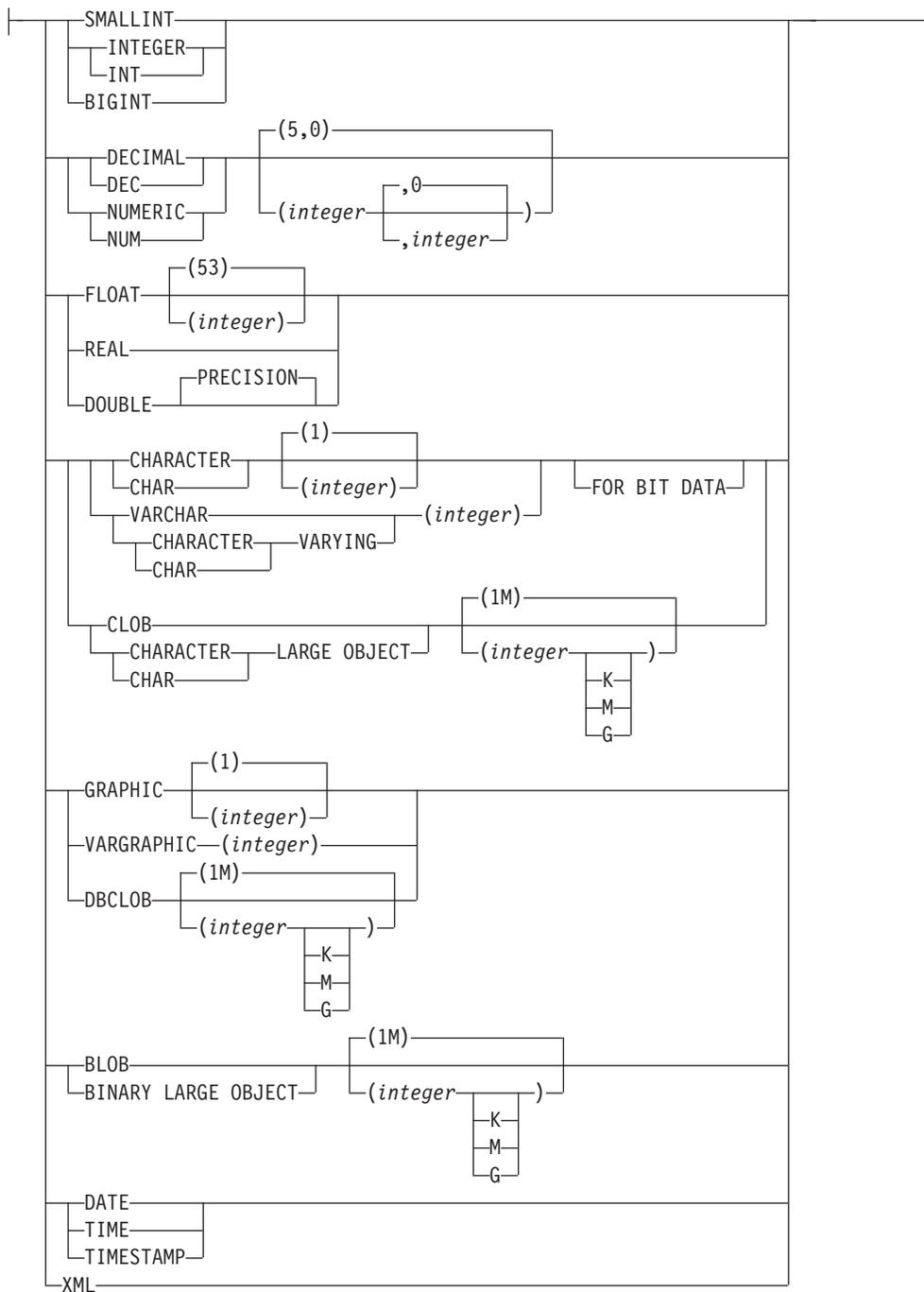


local-data-type:



built-in-type:

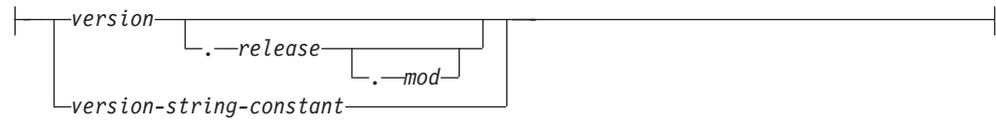
CREATE TYPE MAPPING



remote-server:



server-version:



注:

- 1 CREATE TYPE MAPPING ステートメントには、 TO キーワードと FROM キーワードの両方を指定する必要があります。

説明

type-mapping-name

データ・タイプ・マッピングに名前を付けます。この名前は、カタログですでに記述されているデータ・タイプ・マッピングを指定するものであってはなりません。 *type-mapping-name* を指定しないと、ユニークな名前が生成されます。

FROM または TO

リバースまたはフォワード・タイプ・マッピングを指定します。

FROM

local-data-type が続く場合はフォワード・タイプ・マッピングを、*remote-server* が続く場合はリバース・タイプ・マッピングを指定します。

TO

remote-server が続く場合はフォワード・タイプ・マッピングを、*local-data-type* が続く場合はリバース・タイプ・マッピングを指定します。

local-data-type

フェデレーテッド・データベースに定義したデータ・タイプを指定します。*local-data-type* がスキーマ名なしで指定される場合、SQL パスでスキーマを検索することにより、タイプ名は解決されます。

パラメーター化データ・タイプには、空の括弧を使用できます。特定の長さ、位取り、または精度を指定して定義可能なデータ・タイプのことを、パラメーター化データ・タイプといいます。フォワード・タイプ・マッピングに CHAR() のような空の括弧を指定すると、長さはリモート表の列の長さから判別されます。リバース・タイプ・マッピングに空の括弧を指定すると、タイプ・マッピングはその長さのデータ・タイプにも適用されます。括弧をすべて省略した場合は、データ・タイプのデフォルト長が使用されます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

local-data-type を、DECFLOAT、LONG VARCHAR、LONG VARGRAPHIC、またはユーザー定義タイプにすることはできません (SQLSTATE 42611)。

SERVER *server-name*

data-source-data-type が定義されているデータ・ソースを指名します。

SERVER TYPE *server-type*

data-source-data-type が定義されているデータ・ソースのタイプを指定します。

VERSION

data-source-data-type が定義されているデータ・ソースのバージョンを指定します。

CREATE TYPE MAPPING

version

バージョン番号を指定します。値は整数でなければなりません。

release

version で示されたバージョンのリリース番号を指定します。値は整数でなければなりません。

mod

release で示されたリリースのモディフィケーション番号を指定します。値は整数でなければなりません。

version-string-constant

バージョンの正式名称を指定します。 *version-string-constant* は単一値 (例えば、'8i') にすることができます。あるいは、*version*、*release*、そして該当する場合は *mod* を連結した値にすることができます (例えば、'8.0.3')。

WRAPPER *wrapper-name*

server-type および *server-version* に示されたタイプおよびバージョンのデータ・ソースと対話するために、フェデレーテッド・サーバーが使用するラッパーの名前を指定します。

TYPE *data-source-data-type*

ローカル・データ・タイプとの間でマッピングされるデータ・ソースのデータ・タイプを指定します。

パラメーター化データ・タイプには、空の括弧を使用できます。フォワード・タイプ・マッピングに CHAR() のような空の括弧を指定すると、タイプ・マッピングはどの長さのデータ・タイプにも適用されます。リバース・タイプ・マッピングに空の括弧を指定すると、長さは透過性 DDL に指定されている列の長さから判別されます。括弧をすべて省略した場合は、データ・タイプのデフォルト長が使用されます。

data-source-data-type は、組み込みデータ・タイプでなければなりません。ユーザー定義タイプを指定することはできません。

server-name がタイプ・マッピングとともに指定されているか、または既存のサーバーがタイプ・マッピングの影響を受ける場合、タイプ・マッピング作成時に *data-source-data-type*、*p*、および *s* が検査されます (SQLSTATE 42611)。

p *p* が指定されている場合、*p* と等しい長さまたは精度を持つデータ・タイプだけがタイプ・マッピングの影響を受けます。

[p1..p2]

フォワード・タイプ・マッピングのみ。10進データ・タイプの場合、*p1* と *p2* は値が取る最小および最大桁数を指定します。ストリング・データ・タイプの場合、*p1* と *p2* は値が取る最小および最大文字数を指定します。いずれにせよ、最大値は最小値以上の値にする必要があります。また、最大値と最小値は両方とも、そのデータ・タイプに関して有効なものでなければなりません。

s *s* が指定されている場合、*s* と等しいスケールを持つデータ・タイプだけがタイプ・マッピングの影響を受けます。

[s1..s2]

フォワード・タイプ・マッピングのみ。10進データ・タイプの場合、*s1* と *s2* は小数点以下の桁数の最小および最大数を指定します。最大値は最小値以上

の値にする必要があります。また、最大値と最小値は両方とも、そのデータ・タイプに関して有効なものでなければなりません。

P [operand] S

10 進データ・タイプの場合、P [operand] S は精度と小数点以下の最大桁数との比較を指定します。たとえば、operand (オペランド) に = を指定すると、精度と小数部分に許容できる最大桁数が同じである場合に、タイプ・マッピングが適用されることを示します。

FOR BIT DATA

data-source-data-type が、ビット・データ用かどうかを示します。データ・ソース・タイプの列にバイナリー値が含まれる場合、これらのキーワードは必須です。この属性が文字データ・タイプで指定されていない場合、データベース・マネージャーがこの属性を決定します。

注

- 所定の作業単位 (UOW) 内の CREATE TYPE MAPPING ステートメントは、以下のいずれかの条件の下では処理できません (SQLSTATE 55007)。
 - ステートメントが 1 つのデータ・ソースを参照していて、次のいずれかがすでに UOW に含まれている。
 - このデータ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント。
 - このデータ・ソース内の表またはビューのニックネーム上のオープン・カーソル。
 - このデータ・ソース内の表またはビューのニックネームに対して発行された INSERT、DELETE、または UPDATE ステートメント。
 - ステートメントがデータ・ソースのカテゴリ (たとえば、特定のタイプおよびバージョンのすべてのデータ・ソースなど) を参照しており、次のいずれかがすでに UOW に含まれている。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネームを参照する SELECT ステートメント。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネーム上のオープン・カーソル。
 - それらのデータ・ソースのいずれかの中の表またはビューのニックネームに対して発行された INSERT、DELETE、または UPDATE ステートメント。
- 複数のタイプ・マッピングが適用できる場合は、最新のマッピングが使用されます。SYSCAT.TYPEMAPPINGS カタログ・ビューの CREATE_TIME 列を照会することにより、タイプ・マッピングの作成時間を検索できます。

例

例 1: Oracle データ・タイプ DATE とデータ・タイプ SYSIBM.DATE との間のフォワード・タイプ・マッピングを作成します。このマッピングが定義された後に作成されるすべてのニックネームについて、データ・タイプ DATE の Oracle 列はデータ・タイプ DATE の DB2 列にマップします。

```
CREATE TYPE MAPPING MY_ORACLE_DATE
FROM LOCAL TYPE SYSIBM.DATE
TO SERVER TYPE ORACLE
REMOTE TYPE DATE
```

CREATE TYPE MAPPING

例 2: データ・タイプ SYSIBM.DECIMAL(10,2) とデータ・ソース ORACLE1 の Oracle データ・タイプ NUMBER([10..38],2) との間のフォワード・タイプ・マッピングを作成します。データ・タイプ NUMBER(11,2) の Oracle 表にある列は、11 が 10 と 38 の間に位置するので、データ・タイプ DECIMAL(10,2) の列にマップされます。

```
CREATE TYPE MAPPING MY_ORACLE_DEC
FROM LOCAL TYPE SYSIBM.DECIMAL(10,2)
TO SERVER ORACLE1
REMOTE TYPE NUMBER([10..38],2)
```

例 3: データ・タイプ SYSIBM.VARCHAR(*p*) とデータ・ソース ORACLE1 の Oracle データ・タイプ CHAR(*p*) との間のフォワード・タイプ・マッピングを作成します (*p* は任意の長さ)。データ・タイプ CHAR(10) の Oracle 表にある列は、データ・タイプ VARCHAR(10) の列にマップされます。

```
CREATE TYPE MAPPING MY_ORACLE_CHAR
FROM LOCAL TYPE SYSIBM.VARCHAR()
TO SERVER ORACLE1
REMOTE TYPE CHAR()
```

例 4: データ・ソース ORACLE2 の Oracle データ・タイプ NUMBER(10,2) とデータ・タイプ SYSIBM.DECIMAL(10,2) との間のリバース・タイプ・マッピングを作成します。透過性 DDL を使用して Oracle 表を作成し、データ・タイプ DECIMAL(10,2) の列を指定すると、DB2 はデータ・タイプ NUMBER(10,2) の列を持つ Oracle 表を作成します。

```
CREATE TYPE MAPPING MY_ORACLE_DEC
TO LOCAL TYPE SYSIBM.DECIMAL(10,2)
FROM SERVER ORACLE2
REMOTE TYPE NUMBER(10,2)
```

CREATE USER MAPPING

CREATE USER MAPPING ステートメントは、フェデレーテッド・データベースを使用する許可 ID と、指定したデータ・ソースで使用する許可 ID およびパスワードとの間のマッピングを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID がデータ・ソースへマップされる許可名とは異なる場合、そのステートメントの許可 ID が持つ特権には SYSADM または DBADM 権限が含まれている必要があります。許可 ID と許可名が一致する場合には、必要とされる特権または権限はありません。

構文

```

▶▶ CREATE USER MAPPING FOR authorization-name SERVER server-name
    USER
▶▶ OPTIONS (
    ADD user-mapping-option-name string-constant
)

```

説明

authorization-name

ユーザーまたはアプリケーションがフェデレーテッド・データベースへ接続するときの、許可名を指定します。 *authorization_name* は REMOTE_AUTHID ユーザー・マッピング・オプションにマップされます。

USER

USER 特殊レジスターの値。USER が指定されている場合、CREATE USER MAPPING ステートメントを出す許可 ID は REMOTE_AUTHID ユーザー・マッピング・オプションにマップされます。

SERVER *server-name*

authorization-name (許可名) がアクセスできるデータ・ソースのサーバー・オブジェクトを指定します。 *server-name* はフェデレーテッド・データベースに登録されているリモート・サーバーのローカル名です。

OPTIONS

ユーザー・マッピングを作成したときに使用可能にされるオプションを指示します。

ADD

1 つ以上のユーザー・マッピング・オプションを使用可能にします。

user-mapping-option-name

オプションの名前を指定します。

string-constant

user-mapping-option-name の設定を、文字ストリング定数として指定します。

注

- ユーザー・マッピングは、DB2 ファミリー製品、Documentum、Informix、Microsoft SQL Server、ODBC、Oracle、Sybase、および Teradata のデータ・ソースにのみ必要です。
- REMOTE_PASSWORD オプションはユーザー・マッピングには常に必要です。

例

例 1: DB2 for z/OS データ・ソース・サーバー・オブジェクト SERVER390 へのユーザー・マッピングを登録します。ローカル・フェデレーテッド・データベースの

CREATE USER MAPPING

許可名を SERVER390 のユーザー ID とパスワードにマップします。許可名は RSPALTEN です。SERVER390 のユーザー ID は SYSTEM です。SERVER390 のパスワードは MANAGER です。

```
CREATE USER MAPPING FOR RSPALTEN
SERVER SERVER390
OPTIONS
(REMOTE_AUTHID 'SYSTEM',
REMOTE_PASSWORD 'MANAGER')
```

例 2: Oracle データ・ソース・サーバー・オブジェクト ORACLE1 へのユーザー・マッピングを登録します。MARCR は、ローカルのフェデレーテッド・データベースの許可名で、ORACLE1 のユーザー ID です。許可名とユーザー ID が同じなので、ユーザー・マッピングに REMOTE_AUTHID オプションを指定する必要はありません。ORACLE1 上の MARCR のパスワードは NZXCZY です。

```
CREATE USER MAPPING FOR MARCR
SERVER ORACLE1
OPTIONS
(REMOTE_PASSWORD 'NZXCZY')
```

CREATE VARIABLE

CREATE VARIABLE ステートメントは、グローバル変数を定義します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

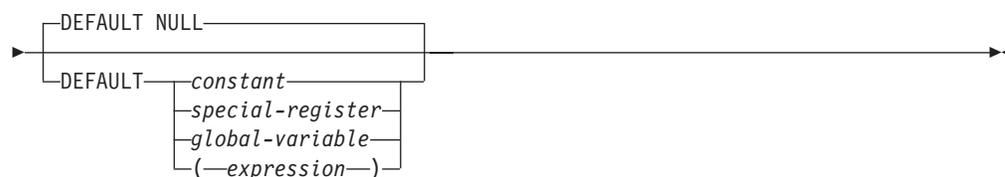
- データベースに対する IMPLICIT_SCHEMA 権限 (変数の暗黙または明示のスキーマ名が存在しない場合)
- スキーマに対する CREATEIN 特権 (変数のスキーマ名が既存のスキーマを指している場合)
- SYSADM または DBADM 権限

さらに、デフォルトの式を実行するために必要なすべての特権。

EXPRESSION 節で参照されるオブジェクトに対する権限を検査する際、グループ特権は考慮されません。

構文

```
▶▶ CREATE VARIABLE variable-name | data-type |
```

**data-type:****説明***variable-name*

グローバル変数の名前を指定します。名前 (暗黙修飾子または明示修飾子を含む) は、現行サーバーに既に存在するグローバル変数を指定するものであってはなりません (SQLSTATE 42710)。修飾子が指定されなければ、暗黙的に現行スキーマが指定されます。

data-type

グローバル変数のデータ・タイプを指定します。データ・タイプは、組み込みデータ・タイプ、特殊タイプ、または参照タイプにすることができます。

built-in-data-type

組み込みデータ・タイプを指定します。組み込みデータ・タイプの説明については、『CREATE TABLE』を参照してください。グローバル変数では、CLOB、DBCLOB、BLOB、LONG VARCHAR、LONG VARGRAPHIC、XML、ARRAY、または構造化タイプを指定できませんのでご注意ください (SQLSTATE 42611)。

FOR BIT DATA は、文字ストリング・データ・タイプの一部として指定することができます。

distinct-type-name

特殊タイプを指定します。グローバル変数の長さ、精度、および位取りは、それぞれ特殊タイプのソース・タイプの長さ、精度、および位取りになります。 *distinct-type-name* がスキーマ名なしで指定される場合、SQL パスでスキーマを検索することにより、*distinct-type-name* は解決されます。組み込みタイプに適用される制限と同じ制限が、特殊タイプのソース・タイプに適用されます。

reference-type-name

参照タイプを指定します。グローバル変数の長さ、精度、および位取りは、それぞれ参照タイプのソース・タイプの長さ、精度、および位取りになります。 *reference-type-name* がスキーマ名なしで指定される場合、SQL パスでスキーマを検索することにより、*reference-type-name* は解決されます。組み込みタイプに適用される制限と同じ制限が、参照タイプのソース・タイプに適用されます。

DEFAULT

グローバル変数のデフォルト値を指定します。値は、定数、特殊レジスター、グ

CREATE VARIABLE

ローカル変数、式、またはキーワード NULL にすることができます。この式は、『式』で説明されているいずれかのタイプの式とすることができます。デフォルト値が指定されていない場合、変数は NULL 値に初期化されます。式の最大サイズは 64K です。

デフォルトの式は、SQL データを変更したり (SQLSTATE 428FL)、外部アクションを実行したり (SQLSTATE 42845) してはなりません。式は、その変数のデータ・タイプと割り当てに互換性があるものでなければなりません。

注

- グローバル変数にはセッション有効範囲があります。これは、グローバル変数はデータベースでアクティブであるすべてのセッションで使用できますが、それらの値は各セッション専用であるということを意味します。
- グローバル変数の値の変更は、トランザクションの制御下にありません。トランザクションが COMMIT または ROLLBACK ステートメントで終了した場合、グローバル変数の値は保存されます。
- **グローバル変数を使用するための特権:** このステートメントで作成されたグローバル変数に対する読み書きを試行するには、このアクションの試行で使用する許可 ID にそのグローバル変数への適切な特権があることが必要です。変数の定義者には暗黙的にその変数へのすべての特権が付与されます。
- **デフォルト値の設定:** 作成されたグローバル変数は、その指定された有効範囲内で最初に参照される時にそのデフォルト値にインスタンス化されます。グローバル変数がステートメントで参照される場合、そのステートメントの制御フローとは独立してインスタンス化される点に注意してください。
- **新規に作成されたセッション・グローバル変数の使用:** グローバル変数がセッション内で作成された場合、その作業単位がコミットするまで、それを他のセッションで使用することはできません。ただし、新規グローバル変数は、作業単位がコミットする前にその変数を作成したセッション内では使用できます。

例

例 1: セッションにどのプリンターを使用するかを示すグローバル変数を作成します。

```
CREATE VARIABLE MYSCHEMA.MYJOB_PRINTER VARCHAR(30)
DEFAULT 'Default printer'
```

例 2: ある従業員が働いている部門を示すグローバル変数を作成します。

```
CREATE VARIABLE SCHEMA1.GV_DEPTNO INTEGER
DEFAULT ((SELECT DEPTNO FROM HR.EMPLOYEES
WHERE EMPUSER = SESSION_USER))
```

例 3: 現行ユーザーのセキュリティー・レベルを示すグローバル変数を作成します。

```
CREATE VARIABLE SCHEMA2.GV_SECURITY_LEVEL INTEGER
DEFAULT (GET_SECURITY_LEVEL (SESSION_USER))
```

CREATE VIEW

CREATE VIEW ステートメントは、1 つまたは複数の表、ビュー、またはニックネームに基づくビューを定義します。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 全選択に指定された表、ビュー、またはニックネームのそれぞれに対して、
 - その表またはビューに対する CONTROL 特権、または
 - その表またはビューに対する SELECT 特権

および以下の少なくとも 1 つ

- データベースに対する IMPLICIT_SCHEMA 権限 (ビューの暗黙または明示のスキーマ名が存在しない場合)
- スキーマに対する CREATEIN 特権 (ビューのスキーマ名が既存のスキーマを指している場合)

サブビューを作成するには、このステートメントの許可 ID が以下の条件に適合している必要があります。

- 表階層のルート表の所有者と同じ名前である。
- サブビューの基礎表に対する SELECT WITH GRANT 権限を持っている。または、スーパービューの SELECT 権限がビュー定義者以外のユーザーに与えられていない。

WITH ROW MOVEMENT を指定する場合は、

- その表またはビューに対する UPDATE 特権
- SYSADM または DBADM 権限

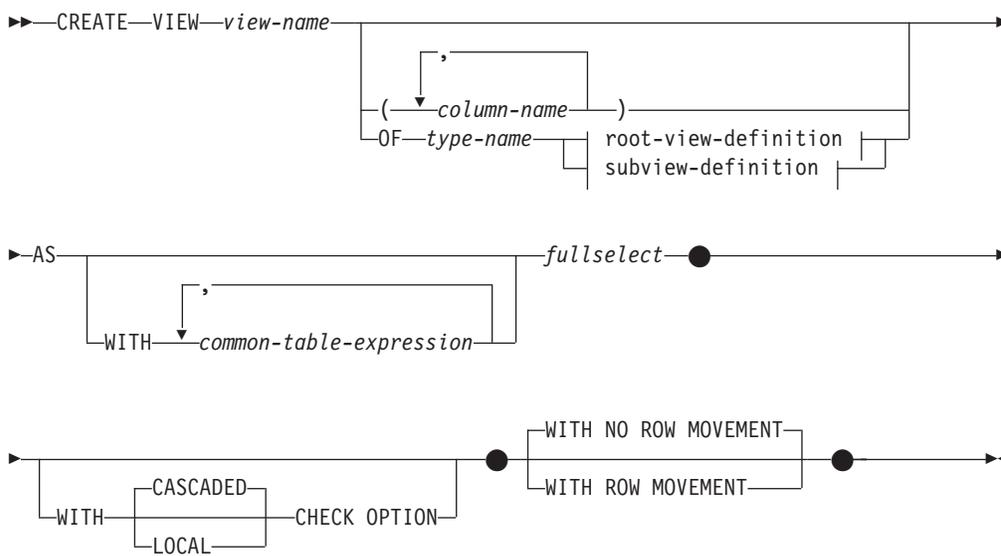
グループ特権は、CREATE VIEW ステートメントで指定された表やビューに対しては考慮されません。

特権は、フェデレーテッド・データベースのニックネームにビューを定義するときには考慮されません。このニックネームで示されている表またはビューのデータ・ソースの許可要件は、照会の処理時に適用されます。ステートメントの許可 ID は、別のリモート許可 ID へマップできます。

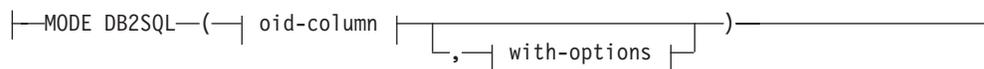
ビューの定義者が SYSADM 権限を持つために、ビューの作成しかできない場合、ビュー作成のため、その定義者には明示的な DBADM 権限が付与されます。

CREATE VIEW

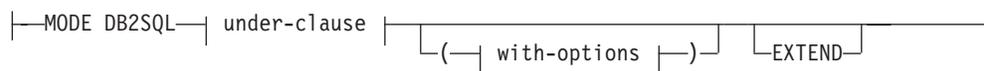
構文



root-view-definition:



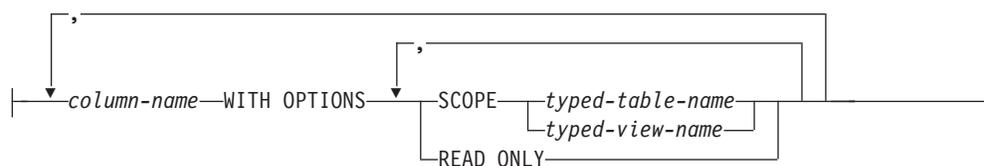
subview-definition:



oid-column:



with-options:



under-clause:



説明

view-name

ビューの名前を指定します。暗黙または明示の修飾子を含む名前は、カタログに記述されている表、ビュー、ニックネーム、または別名を指定するものであってはなりません。修飾子は、SYSIBM、SYSCAT、SYSFUN、または SYSSTAT であってはなりません (SQLSTATE 42939)。

この名前は、作動不能なビューの名前と同じであっても構いません (『作動不能ビュー』を参照)。このような場合、作動不能なビューは、CREATE VIEW ステートメントに指定した新しいビューによって置き換えられます。作動不能なビューが置き換えられると、ユーザーに警告 (SQLSTATE 01595) が出されます。BIND オプション SQLWARN を NO に設定してアプリケーションがバインドされた場合は、警告は戻されません。

column-name

ビューの列の名前を指定します。列名のリストを指定する場合、リスト中の列の名前の数は、全選択の結果表の列の数と同じ数でなければなりません。各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。列名のリストの指定がない場合、ビューの列は、全選択の結果表の列名を継承します。

全選択の結果表の列名が重複している場合、または無名の列がある場合には、列名のリストを指定する必要があります (SQLSTATE 42908)。無名列とは、定数、関数、式、またはセット演算から派生した列で、選択リストの AS 節によって名前が指定されていない列を指します。

OF *type-name*

ビューの列が *type-name* で指定される構造化タイプの属性に基づいていることを指定します。 *type-name* の指定にスキーマ名が含まれていない場合、そのタイプ名は SQL パス上のスキーマを探索することによって決まります (このパスは、静的 SQL の場合は FUNCPATH プリプロセス・オプションによって、動的 SQL の場合は CURRENT PATH レジスターによって定義されます)。ここに指定するタイプ名は、既存のユーザー定義タイプ名で (SQLSTATE 42704)、かつインスタンス化の可能な構造化タイプでなければなりません (SQLSTATE 428DP)。

MODE DB2SQL

この節は、型付きビューのモードを指定するために使用されます。これは、現在サポートされている唯一有効なモードです。

UNDER *superview-name*

このビューが *superview-name* のサブビューであることを指定します。スーパービューは既存のビューでなければならず (SQLSTATE 42704)、このビューは *type-name* のすぐ上位にあるスーパータイプである構造化タイプで定義する必要があります (SQLSTATE 428DB)。 *view-name* と *superview-name* のスキーマ名は、同じでなければなりません (SQLSTATE 428DQ)。 *superview-name* で指定されるビューには、 *type-name* で既に定義された既存のサブビューを含めることはできません (SQLSTATE 42742)。

表の列には、スーパービューのオブジェクト ID 列が含まれています。オブジェクト ID 列のタイプは `REF(type-name)` に変更されており、`type-name` の属性に基づく列が続きます (ここでいうタイプには、スーパータイプの属性も含まれていることを念頭に置いてください)。

INHERIT SELECT PRIVILEGES

スーパービューに対して `SELECT` 特権を持つユーザーやグループはすべて、新しく作成したサブビューに対しても同様の特権を付与されます。この特権は、サブビュー定義者によって付与されたものと見なされます。

OID-column

型付きビューのオブジェクト ID 列を定義します。

REF IS OID-column-name USER GENERATED

オブジェクト ID (OID) 列をビューの最初の列として定義することを指定します。ビュー階層のルート・ビューには、OID が必須です (SQLSTATE 428DX)。このビューはサブビュー以外の型付きビュー (OF 節が必須) でなければなりません (SQLSTATE 42613)。この列の名前は `OID-column-name` という形式で定義されますが、構造化タイプ `type-name` のどの属性の名前とも同一にすることはできません (SQLSTATE 42711)。 `fullselect` で指定した最初の列は、`REF(type-name)` というタイプでなければなりません (キャストして適切なタイプにする必要があるかもしれません)。 `UNCHECKED` を指定しない場合、索引 (主キー、ユニーク制約、ユニーク索引、または OID 列) を使用してユニーク性を強制できる列 (NULL 可能ではない) に基づいている必要があります。この列をオブジェクト ID 列 または OID 列 といいます。 `USER GENERATED` というキーワードは、行を挿入する際にユーザーが OID 列の初期値を提供しなければならないことを指しています。行を挿入した後は、OID 列を更新することはできません (SQLSTATE 42808)。

UNCHECKED

固有であることをシステムが証明できない場合でも、型付きビュー定義のオブジェクト ID の列を固有であると見なすように定義します。この属性は、次のような型付きビュー階層に定義されている表またはビューでの使用を想定しています。すなわち、そのデータが固有性規則に準拠しているものの、システムが固有性を証明できる規則には準拠していないことをユーザーが認識しているという場合です。 `UNCHECKED` オプションは、複数の階層や従来型の表またはビューにわたる範囲を持つビュー階層には必須のオプションです。 `UNCHECKED` を指定する場合、ユーザーの責任でビューの各行にユニークな OID が確実にあるようにします。ユーザーがこの特性を保証しなかったために、ビューに重複した OID 値が入ってしまうと、ユニークでない OID 値のどれかを含むパスの式または `DEREF` 演算子はエラーになります (SQLSTATE 21000)。

with-options

型付きビューの列に適用される追加オプションを定義します。

column-name WITH OPTIONS

追加オプションを指定する列の名前を指定します。 `column-name` は、ビューの `type-name` に定義されている (継承されていない) 属性名に対応していなければなりません。この列は参照タイプである必要があります (SQLSTATE 42842)。また、すでにスーパービューに存在する列に対応する

ことはできません (SQLSTATE 428DJ)。列名は、ステートメント内の 1 つの WITH OPTIONS SCOPE 節に 1 回しか指定できません (SQLSTATE 42613)。

SCOPE

参照タイプ列の有効範囲を指定します。間接参照演算子の左オペランド、または Deref 関数の引数として使用する列には、すべて有効範囲を指定する必要があります。

ターゲット表またはターゲット・ビューが定義するために、後続する ALTER VIEW ステートメント (有効範囲が継承されていない場合) まで、参照タイプ列の有効範囲指定を遅らせることができます (通常は、相互参照表および相互参照ビューの場合に適用する)。ビューの参照タイプ列で有効範囲が指定されていないのに、基礎表またはビュー列の有効範囲が指定された場合、基礎列の有効範囲は参照タイプ列によって継承されます。基礎表またはビューの列に有効範囲がない場合には、この列に有効範囲は指定されません。有効範囲と参照タイプ列についての詳細は、643 ページの『注』を参照してください。

typed-table-name

型付き表の名前。この表は既に存在しているものか、作成する表と同じ名前のものでなければなりません (SQLSTATE 42704)。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-table-name* のタイプを表します (SQLSTATE 428DM)。値が *typed-table-name* の既存行を実際に参照していることを確認するための、 *column-name* の既存値の検査は行われません。

typed-view-name

型付きビューの名前。このビューは既に存在しているものか、作成するビューと同じ名前のものでなければなりません (SQLSTATE 42704)。 *column-name* のデータ・タイプは REF(S) でなければなりません。 S は *typed-view-name* のタイプを表します (SQLSTATE 428DM)。値が *typed-view-name* の既存行を実際に参照していることを確認するための、 *column-name* の既存値の検査は行われません。

READ ONLY

列を読み取り専用列として指定します。このオプションは、列を読み取り専用指定し、サブビュー定義は、同じ列の式を暗黙的に読み取り専用指定することができます。

AS

ビュー定義を指定します。

WITH *common-table-expression*

後続の fullselect で使用する共通表式を定義します。型付きビューを定義するときには、共通表式は指定できません。

fullselect

ビューを定義します。ビューは常に、SELECT が実行された場合の結果となる複数行で構成されます。全選択でホスト変数、パラメーター・マーカー、または宣言済み一時表を参照することはできません。ただし、パラメーター化されたビューを SQL 表関数として作成することは可能です。

全選択では、FROM 節に SQL データ変更ステートメントを組み込みません (SQLSTATE 428FL)。

型付きビューおよびサブビューの場合: *fullselect* は、以下の規則に準拠していなければなりません。そうでない場合、エラーが戻されます (特に他の指定がなければ、SQLSTATE 428EA)。

- 全選択に、DBPARTITIONNUM または HASHEDVALUE 関数、非 deterministic 関数、または外部アクションを持つように定義されている関数への参照を含めることはできません。
- ビューの本体は、単一の副選択か複数の副選択の UNION ALL で構成する必要があります。ビューの本体に直接加わっている各副選択を、ビューの分岐と呼びます。ビューには、1 つかそれ以上の分岐があります。
- 各分岐の FROM 節は、単一の表またはビュー (その分岐の基礎 表またはビューといい、必ずしも型付きではない) で構成される必要があります。
- 各分岐の基礎表またはビューは、別々の階層にする必要があります (つまり、ビューは、同じ階層内の基礎表またはビューが付いた複数の分岐を持つことはできません)。
- 型付きビュー定義の分岐はいずれも GROUP BY または HAVING を指定できません。
- ビューの本体に UNION ALL が含まれる場合、階層内にあるルート・ビューの OID 列に UNCHECKED オプションを指定する必要があります。

ビューおよびサブビューの階層の場合: BR1 および BR2 が、階層内のビュー定義に現れる分岐になるようにします。T1 を BR1 の基礎表またはビューに、T2 を BR2 の基礎表またはビューにします。この場合は以下のようになります。

- T1 および T2 が同じ階層でない場合、ビュー階層にあるルート・ビューの OID 列に UNCHECKED オプションを指定する必要があります。
- T1 および T2 が同じ階層にある場合、行セットが結合しないことを十分保証する述部または ONLY 節を、BR1 および BR2 に含める必要があります。

EXTEND AS を使って定義された型付きのビューの場合: サブビューの本体内の各分岐について:

- 各分岐の基礎表は、即時スーパービューのいくつかの基礎表の副表 (必ずしも適切ではない) でなければなりません。
- SELECT リストの式は、サブビューの非継承列に割り当てることができるものでなければなりません (SQLSTATE 42854)。

AS (EXTEND なし) を使って定義された型付きサブビューの場合:

- サブビューの本体内にあるそれぞれの分岐について、SELECT リストにある式は、サブビューの継承列と非継承列の宣言済みタイプに割り当てられるようにする必要があります (SQLSTATE 42854)。
- サブビューで指定した階層上のそれぞれの分岐の OID 式は、ルート・ビュー内の同じ階層上の分岐の OID 式と同じでなければなりません (キャスト以外)。
- スーパービュー内の READ ONLY として (暗黙的または明示的に) 指定されていない列の式は、そのサブビュー内の同じ基礎階層上のすべての分岐と同じでなければなりません。

WITH CHECK OPTION

ビューによって挿入または更新される行すべてが、ビューの定義に従っていないければならないという制約を指定します。ビューの定義に従わない行とは、ビューの検索条件を満たしていない行です。

WITH CHECK OPTION は、以下のいずれかの条件が真である場合には指定できません。

- ビューが読み取り専用である場合 (SQLSTATE 42813)。挿入が許されていない更新可能なビューに対して WITH CHECK OPTION を指定すると、制約は更新にのみ適用されます。
- ビューが、DBPARTITIONNUM または HASHEDVALUE 関数、非 deterministic 関数、または外部アクションを伴う関数を参照する場合 (SQLSTATE 42997)。
- ニックネームがビューの更新の対象である場合。
- ビューの更新の対象である INSTEAD OF トリガーが定義されているビューである場合 (SQLSTATE 428FQ)。

WITH CHECK OPTION を省略すると、ビューを使用するどのような挿入操作または更新操作のチェックにおいても、ビューの定義は使用されません。ただし、ビューが WITH CHECK OPTION が指定された他のビューに直接または間接的に従属する場合には、挿入操作または更新操作の過程で、何らかのチェックが行われる場合があります。この場合、ビューの定義が使用されるわけではないため、ビューの定義に従っていないビューを介して、行が挿入または更新される可能性があります。

CASCADED

ビュー *V* に対する WITH CASCADED CHECK OPTION 制約は、*V* が従属するいずれかの更新可能ビューから、制約としての検索条件を *V* が継承することを意味します。さらに、*V* に従属するすべての更新可能ビューも、このような制約の対象になります。したがって、*V* の検索条件と、*V* が従属している各ビューの検索条件との AND を取ったものが、*V* あるいは *V* に従属するいずれかのビューの挿入または更新に対して適用される制約となります。

LOCAL

ビュー *V* に対する WITH LOCAL CHECK OPTION 制約は、*V* の検索条件が、*V* または *V* に従属するいずれかのビューの挿入あるいは更新に対する制約として適用されることを意味しています。

次の例は、CASCADED と LOCAL の差異を示しています。次のような更新可能なビューを想定します (Y は、下記の表の列見出しに示しているように、LOCAL または CASCADED に置き換えます)。

```
V1 defined on table T
V2 defined on V1 WITH Y CHECK OPTION
V3 defined on V2
V4 defined on V3 WITH Y CHECK OPTION
V5 defined on V4
```

次の表は、挿入または更新された行を検査するのに使われる検索条件を示しています。

CREATE VIEW

	Y が LOCAL の場合	Y が CASCADED の場合
V1 でのチェック条件:	対象となるビューなし	対象となるビューなし
V2 でのチェック条件:	V2	V2、V1
V3 でのチェック条件:	V2	V2、V1
V4 でのチェック条件:	V2、V4	V4、V3、V2、V1
V5 でのチェック条件:	V2、V4	V4、V3、V2、V1

また、次のような更新可能ビューについても考えてみます。これは、デフォルトの CASCADED オプションを使用した場合の WITH CHECK OPTION の効果を示しています。

```
CREATE VIEW V1 AS SELECT COL1 FROM T1 WHERE COL1 > 10
```

```
CREATE VIEW V2 AS SELECT COL1 FROM V1 WITH CHECK OPTION
```

```
CREATE VIEW V3 AS SELECT COL1 FROM V2 WHERE COL1 < 100
```

次の INSERT ステートメントは V1 を使用するものですが、V1 に WITH CHECK OPTION が指定されておらず、また V1 が、WITH CHECK OPTION の指定された他のどのビューにも従属していないため、このステートメントは成功します。

```
INSERT INTO V1 VALUES(5)
```

次の INSERT ステートメントは V2 を使用するものですが、V2 に WITH CHECK OPTION が指定されており、挿入 (INSERT) によって V2 の定義に従っていない行が作成されるため、このステートメントはエラーになります。

```
INSERT INTO V2 VALUES(5)
```

次の INSERT ステートメントでは V3 を使用しています。V3 に WITH CHECK OPTION は指定されていませんが、これは、WITH CHECK OPTION の指定された V2 の従属であるため、エラーになります (SQLSTATE 44000)。

```
INSERT INTO V3 VALUES(5)
```

次の INSERT ステートメントも、V3 を使用しています。これは V3 の定義に準拠していませんが、成功します (V3 には WITH CHECK OPTION が指定されていません)。これは、WITH CHECK OPTION の指定された V2 の定義に従ったものになっています。

```
INSERT INTO V3 VALUES(200)
```

WITH NO ROW MOVEMENT または WITH ROW MOVEMENT

基礎表のチェック制約に違反する方法で行が更新されたときに、更新可能 UNION ALL ビューに対して行うアクションを指定します。デフォルトは WITH NO ROW MOVEMENT です。

WITH NO ROW MOVEMENT

基礎表のチェック制約に違反する方法で行が更新されたときに、エラー (SQLSTATE 23513) を戻すよう指定します。

WITH ROW MOVEMENT

表のチェック制約に対する違反があっても、更新された行を該当する基礎表に移動させるよう指定します。

行の移動には、チェック制約に違反する行の削除と、それらの行のビューへの再挿入が関係します。WITH ROW MOVEMENT 節を指定できるのは、UNION ALL ビューの列がすべて更新可能になっている場合だけです。

(SQLSTATE 429BJ)。行が削除されたのと同じ基礎表に行が挿入される (おそらく、トリガー起動の後) 場合は、エラーが戻されます (SQLSTATE 23524)。WITH ROW MOVEMENT 節を使用して定義されたビューには、最外部の全選択を除き、ネストされた UNION ALL 操作を含めることはできません (SQLSTATE 429BJ)。

注

- まだ存在していないスキーマ名を用いてビューを作成すると、ステートメントの許可 ID に IMPLICIT_SCHEMA 権限がある場合に限り、そのスキーマが暗黙的に作成されます。そのスキーマの所有者は SYSIBM です。スキーマに対する CREATEIN 特権が PUBLIC に付与されます。
 - ビュー列は、列が式から派生するとき以外は、基本表またはビューの NOT NULL WITH DEFAULT 属性を継承します。基本表に制約が定義されている場合、更新可能ビューに行が挿入または更新される時、それらの制約 (主キー、参照整合性性、およびチェック制約) に対するチェックが行われます。
 - 定義の中で作動不能ビューを使用して新しいビューを作成することはできません (SQLSTATE 51024)。
 - このステートメントでは、宣言済み一時表をサポートしていません (SQLSTATE 42995)。
 - **削除可能ビュー:** 削除操作の INSTEAD OF トリガーがビューに定義されている場合、または下記のすべてが当てはまる場合、ビューは**削除可能**です。
 - 外部全選択の各 FROM 節には、基本表 (OUTER 節なし)、削除可能ビュー (OUTER 節なし)、削除可能なネストした表式、または削除可能な共通表式 (ニックネームが識別不能) のいずれかが 1 つだけ指定されている
 - 外部全選択に VALUES 節が含まれない
 - 外部全選択に GROUP BY 節も HAVING 節も含まれない
 - 外部全選択の選択リストに列関数が含まれない
 - 外部全選択に、UNION ALL を除くセット演算 (UNION、EXCEPT、または INTERSECT) が含まれない
 - UNION ALL のオペランドの基本表が同じ表ではなく、各オペランドが削除可能
 - 外部全選択の選択リストに DISTINCT が含まれない
 - **更新可能ビュー:** 更新操作の INSTEAD OF トリガーがビューに定義されている場合、または下記のすべてが当てはまる場合、ビューの列は**更新可能**です。
 - ビューが削除可能 (削除の INSTEAD OF トリガーとは無関係) であり、列の解決結果が基本表の列 (間接参照操作は使用しない) となり、READ ONLY オプションが指定されない
 - ビューの全選択に UNION ALL が含まれる場合、UNION ALL のオペランドの対応するすべての列のデータ・タイプおよびデフォルト値が正確に一致する (長さ、または精度と位取りを含む)
- ビューのいずれかの列が更新可能なら、ビューは更新可能です。
- 挿入可能なビュー

CREATE VIEW

- 挿入操作の INSTEAD OF トリガーがビューに定義されている場合、またはビューの少なくとも 1 つの列が更新可能であり、ビューの全選択に UNION ALL が含まれない場合、そのビューは挿入可能です。
- 特定の行が基本表のうちの正確に 1 つのチェック制約を満たしている場合のみ、その行をビュー (UNION ALL を含む) に挿入できます。
- 更新不可の列が含まれているビューに挿入するには、それらの列を列リストから除外する必要があります。
- **読み取り専用ビュー:** ビューが読み取り専用 であるのは、削除可能でも、更新可能でも、挿入可能でもない 場合です。

ビューが読み取り専用かどうかは、SYSCAT.VIEWS カタログ・ビューの READONLY 列に示され、INSTEAD OF トリガーが考慮されることはありません。

- 共通表式とネストした表式は、削除可能かどうか、更新可能かどうか、挿入可能かどうか、または読み取り専用かどうかを判別する上で、同じ一連の規則に従います。
- **作動不能ビュー:** 作動不能ビュー とは、SQL ステートメントで使用できなくなったビューのことです。ビューは、次の場合に作動不能になります。
 - ビュー定義が従属している特権が取り消された場合
 - ビュー定義が従属している表、ニックネーム、別名、または関数などのオブジェクトがドロップされた場合
 - ビュー定義が従属しているビューが作動不能になった場合
 - ビュー定義のスーパービューであるビュー (サブビュー) が作動不能になった場合

実際には、作動不能ビューとは、ビュー定義が間違っってドロップされたビューです。例えば、別名がドロップされると、その別名を使用して定義されているビューすべてが作動不能になります。それに従属するすべてのビューも作動不能になり、そのビューに従属するパッケージは無効になります。

作動不能ビューを明示的に再作成するか、あるいはドロップされるまで、その作動不能ビューを使用するステートメントのコンパイルはできません (SQLSTATE 51024)。ただし、CREATE ALIAS、CREATE VIEW、DROP VIEW、および COMMENT ON TABLE の各ステートメントは例外です。作動不能ビューが明示的にドロップされるまで、その修飾名を使って別の表や別名を作成することはできません (SQLSTATE 42710)。

作動不能ビューは、作動不能ビューの定義テキストを使用して、CREATE VIEW ステートメントを発行することにより、再作成することができます。このビュー定義テキストは、SYSCAT.VIEWS カタログの TEXT 列に保管されます。作動不能ビューを再作成する場合は、他のユーザーがそのビューに対して必要となる特権すべてを明示的に付与する必要があります。これは、ビューが作動不能と見なされると、ビューのすべての許可レコードが削除されるためです。作動不能ビューを再作成するために、それを明示的にドロップする必要はありません。作動不能ビューと同じ *view-name* を指定して CREATE VIEW ステートメントを発行すると、作動不能ビューは置き換えられ、CREATE VIEW ステートメントは警告を戻します (SQLSTATE 01595)。

作動不能ビューであることは、SYSCAT.VIEWS カタログ・ビューの VALID 列が X、また SYSCAT.TABLES カタログ・ビューの STATUS 列が X であることによって示されます。

• **特権:**

ビューの定義者は、ビューに対する SELECT 特権と、ビューをドロップする権利を常に与えられます。ビューの定義者は、その定義者が全選択で指定されたすべての基本表、ビューまたはニックネームに対する CONTROL 特権を持っている場合、あるいは定義者が SYSADM 権限または DBADM 権限を持っている場合にのみ、そのビューに対する CONTROL 特権が付与されます。

ビューの定義者は、そのビューが読み取り専用でなく、定義者が基礎となるオブジェクトに対して対応する特権を持っている場合に、そのビューに対する INSERT、UPDATE、列レベルの UPDATE、または DELETE の特権を与えられます。

WITH ROW MOVEMENT でビューが定義された場合は、定義者がビューのすべての列に対する UPDATE 特権を持っており、またすべての基礎表およびビューに対する INSERT および DELETE 特権を持っているなら、その定義者に、そのビューのみでの UPDATE 特権が与えられます。

ビューの定義者にそれらの特権が与えられるのは、それらの特権の派生元の特権がビューの作成時に存在している場合に限りです。定義者は、これらの特権を直接持っているか、または PUBLIC の特権として持っていることが必要です。特権は、フェデレーテッド・サーバーのニックネームにビューを定義するときには考慮されません。ただし、ニックネームにビューを使用する場合、ユーザーの許可 ID には、そのニックネームがデータ・ソースで参照する表またはビューに対する適切な選択特権がなければなりません。もしその特権がなければ、エラーが戻されます。ビューの定義者がメンバーであるグループを持つ特権は考慮されません。

サブビューが作成されると、すぐ上のスーパービューに対して持っている SELECT 特権が自動的にサブビューに対しても付与されます。

• **有効範囲および REF 列:**

ビュー定義の全選択で参照タイプ列を選択する際には、必要なターゲット・タイプと有効範囲について考慮してください。

- 必要なターゲット・タイプおよび有効範囲が基礎表または基礎ビューのものと同じ場合には、参照列をそのまま選択することができます。
- 有効範囲を変更する必要がある場合には、WITH OPTIONS SCOPE 節を使って、必要な有効範囲の表とビューを定義します。
- 参照のターゲット・タイプを変更する必要がある場合には、まず列を参照の対象となっている参照タイプにキャストしてから、新規の参照タイプへもキャストする必要があります。この場合、有効範囲は参照タイプへのキャストで指定できますし、WITH OPTIONS SCOPE 節を使っても指定できます。例えば、REF(TYP1) SCOPE TAB1 として定義された Y 列を選択したとしましょう。そして、この列を REF(VTYP1) SCOPE VIEW1 として定義するとします。この場合、選択リスト項目は次のようになります。

```
CAST(CAST(Y AS VARCHAR(16)) FOR BIT DATA) AS REF(VTYP1) SCOPE VIEW1)
```

- **ID 列:** ビューの列が ID 列と見なされるのは、ビュー定義の全選択内における対応する列のエレメントが、基本表の ID 列の名前に直接または間接的にマップされた、表の ID 列の名前またはビューの列の名前である場合です。

これ以外の場合すべて、ビューの列は ID のプロパティを取得しません。以下に例を示します。

- ビュー定義の選択リストに ID 列の名前のインスタンスが複数含まれている (つまり、同じ列を複数回選択している) 場合
- ビュー定義に結合が関与している場合
- ビュー定義の列に ID 列を参照する式が含まれている場合
- ビュー定義に UNION が含まれている場合

挿入先のビューにおいて、ビュー定義の選択リストに直接または間接的に基本表の ID 列の名前が含まれている場合は、INSERT ステートメントが基本表の ID 列を直接参照する場合と同じ規則が適用されます。

- **フェデレーテッド・ビュー:** フェデレーテッド・ビューとは、全選択内にあるいずれかのニックネームへの参照を含むビューのことです。この種のニックネームが存在する場合、そのビューで使用する許可モデルは、ビューが照会で順番に参照されるときに変更されます。

ビューを作成しても、ビュー定義者がニックネームの基礎データ・ソース表またはビューにアクセスできるかどうかを判別する特権検査は行われません。フェデレーテッド・データベースでの表またはビューに対する特権検査は、最低でもそうしたオブジェクトに対する SELECT 特権をビュー定義者に要求することにより、通常どおり行われます。

フェデレーテッド・ビューが照会で順番に参照される場合、その照会を発行したデータ・ソースおよび許可 ID (または、その照会がマッピングするリモート許可 ID) に対する照会になるニックネームには、データ・ソース表またはビューにアクセスするのに必要な特権がなければなりません。フェデレーテッド・ビューを参照する照会を発行する許可 ID には、フェデレーテッド・サーバーに存在する (フェデレーテッドでない) 表またはビューに対する追加の特権は必要ありません。

- **ROW MOVEMENT、トリガー、および制約:** WITH ROW MOVEMENT 節を使用して定義されたビューが更新される場合、トリガーや制約の操作のシーケンスは次のようになります。
 1. 最終的には移動される行も含め、更新されるすべての行に対して BEFORE UPDATE トリガーが活動化されます。
 2. 更新操作が処理されます。
 3. 更新されるすべての行に対して制約が処理されます。
 4. 更新操作の後、制約を満たしたすべての行に対し、作成された順番で AFTER UPDATE トリガーが (行レベルとステートメント・レベルの両方で) 活動化されます。これは UPDATE ステートメントであるため、すべての基礎表に対して、UPDATE ステートメント・レベルのすべてのトリガーが活動化されます。
 5. 更新操作の後、制約を満たさなかったすべての行 (移動される行) に対して BEFORE DELETE トリガーが活動化されます。

6. 削除操作が処理されます。
 7. 削除されるすべての行に対して制約が処理されます。
 8. 削除されるすべての行に対し、作成された順番で AFTER DELETE トリガーが (行レベルとステートメント・レベルの両方で) 活動化されます。ステートメント・レベルのトリガーは、削除操作に関係する表に対してのみ活動化されます。
 9. 挿入されるすべての行 (つまり、移動される行) に対して BEFORE INSERT トリガーが活動化されます。BEFORE INSERT トリガーの新しい遷移表には、ユーザーからの入力データが含まれています。
 10. 入力操作が処理されます。
 11. 挿入されるすべての行に対して制約が処理されます。
 12. 挿入されるすべての行に対し、作成された順番で AFTER INSERT トリガーが (行レベルとステートメント・レベルの両方で) 活動化されます。ステートメント・レベルのトリガーは、挿入操作に関係する表に対してのみ活動化されます。
- **ネストされた UNION ALL ビュー:** UNION ALL で定義されたビュー、および同じく UNION ALL で定義され、直接的または間接的に基礎となるビューは、いずれかのビューが WITH ROW MOVEMENT 節を使用して定義されていると、更新できません (SQLSTATE 429BK)。
 - **暗黙的な隠し列に関する考慮事項:** 暗黙的な隠し列として定義された基本表の列が、全選択の結果表に含まれることがあります。暗黙的な隠し列がビュー定義の全選択で明示的に参照されている場合に、これが発生します。ただし、ビューの対応する列は、暗黙的な隠し列としての属性を継承しません。ビューの列を隠し列として定義することはできません。
 - **互換性:**
 - 以前のバージョンの DB2 との互換性:
 - FEDERATED キーワードは、キーワード CREATE および VIEW の間に指定できます。しかし FEDERATED キーワードは無視されます。これはフェデレーテッド・オブジェクトがビュー定義で使用される場合は、警告は戻されないからです。

例

例 1: PROJECT 表に基づくビュー MA_PROJ を作成します。このビューには、文字 'MA' で始まるプロジェクト番号 (PROJNO) を持つ行だけを入れます。

```
CREATE VIEW MA_PROJ AS SELECT *
FROM PROJECT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 2: 例 1 と同様にビューを作成します。ただし、プロジェクト番号 (PROJNO)、プロジェクト名 (PROJNAME)、およびプロジェクトの責任者 (RESPEMP) の列だけを選択します。

```
CREATE VIEW MA_PROJ
AS SELECT PROJNO, PROJNAME, RESPEMP
FROM PROJECT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

CREATE VIEW

例 3: 例 2 と同様のビューを作成します。ただし、ビューの中でプロジェクトの責任者の列を IN_CHARGE と呼びます。

```
CREATE VIEW MA_PROJ
(PROJNO, PROJNAME, IN_CHARGE)
AS SELECT PROJNO, PROJNAME, RESPEMP
FROM PROJECT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

注: 列名のいずれか 1 つだけを変更する場合でも、ビューの 3 つの列すべての名前を MA_PROJ の後の括弧内に指定する必要があります。

例 4: PRJ_LEADER という名前のビューを作成します。このビューには、PROJECT 表の最初の 4 つの列 (PROJNO、PROJNAME、DEPTNO、RESPEMP) と、プロジェクトの責任者 (RESPEMP) のラストネーム (LASTNAME) を入れます。ラストネームは、EMPLOYEE 表の EMPNO を PROJECT 表の RESPEMP と突き合わせることによって、EMPLOYEE 表から入手します。

```
CREATE VIEW PRJ_LEADER
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME
FROM PROJECT, EMPLOYEE
WHERE RESPEMP = EMPNO
```

例 5: 例 4 と同様のビューを作成します。ただし、列 PROJNO、PROJNAME、DEPTNO、RESPEMP、および LASTNAME に加えて、担当者の給与総額 (SALARY + BONUS + COMM) を入れます。また、平均スタッフ数 (PRSTAFF) が 1 より大きいプロジェクトだけを選択します。

```
CREATE VIEW PRJ_LEADER
(PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, TOTAL_PAY )
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP, LASTNAME, SALARY+BONUS+COMM
FROM PROJECT, EMPLOYEE
WHERE RESPEMP = EMPNO
AND PRSTAFF > 1
```

全選択で、式 SALARY+BONUS+COMM に TOTAL_PAY という名前を付けることによって、次のように、列名リストの指定を省略することができます。

```
CREATE VIEW PRJ_LEADER
AS SELECT PROJNO, PROJNAME, DEPTNO, RESPEMP,
LASTNAME, SALARY+BONUS+COMM AS TOTAL_PAY
FROM PROJECT, EMPLOYEE
WHERE RESPEMP = EMPNO AND PRSTAFF > 1
```

例 6: 次の図に示すような表とビューのセットがあるとします。

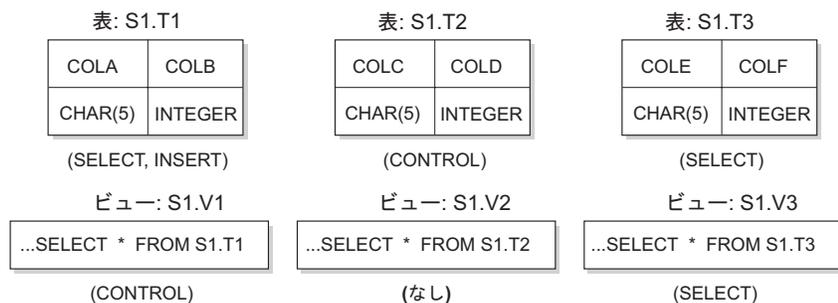


図 1. 例 6 の表とビュー

ユーザー ZORPIE (DBADM または SYSADM のいずれの権限も持たない) は、各オブジェクトの下の大括弧内に示している権限を与えられています。

1. 次のステートメントによって作成するビューに対して、ZORPIE は CONTROL 特権を獲得します。

```
CREATE VIEW VA AS SELECT * FROM S1.V1
```

これは、ZORPIE が S1.V1 に対して CONTROL 権限を持っているからです。DBADM または SYSADM のいずれかの権限を持つユーザーが、S1.V1 に対する CONTROL 権限を ZORPIE に与えている必要があります。基礎となる基本表に対して、どのような特権を与えられているかは関係ありません。

2. ZORPIE は、次のようなビューの作成は許されません。

```
CREATE VIEW VB AS SELECT * FROM S1.V2
```

これは、ZORPIE には、S1.V2 に対する CONTROL も SELECT も与えられていないからです。基礎となる基本表 (S1.T2) に対して CONTROL を与えられているかどうかは、関係ありません。

3. 次のステートメントによって作成するビューに対して、ZORPIE は CONTROL 特権を獲得します。

```
CREATE VIEW VC (COLA, COLB, COLC, COLD)
AS SELECT * FROM S1.V1, S1.T2
WHERE COLA = COLC
```

これは、ZORPIE.VC の全選択では、ビュー S1.V1 と S1.T2 を参照しており、ZORPIE はその両方に対する CONTROL を持っているからです。ビュー VC は読み取り専用で、INSERT、UPDATE、または DELETE のいずれの権限も ZORPIE には与えられないことに注意してください。

4. 次のステートメントによって作成するビューに対して、ZORPIE は SELECT 特権を入手します。

```
CREATE VIEW VD (COLA,COLB, COLE, COLF)
AS SELECT * FROM S1.V1, S1.V3
WHERE COLA = COLE
```

これは、ZORPIE.VD の全選択で 2 つのビュー S1.V1 および S1.V3 を参照しており、ZORPIE はその 1 つに対しては SELECT 特権を、もう 1 つに対しては CONTROL 特権を与えられているからです。ZORPIE.VD に対する特権として、2 つの特権のうち低い方の特権である SELECT が ZORPIE に与えられます。

5. 以下のビュー定義では、ZORPIE はビュー VE に対して WITH GRANT OPTION を伴う INSERT、UPDATE および DELETE の各特権と、SELECT 特権を与えられます。

```
CREATE VIEW VE
AS SELECT * FROM S1.V1
WHERE COLA > ANY
(SELECT COLE FROM S1.V3)
```

ZORPIE の VE に対する特権は、主として S1.V1 に対する特権によって決定されます。S1.V3 は副照会で参照されるだけなので、ビュー VE の作成には S1.V3 に対する SELECT 特権のみ必要となります。ビューの定義者は、ビュー定義で参照されるすべてのオブジェクトに対して CONTROL を持っている場合

CREATE VIEW

のみ、そのビューに対する CONTROL を得ます。ZORPIE は S1.V3 に対する CONTROL を持っていないため、VE に対する CONTROL は得られません。

CREATE WORK ACTION SET

CREATE WORK ACTION SET ステートメントは、作業アクション・セットおよびその中の作業アクションを定義します。

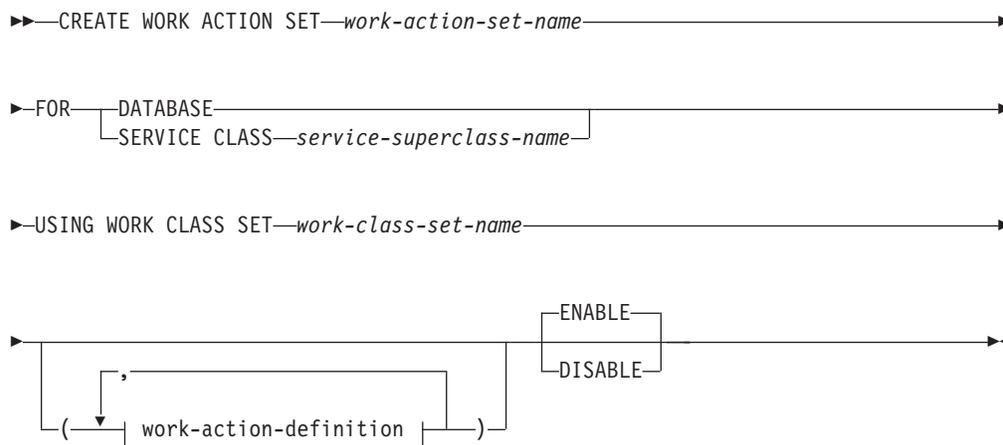
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

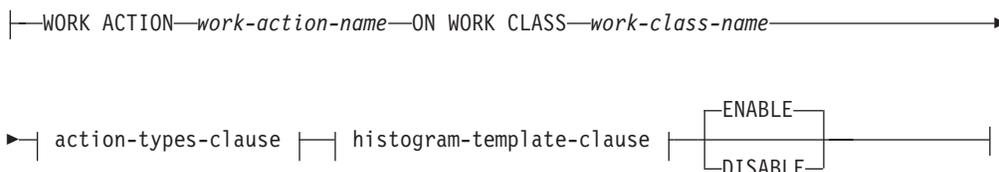
許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文

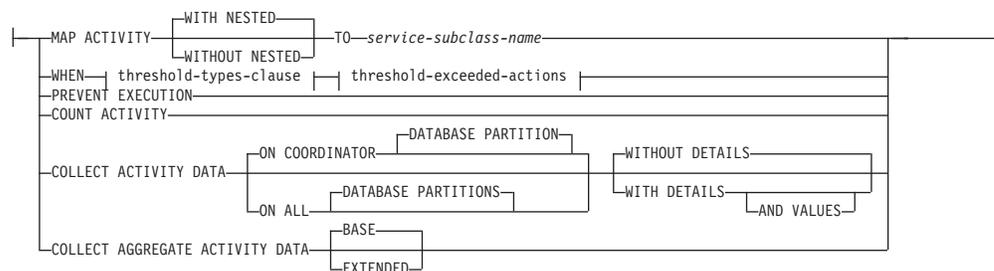


work-action-definition:

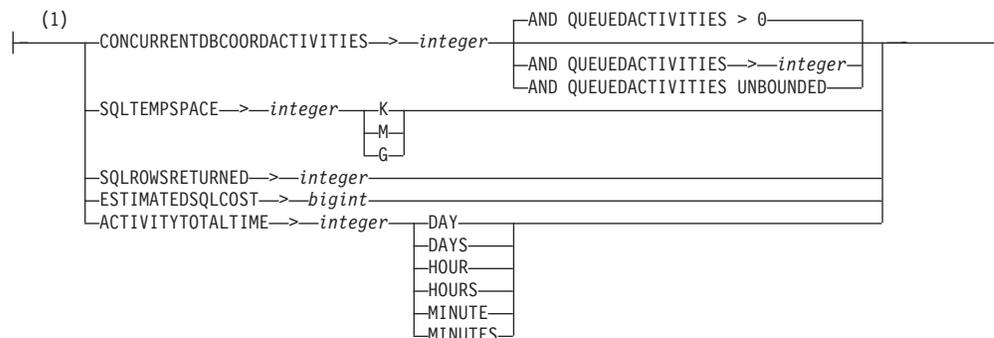


action-types-clause:

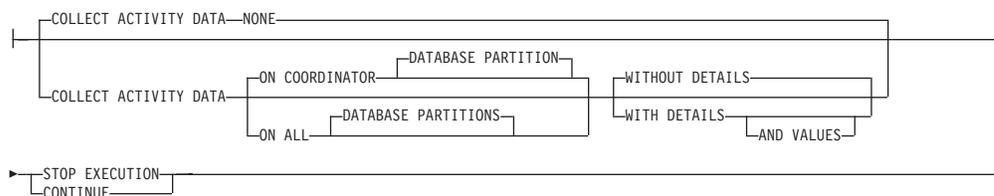
CREATE WORK ACTION SET



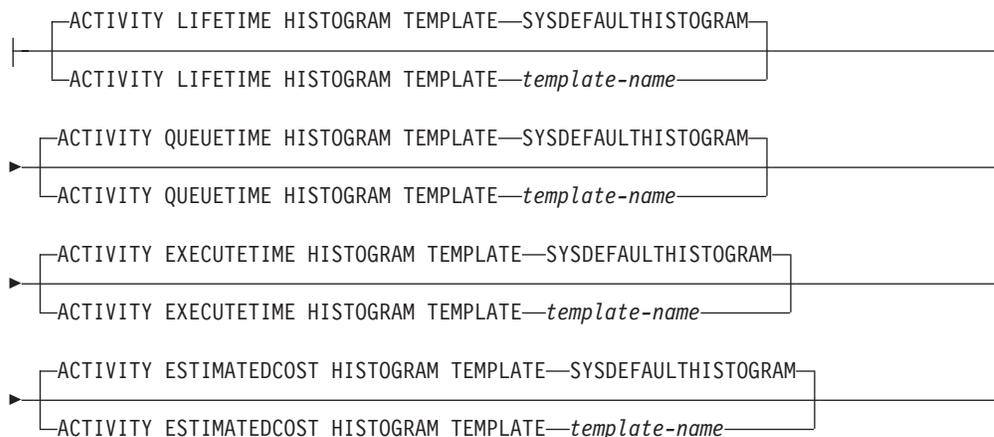
threshold-types-clause:



threshold-exceeded-actions:



histogram-template-clause:



CREATE WORK ACTION SET



注:

- 1 つの作業クラスに一度に適用できる同一しきい値タイプの作業アクションは、1 つだけです。

説明

work-action-set-name

作業アクション・セットの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *work-action-set-name* は、現行のサーバー上に既に存在する作業アクション・セットを識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

FOR

この作業アクション・セットのアクションが適用されるデータベース・マネージャー・オブジェクトを指定します。データベース・マネージャー・オブジェクトにはそれぞれ作業アクション・セットを 1 つだけ定義できます (SQLSTATE 5U017)。

DATABASE

この作業アクション・セットのアクションはデータベースに適用されます。DATABASE が指定された場合、MAP ACTIVITY アクションは指定できません (SQLSTATE 5U034)。

SERVICE CLASS *service-superclass-name*

この作業アクション・セットのアクションは *service-superclass-name* に適用されます。SERVICE CLASS が指定された場合、しきい値アクションは指定できません (SQLSTATE 5U034)。 *service-superclass-name* は、現行のサーバーに存在するものでなければなりません (SQLSTATE 42704)。 *service-superclass-name* はサービス・サブクラスであってはならず、以下のどのクラスになることもできません (SQLSTATE 5U032)。

- システム・サービス・クラス (SYSDEFAULTSYSTEMCLASS)
- 保守サービス・クラス (SYSDEFAULTMAINTENANCECLASS)
- デフォルト・ユーザー・サービス・クラス (SYSDEFAULTUSERCLASS)

USING WORK CLASS SET *work-class-set-name*

アクションの実行対象データベース・アクティビティを分類する作業クラスを含む作業クラス・セットを指定します。 *work-class-set-name* は、現行のサーバーに存在するものでなければなりません (SQLSTATE 42704)。

work-action-definition

作業アクションの定義を指定します。

WORK ACTION *work-action-name*

作業アクションの名前を指定します。 *work-action-name* は、現行のサーバーのこの作業アクション・セットの下に既存の作業アクションを識別するものであってはなりません (SQLSTATE 42710)。 *work-action-name* の先頭に 'SYS' を使用することはできません (SQLSTATE 42939)。

ON WORK CLASS *work-class-name*

この作業アクションが適用されるデータベース・アクティビティを識別する作業クラスを指定します。 *work-class-name* は、現行のサーバーの *work-class-set-name* に存在するものでなければなりません (SQLSTATE 42704)。

MAP ACTIVITY

アクティビティのマッピングの作業アクションを指定します。このアクションは、この作業アクション・セットが定義されているオブジェクトがサービス・スーパークラスである場合にのみ指定できます (SQLSTATE 5U034)。

WITH NESTED または WITHOUT NESTED

このアクティビティの下にネストされているアクティビティをサービス・サブクラスにマップするかどうかを指定します。デフォルトは WITH NESTED です。

WITH NESTED

作業クラスに分類されるすべてのデータベース・アクティビティのうちネ스팅・レベルがゼロのもの、およびこのアクティビティの下でネストされているすべてのデータベース・アクティビティは、サービス・サブクラスにマップされます。つまり、ネ스팅・レベルがゼロより大きいアクティビティは、ネ스팅・レベルがゼロのアクティビティと同じサービス・クラスの下で実行されます。

WITHOUT NESTED

作業クラスの下に分類されているデータベース・アクティビティのうち、ネ스팅・レベルがゼロのものだけがサービス・サブクラスにマップされます。このアクティビティの下にネストされているデータベース・アクティビティは、そのアクティビティ・タイプに従って処理されます。

TO *service-subclass-name*

アクティビティのマッピング先となるサービス・サブクラスを指定します。 *service-subclass-name* は、現行サーバーの *service-superclass-name* に既に存在するものでなければなりません (SQLSTATE 42704)。 *service-subclass-name* にデフォルトのサービス・サブクラス SYSDEFAULTSUBCLASS を指定することはできません (SQLSTATE 5U018)。

WHEN

この作業アクションが定義されている作業クラスに関連するデータベース・アクティビティに適用するしきい値を指定します。しきい値は、この作業アクション・セットが定義されているデータベース・マネージャー・オブジェクトがデータベースである場合にのみ指定できます (SQLSTATE 5U034)。これらのしきい値は、データベース・マネージャーによって開始された内部のデータベース・アクティビティ、または管理 SQL ルーチンによって生成されたデータベース・アクティビティには適用されません。

threshold-types-clause

有効なしきい値タイプについての説明は、『CREATE THRESHOLD』ステートメントの項を参照してください。

CREATE WORK ACTION SET

threshold-exceeded-actions

しきい値を超過した場合の有効なアクションについて説明は、『CREATE THRESHOLD』ステートメントを参照してください。

PREVENT EXECUTION

この作業アクションが定義されている作業クラスに関連するデータベース・アクティビティーの実行を許可しないことを指定します (SQLSTATE 5U033)。

COUNT ACTIVITY

この作業アクションが定義されている作業クラスと関連付けられたデータベース・アクティビティーすべてを実行し、実行されるたびに作業クラスのカウンターを増分することを指定します。

COLLECT ACTIVITY DATA

この作業アクションが定義されている作業クラスに関連する各アクティビティーについてのデータを、アクティビティー完了時に、アクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。デフォルトは COLLECT ACTIVITY DATA WITHOUT DETAILS です。

ON COORDINATOR DATABASE PARTITION

アクティビティーのコーディネーターのデータベース・パーティションでのみアクティビティー・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティーが処理されるすべてのデータベース・パーティションでアクティビティー・データを収集することを指定します。予測しきい値の場合、しきい値を超過した場合の CONTINUE アクションも指定した場合にのみ、アクティビティー情報がすべてのパーティションで収集されます。反応しきい値の場合、ON ALL DATABASE PARTITIONS 節を指定しても効果はなく、アクティビティー情報は常にコーディネーター・パーティションでのみ収集されます。予測しきい値および反応しきい値の両方について、アクティビティーの詳細または値は、コーディネーター・パーティションでのみ収集されます。

WITHOUT DETAILS

この作業アクションが定義されている作業クラスに関連する各アクティビティーについてのデータを、アクティビティー完了時に、アクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境の情報はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティーについて、これらの情報をアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティーに入力データ値が含まれている場合、それを

アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。

COLLECT AGGREGATE ACTIVITY DATA

この作業アクションが定義されている作業クラスに関連するアクティビティについて、集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。デフォルトは COLLECT AGGREGATE ACTIVITY DATA BASE です。この節は、データベースに適用される作業アクション・セットで定義されている作業アクションには指定できません。

BASE

この作業アクションが定義されている作業クラスに関連するアクティビティについて、基礎集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。基礎集約アクティビティ・データには以下のものが含まれます。

- アクティビティ・コストの最高水準点の見積もり
- 戻り行数の最高水準点
- TEMPORARY 表スペース使用量の最高水準点
- アクティビティ存続時間のヒストグラム
- アクティビティ・キュー時間のヒストグラム
- アクティビティ実行時間のヒストグラム

EXTENDED

この作業アクションが定義されている作業クラスに関連するアクティビティについて、すべての集約アクティビティ・データをキャプチャーし、統計イベント・モニター (アクティブになっている場合) に送信することを指定します。これには、すべての基礎集約アクティビティ・データに加えて、以下のものが含まれます。

- アクティビティ・データ操作言語 (DML) の見積コスト・ヒストグラム
- アクティビティ DML の到着間隔時間のヒストグラム

ENABLE または DISABLE

データベース・アクティビティをサブミットする際にこの作業アクションを考慮するかどうかを指定します。デフォルトは ENABLE です。

ENABLE

この作業アクションが有効であり、データベース・アクティビティのサブミット時に考慮することを指定します。

DISABLE

作業アクションが無効であり、データベース・アクティビティのサブミット時に考慮の対象にならないことを指定します。

CREATE WORK ACTION SET

ENABLE または DISABLE

データベース・アクティビティーをサブミットする際にこの作業アクション・セットを考慮するかどうかを指定します。デフォルトは ENABLE です。

ENABLE

この作業アクション・セットが有効であり、データベース・アクティビティーのサブミット時に考慮することを指定します。

DISABLE

この作業アクション・セットが無効であり、データベース・アクティビティーのサブミット時に考慮の対象とならないことを指定します。

histogram-template-clause

この作業アクションの割り当て先の作業クラスに関連するアクティビティーの集約アクティビティー・データを収集する際に使用するヒストグラム・テンプレートを指定します。作業クラスの集約アクティビティー・データが収集されるのは、作業アクション・タイプが COLLECT AGGREGATE ACTIVITY DATA となっている場合だけです。

ACTIVITY LIFETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で実行された (この作業アクションの割り当て先作業クラスに関連する) DB2 アクティビティーの所要時間 (マイクロ秒) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。この時間には、キューに入っていた時間と実行時間の両方が含まれます。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY QUEUETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で (この作業アクションが割り当てられている作業クラスに関連する) DB2 アクティビティーがキューに入れられていた時間の長さ (マイクロ秒単位) に関する、統計データの収集に使用されるヒストグラムを記述する、テンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY EXECUTETIME HISTOGRAM TEMPLATE *template-name*

特定のインターバルの中で (この作業アクションが割り当てられている作業クラスに関連する) DB2 アクティビティーが実行されている時間の長さ (マイクロ秒単位) に関する、統計データの収集に使用されるヒストグラムを記述する、テンプレートを指定します。この時間には、アクティビティーがキューに入っていた時間は含まれません。このヒストグラムでは、アクティビティーが実行される各データベース・パーティションごとにアクティビティーの実行時間が収集されます。アクティビティーのコーディネーターのデータベース・パーティションの場合、これはエンドツーエンドの実行時間です (つまり、存続時間からキューに入っていた時間を差し引いた時間)。コーディネーター・データベース・パーティション以外の場合、これはこれらのパーティションがアクティビティーの代わりに費やした時間です。特定のアクティビティーの実行中、DB2 は、リモート・データベース・パーティショ

ンに対して作業を複数回提示することがあります。リモート・パーティションはそのたびにアクティビティーのオカレンスの実行時間を収集します。そのため、実行時間のヒストグラム数は、データベース・パーティションで実行された固有アクティビティーの実際の数とは異なる場合があります。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節と、その BASE または EXTENDED のどちらかのオプションが指定されている場合にのみ収集されます。

ACTIVITY ESTIMATEDCOST HISTOGRAM TEMPLATE *template-name*

この作業アクションの割り当て先作業クラスに関連する DML アクティビティーの見積コスト (timeron 単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節とその EXTENDED オプションが指定されている場合にのみ収集されます。

ACTIVITY INTERARRIVALTIME HISTOGRAM TEMPLATE *template-name*

この作業アクションの割り当て先作業クラスに関連するすべてのアクティビティーについて、1 つの DML アクティビティーの到着から次の DML アクティビティーの到着までの間の時間の長さ (マイクロ秒単位) に関する統計データを収集するために使用されるヒストグラムを記述するテンプレートを指定します。デフォルトは SYSDEFAULTHISTOGRAM です。この情報は、COLLECT AGGREGATE ACTIVITY DATA 節とその EXTENDED オプションが指定されている場合にのみ収集されます。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

CREATE WORK ACTION SET

例

例 1: すべてのデータベース・アクティビティに適用する DATABASE_ACTIONS という名前の作業アクション・セットを作成します。LARGE_QUERIES 作業クラス・セットを使用し、次の作業アクションを定義します。作業アクション ONE_CONCURRENT_QUERY は 1 つのしきい値アクションを持ちます。このアクションは、照会が LARGE_ESTIMATED_COST 作業クラス内にある場合にシステムで一度に 1 つずつ並行照会を実行することができます。そのしきい値を超過するとデータベース・マネージャーはアクティビティをキューに入れますが、一度に複数のデータベース・アクティビティがキューに入れられないようにします。キューのしきい値を超過した場合はデータベース・アクティビティを実行できなくなります。作業アクション TWO_CONCURRENT_QUERIES は 1 つのしきい値アクションを持ちます。このアクションは、照会が LARGE_CARDINALITY 作業クラス内にある場合に 2 つの並行照会を同時に実行することができます。照会が 2 つ以下の場合にはキューに入れます。2 つを超える照会がキューに入れられる場合、データベース・アクティビティはキューに照会を入れ続け、データベース・アクティビティ・データをアクティビティ・イベント・モニター (アクティブになっている場合) で収集します。

```
CREATE WORK ACTION SET DATABASE_ACTIONS
FOR DATABASE USING WORK CLASS SET LARGE_QUERIES
(WORK ACTION ONE_CONCURRENT_QUERY ON WORK CLASS LARGE_ESTIMATED_COST
WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 1
STOP EXECUTION,
WORK ACTION TWO_CONCURRENT_QUERIES ON WORK CLASS LARGE_CARDINALITY
WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 2
COLLECT ACTIVITY DATA CONTINUE)
```

例 2: MAP_SELECTS という名前の 1 つの作業アクションを持つ ADMIN_APPS_ACTIONS という名前の作業アクション・セットを作成します。これはサービス・スーパークラス ADMIN_APPS の下実行されるデータベース・アクティビティに適用されます。作業アクションは、SELECT_CLASS 作業クラス内にあるすべてのデータベース・アクティビティをサービス・サブクラス SELECTS_SERVICE_CLASS (DML_SELECTS 作業クラス・セットにある) にマップします。

```
CREATE WORK ACTION SET ADMIN_APPS_ACTIONS
FOR SERVICE CLASS ADMIN_APPS USING
WORK CLASS SET DML_SELECTS
(WORK ACTION MAP_SELECTS ON WORK CLASS SELECT_CLASS
MAP TO SELECTS_SERVICE_CLASS)
```

CREATE WORK CLASS SET

CREATE WORK CLASS SET ステートメントは、作業クラス・セットを定義します。

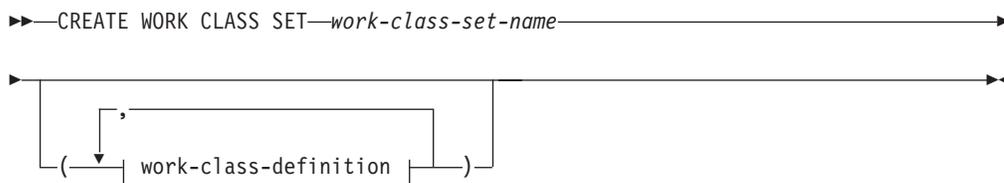
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

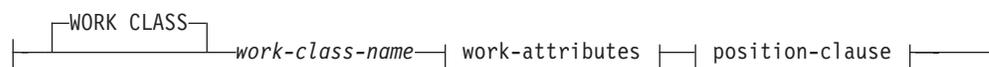
許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

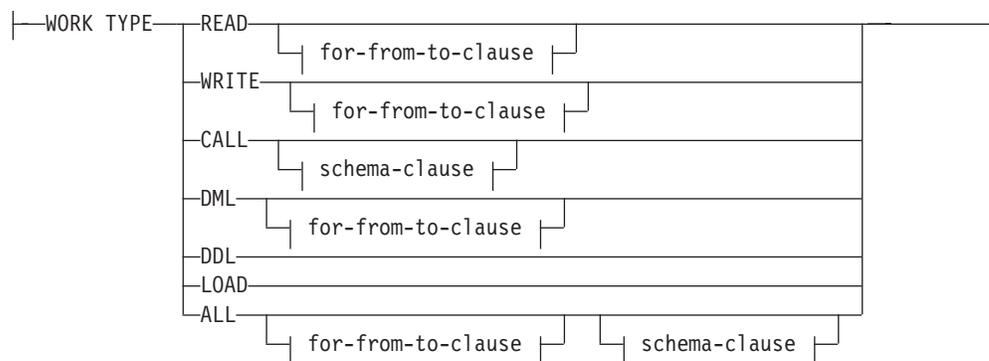
構文



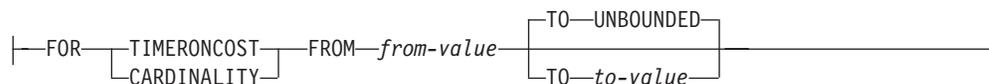
work-class-definition:



work-attributes:



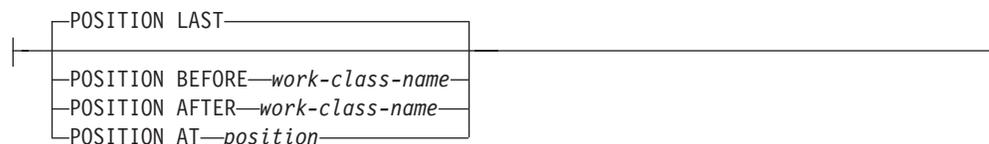
for-from-to-clause:



schema-clause:



position-clause:



説明

work-class-set-name

作業クラス・セットの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *work-class-set-name* は、現行のサーバー上に既に存在する作業クラス・セットを識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

work-class-definition

作業クラスの定義を指定します。

WORK CLASS *work-class-name*

作業クラスの名前を指定します。 *work-class-name* は、現行のサーバー上の作業クラス・セット内に既に存在する作業クラスを識別するものであってはなりません (SQLSTATE 42710)。 *work-class-name* を 'SYS' で始めることはできません (SQLSTATE 42939)。

work-attributes

データベース・アクティビティの属性は、アクティビティがこの作業クラスと関連付けられる場合、この作業クラスで指定されているすべての属性と一致する必要があります。

WORK TYPE

データベース・アクティビティのタイプを指定します。

READ

このアクティビティには次のステートメントが含まれます。

- DELETE、INSERT、MERGE、または UPDATE ステートメントを含まないすべての SELECT または SELECT INTO ステートメント、およびすべての VALUES INTO ステートメント
- すべての XQuery ステートメント

WRITE

このアクティビティには次のステートメントが含まれます。

- UPDATE
- DELETE
- INSERT
- MERGE
- DELETE、INSERT、または UPDATE ステートメントを含むすべての SELECT ステートメント、およびすべての VALUES INTO ステートメント
- すべての XQuery ステートメント

CALL

CALL ステートメントが含まれます。 CALL ステートメントは、作業クラスの作業タイプが CALL または ALL の場合に考慮されます。

DML

READ と WRITE でリストされているステートメントが含まれます。

DDL

このアクティビティーには次のステートメントが含まれます。

- ALTER
- CREATE
- COMMENT
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- FLUSH PACKAGE CACHE
- GRANT
- REFRESH TABLE
- RENAME
- REVOKE
- SET INTEGRITY

LOAD

DB2 のロード操作。

ALL

上記のキーワードのいずれかに該当する、認識されるすべてのワークロード管理 (WLM) アクティビティー。

FOR

FROM *from-value* TO *to-value* 節で指定されている情報のタイプを示します。FOR 節は、以下の作業タイプでのみ使用されます。

- READ
- WRITE
- DML
- ALL

TIMERONCOST

作業の見積コスト (timeron 単位)。この値は、作業が FROM *from-value* TO *to-value* 節で指定された範囲に入るかどうかを判断するのに使用されます。

CARDINALITY

作業の見積カーディナリティー。この値は、作業が FROM *from-value* TO *to-value* 節で指定された範囲に入るかどうかを判断するのに使用されます。

FROM *from-value* TO UNBOUNDED または FROM *from-value* TO *to-value*

timeron 値 (見積コストの場合) またはカーディナリティーの範囲を指定します。データベース・アクティビティーがこの作業クラスに属するためには、この範囲に収まっていなければなりません。この範囲には、*from-value* および *to-value* が含まれています。作業クラスでこの節が指定されていない場合、指定された作業タイプに該当するすべての作業が含まれます (つまり、デフォルトは FROM 0 TO UNBOUNDED です)。この範囲は以下の作業タイプでのみ使用されます。

CREATE WORK CLASS SET

- READ
- WRITE
- DML
- ALL

FROM *from-value* TO UNBOUNDED

from-value は、ゼロまたは正の DOUBLE 値でなければなりません (SQLSTATE 5U019)。範囲に上限はありません。

FROM *from-value* TO *to-value*

from-value はゼロまたは正の DOUBLE 値でなければならず、*to-value* は正の DOUBLE 値でなければなりません。 *from-value* は *to-value* 以下でなければなりません (SQLSTATE 5U019)。

schema-clause

ROUTINES IN SCHEMA *schema-name*

CALL ステートメントが呼び出すプロシージャのスキーマ名を指定します。この節は、作業タイプが CALL または ALL で、データベース・アクティビティーが CALL ステートメントの場合にのみ使用されます。値が指定されていない場合、すべてのスキーマが含まれます。

position-clause

POSITION

この作業クラスを作業クラス・セット内のどの位置に配置するかを指定します。この位置によって、作業クラスが評価される順序が決まります。実行時に作業クラスの割り当てを実行する際、データベース・マネージャーは、まずデータベースまたはサービス・スーパークラスのどちらかのオブジェクトに関連付けられている作業クラス・セットを判別します。次に、その作業クラス・セット内で最初に一致する作業クラスが選択されます。このキーワードが指定されていない場合、作業クラスは最後尾に配置されます。

LAST

作業クラスを、作業クラス・セット内で作業クラスの番号付きリストの最後尾に配置することを指定します。これはデフォルトです。

BEFORE *work-class-name*

作業クラスを、リストの作業クラス *work-class-name* の前に配置することを指定します。 *work-class-name* は、現行のサーバー上に存在する作業クラス・セットの作業クラスを識別するものでなければなりません (SQLSTATE 42704)。

AFTER *work-class-name*

作業クラスを、リストの作業クラス *work-class-name* の後に配置することを指定します。 *work-class-name* は、現行のサーバー上に存在する作業クラス・セットの作業クラスを識別するものでなければなりません (SQLSTATE 42704)。

AT *position*

作業クラス・セット内で作業クラスを配置する位置を、作業クラスの番号付きリストの中での絶対位置として指定します。この値に

は、任意の正整数値を指定できます (SQLSTATE 42615)。 *position* が既存の作業クラスの数に 1 を足した値より大きい場合、その作業クラスは作業クラス・セットの最後尾に配置されます。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。 WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)
 - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
 - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。

例

例 1: LARGE_QUERIES という名前の作業クラス・セットを作成します。これは 9999 より大きい見積コストと 1000 より大きい見積カーディナリティーを持つすべての DML を表す作業クラスのセットを持ちます。

```
CREATE WORK CLASS SET LARGE_QUERIES
(WORK CLASS LARGE_ESTIMATED_COST WORK TYPE DML
FOR TIMERONCOST FROM 9999 TO UNBOUNDED,
WORK CLASS LARGE_CARDINALITY WORK TYPE DML
FOR CARDINALITY FROM 1000 TO UNBOUNDED)
```

例 2: DML_SELECTS という名前の作業クラス・セットを作成します。これは DELETE、INSERT、MERGE、または UPDATE ステートメントを含まないすべての DML SELECT ステートメントを表す作業クラスを持ちます。

```
CREATE WORK CLASS SET DML_SELECTS
(WORK CLASS SELECT_CLASS WORK TYPE READ)
```

CREATE WORKLOAD

CREATE WORKLOAD ステートメントは、ワークロードを定義します。

CREATE WORKLOAD

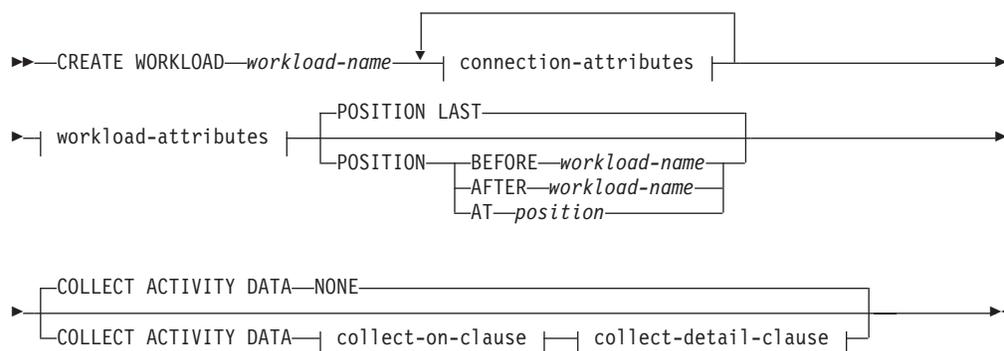
呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

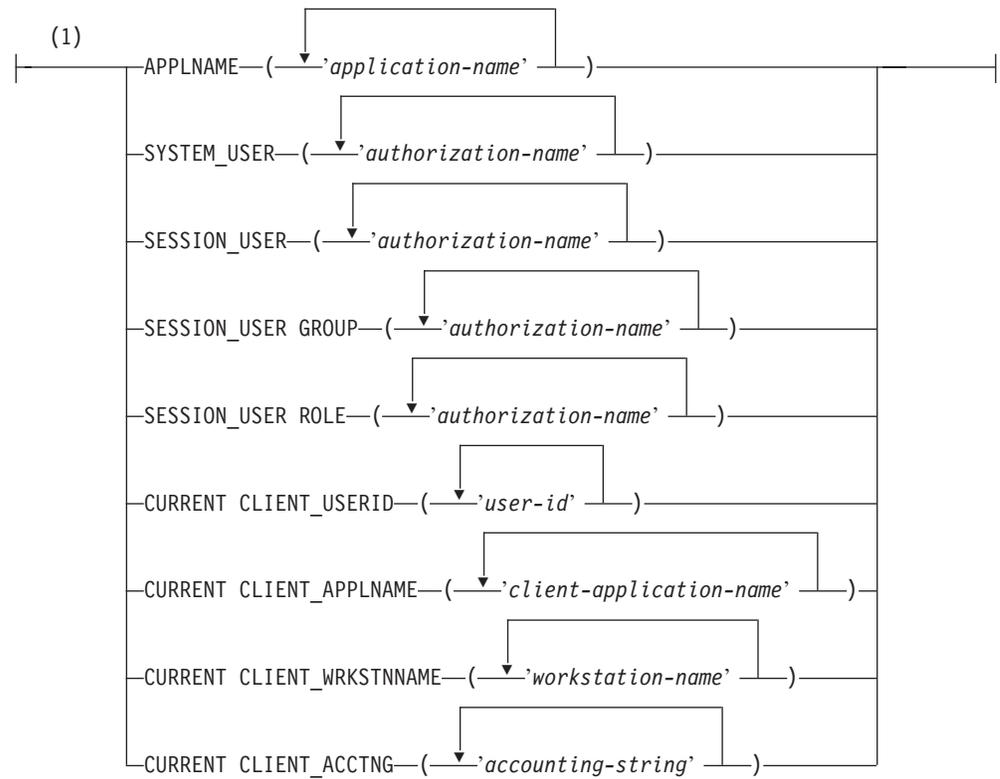
許可

このステートメントの許可 ID が持つ特権には、 SYSADM または DBADM 権限が含まれている必要があります。

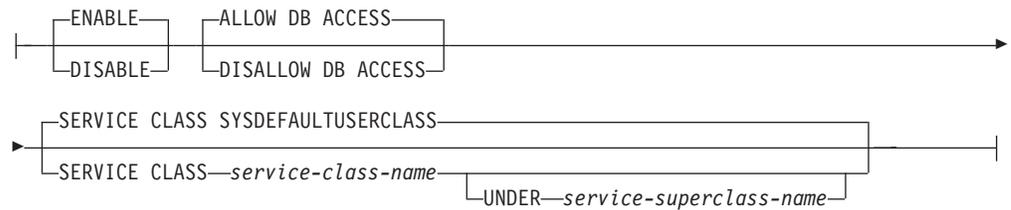
構文



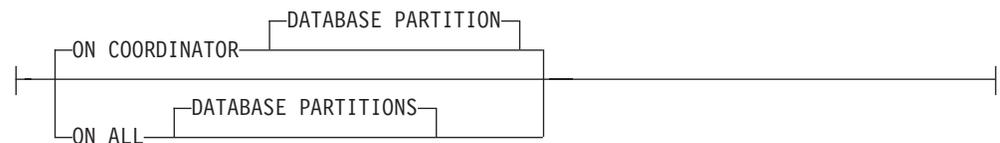
connection-attributes:



workload-attributes:



collect-on-clause:



collect-detail-clause:



CREATE WORKLOAD

注:

- 1 各接続属性節は、一度しか指定できません。

説明

workload-name

ワークロードの名前を示します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *workload-name* は、現行のサーバー上の既存のワークロードを識別するものであってはなりません (SQLSTATE 42710)。名前を文字 'SYS' で始めることはできません (SQLSTATE 42939)。

connection-attributes

接続の確立時にこのワークロードに接続属性を関連付ける場合、接続属性は、このワークロード定義で指定されているすべての属性と一致している必要があります。ワークロード定義の接続属性に値のリストが指定されている場合、対応する接続属性は、リストに含まれている値のうち、少なくとも 1 つと一致している必要があります。ワークロード定義で接続属性が指定されない場合、その接続に関して対応する接続属性は任意の値で問題ありません。

APPLNAME ('*application-name*', ...)

APPLNAME 接続属性に 1 つ以上のアプリケーションを指定します。1 つのアプリケーション名をリストの中で複数回指定することはできません (SQLSTATE 42713)。 *application-name* は、大文字と小文字を区別して扱います。この値は、システム・モニター出力や LIST APPLICATIONS コマンドからの出力の「アプリケーション名」フィールドに表示される値と等しくなります。

SYSTEM_USER ('*authorization-name*', ...)

SYSTEM_USER 接続属性に 1 つ以上の許可 ID を指定します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

SESSION_USER ('*authorization-name*', ...)

SESSION_USER 接続属性に 1 つ以上の許可 ID を指定します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

SESSION_USER GROUP ('*authorization-name*', ...)

SESSION_USER GROUP 接続属性に 1 つ以上の許可 ID を指定します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

SESSION_USER ROLE ('*authorization-name*', ...)

SESSION_USER ROLE 接続属性に 1 つ以上の許可 ID を指定します。このコンテキストでの SESSION 許可 ID のロールは、どのように取得されたロールであるかに関係なく、SESSION 許可 ID に使用可能なすべてのロールを参照します。1 つの許可 ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

CURRENT_CLIENT_USERID ('*user-id*', ...)

CURRENT_CLIENT_USERID 接続属性に 1 つ以上のクライアント・ユーザー ID を指定します。1 つのクライアント・ユーザー ID をリストの中で複数回指定することはできません (SQLSTATE 42713)。

CURRENT_CLIENT_APPLNAME ('*client-application-name*', ...)

CURRENT_CLIENT_APPLNAME 接続属性に 1 つ以上のアプリケーション

を指定します。1つのアプリケーション名をリストの中で複数回指定することはできません (SQLSTATE 42713)。 *client-application-name* は、大文字と小文字を区別して扱います。この値は、システム・モニター出力の「TPモニター・クライアント・アプリケーション名」フィールドに表示される値と等しくなります。

CURRENT CLIENT_WRKSTNNAME (*'workstation-name'*, ...)

CURRENT CLIENT_WRKSTNNAME 接続属性に1つ以上のクライアント・ワークステーション名を指定します。1つのクライアント・ワークステーション名をリストの中で複数回指定することはできません (SQLSTATE 42713)。

CURRENT CLIENT_ACCTNG (*'accounting-string'*, ...)

CURRENT CLIENT_ACCTNG 接続属性に1つ以上のクライアント会計情報ストリングを指定します。1つのクライアント会計情報ストリングをリストの中で複数回指定することはできません (SQLSTATE 42713)。

workload-attributes

ワークロードの属性を指定します。

ENABLE または **DISABLE**

ワークロードを選択する際にこのワークロードを考慮するかどうかを指定します。デフォルトは ENABLE です。

ENABLE

このワークロードを有効にし、ワークロードを選択する際にこのワークロードを考慮することを指定します。

DISABLE

このワークロードを無効にし、ワークロードを選択する際にこのワークロードを考慮しないことを指定します。

ALLOW DB ACCESS または **DISALLOW DB ACCESS**

このワークロードに関連付けられているワークロード・オカレンスにデータベースへのアクセスを許可するかどうかを指定します。デフォルトは ALLOW DB ACCESS です。

ALLOW DB ACCESS

このワークロードに関連付けられているワークロード・オカレンスにデータベースへのアクセスを許可することを指定します。

DISALLOW DB ACCESS

このワークロードに関連付けられているワークロード・オカレンスにデータベースへのアクセスを許可しないことを指定します。このワークロードに関連付けられている次の作業単位は拒否されます (SQLSTATE 5U020)。既に実行中のワークロード・オカレンスは完了まで実行できません。

SERVICE CLASS*service-class-name*

このワークロードに関連付けられている要求をサービス・クラス *service-class-name* で実行することを指定します。*service-class-name* には、現行のサーバー上の既存のサービス・クラスを指定する必要があります (SQLSTATE 42704)。*service-class-name* を 'SYSDEFAULTSUBCLASS'、'SYSDEFAULTSYSTEMCLASS'、または

CREATE WORKLOAD

'SYSDEFAULTMAINTENANCECLASS' にすることはできません (SQLSTATE 5U032)。デフォルトは SYSDEFAULTUSERCLASS です。

UNDER *service-superclass-name*

この節は、サービス・サブクラスを指定するときに使用されます。
service-superclass-name は、*service-class-name* のサービス・スーパークラスを識別します。*service-superclass-name* には、現行のサーバー上の既存のサービス・スーパークラスを指定する必要があります (SQLSTATE 42704)。*service-superclass-name* を 'SYSDEFAULTSYSTEMCLASS' または 'SYSDEFAULTMAINTENANCECLASS' にすることはできません (SQLSTATE 5U032)。

POSITION

ワークロードの番号付きリストの中でこのワークロードをどの位置に配置するかを指定します。実行時には、このリストの順番で、必須の接続属性と一致する最初のワークロードが検索されます。デフォルトは LAST です。

LAST

ワークロードが、デフォルトのワークロード
SYSDEFAULTUSERWORKLOAD および SYSDEFAULTADMWORKLOAD
の前の、リストで最後のワークロードであることを指定します。

BEFORE *relative-workload-name*

リストの中でワークロードをワークロード *relative-workload-name* の前に配置することを指定します。*relative-workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。*relative-workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、BEFORE オプションを指定できません (SQLSTATE 42832)。

AFTER *relative-workload-name*

リストの中でワークロードをワークロード *relative-workload-name* の後に配置することを指定します。*relative-workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。*relative-workload-name* が 'SYSDEFAULTUSERWORKLOAD' または 'SYSDEFAULTADMWORKLOAD' の場合は、AFTER オプションを指定できません (SQLSTATE 42832)。

AT *position*

リストの中でのワークロードの絶対位置を指定します。この値には、任意の正整数値を指定できます (SQLSTATE 42615)。*position* が既存のワークロードの数に 1 を足した値より大きい場合、ワークロードはリストの最後、SYSDEFAULTUSERWORKLOAD および SYSDEFAULTADMWORKLOAD のすぐ前に置かれます。

COLLECT ACTIVITY DATA

このワークロードに関連付けられている各アクティビティに関するデータを、アクティビティが完了したときに、アクティビティ・イベント・モニター (アクティブになっている場合) に送信することを指定します。デフォルトは COLLECT ACTIVITY DATA NONE です。

collect-on-clause

どこでアクティビティー・データを収集するかを指定します。デフォルトは ON COORDINATOR DATABASE PARTITION です。

ON COORDINATOR DATABASE PARTITION

アクティビティーのコーディネーターのデータベース・パーティションでのみアクティビティー・データを収集することを指定します。

ON ALL DATABASE PARTITIONS

アクティビティーが処理されるすべてのデータベース・パーティションでアクティビティー・データを収集することを指定します。ただし、アクティビティーの詳細や値は、アクティビティーのコーディネーターのデータベース・パーティションでのみ収集されます。

NONE

このワークロードに関連付けられている各アクティビティーについてはアクティビティー・データを収集しないことを指定します。

collect-detail-clause

どのタイプのアクティビティー・データを収集するかを指定します。デフォルトは WITHOUT DETAILS です。

WITHOUT DETAILS

このワークロードに関連付けられている各アクティビティーに関するデータを、アクティビティーの実行が完了したときに、アクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。ステートメントおよびコンパイル環境はイベント・モニターに送信されません。

WITH DETAILS

ステートメントやコンパイル環境の情報を含むアクティビティーについて、これらの情報をアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

AND VALUES

アクティビティーに入力データ値が含まれている場合、それをアクティビティー・イベント・モニター (アクティブになっている場合) に送信することを指定します。

規則

- ワークロード管理 (WLM) 排他 SQL ステートメントの後は COMMIT または ROLLBACK ステートメントでなければなりません (SQLSTATE 5U021)。 WLM 排他 SQL ステートメントは次のとおりです。
 - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
 - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
 - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
 - CREATE WORK ACTION SET、ALTER WORK ACTION SET、または DROP (WORK ACTION SET)
 - CREATE WORK CLASS SET、ALTER WORK CLASS SET、または DROP (WORK CLASS SET)

CREATE WORKLOAD

- CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
- GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- WLM 排他 SQL ステートメントをグローバル・トランザクション (例えば XA トランザクション) 内で発行することはできません (SQLSTATE 51041)。

注

- 変更はシステム・カタログに書き込まれますが、コミットされるまでは有効になりません。これは、ステートメントを発行する接続の場合でも当てはまります。
- データベース接続が確立されると、データベース・マネージャーは、POSITION 節で指定された接続属性に基づいて (指定された順番で) 一致するワークロードを検索します。一致するワークロードが見つかり、データベース・マネージャーは、現行のセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを確認します。セッション・ユーザーにそのワークロードに対する USAGE 特権がない場合、データベース・マネージャーは次の一致するワークロードを検索します。セッション・ユーザーにこのワークロードに対する USAGE 特権がある場合は、接続はそのワークロードに関連付けられます。一致するワークロードが見つからない場合は、接続はデフォルト・ユーザー・ワークロード SYSDEFAULTUSERWORKLOAD に関連付けられます。セッション・ユーザーに SYSDEFAULTUSERWORKLOAD に対する USAGE 特権がない場合は、エラーが返されます (SQLSTATE 42501)。
- データベース・マネージャーが以下のいずれかの状態を検出した場合は、新規作業単位が開始されるごとにワークロードの関連付けが再評価されます。
 - 接続属性が変更されている。これは、次のいずれかのイベントが発生した場合に起きることがあります。
 - クライアント情報設定 API (sqlseti) が呼び出され、それによってワークロード定義に含まれている接続属性が変更された。ワークロードの再評価を開始できるようにエンド・ユーザーがクライアント情報を設定できたとしても、セッション・ユーザーにそのワークロードに対する USAGE 特権がなければ、ワークロードの再マップ自体は不可能であることに注意してください。
 - SET SESSION AUTHORIZATION ステートメントが呼び出され、それによって現行セッション・ユーザーが変更された。
 - セッション・ユーザーが使用できるロールが変更された。
 - ワークロードが作成されている。
 - ワークロードがドロップされている。
 - ワークロードが変更されている。
 - ワークロードに対する USAGE 特権がユーザー・グループ、またはロールに付与されている。
 - ワークロードに対する USAGE 特権がユーザー・グループ、またはロールから取り消されている。

ワークロードの再評価によって再割り当てされるワークロードがない場合、結果として現行のワークロード・オカレンスが引き続き実行されます。つまり、新規のワークロード・オカレンスは開始されません。

- アクティビティーがまだアクティブである間は、別のワークロードに接続を再割り当てできません。このようなアクティビティーの例としては、オープン WITH

HOLD カーソルのような、複数の作業単位にまたがってリソースを保守するロード操作、実行中のプロシージャ、またはステートメントが挙げられます。現行のワークロード・オカレンスは、実行中のアクティビティがすべて完了するまで引き続き実行されます。ワークロードの再アサインは、次の作業単位の開始時に実行されます。

- ワークロードによって参照されているサービス・クラスは、ワークロードによって参照されなくなる時点までドロップできません。ワークロードからサービス・クラスの参照を除去するには、次のいずれかのアクションを実行できます。
 - ワークロードに変更を加えてサービス・クラス名を変える
 - ワークロードをドロップする
- ワークロードによって参照されているロールは、ワークロードによって参照されなくなるまでドロップできません。ワークロードからロールの参照を除去するには、次のいずれかのアクションを実行できます。
 - ワークロードに変更を加えてロールを除去する
 - ワークロードをドロップする

例

例 1: グループ FINANCE に属するセッション・ユーザーによってサブミットされた要求に対して、CAMPAIGN という名前のワークロードを作成します。これらの要求は、デフォルト・ユーザー・サービス・クラス SYSDEFAULTUSERCLASS で実行されます。

```
CREATE WORKLOAD CAMPAIGN
SESSION_USER GROUP ('FINANCE')
```

例 2: ロール HR のセッション・ユーザー用に、CURRENT CLIENT_APPLNAME 特殊レジスターを SALARYSYS に設定した PAYROLL という名前のワークロードを作成します。このワークロードに関連付けられている作業単位は、サービス・スーパークラス HRSC の下のサービス・クラス MEDIUMSC で実行されます。実行時にワークロードの選択が実行される際、このワークロードは、必ずワークロード CAMPAIGN が評価され、一致しないことが確認された後で評価される必要があります。

```
CREATE WORKLOAD PAYROLL
SESSION_USER ROLE ('HR')
CURRENT_CLIENT_APPLNAME ('SALARYSYS') SERVICE CLASS MEDIUMSC
UNDER HRSC POSITION AFTER CAMPAIGN
```

例 3: ワークロード CAMPAIGN のオカレンス (例 1 より) は、現在システムで実行中です。グループ FINANCE に属するセッション・ユーザーがサブミットした要求に対しても、NEWCAMPAIGN という名前のワークロードを作成しますが、このワークロードではアプリケーション DB2BP.EXE を通してサブミットされた要求だけを扱います。このワークロードに関連付けられた要求は、サービス・クラス MARKETINGSC で実行されます。NEWCAMPAIGN は CAMPAIGN よりも先に評価されます。

```
CREATE WORKLOAD NEWCAMPAIGN
SESSION_USER GROUP ('FINANCE')
APPLNAME ('DB2BP.EXE') SERVICE CLASS MARKETINGSC
POSITION BEFORE CAMPAIGN
```

CREATE WORKLOAD

CAMPAIGN の実行中のワークロード・オカレンスは、現在の作業単位が完了するまで引き続き実行されます。そのときに、ワークロードの再評価が行われ、接続はワークロード NEWCAMPAIGN に再マップできるようになります。

例 4: システム・ユーザー BOB および MARY によってアプリケーション app11、app12、または app13 からサブミットされた要求に対して、REPORTS という名前のワークロードを作成します。

```
CREATE WORKLOAD REPORTS
  APPLNAME ('app11', 'app12', 'app13')
  SYSTEM_USER ('BOB', 'MARY')
```

CREATE WRAPPER

CREATE WRAPPER ステートメントは、ラッパーをフェデレーテッド・サーバーに登録します。ラッパーは、フェデレーテッド・サーバーがデータ・ソースの特定のタイプと対話するためのメカニズムです。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

構文

```
CREATE WRAPPER wrapper-name [LIBRARY library-name]
[OPTIONS (ADD wrapper-option-name string-constant)]
```

説明

wrapper-name

ラッパーの名前を指定します。次のような名前にすることができます。

- 事前定義名。事前定義名を指定すると、フェデレーテッド・サーバーは *library-name* に自動的にデフォルト値を割り当てます。
- ユーザーが指定する名前。ユーザー指定の名前を提供する場合は、そのラッパーおよびオペレーティング・システムとともに使用する適切な *library-name* も指定する必要があります。

LIBRARY *library-name*

ラッパー・ライブラリー・モジュールを含むファイルの名前を指定します。

ライブラリー名は、絶対パス名または単にベース名 (パスなし) として指定できます。ベース名だけを指定する場合は、ライブラリーは DB2 インストール・パスの `lib` (UNIX)、または `bin` (Windows) サブディレクトリーに存在していなければなりません。 `library-name` は単一引用符で囲む必要があります。

`LIBRARY` オプションが必要なのは、ユーザーが指定した `wrapper-name` を使用する場合だけです。事前定義された `wrapper-name` を指定する場合には、このオプションは使用しません。

OPTIONS (ADD *wrapper-option-name string-constant*, ...)

ラッパー・オプションは、ラッパーを構成するため、または DB2 によるラッパーの使用法を定義するために使用します。 `wrapper-option-name` はオプションの名前です。 `string-constant` は、ラッパー・オプションの設定を指定します。 `string-constant` は単一引用符で囲む必要があります。すべてのラッパーによって使用されるラッパー・オプションもあれば、特定のラッパー固有のオプションもあります。

例

例 1: Oracle データ・ソースにアクセスするために、フェデレーテッド・サーバー上の NET8 ラッパーを登録します。Oracle データ・ソースへのアクセスに使用できるラッパーは 2 つあり、`NET8` はそのうちの 1 つの事前定義名です。

```
CREATE WRAPPER NET8
```

例 2: ODBC データ・ソースにアクセスするために、Linux オペレーティング・システムを使用する DB2 フェデレーテッド・サーバー上にラッパーを登録します。フェデレーテッド・データベースに登録されているラッパーに `odbc` という名前を割り振ります。ODBC Driver Manager を含むライブラリーの絶対パスは、ラッパー・オプション `MODULE '/usr/lib/odbc.so'` に定義されています。

```
CREATE WRAPPER odbc OPTIONS (MODULE '/usr/lib/odbc.so')
```

例 3: ODBC データ・ソースにアクセスするために、Windows オペレーティング・システムを使用する DB2 フェデレーテッド・サーバー上にラッパーを登録します。ODBC ラッパーのライブラリー名は `'db2rcodbc.dll'` です。

```
CREATE WRAPPER odbc LIBRARY 'db2rcodbc.dll'
```

例 4: Entrez データ・ソースにアクセスするために、AIX オペレーティング・システムを使用する DB2 フェデレーテッド・サーバー上にラッパーを登録します。ラッパーの名前を `entrez_wrapper` と指定します。AIX フェデレーテッド・サーバー上で、Entrez ラッパーのライブラリー・ファイルは `libdb2lsentrez.a` です。Entrez ラッパーをフェデレーテッド・サーバーに登録する場合は、`EMAIL` オプションが必要です。

```
CREATE WRAPPER entrez_wrapper LIBRARY 'libdb2lsentrez.a'
OPTIONS (EMAIL 'jeff@someplace.com')
```

DECLARE CURSOR

`DECLARE CURSOR` ステートメントは、カーソルを定義します。

DECLARE CURSOR

呼び出し

対話式 SQL 機能には外見上対話式の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

許可

「カーソルの SELECT ステートメント」という用語は、以下の許可規則を示すために使用されます。カーソルの SELECT ステートメントは、次のいずれかです。

- *statement-name* (ステートメント名) によって識別され、準備される選択ステートメント。
- 指定された *select-statement* (選択ステートメント)

カーソルの SELECT ステートメントに (直接的にか、または別名を使用するかのいずれか) 指定する表またはビューのそれぞれについて、ステートメントの許可 ID の特権に、以下の特権が少なくとも 1 つ含まれている必要があります。

- *select-statement* で指定された表またはビューのそれぞれに対する以下のいずれかの特権。
 - 表またはビューに対する SELECT 特権、または
 - 表またはビューに対する CONTROL 特権
- SYSADM または DBADM 権限

select-statement に SQL データ変更ステートメントが含まれている場合は、そのステートメントの許可要件も DECLARE CURSOR ステートメントに適用されます。

statement-name を指定した場合:

- ステートメントの許可 ID は、ランタイム許可 ID になります。
- 許可検査は、選択ステートメントが準備される時点で行われます。
- 選択ステートメントの準備が成功しない限り、カーソルはオープンされません。

select-statement を指定した場合:

- GROUP 特権は検査されません。
- ステートメントの許可 ID は、プログラム作成時に指定される許可 ID になります。

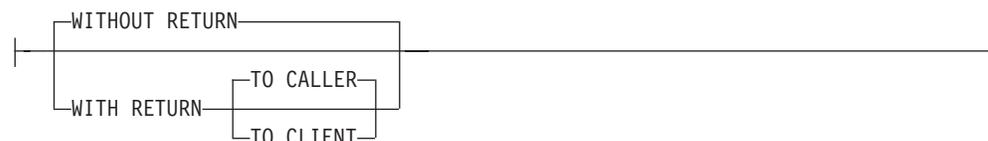
構文

```
▶▶ DECLARE cursor-name CURSOR ● | holdability | ● | returnability | ● ▶▶
▶ FOR { select-statement | statement-name } ▶▶
```

holdability:



returnability:



説明

cursor-name

ソース・プログラムの実行時に作成されるカーソルの名前を指定します。この名前は、ソース・プログラムに宣言されている他のカーソルの名前と同じではありません。カーソルは、その使用に先立ってオープンする必要があります。

WITHOUT HOLD または WITH HOLD

コミット操作の結果としてカーソルをクローズすることを回避するかどうかを指定します。

WITHOUT HOLD

コミット操作の結果としてカーソルをクローズすることを回避しません。これはデフォルトです。

WITH HOLD

複数の作業単位を通してリソースを維持します。WITH HOLD カーソル属性の効果は次のとおりです。

- COMMIT で終了する作業単位の場合:
 - WITH HOLD として定義されたオープン・カーソルはオープンされたままです。カーソルは、結果表の次の論理行の前に位置づけられます。

WITH HOLD カーソルとの接続に対して、COMMIT ステートメントの後で DISCONNECT ステートメントが出された場合、保留されたカーソルを明示的にクローズする必要があります。そうしない場合、(SQL ステートメントが全く発行されていない場合でも単に WITH HOLD カーソルをオープンしたままにすることによって) その接続が作業を行っているとは想定され、その DISCONNECT ステートメントはエラーになります。

- オープンされている WITH HOLD カーソルの現行カーソル位置を保護するロック以外のすべてのロックが解放されます。保留されるロックには、表に対するロックと、並列環境の場合はカーソルが現在位置している行に対するロックがあります。パッケージと動的 SQL セクション (存在する場合) に対するロックは保留されます。

DECLARE CURSOR

- WITH HOLD の定義されたカーソルに対して、COMMIT 要求の直後に有効な操作は、次のとおりです。
 - FETCH: カーソルの次の行を取り出します。
 - CLOSE: カーソルをクローズします。
- UPDATE および DELETE CURRENT OF CURSOR は、同一作業単位内で取り出された行に対してのみ有効です。
- LOB ロケータは解放されます。
- 以下によって変更された行のセットがコミットされます。
 - データ変更ステートメント
 - オープン WITH HOLD カーソルに組み込まれている、SQL データを変更するルーチン
- ROLLBACK で終了する作業単位の場合:
 - オープン・カーソルはすべてクローズされます。
 - その作業単位の過程で獲得したロックはすべて解除されます。
 - LOB ロケータは解放されます。
- 特殊な COMMIT の場合:
 - パッケージは、パッケージをバインドすることによって明示的に再作成されるか、または無効になった後、それが初めて参照されるときに動的に再作成されることにより暗黙のうちに再作成されます。保留されたカーソルはすべて、パッケージの再バインド時にはクローズされます。そのような場合、それ以後の実行時にエラーになる場合があります。

WITHOUT RETURN または WITH RETURN

カーソルの結果表を、プロシージャから戻される結果セットとして使用するかどうかを指定します。

WITHOUT RETURN

カーソルの結果表を、プロシージャから戻される結果セットとして使用しないことを指定します。

WITH RETURN

カーソルの結果表を、プロシージャから戻される結果セットとして使用することを指定します。WITH RETURN が使用されるのは、DECLARE CURSOR ステートメントにプロシージャのソース・コードが含まれている場合だけです。これ以外の場合は、プリコンパイラがこの節を受け入れても、この節は効力を持ちません。

SQL プロシージャでは、WITH RETURN 節を使用して宣言されたカーソルは SQL プロシージャの終了後もクローズされずに残り、SQL プロシージャからの結果セットを定義します。そして、その他のオープン・カーソルは、SQL プロシージャが終了するときにすべてクローズされます。外部プロシージャ (LANGUAGE SQL を使用して定義されていないもの) では、すべてのカーソルのデフォルトは WITH RETURN TO CALLER です。したがって、外部プロシージャの終了時に残っているオープン・カーソルがすべて結果セットと見なされます。プロシージャから戻されるカーソルを両方向スクロール・カーソルとして宣言することはできません。

TO CALLER

カーソルが呼び出し側に結果セットを返すよう指定します。たとえば、他のプロシージャから呼び出しが行われた場合は、そのプロシージャに結果セットが返されます。また、呼び出し側がクライアント・アプリケーションであるなら、そのクライアント・アプリケーションに結果セットが返されます。

TO CLIENT

カーソルがクライアント・アプリケーションに結果セットを返すよう指定します。このカーソルは、中間にネストされたプロシージャからは認識されません。関数、メソッド、またはトリガーがプロシージャを直接または間接的に呼び出した場合は、結果セットをクライアントに返すことができず、プロシージャの終了後にカーソルがクローズされます。

select-statement

カーソルの **SELECT** ステートメントを指定します。その *select-statement* には、パラメーター・マーカーを含めることはできませんが、ホスト変数への参照は含めることができます。参照されるホスト変数の宣言は、ソース・プログラムにおいて **DECLARE CURSOR** ステートメントよりも前になければなりません。

statement-name

カーソルの **SELECT** ステートメントは、カーソルのオープン時に *statement-name* によって指定される準備済み **SELECT** ステートメントです。*statement-name* は、ソース・プログラムの他の **DECLARE CURSOR** ステートメントに指定されている *statement-name* と同じであってはなりません。

準備済み **SELECT** ステートメントの説明については、『**PREPARE**』を参照してください。

注

- 他のプログラムから呼び出されたプログラム、または同じプログラムの別のソース・ファイルから呼び出されたプログラムで、呼び出し側プログラムによってオープンされたカーソルを使用することはできません。
- **SQL** 以外の **LANGUAGE** を使用する、ネストされていないプロシージャには、**WITH RETURN** 節を使用せずに **DECLARE CURSOR** が指定されるとデフォルトで **WITH RETURN TO CALLER** を使用し、カーソルをクローズせずにプロシージャに残すという性質があります。このようにすることによって、適当なクライアント・アプリケーションに結果セットを返すことができる以前のバージョンのプロシージャにも対応することができます。この性質を無効にするには、プロシージャでオープンされているカーソルをすべてクローズしてください。
- カーソルの **SELECT** ステートメントが **CURRENT DATE**、**CURRENT TIME**、または **CURRENT TIMESTAMP** を含む場合、これらの特殊レジスターを参照すると、それぞれの **FETCH** でそれぞれの同一の日時値が与えられます。この値は、カーソルがオープンされた時点で決まります。この値は、カーソルのオープン時に決まります。
- データをより効率的に処理するために、データベース・マネージャーでは、リモート・サーバーからデータを検索するときに、読み取り専用カーソルに対しては

DECLARE CURSOR

データ変更を禁止することができます。FOR UPDATE 節を使用するなら、データベース・マネージャーで、カーソルが更新可能かどうかを決めることができます。更新可能性は、アクセス・パス選択を決めるためにも使用されます。カーソルを位置指定 UPDATE または DELETE ステートメントで使用しない場合は、FOR READ ONLY として宣言してください。

- オープン状態のカーソルは、結果表と、その表の行に対する相対位置を示します。表は、カーソルの SELECT ステートメントによって指定される結果表です。
- カーソルは、以下の各項目が真となる場合に削除可能 です。
 - 外部全選択の各 FROM 節に、OUTER 節を使用しないで、基本表または削除可能ビュー (ネストした表式や共通表式またはニックネームを指定できない) が指定されている
 - 外部全選択に VALUES 節が含まれない
 - 外部全選択に GROUP BY 節も HAVING 節も含まれない
 - 外部全選択の選択リストに列関数が含まれない
 - 外部全選択に、UNION ALL を除くセット演算 (UNION、EXCEPT、または INTERSECT) が含まれない
 - 外部全選択の選択リストに DISTINCT が含まれない
 - 外部全選択に ORDER BY 節が含まれておらず (ORDER BY 節がビューにネストされていてもよい)、FOR UPDATE 節が指定されていない
 - 選択ステートメントに FOR READ ONLY 節が含まれない
 - 最外部の全選択の FROM 節に *data-change-table-reference* が含まれない
 - 次の 1 つまたは複数が真である
 - FOR UPDATE 節が指定されている
 - カーソルが静的に定義されており、STATICREADONLY BIND オプションが YES になっていない
 - LANGLEVEL BIND オプションが MIA または SQL92E である

カーソルに関連する外部全選択の選択リスト内の列は、以下の各項目が真となる場合に、更新可能 です。

- カーソルが削除可能である
- 列の解決結果が基本表の列となる
- LANGLEVEL BIND オプションが MIA の場合、SQL92E または *select-statement* が FOR UPDATE 節を含んでいる (列が FOR UPDATE 節で明示的または暗黙的に指定されている必要があります)

カーソルが読み取り専用 であるのは、削除可能でない場合です。

カーソルは、以下の各項目が真となる場合に未確定 です。

- 選択ステートメントが動的に準備される
- 選択ステートメントに FOR READ ONLY 節も FOR UPDATE 節も含まれていない
- LANGLEVEL BIND オプションが SAA1 である
- それ以外の点では、カーソルは削除可能カーソルの条件を満たしている

未確定カーソルは、BLOCKING BIND オプションが ALL の場合には読み取り専用と見なされます。そうでない場合は、更新可能と見なされます。

- CLI を使用して作成されたアプリケーション・プログラムによって呼び出されるプロシージャの中のカーソルは、クライアント・アプリケーションに直接返される結果セットを定義するために使用することができます。また、SQL プロシージャが WITH RETURN 節を使用して定義される場合に限り、そのプロシージャの中のカーソルを呼び出し側の SQL プロシージャに返すこともできます。
- WITH HOLD を宣言したカーソルから直接または間接的に呼び出されるルーチン内で宣言されるカーソルは、WITH HOLD オプションを継承しません。したがって、ルーチン内のカーソルが明示的に WITH HOLD と定義されない限り、カーソルはアプリケーションの COMMIT によってクローズされます。

次のようなアプリケーションと 2 つの UDF について考慮します。

アプリケーション:

```
DECLARE APPCUR CURSOR WITH HOLD FOR SELECT UDF1() ...
OPEN APPCUR
FETCH APPCUR ...
COMMIT
```

UDF1:

```
DECLARE UDF1CUR CURSOR FOR SELECT UDF2() ...
OPEN UDF1CUR
FETCH UDF1CUR ...
```

UDF2:

```
DECLARE UDF2CUR CURSOR WITH HOLD FOR SELECT UDF2() ...
OPEN UDF2CUR
FETCH UDF2CUR ...
```

アプリケーションがカーソル APPCUR を取り出した後は、3 つのカーソルすべてがオープンになります。アプリケーションが COMMIT ステートメントを発行すると、APPCUR は、WITH HOLD と宣言されているのでオープンのままになります。しかし、UDF1 では、カーソル UDF1CUR は、WITH HOLD オプションを指定して定義されていないのでクローズされます。カーソル UDF1CUR がクローズされると、対応する選択ステートメント内のすべてのルーチン呼び出しが完了します (最終呼び出しを受け取るように定義されている場合は、それを受け取ります)。UDF2 が完了し、UDF2CUR がクローズされます。

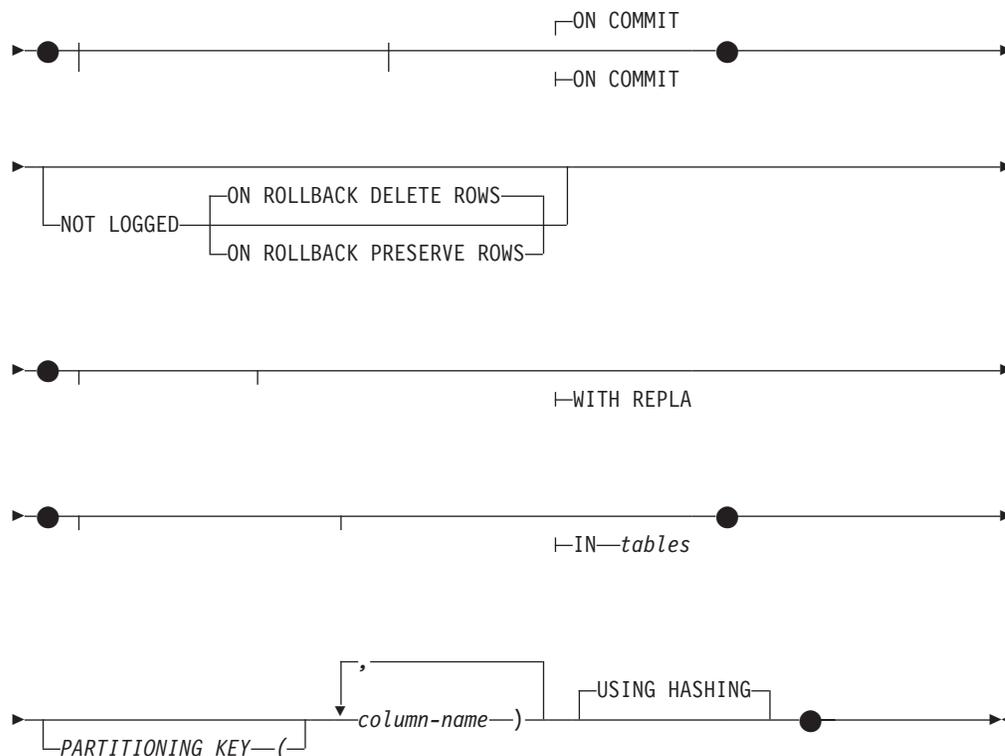
例

例 1: DECLARE CURSOR ステートメントは、SELECT の結果にカーソル名 C1 を関連付けます。

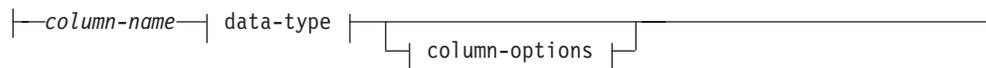
```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT DEPTNO, DEPTNAME, MGRNO
FROM DEPARTMENT
WHERE ADMRDEPT = 'A00';
```

例 2: EMPLOYEE 表が、生成された列 WEEKLYPAY (年間の給与に基づいて週ごとの支払いを計算する) を追加するように調整されていると想定します。カーソルを宣言して、挿入される行からシステムが生成した列の値を取り出します。

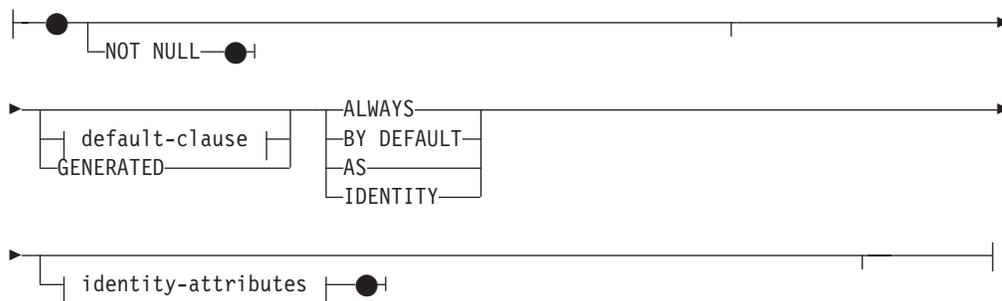
DECLARE GLOBAL TEMPORARY TABLE



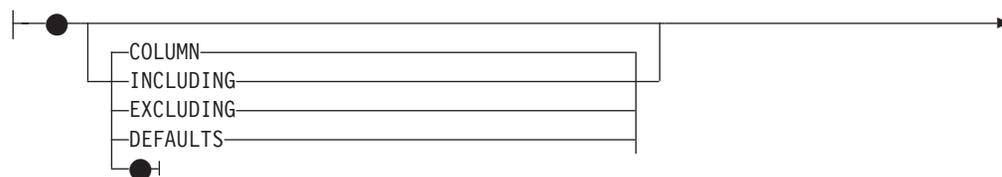
column-definition:



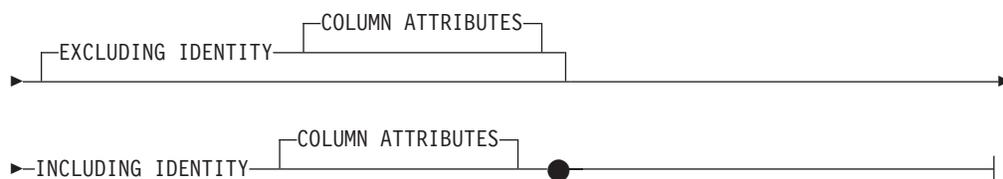
column-options:



copy-options:



DECLARE GLOBAL TEMPORARY TABLE



説明

table-name

一時表の名前を示します。修飾子を明示的に指定する場合は、SESSION でなければなりません。そうしないと、エラーになります (SQLSTATE 428EK)。修飾子が指定されなければ、暗黙的に SESSION が指定されます。

同じ *table-name* の宣言済みグローバル一時表を定義するセッションであっても、その宣言済みグローバル一時表の記述はそれぞれのセッションによって異なります。 *table-name* を使用する宣言済み一時表がセッション内にすでに存在している場合は、 WITH REPLACE 節を指定する必要があります (SQLSTATE 42710)。

表、ビュー、別名、またはニックネームについては、同じ名前および同じスキーマ名 (SESSION) を持つものがカタログ内にすでに存在していても構いません。このような場合には、次のような処理が行われます。

- 宣言されているグローバル一時表 *table-name* は、正常に定義されます (エラーや警告は戻されません)。
- SESSION.*table-name* への参照はすべて、カタログ内ですでに定義されている SESSION.*table-name* ではなく、宣言済みグローバル一時表に対して行われます。

column-definition

一時表の列の属性を定義します。

column-name

表を構成する列の名前を指定します。名前を修飾したり、表の複数の列に対して同じ名前を使用することはできません (SQLSTATE 42711)。

表には、以下のものを指定できます。

- 4K ページ・サイズの場合、最大 500 列。列のバイト・カウントは 4 005 を超えてはなりません。
- 8K ページ・サイズの場合、最大 1 012 列。列のバイト・カウントは 8 101 を超えてはなりません。
- 16K ページ・サイズの場合、最大 1 012 列。列のバイト・カウントは 16 293 を超えてはなりません。
- 32K ページ・サイズの場合、最大 1 012 列。列のバイト・カウントは 32 677 を超えてはなりません。

詳細については、『CREATE TABLE』の『行サイズ』を参照してください。

data-type

許容されるタイプについては、『CREATE TABLE』の *data-type* を参照してください。宣言済みグローバル一時表では、BLOB、CLOB、

DECLARE GLOBAL TEMPORARY TABLE

DBCLOB、LONG VARCHAR、LONG VARGRAPHIC、XML、参照、および構造化タイプを使用できませんのでご注意ください (SQLSTATE 42962)。この制限には、これらのタイプをソースとする特殊タイプが含まれます。

FOR BIT DATA は、文字ストリング・データ・タイプの一部として指定することができます。

column-options

表の列に関連した追加オプションを定義します。

NOT NULL

列に NULL 値が入るのを防止します。NULL 値の指定については、『CREATE TABLE』の NOT NULL を参照してください。

default-clause

デフォルトの指定については、『CREATE TABLE』の *default-clause* を参照してください。

IDENTITY および *identity-attributes*

ID 列の指定については、『CREATE TABLE』の IDENTITY および *identity-attributes* を参照してください。

LIKE *table-name1* または *view-name*

表の列の名前と記述が、指定された表 (*table-name1*)、ビュー (*view-name*)、またはニックネーム (*nickname*) の列とまったく同じであることを示します。LIKE の後に指定する名前は、カタログに存在している表、ビュー、またはニックネーム、あるいは宣言済み一時表を識別するものでなければなりません。型付き表または型付きビューを指定することはできません (SQLSTATE 428EC)。

LIKE は、*n* 列の暗黙的な定義で使用します。*n* は、指定した表 (暗黙的な隠し列を含む) またはビューに含まれる列の数を表します。既存の表の暗黙的な隠し列に対応する新規表の列も暗黙的な隠し列として定義されます。

- 表が特定されると、暗黙的な定義には *table-name1* のそれぞれの列の列名、データ・タイプ、および NULL 可能特性が入ります。EXCLUDING COLUMN DEFAULTS を指定しないと、列のデフォルト値も入ります。
- ビューが特定されると、暗黙的な定義には *view-name* に指定した全選択のそれぞれの結果列の列名、データ・タイプ、および NULL 可能特性が入ります。

copy-attributes 節に基づいて、列のデフォルトと ID 列属性を組み込んだり除外したりすることができます。

保護表が LIKE 節内で定義されている場合、新規表は保護表になりません。

表が LIKE 節内で定義されていて、その表に ROW CHANGE TIMESTAMP 列が含まれている場合、新規表の対応する列は ROW CHANGE TIMESTAMP 列のデータ・タイプのみを継承します。新規列は生成された列とは見なされません。

table-name1 や *view-name* には、作成するグローバル一時表と同じ名前を使用することはできません (SQLSTATE 428EC)。

AS (*fullselect*) DEFINITION ONLY

表の列の名前と記述が、全選択を実行した場合に全選択の派生結果表に現れる列

DECLARE GLOBAL TEMPORARY TABLE

と同じになるよう指定します。AS (*fullselect*) は、宣言済みグローバル一時表に対する n 列の暗黙的な定義で使用されます。 n は、全選択の結果として得られる列の数を表します。

暗黙的な定義には、 n 列の以下の属性が含まれます (データ・タイプに該当する場合):

- 列名
- データ・タイプ、長さ、精度、および位取り
- NULL 可能

以下の属性は含まれません (デフォルト値および識別属性は、*copy-options* を使用して含めることができます):

- デフォルト値
- 識別属性
- ROW CHANGE TIMESTAMP

全選択で参照される表やビューの他のオプションの属性は、暗黙的な定義には含まれません。

選択リストの各エレメントの名前は、それぞれユニークなものでなければなりません (SQLSTATE 42711)。SELECT 節で AS 節を使用すると、それぞれのエレメントにユニークな名前を付けることができます。全選択の結果が LOB データ・タイプを持つ列となってはなりません。全選択が、ホスト変数を参照したり、パラメーター・マーカを含んでいたりはなりません。

copy-options

これらのオプションでは、ソースの結果表定義 (表、ビュー、または全選択) から付加的な属性をコピーするかどうかを指定します。

INCLUDING COLUMN DEFAULTS

ソース結果表の定義の更新可能な各列の列デフォルトをコピーします。更新可能でない列では、作成される表の対応列にデフォルトが定義されないこととなります。

LIKE *table-name1* が指定されており、かつ *table-name1* が基本表か宣言済み一時表である場合に限り、この INCLUDING COLUMN DEFAULTS がデフォルトとして使用されます。

EXCLUDING COLUMN DEFAULTS

列のデフォルトは、ソースの結果表定義からコピーされません。

この節がデフォルトです。ただし、LIKE *table-name* が指定され、かつ *table-name* が基本表または宣言済み一時表を示す場合を除きます。

INCLUDING IDENTITY COLUMN ATTRIBUTES

この節を使用すると、ソースの結果表定義から ID 列の属性 (START WITH、INCREMENT BY、および CACHE の値) がコピーされます。これらの属性をコピーできるのは、表、ビュー、または全選択内の対応する列のエレメントが、ID のプロパティが含まれている基本表の列名に直接または間接的にマップされた表の列の名前、またはビューの列の名前である場合です。これ以外の場合は、新しい一時表の列に ID のプロパティは定義されません。以下に例を示します。

DECLARE GLOBAL TEMPORARY TABLE

- 全選択の選択リストに ID 列の名前のインスタンスが複数含まれている (つまり、同じ列を複数回選択している) 場合
- 全選択の選択リストに複数の ID 列が含まれている (つまり、結合が関与している) 場合
- ID 列が選択リスト内の式に組み込まれている場合
- 全選択にセット演算 (UNION (合併)、EXCEPT (差)、または INTERSECT (論理積)) が含まれている場合

EXCLUDING IDENTITY COLUMN ATTRIBUTES

ソース結果表の定義から ID 列属性はコピーされません。

ON COMMIT

COMMIT 操作の実行時にグローバル一時表で行うアクションを指定します。

DELETE ROWS

表にオープンされている WITH HOLD カーソルがなければ、すべての行が表から削除されます。これはデフォルトです。

PRESERVE ROWS

表の行が保存されます。

NOT LOGGED

表に対する挿入、更新、または削除操作をログに記録せず、表の作成またはドロップをログに記録するよう指定します。ROLLBACK (または ROLLBACK TO SAVEPOINT) 操作中に:

- 表が作業単位 (またはセーブポイント) 内で作成された場合、表はドロップされます。
- 表が作業単位 (またはセーブポイント) 内でドロップされた場合、表は再作成されますが、データは消失します。

ON ROLLBACK

ROLLBACK (または ROLLBACK TO SAVEPOINT) 操作が実行されるときに、記録されていないグローバル一時表に対して取られるアクションを指定します。

DELETE ROWS

表データが変更されている場合は、すべての行が削除されます。これはデフォルトです。

PRESERVE ROWS

表の行が保存されます。

WITH REPLACE

ユーザーが指定した名前を持つ宣言済みグローバル一時表がすでに存在している場合は、既存の表をこのステートメントで定義した一時表で置き換える (および既存の表の行をすべて削除する) よう指示します。

WITH REPLACE が指定されていない場合は、現行セッションにすでに存在している宣言済みグローバル一時表の名前を指定することはできません (SQLSTATE 42710)。

IN *tablespace-name*

グローバル一時表をインスタンス化する表スペースを指定します。ここでは、既存の USER TEMPORARY 表スペースを指定する必要があります (SQLSTATE 42838)。また、ステートメントの許可 ID にはその表スペースに対する USE 特

DECLARE GLOBAL TEMPORARY TABLE

権が含まれていなければなりません (SQLSTATE 42501)。この節が指定されない場合、表をインスタンス化する表スペースは USER TEMPORARY 表スペースの中から選択され、その中のステートメントの許可 ID に USE 特権が含まれている表スペースで、かつ必要なページ・サイズに最も適したサイズの表スペースが使用されます。複数の表スペースがそれにあてはまる場合、以下のどれに USE 特権が付与されているかに応じて優先順位が決められます。

1. 許可 ID
2. 許可 ID を保有するグループ
3. PUBLIC

それでも複数の表スペースがそれにあてはまる場合は、最終選択はデータベース・マネージャーによって行われます。条件に合う USER TEMPORARY 表スペースがない場合はエラーが戻されます (SQLSTATE 42727)。

表スペースの決定は、以下の時点で変更することができます。

- 表スペースをドロップまたは作成するとき
- USE 特権を付与または取り消すとき

十分な表のページ・サイズは、行のバイト・カウントか列の数のいずれかによって決まります。詳細については、『CREATE TABLE』の『行サイズ』を参照してください。

PARTITIONING KEY (*column-name*,...)

表内のデータを分散するときに使用される分散キーを指定します。各 *column-name* (列名) は、表の列を指定するものでなければなりません。また、同じ列を複数回指定することはできません。

この節の指定がなく、この表が複数パーティション・データベースのパーティション・グループに存在する場合は、その分散キーが宣言済み一時表の最初の列として定義されます。

宣言済み一時表では、単一パーティションのデータベース・パーティション・グループに定義された表スペースにおいて、すべての列の集合を分散キーの定義に使用することができます。このパラメーターの指定がない場合、分散キーは作成されません。

USING HASHING

データ配分の方式として、ハッシュ関数を使用することを指定します。これは、サポートされる唯一の配分方式です。

注

- USER TEMPORARY 表スペースは、ユーザー定義の一時表が宣言される前に、存在しなくてはなりません (SQLSTATE 42727)。
- **宣言済みグローバル一時表の参照:** 宣言済みグローバル一時表の記述は DB2 カタログ (SYSCAT.TABLES) に現れないため、この記述は永続的なものではなく、またデータベース接続によって共有することもできません。従って、同じ *table-name* という宣言済みグローバル一時表を定義するセッションであっても、その宣言済みグローバル一時表の記述はそれぞれのセッションによって異なる可能性があります。

SQL ステートメント (DECLARE GLOBAL TEMPORARY TABLE ステートメントは除く) を使用して宣言済みグローバル一時表を参照するためには、その表を

DECLARE GLOBAL TEMPORARY TABLE

スキーマ名 `SESSION` で明示的または暗黙的に修飾する必要があります。
`table-name` が `SESSION` で修飾されていない場合、宣言済みグローバル一時表は参照を決定する際に認識されません。

グローバル一時表が名前によって宣言されていない接続で `SESSION.table-name` を参照する場合は、カタログ内の持続オブジェクトから参照先が決定されます。そのオブジェクトが存在しない場合はエラーが戻されます (SQLSTATE 42704)。

- バインドしているパッケージに、`SESSION` によって暗黙的または明示的に修飾された表を参照する静的 SQL ステートメントが含まれている場合、それらのステートメントは静的にはバインドされません。これらのステートメントは、呼び出されると、パッケージのバインドにおいて `VALIDATE` オプションが選択されているかどうかにかかわらず、バインドされ追加されていきます。ステートメントの実行時には、宣言済み一時表が存在する場合はその一時表に対して、存在しない場合は永続表に対して各表の参照が行われます。このどちらも存在しない場合はエラーが戻されます (SQLSTATE 42704)。
- **特権:** 宣言済みグローバル一時表を定義する場合、その表を定義するユーザーには、表をドロップする権限も含めて、その表に対するすべての表特権が付与されます。加えて、`PUBLIC` に対しても同様の特権が `GRANT` されます。(`GRANT` オプションによって `GRANT` される特権はありません。また、これらの特権はいずれもカタログ表には現れません。) これらの特権を持つユーザーは、すでに定義されている宣言済みグローバル一時表を参照するセッションで、すべての SQL ステートメントを実行することができます。
- **インスタンス化と終了:** 以下の説明において、それぞれ `P` はセッションを、`T` はセッション `P` の中の宣言済みグローバル一時表を示しています。
 - `T` の空のインスタンスは、`P` で実行される `DECLARE GLOBAL TEMPORARY TABLE` ステートメントの結果として作成されます。
 - `P` で実行されるすべての SQL ステートメントでは `T` を参照することができます。そして、`P` で `T` を参照する場合は、すべてその同じインスタンスが参照されます。
 - SQL プロシージャのコンパウンド・ステートメント (`BEGIN` と `END` で定義される) で `DECLARE GLOBAL TEMPORARY TABLE` ステートメントを指定すると、宣言済みグローバル一時表の有効範囲がコンパウンド・ステートメントだけでなく接続にまで広がり、表はコンパウンド・ステートメントの外部からも認識されるようになります。表はコンパウンド・ステートメントの `END` で暗黙的にドロップされません。宣言済みグローバル一時表は、その表が明示的にドロップされない限り、同じ名前を使用してセッション内の他のコンパウンド・ステートメントで複数回定義することはできません。
 - `ON COMMIT DELETE ROWS` 節が暗黙的または明示的に指定された場合は、`P` においてコミット操作が作業単位で終了し、`T` に属する `WITH HOLD` カーソルが 1 つも `P` にオープンされていない状態になると、操作 `DELETE FROM SESSION.T` がコミットに組み込まれます。
 - `P` において、作業単位またはセーブポイントでロールバック操作が終了し、その作業単位またはセーブポイントに `SESSION.T` への変更が含まれている場合、次のようになります。

DECLARE GLOBAL TEMPORARY TABLE

- NOT LOGGED が指定されている場合、ロールバックには SESSION.T からの DELETE 操作も含まれます (ON ROLLBACK PRESERVE ROWS も指定されている場合を除く)。
- NOT LOGGED が指定されていない場合、T への変更は取り消されます。

P において、作業単位またはセーブポイントでロールバック操作が終了し、その作業単位またはセーブポイントに SESSION.T への宣言が含まれている場合、このロールバック操作には DROP SESSION.T 操作が含まれます。

P において、作業単位またはセーブポイントでロールバック操作が終了し、その作業単位またはセーブポイントに宣言済み一時表 SESSION.T のドロップが含まれている場合、このロールバック操作によって表のドロップは取り消されます。NOT LOGGED が指定されている場合は、表も空にされます。

- アプリケーションによって、宣言された T が終了されたり、データベースとの接続が切断された場合、T はドロップされ、そのインスタンス化された行は破棄されます。
- T の宣言が行われたサーバーへの接続が終了すると、T はドロップされ、そのインスタンス化された行は破棄されます。
- **宣言済みグローバル一時表の使用に関する制限:** 宣言済みグローバル一時表には、以下のような使用上の制限があります。
 - ALTER、COMMENT、GRANT、LOCK、RENAME、または REVOKE ステートメントでこの一時表を指定することはできません (SQLSTATE 42995)。
 - CREATE ALIAS、CREATE FUNCTION (SQL スカラー、表、または行)、CREATE TRIGGER、または CREATE VIEW ステートメントでこの一時表を参照することはできません (SQLSTATE 42995)。
 - 参照制約でこの一時表を指定することはできません (SQLSTATE 42995)。
- **互換性**
 - DB2 for z/OS との互換性:
 - 以下の構文はデフォルトの振る舞いとして受け入れられます。
 - CCSID ASCII
 - CCSID UNICODE

DELETE

DELETE ステートメントは、表、ニックネーム、またはビュー、あるいは指定した全選択の基礎表、ニックネーム、またはビューから、行を削除します。ニックネームから行を削除すると、そのニックネームの参照先のデータ・ソース・オブジェクトから行を削除することになります。ビューから行を削除すると、このビューに対する削除操作に INSTEAD OF トリガーが定義されていない場合は、そのビューの基本となる表から行を削除することになります。このようなトリガーが定義されている場合は、トリガーが代わりに実行されます。

このステートメントには、以下の 2 つの形式があります。

- 検索 (*Searched*) DELETE 形式は、1 つまたは複数の行を削除するのに使用します (削除する行は検索条件によって、自由に限定できます)。

- 位置指定 (*Positioned*) DELETE 形式は、1 行だけを削除する場合に使用します (削除される行は、カーソルの現在位置によって決まります)。

呼び出し

DELETE ステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

許可

このステートメントのどの形式を実行する場合も、ステートメントの許可 ID に、以下の特権が少なくとも 1 つ含まれている必要があります。

- 削除する行を含む表、ビュー、またはニックネームに対する DELETE 特権
- 削除する行を含む表、ビュー、またはニックネームに対する CONTROL 特権
- SYSADM または DBADM 権限

検索 DELETE ステートメントを実行する場合、副照会で参照される表、ビュー、またはニックネームのそれぞれに対して、ステートメントの許可 ID に、以下の特権が少なくとも 1 つ含まれている必要があります。

- SELECT 特権
- CONTROL 特権
- SYSADM または DBADM 権限

ステートメントを処理するために使用されるパッケージが SQL92 規則を使用してプリコンパイルされる場合 (SQL92E または MIA の値を指定したオプション LANGLEVEL) で、検索 DELETE ステートメント形式の *search-condition* に表またはビューの列への参照が含まれている場合には、このステートメントの許可 ID の特権には以下の特権のうち少なくとも 1 つが含まれている必要があります。

- SELECT 特権
- CONTROL 特権
- SYSADM または DBADM 権限

指定した表またはビューが ONLY キーワードの後にくる場合、ステートメントの許可 ID が持つ特権にも、指定した表またはビューの副表またはサブビューごとに SELECT 特権が含まれている必要があります。

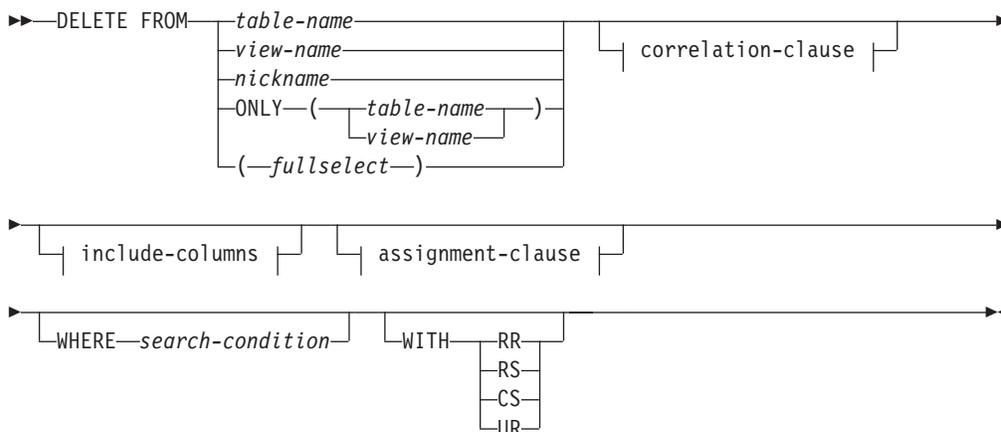
静的 DELETE ステートメントの場合、グループ特権は検査されません。

削除操作の対象がニックネームの場合は、データ・ソースのオブジェクトに対する特権は、ステートメントがデータ・ソースで実行されるまで考慮されません。この時点で、データ・ソースに接続するために使用される許可 ID は、データ・ソースのオブジェクトに対して操作を行うのに必要な特権を持っている必要があります。ステートメントの許可 ID は、データ・ソースの別の許可 ID へマップできます。

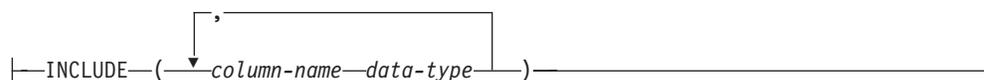
DELETE

構文

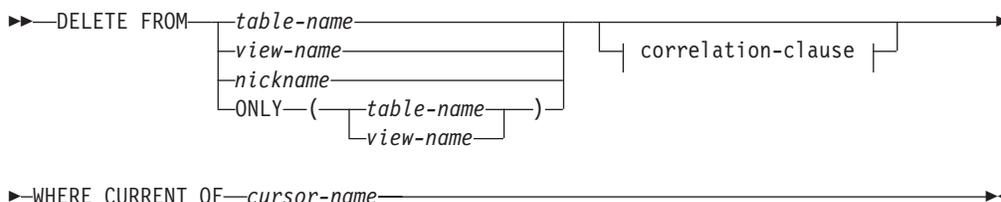
検索削除:



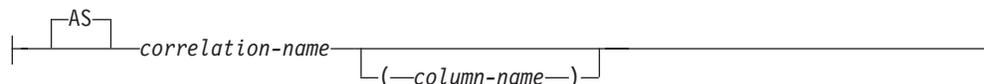
include-columns:



位置指定削除:



correlation-clause:



説明

FROM *table-name*、*view-name*、*nickname*、または (*fullselect*)

削除操作の対象のオブジェクトを指定します。この名前は、カタログに存在する表またはビューを指定するものでなければならず、カタログ表、カタログ・ビュー、システムによって保守されるマテリアライズ照会表、または読み取り専用のビューは指定できません。

table-name が型付き表である場合は、このステートメントを使用して、その表またはそれに関する副表の行を削除できます。

view-name が型付きビューである場合は、このステートメントを使用して、その基礎表の行またはそのビューに関係するサブビューの基礎表を削除できます。*view-name* が基礎表 (型付き表) を伴う通常のビューである場合、このステートメントを使用して、その型付き表またはそれに関係する副表の行を削除できません。

削除操作のオブジェクトが全選択である場合、全選択は、CREATE VIEW ステートメントの説明の『注』にある、『削除可能ビュー』の項目で定義されているように、削除可能になっている必要があります。

WHERE 節内で参照できるのは、指定された表の列だけです。位置指定 DELETE の場合は、FROM 節に指定されているのと同じ表またはビューを、関連するカーソルにも ONLY を使用せずに指定しなければなりません。

FROM ONLY (*table-name*)

型付き表の場合に適用できます。ONLY キーワードは、指定された表のデータだけにステートメントを適用し、その表に関係する副表の行は削除されないことを指定します。位置指定 DELETE の場合は、FROM 節に指定されているのと同じ表を、関連するカーソルにも ONLY を使用して指定しなければなりません。*table-name* が型付き表でない場合は、このステートメントに ONLY キーワードを使用しても効果はありません。

FROM ONLY (*view-name*)

このステートメントは型付きビューのみに適用されます。ONLY キーワードは、指定されたビューのデータだけにこのステートメントが適用されることを指定します。サブビューの行は、このステートメントでは削除されません。位置指定 DELETE の場合は、FROM 節に指定されているのと同じビューを、関連するカーソルにも ONLY を使用して指定しなければなりません。*view-name* が型付きビューでない場合は、このステートメントに ONLY キーワードを使用しても効果はありません。

correlation-clause

search-condition で、表、ビュー、ニックネームまたは全選択を指定するのに使用できます。*correlation-clause* についての説明は、『副選択』の説明にある『*table-reference*』を参照してください。

include-columns

全選択の FROM 節にネストされているとき、*table-name* や *view-name* などの列と一緒に DELETE ステートメントの中間結果表に組み込まれている列セットを指定します。*include-columns* は、*table-name* や *view-name* で指定されている列のリストの最後に付加されます。

INCLUDE

DELETE ステートメントの中間結果表に組み込まれる列のリストを指定します。

column-name

DELETE ステートメントの中間結果表の列を指定します。名前は、他の組み込み列や、*table-name* または *view-name* の列と同じ名前であってはなりません (SQLSTATE 42711)。

data-type

組み込み列のデータ・タイプを指定します。データ・タイプは、CREATE TABLE ステートメントでサポートされているものでなければなりません。

DELETE

assignment-clause

UPDATE ステートメントの *assignment-clause* の説明を参照してください。同じ規則が適用されます。 *include-columns* は、 *assignment-clause* を使用して設定できる唯一の列です (SQLSTATE 42703)。

WHERE

削除する行を選択する条件を指定します。この節は、省略するか、検索条件を指定するか、あるいはカーソルの名前を指定できます。この節を省略すると、表またはビューのすべての行が削除されます。

search-condition

副照会以外の検索条件の各 *column-name* (列名) は、表またはビューの列を指定していなければなりません。

search-condition は、該当の表、ビュー、またはニックネームの各行に適用されます。 *search-condition* の結果が「真」の行だけが削除されます。

検索条件に副照会が含まれる場合、その副照会は、 *search-condition* (検索条件) が行に適用されるたびに実行され、その結果が *search-condition* (検索条件) の適用の対象として使用されます。実際には、相関参照が含まれていない副照会は一度実行されるのに対し、相関参照の含まれている副照会は各行ごとに一度ずつ実行しなければならない場合があります。副照会で DELETE ステートメントの対象の表、または削除規則が CASCADE あるいは SET NULL の従属表が参照されている場合、その副照会は行が削除される前に完全に評価されます。

CURRENT OF *cursor-name*

プログラムの DECLARE CURSOR ステートメントで定義されたカーソルを指定します。 DECLARE CURSOR ステートメントは、DELETE ステートメントよりも前になければなりません。

指定する表、ビュー、またはニックネームは、そのカーソルの SELECT ステートメントの FROM 節でも指定されていなければならないが、またそのカーソルの結果表が読み取り専用であってはなりません。(読み取り専用の結果表については、『DECLARE CURSOR』を参照してください。)

DELETE ステートメントが実行される場合、カーソルは行の位置になければなりません。その行が削除されます。削除後、カーソル位置はその結果表の次の行の前になります。次の行がない場合、カーソル位置は最後の行の後になります。

WITH

削除する行を検索しているときに使用される分離レベルを指定します。

RR

反復可能読み取り

RS

読み取り固定

CS

カーソル固定

UR

非コミット読み取り

ステートメントのデフォルト分離レベルは、ステートメントがバインドされているパッケージの分離レベルです。WITH 節はニックネームには影響を与えません。ニックネームは常にステートメントのデフォルトの分離レベルを使用します。

規則

- **トリガー:** DELETE ステートメントによってトリガーの実行が引き起こされる場合があります。トリガーが他のステートメントの実行を引き起こす場合や、削除された行に起因するエラーが発生する場合があります。ビューの DELETE ステートメントが INSTEAD OF トリガーの発生を引き起こす場合は、参照保全是、トリガーの発生を引き起こしたビューの基礎表に対してではなく、トリガー内で実行される更新に対して検査されます。
- **参照保全:** 指定する表または指定するビューの基本表が親である場合、削除のために選択する行は RESTRICT の削除規則との関係において従属であってはならず、DELETE は RESTRICT の削除規則との関係において従属である下層行にカスケードしてはなりません。

削除操作が RESTRICT の削除規則によって禁止されていなければ、選択された行は削除されます。選択された行の従属行もすべて影響を受けます。

- 削除規則が SET NULL の関係において、すべての従属行の外部キーの NULL 可能列は、NULL 値に設定されます。
- 削除規則が CASCADE のリレーションシップにおいて、すべての従属行も削除され、上記の規則はこれらの行にも適用されます。

他の参照制約が実施された後で、非 NULL の外部キーが既存の親行を指すようにするために、NO ACTION の削除規則が検査されます。

- **セキュリティ・ポリシー:** 指定された表または指定されたビューの基本表がセキュリティ・ポリシーで保護されている場合、セッション許可 ID は、以下を許可するラベル・ベースのアクセス制御 (LBAC) クレデンシャルを持っている必要があります。
 - すべての保護された列に対する書き込みアクセス (SQLSTATE 42512)
 - 削除のために選択されたすべての行に対する読み取りおよび書き込みアクセス (SQLSTATE 42519)

注

- 複数行の DELETE の実行中にエラーが起こった場合、データベースは変更されません。
- 適切なロックがすでに存在するのでない限り、正常な DELETE ステートメントの実行中には、1 つまたは複数の排他ロックが獲得されます。COMMIT ステートメントまたは ROLLBACK ステートメントを発行すると、それらのロックは解放されます。ロックがコミットまたはロールバック操作によって解放される時まで、削除操作の効果は次のものにしか認識されません。
 - 削除を実行したアプリケーション・プロセス
 - 分離レベル UR を使用する別のアプリケーション・プロセス
 ロックにより、他のアプリケーション・プロセスが、表に対して操作を実行するのを防ぐことができます。

DELETE

- アプリケーション・プロセスがそのカーソルのいずれかがある行を削除すると、それらのカーソルの位置はその結果表の中の次の行の前になります。C をカーソルとし、それが (OPEN、C による DELETE、その他の何らかのカーソルによる DELETE、または検索 DELETE の結果として) 行 R の前の位置にあるとします。R の派生元の基本表に影響する INSERT、UPDATE、および DELETE 操作があると、C を参照する次の FETCH 操作では、必ずしも C の位置が R にある必要はありません。たとえば、この操作によって C が R' の位置になることがあります (R' は操作の結果表で次の行となった新しい行)。
- 削除操作の対象となる行数が SQLCA の SQLERRD(3) に示されます。SQL プロシージャ・ステートメントでは、値は GET DIAGNOSTICS ステートメントの ROW_COUNT 変数を使用して検索できます。参照制約による影響およびトリガーにより実行されたステートメントによる影響を受けた行の数は、SQLCA の SQLERRD(5) に示されます。これには、CASCADE 削除規則の結果として削除された行と、SET NULL 削除規則の結果として外部キーが NULL に設定された行とが含まれます。トリガーにより実行されたステートメントについては、挿入、更新、または削除された行数が含まれます。
- あるエラーが起きたために検索条件に合う行の削除がすべて完了しなかった場合、および既存の参照制約に必要なすべての操作が行われなかった場合、表は変更されずエラーが戻されます。
- ニックネームの場合は、外部サーバー・オプション iud_app_svpt_enforce によってさらに制限が加えられます。詳細については、フェデレーテッド・システムの資料を参照してください。
- 一部のデータ・ソースの場合、データが矛盾している可能性があるために、ニックネームに対する削除を行うと SQLCODE -20190 が戻される場合があります。詳細については、フェデレーテッド・システムの資料を参照してください。

例

例 1: DEPARTMENT 表から部門 (DEPTNO) 'D11' を削除します。

```
DELETE FROM DEPARTMENT
WHERE DEPTNO = 'D11'
```

例 2: DEPARTMENT 表からすべての部門を削除します (つまり、表を空にします)。

```
DELETE FROM DEPARTMENT
```

例 3: EMPLOYEE 表から 1995 年中に売上が 1 つもなかった営業担当員または現場担当員を削除します。

```
DELETE FROM EMPLOYEE
WHERE LASTNAME NOT IN
  (SELECT SALES_PERSON
   FROM SALES
   WHERE YEAR(SALES_DATE)=1995)
AND JOB IN ('SALESREP','FIELDREP')
```

例 4: EMPLOYEE 表から、重複している従業員の行をすべて削除します。従業員の行は、ラストネームが一致していれば、重複していると考えられます。従業員の行のファーストネームは、字句順に、できるだけ短くしておきます。

```
DELETE FROM
  (SELECT ROWNUMBER() OVER (PARTITION BY LASTNAME ORDER BY FIRSTNAME)
   FROM EMPLOYEE) AS E(RN)
WHERE RN = 1
```

DESCRIBE

DESCRIBE ステートメントは、オブジェクトについての情報を入手します。このステートメントで入手できる 2 タイプの情報があります。これらのそれぞれについて、個々に説明します。

- 準備済みステートメントの入力パラメーター・マーカー。準備済みステートメント内の入力パラメーター・マーカーについての情報を入手します。この情報は記述子に入れられます。
- 準備済みステートメントの出力。準備済みステートメントに関する情報、または準備済み SELECT ステートメント内の選択リスト列に関する情報を入手します。この情報は記述子に入れられます。

DESCRIBE INPUT

DESCRIBE INPUT ステートメントでは、準備済みステートメントの入力パラメーター・マーカーについての情報を取得します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

許可

必要ありません。

構文

```
►►—DESCRIBE INPUT—statement-name—INTO—descriptor-name—◄◄
```

説明

statement-name

準備済みステートメントを指定します。DESCRIBE INPUT ステートメントを実行する時点で、この名前は、現行のサーバーのアプリケーション・プロセスによってすでに準備されているステートメントを指定していなければなりません。

CALL ステートメントの場合は、プロシージャの入力パラメーター (IN または INOUT として定義されているパラメーター) に関する情報が戻されます。入力パラメーター・マーカーは、使用法に関係なく常に NULL 可能と見なされません。

INTO *descriptor-name*

SQL 記述子域 (SQLDA) を指定します。DESCRIBE INPUT ステートメントを実行する前に、SQLDA 内の以下の変数を設定しておく必要があります。

DESCRIBE INPUT

SQLN SQLDA に用意する **SQLVAR** のオカレンス数を指定します。

DESCRIBE INPUT ステートメントを実行する前に、**SQLN** にゼロ以上の値を設定する必要があります。

DESCRIBE INPUT ステートメントを実行すると、データベース・マネージャーは、以下のように **SQLDA** の変数に値を割り当てます。

SQLDAID

最初の 6 バイトは '**SQLDA**' に設定されます (5 文字の英字の後、6 文字目はスペース文字です)。

7 番目のバイト (**SQLDOUBLED** として定義されているバイト) は、記述されているパラメーター・マーカに基づいて設定されます。

- **SQLDA** に各入力パラメーターの **SQLVAR** 項目が 2 つ含まれている場合は、7 番目のバイトが '2' に設定されます。この技法によって、LOB または構造化タイプの入力パラメーターに対応できます。
- それ以外の場合、7 番目のバイトはスペース文字に設定されます。

SQLDA の中にすべての入力パラメーター・マーカの説明を入れるだけの余地がない場合は、7 番目のバイトがスペース文字に設定されます。

8 番目のバイトは、スペース文字に設定されます。

SQLDABC

SQLDA の長さ (バイト単位)。

SQLD プロシーチャーの **IN** パラメーターと **INOUT** パラメーターの数。

SQLVAR

SQLD の値が 0 の場合、または **SQLD** の値が **SQLN** の値より大きい場合は、**SQLVAR** のエレメントには値は割り当てられません。

SQLD の値が n (n は 0 より大きく **SQLN** の値以下) の場合は、**SQLVAR** の最初の n 個のオカレンスに値が割り当てられます。これらの値では、プロシーチャーの入力パラメーターのパラメーター・マーカを記述します。**SQLVAR** の最初のオカレンスでは、最初の入力パラメーター・マーカを記述し、**SQLVAR** の 2 番目のオカレンスでは、2 番目の入力パラメーター・マーカを記述する、といった具合になります。

基本 **SQLVAR**

SQLTYPE

パラメーターのデータ・タイプと、その列に **NULL** 値が入るかどうかを示すコード。

SQLLEN

パラメーターのデータ・タイプによって決まる長さを示す値。LOB データ・タイプの場合、**SQLLEN** は 0 になります。

SQLNAME

SQLVAR が、プロシーチャーのパラメーター・リスト内の式の一部分ではないパラメーター・マーカに対応する場合は、**CREATE PROCEDURE** ステートメントでパラメーターが指定されていれば、**sqlname** にはそのパラメーターの名前が含まれます。

2次 **SQLVAR**

これらの変数は、LOB、特殊タイプ、構造化タイプ、または参照タイプのパラメーターを含めることができるよう、SQLVAR の項目の数が 2 倍にされた場合にのみ使用されます。

SQLLONGLEN

BLOB、CLOB、または DBCLOB のパラメーターの長さ属性。

SQLDATATYPE_NAME

データベース・マネージャーは、すべてのユーザー定義タイプ (特殊タイプまたは構造化タイプ) のパラメーターで、この名前を完全修飾ユーザー定義タイプ名に設定します。また、参照タイプのパラメーターでは、データベース・マネージャーは、この名前を参照のターゲット・タイプの完全修飾ユーザー定義タイプ名に設定します。それ以外の場合、スキーマ名は SYSIBM となり、タイプ名は SYSCAT.DATATYPES カタログ・ビューの TYPENAME 列に含まれている名前になります。

注

- **SQLDA の準備:** DESCRIBE INPUT ステートメントを実行する前に、SQLDA を割り振り、SQLN の値をゼロ以上の値に設定して、SQLDA 中の SQLVAR のオカレンスの数を示す必要があります。SQLN のオカレンスを格納するために、十分なストレージを割り振る必要があります。準備済みステートメントの入力パラメーター・マーカの説明を取得するには、SQLVAR のオカレンス数が入力パラメーター・マーカの数を下回ってはなりません。さらに、入力パラメーター・マーカに LOB または構造化タイプが含まれている場合は、SQLVAR のオカレンス数が入力パラメーター・マーカの数の 2 倍になっている必要があります。
- 拡張 UNIX コード (EUC) コード・ページと DBCS コード・ページの間、または Unicode コード・ページと非 Unicode コード・ページの間でコード・ページ変換を行うと、結果の文字長が変化する場合があります。
- 構造化タイプが選択されているのに、FROM SQL トランスフォームが定義されていない (CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターで指定された TRANSFORM GROUP がない (SQLSTATE 428EM) か、あるいは指定されたグループに FROM SQL 変換機能が定義されていないため (SQLSTATE 42744)) 場合は、エラーが戻されます。
- **SQLDA の割り振り:** SQLDA を割り振るための可能な方法としては、以下の 3 とおりがあります。

方式 1: アプリケーションで処理する必要のある選択リストが入るだけの十分な数の SQLVAR のオカレンスを含む SQLDA を割り振ります。表に LOB、特殊タイプ、構造化タイプ、または参照タイプの列が含まれている場合には、SQLVAR の数を最大列数の 2 倍にしてください。それ以外の場合は、その数を最大列数と同じにします。割り振りを行ったなら、アプリケーションでこの SQLDA を繰り返し使用できるようになります。

このテクニックでは、大量のストレージを使用し、そのストレージのほとんどが特定の選択リストで使用されるわけではない場合でも決して割り振り解除されることがありません。

DESCRIBE INPUT

方式 2: 選択リストを処理するたびに、以下の 2 つのステップを繰り返し実行します。

1. SQLVAR のオカレンスのない SQLDA (SQLN を 0 にした SQLDA) を指定した DESCRIBE INPUT ステートメントを実行します。SQLD の戻り値は、結果表の列数となります。これは、必要な SQLVAR のオカレンス数か、またはその数の半分のいずれかになります。SQLVAR 項目がないので、SQLSTATE 01005 の警告が出されます。その警告の SQLCODE が +237、+238、または +239 のいずれかである場合、SQLVAR 項目の数は SQLD の戻り値の 2 倍でなければなりません。(上記の正の SQLCODE の戻り値は、SQLWARN BIND オプションが YES (正の SQLCODE を戻す) に設定されていることが前提となっています。SQLWARN が NO に設定されている場合でも +238 が戻されて、SQLVAR 項目の数が SQLD の戻り値の 2 倍でなければならないことを示します。)
2. SQLVAR のオカレンス数として十分大きい数を指定した SQLDA を割り振ります。この新しい SQLDA を使用して、DESCRIBE ステートメントをもう一度実行します。

この方式では、方式 1 よりもストレージを効率的に管理できます。しかし、DESCRIBE INPUT ステートメントの数は 2 倍になります。

方式 3: 選択リストのほとんど (そしておそらくは全部) を処理できるほどの大きさではあるが、適度に小さい SQLDA を割り振ります。DESCRIBE INPUT を実行して SQLD 値を調べます。必要なら、SQLVAR のオカレンス数として SQLD 値を使用することにより、もっと大きな SQLDA を割り振ります。

この方式は、最初の 2 つの方式の折衷方式です。その効果は、元の SQLDA サイズを適切に選択することに依存しています。

例

準備済みステートメントに組み込める最大数の入力パラメーターを記述できるように、十分な数の SQLVAR オカレンスを設定した SQLDA を指定して、DESCRIBE INPUT ステートメントを実行します。最大で 5 つのパラメーター・マーカを記述する必要があり、入力データに LOB が含まれていないと想定すると、次のようになります。

```
/* STMT1_STR contains INSERT statement with VALUES clause */
EXEC SQL PREPARE STMT1_NAME FROM :STMT1_STR;
... /* code to set SQLN to 5 and to allocate the SQLDA */
EXEC SQL DESCRIBE INPUT STMT1_NAME INTO :SQLDA;
.
.
.
```

この例では、『DESCRIBE OUTPUT』の『SQLDA の割り振り』にある方式 1 を使用しています。

DESCRIBE OUTPUT

DESCRIBE OUTPUT ステートメントは、準備されたステートメントについての情報を入手します。

呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

許可

必要ありません。

構文

```

▶▶ DESCRIBE OUTPUT statement-name INTO descriptor-name

```

説明

statement-name

準備済みステートメントを指定します。 DESCRIBE OUTPUT ステートメントを実行する時点で、この名前は現行のサーバーでアプリケーション・プロセスにより準備されたステートメントを指定していなければなりません。

準備済みステートメントが SELECT または VALUES INTO ステートメントである場合、戻される情報は、その結果表の中の列数を示します。準備済みステートメントが CALL ステートメントである場合、戻される情報は、プロシージャの OUT または INOUT として定義される出力パラメーターを示します。

INTO *descriptor-name*

SQL 記述子域 (SQLDA) を指定します。 DESCRIBE OUTPUT ステートメントを実行する前に、SQLDA 内の以下の変数を設定しておく必要があります。

SQLN SQLDA に用意する SQLVAR のオカレンス数を指定します。

DESCRIBE OUTPUT ステートメントを実行する前に、SQLN にゼロ以上の値を設定する必要があります。

DESCRIBE OUTPUT ステートメントを実行すると、データベース・マネージャは、以下のように SQLDA の変数に値を割り当てます。

SQLDAID

最初の 6 バイトは 'SQLDA' に設定されます (5 文字の英字の後、6 文字目はスペース文字です)。

SQLDOUBLED として定義される 7 番目のバイトは、示された結果列またはパラメーター・マーカに基づいて設定されます。

- すべての列または出力パラメーターに対して SQLDA に 2 つの SQLVAR 項目が含まれている場合、7 番目のバイトは '2' に設定されます。LOB、特殊タイプ、構造化タイプ、参照タイプの列、または出力パラメーターを可能にするために、このようになっています。
- それ以外の場合、7 番目のバイトはスペース文字に設定されます。

SQLDA の中にすべての結果列または出力パラメーター・マーカの記述が入るだけの十分なスペースがない場合、7 番目のバイトはスペース文字に設定されます。

8 番目のバイトは、スペース文字に設定されます。

DESCRIBE OUTPUT

SQLDABC

SQLDA の長さ (バイト単位)。

SQLD 準備済みステートメントが SELECT である場合は、SQLD は結果表の中の列数に設定されます。準備済みステートメントが CALL ステートメントである場合は、SQLD はプロシージャーの OUT および INOUT パラメーターの数に設定されます。それ以外の場合、SQLD は 0 に設定されます。

SQLVAR

SQLD の値が 0 の場合、または SQLD の値が SQLN の値より大きい場合は、SQLVAR のエレメントには値は割り当てられません。

SQLD の値が n (n は 0 より大きく、SQLN の値以下) の場合、SQLVAR の先頭の n オカレンスに対して、値は SQLTYPE、SQLLEN、SQLNAME、SQLLONGLEN、および SQLDATATYPE_NAME に割り当てられます。これらの値は、結果表の列、またはプロシージャーの出力パラメーターのパラメーター・マーカのいずれかを示します。SQLVAR の最初のオカレンスは最初の列または出力パラメーター・マーカを示し、SQLVAR の 2 番目のオカレンスは 2 番目の列または出力パラメーター・マーカを示し、というように続きます。

基本 SQLVAR

SQLTYPE

列またはパラメーターのデータ・タイプと、その列に NULL 値が入るかどうを示すコード。

SQLLEN

列またはパラメーターのデータ・タイプによって決まる長さを示す値。LOB データ・タイプの場合、SQLLEN は 0 になります。

SQLNAME

sqlname は、以下のように導き出されます。

- SQLVAR が、SELECT ステートメントの選択リスト内の単純な列参照用の派生列に対応する場合は、sqlname はその列の名前です。
- SQLVAR が、プロシージャーのパラメーター・リスト内の式の一部ではないパラメーター・マーカに対応する場合は、CREATE PROCEDURE でパラメーターが指定されていれば、sqlname にはそのパラメーターの名前が含まれます。
- CREATE PROCEDURE でパラメーターが指定されていなければ、sqlname には SQLDA 内の SQLVAR の位置を表す ASCII 数値リテラル値が含まれます。

2次 SQLVAR

これらの変数は、LOB、特殊タイプ、構造化タイプ、または参照タイプの列またはパラメーターを含めることができるよう、SQLVAR の項目の数が 2 倍にされた場合にのみ使用されます。

SQLLONGLEN

BLOB、CLOB、または DBCLOB の列またはパラメーターの長さ属性。

SQLDATATYPE_NAME

データベース・マネージャは、すべてのユーザー定義タイプ (特殊タイプまたは構造化タイプ) の列またはパラメーターで、この名前を完全修飾ユーザー定義タイプ名に設定します。また、参照タイプの列またはパラメーターでは、データベース・マネージャは、この名前を参照のターゲット・タイプの完全修飾ユーザー定義タイプ名に設定します。それ以外の場合、スキーマ名は `SYSIBM` となり、タイプ名は `SYSCAT.DATATYPES` カタログ・ビューの `TYPENAME` 列に含まれている名前になります。

注

- `DESCRIBE OUTPUT` ステートメントを実行する前に、`SQLN` の値を `SQLDA` の中の `SQLVAR` のオカレンスの数に設定し、`SQLN` 個のオカレンスが入るだけの十分なストレージを割り振っておく必要があります。例えば、準備済み `SELECT` ステートメントの結果表の列の記述を入手するには、`SQLVAR` のオカレンス数は列数以上でなければなりません。
- 大きいサイズの `LOB` が予想される場合、このラージ・オブジェクトの処理がアプリケーション・メモリーに与える影響について考慮してください。そのような状況では、ロケータまたはファイル参照変数を使用することを考えてください。`DESCRIBE OUTPUT` ステートメントを実行してからストレージを割り振るまでの間に、`SQLDA` を修正して、`SQLLEN` などの他のフィールドへ対応する変更を使用して、`SQL_TYP_xLOB` の `SQLTYPE` を `SQL_TYP_xLOB_LOCATOR` または `SQL_TYP_xLOB_FILE` に変更してください。その後、`SQLTYPE` に基づいてストレージを割り振ってから、処理を続けます。
- 拡張 `UNIX` コード (`EUC`) コード・ページと `DBCS` コード・ページとの間、または `Unicode` と非 `Unicode` コード・ページとの間でコード・ページ変換を行うと、結果の文字長が変化する場合があります。
- 構造化タイプが選択されているのに、`FROM SQL` トランスフォームが定義されていない (`CURRENT DEFAULT TRANSFORM GROUP` 特殊レジスターで指定された `TRANSFORM GROUP` がない (`SQLSTATE 428EM`) か、あるいは指定されたグループに `FROM SQL` 変換機能が定義されていないため (`SQLSTATE 42744`) 場合は、エラーが戻されます。
- **`SQLDA` の割り振り:** `SQLDA` を割り振るための可能な方法としては、以下の 3 とおりがあります。

方式 1: アプリケーションで処理する必要のある選択リストが入るだけの十分な数の `SQLVAR` のオカレンスを含む `SQLDA` を割り振ります。表に `LOB`、特殊タイプ、構造化タイプ、または参照タイプの列が含まれている場合には、`SQLVAR` の数を最大列数の 2 倍にしてください。それ以外の場合は、その数を最大列数と同じにします。割り振りを行ったなら、アプリケーションでこの `SQLDA` を繰り返し使用できるようになります。

このテクニックでは、大量のストレージを使用し、そのストレージのほとんどが特定の選択リストで使用されるわけではない場合でも決して割り振り解除されることがありません。

方式 2: 選択リストを処理するたびに、以下の 2 つのステップを繰り返し実行します。

DESCRIBE OUTPUT

1. SQLVAR のオカレンスのない SQLDA (SQLN を 0 にした SQLDA) を指定した DESCRIBE OUTPUT ステートメントを実行します。SQLD の戻り値は、結果表の列数となります。これは、必要な SQLVAR のオカレンス数か、またはその数の半分のいずれかになります。SQLVAR 項目がないので、SQLSTATE 01005 の警告が出されます。その警告の SQLCODE が +237、+238、または +239 のいずれかである場合、SQLVAR 項目の数は SQLD の戻り値の 2 倍でなければなりません。(上記の正の SQLCODE の戻り値は、SQLWARN BIND オプションが YES (正の SQLCODE を戻す) に設定されていることが前提となっています。SQLWARN が NO に設定されている場合でも +238 が戻されて、SQLVAR 項目の数が SQLD の戻り値の 2 倍でなければならないことを示します。)
2. SQLVAR のオカレンス数として十分大きい数を指定した SQLDA を割り振ります。この新しい SQLDA を使用して、DESCRIBE OUTPUT ステートメントをもう一度実行します。

この方式では、方式 1 よりもストレージを効率的に管理できます。しかし、DESCRIBE OUTPUT ステートメントの数は 2 倍になります。

方式 3: 選択リストのほとんど (そしておそらくは全部) を処理できるほどの大きさではあるが、適度に小さい SQLDA を割り振ります。DESCRIBE を実行して SQLD 値を調べます。必要なら、SQLVAR のオカレンス数として SQLD 値を使用することにより、もっと大きな SQLDA を割り振ります。

この方式は、最初の 2 つの方式の折衷方式です。その効果は、元の SQLDA サイズを適切に選択することに依存しています。

- **暗黙的な隠し列についての考慮事項:** DESCRIBE OUTPUT ステートメントで暗黙的な隠し列についての情報が戻されるのは、記述対象の照会最終的な結果表の SELECT リスト内にその列が明示的に指定されている場合のみです。暗黙的な隠し列が照会の結果表に含まれない場合、DESCRIBE OUTPUT ステートメントによって戻されるその照会に関する情報には、暗黙的な隠し列の情報が含まれません。

例

C プログラムの中で、SQLVAR オカレンスのない SQLDA を指定して DESCRIBE OUTPUT ステートメントを実行します。SQLD が 0 より大きい場合、その値を使って必要な数の SQLVAR のオカレンスを含む SQLDA を割り振り、その SQLDA を使って DESCRIBE ステートメントを実行します。

```
EXEC SQL BEGIN DECLARE SECTION;
       char stmt1_str[200];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLDA;
EXEC SQL DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

... /* code to prompt user for a query, then to generate */
     /* a select-statement in the stmt1_str          */
EXEC SQL PREPARE STMT1_NAME FROM :stmt1_str;

... /* code to set SQLN to zero and to allocate the SQLDA */
EXEC SQL DESCRIBE STMT1_NAME INTO :sqlda;

... /* code to check that SQLD is greater than zero, to set */
     /* SQLN to SQLD, then to re-allocate the SQLDA          */
```

```

EXEC SQL DESCRIBE STMT1_NAME INTO :sqllda;

... /* code to prepare for the use of the SQLDA          */
    /* and allocate buffers to receive the data          */
EXEC SQL OPEN DYN_CURSOR;

... /* loop to fetch rows from result table              */
EXEC SQL FETCH DYN_CURSOR USING DESCRIPTOR :sqllda;
.
.
.

```

DISCONNECT

DISCONNECT ステートメントは、アクティブな作業単位がない場合に (つまり、コミットまたはロールバックの操作の後)、1 つまたは複数の接続を破棄します。

DISCONNECT ステートメントの対象が単一の接続の場合、その接続は、アクティブな作業単位があるかどうかにかかわらず、そのデータベースが既存の作業単位に関係していない場合にのみ破棄されます。たとえば、他のいくつかのデータベースの作業が終了し、ステートメントの対象については終了していない場合、接続を破棄せずに切断される場合があります。

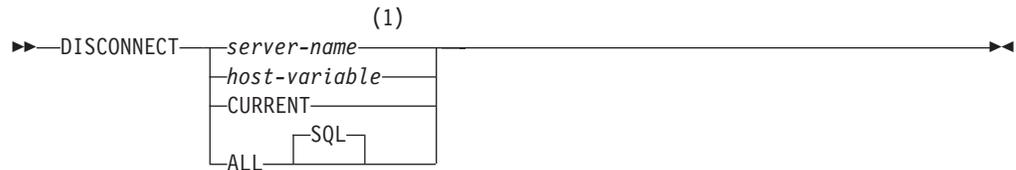
呼び出し

対話式 SQL 機能には外見上対話式的の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

許可

必要ありません。

構文



注:

- 1 CURRENT または ALL という名前のアプリケーション・サーバーは、ホスト変数によってのみ指定することができます。

説明

server-name または *host-variable*

server-name (サーバー名) またはその *server-name* を含む *host-variable* (ホスト変数) によって、アプリケーション・サーバーを指定します。

DISCONNECT

host-variable (ホスト変数) を指定する場合、それは、長さ属性が 8 以下の文字ストリング変数でなければならず、標識変数を含めることはできません。その *host-variable* に入る *server-name* は、左寄せする必要があり、引用符で区切ることはできません。

server-name は、アプリケーション・サーバーを指定するデータベース別名である点に注意してください。この名前は、アプリケーション・リクエスターのローカル・ディレクトリーにリストされている必要があります。

指定されたデータベース別名、またはホスト変数に含まれているデータベース別名は、そのアプリケーション・プロセスの既存の接続を指定するものでなければなりません。データベース別名が既存の接続を指定していない場合、エラー (SQLSTATE 08003) になります。

CURRENT

アプリケーション・プロセスの現行接続を指定します。アプリケーション・プロセスは、接続された状態でなければなりません。接続されていない場合、エラー (SQLSTATE 08003) になります。

ALL

アプリケーション・プロセスの既存の接続を全部破棄することを指定します。ステートメント実行時に接続が存在していない場合でも、エラーまたは警告のメッセージは出されません。任意に選択できるキーワードである SQL は RELEASE ステートメントの構文との一貫性を持たせるために含まれています。

規則

- 一般に、DISCONNECT ステートメントは作業単位の中では実行できません。実行すると、エラー (SQLSTATE 25000) になります。この規則の例外は、単一の接続を切断することを指定し、データベースが既存の作業単位に加わっていない場合です。この場合、DISCONNECT ステートメントが発行される時にアクティブな作業単位があるかどうかは問題になりません。
- DISCONNECT ステートメントは、トランザクション処理 (TP) モニター環境の中では全く実行できません (SQLSTATE 25000)。DB2 トランザクション・マネージャを使用する場合にのみ有効です。

注

- DISCONNECT ステートメントが正常に処理されると、指定されたそれぞれの接続は破棄されます。

DISCONNECT ステートメントが正常に処理されない場合、アプリケーション・プロセスの接続状態とその接続の状態は変更されません。

- DISCONNECT を使って現行接続を破棄する場合、その次に実行する SQL ステートメントは、CONNECT または SET CONNECTION でなければなりません。
- タイプ 1 CONNECT セマンティクスでは、DISCONNECT を使用できないわけではありません。ただし、DISCONNECT CURRENT と DISCONNECT ALL は、使用することはできますが、CONNECT RESET ステートメントの場合と違って、コミット操作は行われません。

タイプ 1 CONNECT では一度に 1 つの接続しかサポートされないため、DISCONNECT ステートメントに *server-name* または *host-variable* を指定する場

合、それは現行接続を指定するものでなければなりません。一般に、『規則』に示されている例外を除き、DISCONNECT は作業単位内で実行すると失敗します。

- リモート接続を作成し保守するには、さまざまなリソースが必要になります。したがって、再使用の予定がないリモート接続は、できるだけ破棄する必要があります。
- 接続は、接続オプションの効果のためにコミット操作中に破棄されることもあります。そのような接続オプションには、AUTOMATIC、CONDITIONAL、またはEXPLICIT があります。それらは、プリコンパイラー・オプションとして設定されたり、実行時に SET CLIENT API によって設定されます。DISCONNECT オプションの指定については、『分散リレーショナル・データベース』を参照してください。

例

例 1: IBMSTHDB への SQL 接続は、アプリケーションではもはや必要でなくなりました。コミットかロールバックの操作を行った後、次のステートメントを実行してその接続を破棄します。

```
EXEC SQL DISCONNECT IBMSTHDB;
```

例 2: 現行の接続は、アプリケーションでもはや必要でなくなりました。コミットかロールバックの操作を行った後、次のステートメントを実行してその接続を破棄します。

```
EXEC SQL DISCONNECT CURRENT;
```

例 3: 既存の接続は、アプリケーションでもはや必要でなくなりました。コミットかロールバックの操作を行った後、次のステートメントを実行して接続をすべて破棄します。

```
EXEC SQL DISCONNECT ALL;
```

DROP

DROP ステートメントは、オブジェクトを削除します。そのオブジェクトに直接または間接的に従属するオブジェクトがある場合、それらも削除されるか、または作動不能になります。オブジェクトを削除すると、その記述がカタログから削除され、そのオブジェクトを参照するパッケージがあれば無効になります。

呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

2 部構成の名前を許可するオブジェクトをドロップする場合には、ステートメントの許可 ID に、以下の特権が少なくとも 1 つ含まれている必要があります。

- そのオブジェクトのスキーマに対する DROPIN 特権

DROP

- オブジェクトのカタログ・ビューの OWNER 列に記録されている、オブジェクトの所有者
- オブジェクトに対する CONTROL 特権 (索引、SPECIFICATION ONLY 指定の索引、ニックネーム、パッケージ、表、およびビューにのみあてはまる)
- SYSCAT.DATATYPES カタログ・ビューの OWNER 列に記録されている、ユーザー定義タイプの所有者 (ユーザー定義タイプに関連したメソッドをドロップしている場合にのみあてはまる)
- SYSADM または DBADM 権限

表またはビューの階層をドロップする場合、ステートメントの許可 ID が持つ特権には、その階層内にあるそれぞれの表またはビューについて、上記の特権のいずれかが含まれている必要があります。

スキーマをドロップする場合、ステートメントの許可 ID は、SYSADM 権限または DBADM 権限を保持しているか、または SYSCAT.SCHEMATA カタログ・ビューの OWNER 列に記録されているスキーマ所有者でなければなりません。

バッファプール、データベース・パーティション・グループ、ヒストグラム・テンプレート、または表スペースをドロップする場合、ステートメントの許可 ID は SYSADM 権限または SYSCTRL 権限を保持している必要があります。

データ・タイプ・マッピング、イベント・モニター、関数マッピング、サーバー定義、サービス・クラス、作業アクション・セット、作業クラス・セット、ワークロード、しきい値、ヒストグラム、またはラッパーをドロップする場合、ステートメントの許可 ID は SYSADM 権限または DBADM 権限を保持している必要があります。

ユーザー・マッピングをドロップする場合、この許可 ID がマッピング内にあるフェデレーテッド・データベースの許可名と異なる場合には、ステートメントの許可 ID が SYSADM または DBADM 権限を保持している必要があります。許可 ID と許可名が一致する場合には、必要とされる特権または権限はありません。

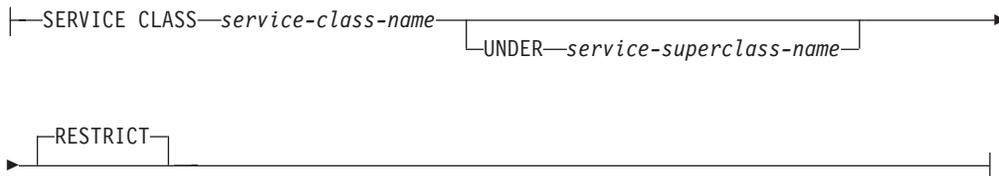
トランスフォームをドロップする場合、ステートメントの許可 ID は SYSADM 権限か DBADM 権限を保持しているか、または *type-name* の所有者でなければなりません。

監査ポリシー、ロール、セキュリティー・ラベル、セキュリティー・ラベル・コンポーネント、セキュリティー・ポリシー、またはトラステッド・コンテキストをドロップする場合、ステートメントの許可 ID は SECADM 権限を保持する必要があります。

構文

▶—DROP—▶

ALIAS— <i>alias-name</i>
AUDIT POLICY— <i>policy-name</i>
BUFFERPOOL— <i>bufferpool-name</i>
EVENT MONITOR— <i>event-monitor-name</i>
FUNCTION— <i>function-name</i> (((<i>data-type</i>)))) RESTRIC
SPECIFIC FUNCTION— <i>specific-name</i> RESTRIC
FUNCTION MAPPING— <i>function-mapping-name</i>
HISTOGRAM TEMPLATE— <i>template-name</i> (1)
INDEX— <i>index-name</i>
INDEX EXTENSION— <i>index-extension-name</i> RESTRIC
METHOD— <i>method-name</i> (((<i>datatype</i>))) FOR— <i>type-name</i> RESTRIC
SPECIFIC METHOD— <i>specific-name</i> RESTRIC
NICKNAME— <i>nickname</i>
DATABASE PARTITION GROUP— <i>db-partition-group-name</i>
PACKAGE— <i>schema-name.</i> <i>package-id</i> VERSION— <i>version-id</i> RESTRIC
PROCEDURE— <i>procedure-name</i> (((<i>data-type</i>))) RESTRIC
ROLE— <i>role-name</i>
SPECIFIC PROCEDURE— <i>specific-name</i> RESTRIC
SCHEMA— <i>schema-name</i> RESTRIC
SECURITY LABEL— <i>security-label-name</i> RESTRIC
SECURITY LABEL COMPONENT— <i>sec-label-comp-name</i> RESTRIC
SECURITY POLICY— <i>security-policy-name</i> RESTRIC
SEQUENCE— <i>sequence-name</i> RESTRIC
SERVER— <i>server-name</i>
TABLE— <i>table-name</i>
TABLE HIERARCHY— <i>root-table-name</i>
TABLESPACE— <i>tablespace-name</i>
TABLESPACES
TRANSFORM—ALL— <i>group-name</i> FOR— <i>type-name</i>
TRANSFORMS
THRESHOLD— <i>threshold-name</i>
TRIGGER— <i>trigger-name</i>
TRUSTED CONTEXT— <i>context-name</i>
TYPE— <i>type-name</i> RESTRIC
TYPE MAPPING— <i>type-mapping-name</i>
USER MAPPING FOR— <i>authorization-name</i> SERVER— <i>server-name</i> USER RESTRIC
VARIABLE— <i>variable-name</i> RESTRIC
VIEW— <i>view-name</i>
VIEW HIERARCHY— <i>root-view-name</i>
WORK ACTION SET— <i>work-action-set-name</i> RESTRIC
WORK CLASS SET— <i>work-class-set-name</i> RESTRIC
WORKLOAD— <i>workload-name</i>
WRAPPER— <i>wrapper-name</i>
XROBJECT— <i>xsubject-name</i>

service-class:**注:**

- 1 *Index-name* には、索引、あるいは SPECIFICATION ONLY 指定の索引のどちらかの名前を指定できます。

説明**ALIAS** *alias-name*

ドロップする別名を指定します。 *alias-name* (別名) は、カタログに記述されている別名を指定する名前であればなりません (SQLSTATE 42704)。指定した別名は削除されます。

別名を参照するすべての表、ビュー、およびトリガーは作動不能になります。(これには、CREATE TRIGGER ステートメントの ON 節で参照されている表と、トリガー SQL ステートメントで参照されているすべての表が含まれます。)

AUDIT POLICY *policy-name*

ドロップする監査ポリシーを指定します。 *policy-name* は、現行のサーバーに存在する監査ポリシーを識別するものでなければなりません (SQLSTATE 42704)。監査ポリシーは、何らかのデータベース・オブジェクトと関連付けられていてはなりません (SQLSTATE 42893)。指定された監査ポリシーはカタログから削除されます。

BUFFERPOOL *bufferpool-name*

ドロップするバッファーク・プールを指定します。 *bufferpool-name* (バッファーク・プール名) は、カタログに記述されているバッファーク・プールを指定していなければなりません (SQLSTATE 42704)。そのバッファーク・プールに表スペースが割り当てられていない場合もあります (SQLSTATE 42893)。IBMDEFAULTBP バッファーク・プールはドロップできません (SQLSTATE 42832)。バッファーク・プール・メモリーはすぐに解放され、DB2 で使用されます。ディスク装置は、次にデータベースへ接続するときまで解放できません。

EVENT MONITOR *event-monitor-name*

ドロップするイベント・モニターを指定します。 *event-monitor-name* (イベント・モニター名) は、すでにカタログに存在するイベント・モニターを指定していなければなりません (SQLSTATE 42704)。

指定したイベント・モニターがアクティブの場合は、エラーが戻されます (SQLSTATE 55034)。それ以外の場合は、イベント・モニターは削除されます。ただし、SET EVENT MONITOR STATE ステートメントを使用してイベント・モニターを活動化した状態で、データベースをいったん非活動化してから再活動

化した場合は、DROP ステートメントを実行する前に、SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを非活動化してください。

ドロップする WRITE TO FILE イベント・モニターのターゲット・パスにイベント・ファイルが存在する場合、そのイベント・ファイルは削除されません。ただし、それと同じターゲット・パスを指定した新しいイベント・モニターが作成されると、それらのイベント・ファイルは削除されます。

WRITE TO TABLE イベント・モニターをドロップする場合、表情報は SYSCAT.EVENTTABLES カタログ・ビューからドロップされますが、表そのものはドロップされません。

FUNCTION

ドロップするユーザー定義関数 (完全な関数または関数テンプレートのいずれか) のインスタンスを指定します。指定する関数インスタンスは、カタログに記述されたユーザー定義関数でなければなりません。CREATE TYPE (特殊) ステートメントによって暗黙に生成された関数はドロップできません。

関数のインスタンスを指定する方法としては、次のようにいくつかの方法があります。

FUNCTION *function-name*

特定の関数を指定します。 *function-name* (関数名) の関数インスタンスが 1 つだけ存在している場合にのみ有効です。このように指定する関数には、任意の数のパラメーターが定義されていても構いません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこの名前関数が存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定したスキーマまたは暗黙のスキーマに、この関数の特定インスタンスが複数存在する場合、エラーが戻されます (SQLSTATE 42725)。

FUNCTION *function-name (data-type,...)*

ドロップする関数を固有に指定する関数シグニチャーを指定します。関数選択のアルゴリズムは使用されません。

function-name

ドロップする関数の関数名を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(data-type,...)

これは、CREATE FUNCTION ステートメント上で (対応する位置に) 指定されたデータ・タイプに一致していなければなりません。データ・タイプの数、およびデータ・タイプを論理的に連結した値が、ドロップする特定の関数インスタンスを識別するのに使用されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

長さ、精度、または位取りをコーディングする場合、その値は、CREATE FUNCTION ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(n) のタイプは、n に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

RESTRICT

RESTRICT キーワードを指定すると、以下のいずれかの従属関係が存在する場合は、関数をドロップしないという規則が適用されます。

- 別のルーチンがその関数に基づいている。
- ビューがその関数を使用している。
- トリガーがその関数を使用している。
- マテリアライズ照会表が、その定義で関数を使用することはできません。

RESTRICT は、デフォルトの動作です。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つ関数がない場合は、エラーが戻されます (SQLSTATE 42883)。

SPECIFIC FUNCTION *specific-name*

関数の作成時に指定された特定関数名、またはデフォルト値として使用された特定関数名を使用して、ドロップする特定のユーザー定義関数を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。*specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定関数のインスタンスを指定していなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42704)。

RESTRICT

RESTRICT キーワードを指定すると、以下のいずれかの従属関係が存在する場合は、関数をドロップしないという規則が適用されます。

- 別のルーチンがその関数に基づいている。
- ビューがその関数を使用している。
- トリガーがその関数を使用している。

RESTRICT は、デフォルトの動作です。

SYSIBM、SYSFUN、または SYSPROC スキーマ (SQLSTATE 42832) の関数はドロップできません。

他のオブジェクトが関数に從属している場合があります。そのような関数をドロップする場合には、その前にそのような從属オブジェクトをすべてドロップしておく必要があります (作動不能としてマークされるパッケージは除く)。從属オブジェクトを伴う関数をドロップしようとする、エラー (SQLSTATE 42893) になります。それらの從属関係のリストについては、727 ページの『規則』 ページを参照してください。

関数がドロップ可能な場合、その関数がドロップされます。

ドロップする特定関数に從属するパッケージがある場合には、それは作動不能としてマークされます。そのようなパッケージが暗黙のうちに再バインドされることはありません。BIND コマンドまたは REBIND コマンドを使って再バインドするか、PREP コマンドを使って再作成する必要があります。

FUNCTION MAPPING *function-mapping-name*

ドロップする関数マッピングを指定します。 *function-mapping-name* は、カタログに記述されているユーザー定義関数マッピングを指定していなければなりません (SQLSTATE 42704)。その関数マッピングはデータベースから削除されます。

デフォルトの関数マッピングは、ドロップできませんが、CREATE FUNCTION MAPPING ステートメントを使用することによって、使用不可にすることができます。デフォルトの関数マッピングをオーバーライドするために作成されたユーザー定義の関数マッピングをドロップすると、デフォルトの関数マッピングが復元されます。

ドロップされる関数マッピングに從属しているパッケージは、無効になります。

HISTOGRAM TEMPLATE *template-name*

ドロップ対象のヒストグラム・テンプレートを指定します。 *template-name* は、現行のサーバーに存在するヒストグラム・テンプレートを識別するものでなければなりません (SQLSTATE 42704)。 *template-name* に SYSDEFAULTHISTOGRAM を指定することはできません (SQLSTATE 42832)。サービス・クラスまたは作業アクションがヒストグラム・テンプレートに依存している場合、そのヒストグラム・テンプレートをドロップすることはできません (SQLSTATE 42893) 指定されたヒストグラム・テンプレートはカタログから削除されます。

INDEX *index-name*

ドロップする索引または SPECIFICATION ONLY 指定の索引を指定します。 *index-name* (索引名) は、カタログに記述されている索引または SPECIFICATION ONLY 指定の索引を識別していなければなりません (SQLSTATE 42704)。索引は、システムに必須の主キーまたはユニーク制約の索引であってはならず、複製されたマテリアライズ照会表または XML 列の索引であってもなりません (SQLSTATE 42917)。指定した索引または SPECIFICATION ONLY 指定の索引は削除されます。

ドロップされる索引または SPECIFICATION ONLY 指定の索引に從属しているパッケージは、無効になります。

INDEX EXTENSION *index-extension-name* **RESTRICT**

ドロップする索引拡張を指定します。 *index-extension-name* (索引拡張名) は、カタログに記述されている索引拡張を指定する名前であればなりません (SQLSTATE 42704)。 **RESTRICT** キーワードは、この索引拡張の定義に従って索引を定義できないという規則を課します (SQLSTATE 42893)。

METHOD

ドロップするメソッド本体を指定します。指定するメソッドの本体は、カタログに記述されているメソッドでなければなりません (SQLSTATE 42704)。

CREATE TYPE ステートメントによって暗黙的に生成されたメソッド本体をドロップすることはできません。

DROP METHOD によって、メソッドの本体は削除されますが、メソッドの指定 (シグニチャー) はサブジェクト・タイプの定義の一部として残されます。メソッドの本体をドロップした後、メソッドの指定は **ALTER TYPE DROP METHOD** を使用してサブジェクト・タイプの定義から削除することができます。

ドロップするメソッド本体は、以下のようないくつかの方法で指定することができます。

METHOD *method-name*

ドロップする特定のメソッドを指定します。この方法は、対象となるタイプ *type-name* に、*method-name* という名前のメソッド・インスタスが 1 つしかないことが明らかな場合にのみ有効です。この方法を用いる場合は、メソッドにいくつのパラメーターが定義されていても構いません。タイプ *type-name* に、指定された名前のメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定されたデータ・タイプに、そのメソッドの特定のインスタスが複数存在する場合も、エラーが戻されます (SQLSTATE 42725)。

METHOD *method-name (data-type,...)*

ドロップするメソッドを一意的に識別できるメソッド・シグニチャーを指定します。メソッド選択のアルゴリズムは使用されません。

method-name

指定したタイプの中から、ドロップするメソッドのメソッド名を指定します。指定する名前は、修飾なしの ID でなければなりません。

(data-type, ...)

データ・タイプを指定します。ここで指定されるデータ・タイプは、**CREATE TYPE** または **ALTER TYPE** ステートメントで、メソッドの指定の対応する位置に指定されたデータ・タイプと一致していなければなりません。データ・タイプの数とデータ・タイプを論理的に連結した値から、ドロップする特定のメソッドが識別されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE TYPE ステートメントで指定された値と完全に一致していなければなりません。

0<n<25 は REAL を意味し、24<n<54 は DOUBLE を意味するので、FLOAT(n) のタイプは、n に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定されたデータ・タイプに、指定されたシグニチャーを持つメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42883)。

FOR *type-name*

指定したメソッドのドロップを行うタイプを指定します。ここで指定される名前は、カタログにすでに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないタイプ名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル / BIND オプションによって、修飾子のないタイプ名に修飾子が暗黙指定されます。

RESTRICT

RESTRICT キーワードを指定すると、以下のいずれかの従属関係が存在する場合は、メソッドをドロップしないという規則が適用されます。

- 別のルーチンがそのメソッドに基づいている。
- ビューがそのメソッドを使用している。
- トリガーがそのメソッドを使用している。

RESTRICT は、デフォルトの動作です。

SPECIFIC METHOD *specific-name*

CREATE TYPE または ALTER TYPE ステートメントにおいてユーザーが指定した名前、もしくはデフォルトで指定された名前を使用して、ドロップする特定のメソッドを識別します。特定名 (*specific-name*) に修飾子が付いていない場合、動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のない特定名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル / BIND オプションにより、修飾子のない特定名に修飾子が暗黙指定されます。specific-name に指定される名前は、メソッドの名前でなければなりません。そうでない場合は、エラーが戻されます (SQLSTATE 42704)。

RESTRICT

RESTRICT キーワードを指定すると、以下のいずれかの従属関係が存在する場合は、メソッドをドロップしないという規則が適用されます。

- 別のルーチンがそのメソッドに基づいている。
- ビューがそのメソッドを使用している。
- トリガーがその関数を使用している。

RESTRICT は、デフォルトのメソッドです。

メソッドに他のオブジェクトが従属している場合があります。そのような場合は、メソッドをドロップする前にそれらの従属関係をすべて除去する必要があります (ただし、そのメソッドがドロップされると作動不能としてマークされるパッケージは例外です)。そのような従属関係を持つメソッドをドロップしようとすると、エラーが戻されます (SQLSTATE 42893)。

ドロップできる状態にあれば、メソッドはドロップされます。

ドロップする特定のメソッドに従属しているパッケージは、そのメソッドがドロップされると、作動不能としてマークされます。そのようなパッケージが暗黙的に再バインドされることはありません。これらのパッケージは、`BIND` コマンドまたは `REBIND` コマンドを使用して再バインドするか、あるいは `PREP` コマンドを使用して再作成する必要があります。

ドロップされる特定のメソッドが別のメソッドをオーバーライドする場合、オーバーライドされるメソッド (および、ドロップされる特定のメソッドのスーパータイプでこのメソッドをオーバーライドするメソッド) に従属したパッケージはすべて無効になります。

NICKNAME *nickname*

ドロップするニックネームを指定します。ニックネームは、カタログにリストされていなければなりません (SQLSTATE 42704)。そのニックネームはデータベースから削除されます。

ニックネームに関連した列および索引に関するすべての情報が、カタログから削除されます。ニックネームに従属したマテリアライズ照会表はドロップされません。ニックネームに従属した `SPECIFICATION ONLY` 指定の索引はドロップされます。ニックネームに従属するビューは、作動不能としてマークされます。ドロップされた `SPECIFICATION ONLY` 指定の索引または作動不能ビューに従属するパッケージはいずれも無効になります。ニックネームが参照するデータ・ソース表は影響を受けません。

SQL 関数またはメソッドがニックネームに依存している場合、そのニックネームはドロップできません (SQLSTATE 42893)。

DATABASE PARTITION GROUP *db-partition-group-name*

ドロップするデータベース・パーティション・グループを指定します。

db-partition-group-name パラメーターは、カタログに記述されているデータベース・パーティション・グループを指定していなければなりません (SQLSTATE 42704)。これは、1 部構成の名前です。

データベース・パーティション・グループをドロップすると、データベース・パーティション・グループで定義されたすべての表スペースがドロップされます。そのような表スペース内の表に対して従属関係がある既存のデータベース・オブジェクト (パッケージや参照制約など) は、ドロップされるか、または無効になり (該当する場合)、従属するビューとトリガーは作動不能になります。

システム定義のデータベース・パーティション・グループはドロップできません (SQLSTATE 42832)。

現在データ再配分が行われているデータベース・パーティション・グループに対して `DROP DATABASE PARTITION GROUP` ステートメントを発行すると、データベース・パーティション・グループのドロップ操作は失敗し、エラーが戻されます (SQLSTATE 55038)。ただし、部分的に再配分されたデータベース・

パーティション・グループはドロップできます。データベース・パーティション・グループは、`REDISTRIBUTE DATABASE PARTITION GROUP` コマンドが完了するまで実行されなかった場合は、部分的に再配分の状態になります。これは、エラーまたは `FORCE APPLICATION ALL` コマンドによって割り込まれた場合に起こる可能性があります。(部分的に再配分されたデータベース・パーティション・グループの場合、`SYSCAT.DBPARTITIONGROUPS` カタログの `REBALANCE_PMAP_ID` は -1 ではありません。)

PACKAGE *schema-name.package-id*

ドロップするパッケージを指定します。スキーマ名が指定されていない場合、パッケージ ID は暗黙的にデフォルト・スキーマで修飾されます。スキーマ名およびパッケージ ID は、明示的または暗黙的に指定されたバージョン ID とともに、カタログに記述されているパッケージを指定していなければなりません (SQLSTATE 42704)。指定したパッケージが削除されます。ドロップするパッケージが、*schema-name.package-id* で指定された唯一のパッケージである場合 (つまり、他のバージョンは存在しない場合)、そのパッケージに対する特権もすべて削除されます。

VERSION *version-id*

ドロップするパッケージ・バージョンを指定します。値が指定されない場合には、空ストリングがバージョンのデフォルトになります。同じパッケージ名が付けられていてもバージョンは異なる、複数のパッケージが存在する場合、`DROP` ステートメントを 1 回呼び出すときに、1 つのパッケージ・バージョンだけをドロップできます。次のような場合は、バージョン ID を二重引用符で区切ってください。

- バージョン ID が `VERSION(AUTO)` プリコンパイラー・オプションによって生成された場合
- バージョン ID が数字で始まる場合
- バージョン ID が小文字であったり、大/小文字混合である場合

ステートメントをオペレーティング・システムのコマンド・プロンプトから呼び出す場合は、各二重引用符の区切り文字の前に円記号を置いて、オペレーティング・システムによって区切り文字が外されないようにします。

PROCEDURE

ドロップするプロシージャのインスタンスを指定します。指定するプロシージャ・インスタンスは、カタログに記述されたプロシージャでなければなりません。

プロシージャ・インスタンスを指定する方法としては、次のようにいくつかの方法があります。

PROCEDURE *procedure-name*

特定のプロシージャを指定します。この方法は、*procedure-name* で指定したプロシージャ・インスタンスがスキーマ内に 1 つしか存在しないことが明らかな場合にのみ有効です。この方法で指定するプロシージャには、パラメーターがいくつ定義されていても構いません。指定したスキーマまたは暗黙のスキーマにこの名前前のプロシージャが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。動的 SQL ステートメントでは、`CURRENT SCHEMA` 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、`QUALIFIER` プ

リコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこのプロシージャの特定インスタンスが複数存在する場合は、エラーが戻されず (SQLSTATE 42725)。

PROCEDURE *procedure-name* (*data-type*,...)

ドロップするプロシージャを一意に識別するプロシージャ・シグニチャーを指定します。プロシージャ選択のアルゴリズムは使用されません。フェデレーテッド・プロシージャの場合は、CREATE PROCEDURE ステートメントにシグニチャー情報を指定しません。その情報は、システム・カタログから入手できます。

procedure-name

ドロップするプロシージャのプロシージャ名を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

(*data-type*,...)

データ・タイプを指定します。ここで指定されるデータ・タイプは、CREATE PROCEDURE ステートメントの対応する位置に指定されたデータ・タイプと一致していなければなりません。ただし、フェデレーテッド・プロシージャの場合は例外です。その場合、データ・タイプは、対応するパラメーターに関してローカル・カタログに格納されている指定内容と一致しなければなりません。データ・タイプの数、およびデータ・タイプを論理的に連結した値を使用して、ドロップする特定のプロシージャのインスタンスが識別されます。

data-type が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE PROCEDURE ステートメントにおける指定に完全に一致していなければなりません。フェデレーテッド・プロシージャの場合は、対応するパラメーターに関してローカル・カタログに格納されている指定内容と完全に一致しなければなりません。

$0 < n < 25$ は REAL を意味し、 $24 < n < 54$ は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

RESTRICT

RESTRICT キーワードを指定すると、トリガー定義、SQL 関数、または SQL メソッドにプロシージャの名前が付けられた CALL ステートメントが含まれる場合に、そのプロシージャはドロップされずに済みます。

RESTRICT は、デフォルトの動作です。

指定したスキーマまたは暗黙のスキーマに、指定されたシグニチャーを持つプロシージャがない場合は、エラーが戻されます (SQLSTATE 42883)。

ROLE *role-name*

ドロップするロールを指定します。 *role-name* は、現行のサーバーにすでに存在するロールを識別するものでなければなりません (SQLSTATE 42704)。あるロールがルーチンに対する EXECUTE 特権またはシーケンスに対する USAGE 特権を持っていて、パッケージ以外の SQL オブジェクトがそのルーチンまたはシーケンスに依存している場合には、*role-name* はそのロール、または *role-name* を含むそのようなロールを識別するものであってはなりません (SQLSTATE 42893)。SQL オブジェクトの所有者は、*authorization-name* または *authorization-name* のメンバーである任意のユーザーのいずれかです (*authorization-name* はロール)。

ドロップ対象のロールが以下のいずれかに該当する場合、DROP ROLE ステートメントは失敗します (SQLSTATE 42893)。

- 接続属性 SESSION_USER ROLE のいずれか 1 つの値が *role-name* であるようなワークロードが存在する
- *role-name* を使用するトラステッド・コンテキストが存在する

指定されたロールはカタログから削除されます。

SPECIFIC PROCEDURE *specific-name*

プロシージャの作成時にユーザーが指定した特定のプロシージャ名か、デフォルト値として与えられたプロシージャ名を使用して、ドロップする特定のプロシージャを識別します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスタが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。*specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定プロシージャのインスタンスを指定していなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42704)。

RESTRICT

RESTRICT キーワードを指定すると、トリガー定義、SQL 関数、または SQL メソッドにプロシージャの名前が付けられた CALL ステートメントが含まれる場合に、そのプロシージャはドロップされずに済みます。

RESTRICT は、デフォルトの動作です。

SYSIBM、SYSFUN、または SYSPROC スキーマのプロシージャはドロップできません (SQLSTATE 42832)。

SCHEMA *schema-name* **RESTRICT**

ドロップする特定のスキーマを指定します。 *schema-name* に指定するスキーマ名は、カタログに記述されているスキーマを識別するものでなければなりません

(SQLSTATE 42704)。RESTRICT キーワードは、データベースから削除するスキーマとして指定したスキーマにオブジェクトを定義できないという規則を課します (SQLSTATE 42893)。

SECURITY LABEL *security-label-name*

ドロップするセキュリティ・ラベルを指定します。名前は、セキュリティ・ポリシーで修飾する必要があり (SQLSTATE 42704)、現在のサーバー上に存在するセキュリティ・ラベルを識別していなければなりません (SQLSTATE 42704)。

RESTRICT

このオプションはデフォルトであり、以下のいずれかの従属関係が存在する場合に、セキュリティ・ラベルがドロップされないようにします (SQLSTATE 42893)。

- 1 つ以上の許可 ID が読み取りアクセス用のセキュリティ・ラベルを現在保持している場合
- 1 つ以上の許可 ID が書き込みアクセス用のセキュリティ・ラベルを現在保持している場合
- 1 つ以上の列を保護するためにセキュリティ・ラベルが現在使用中になっている場合

SECURITY LABEL COMPONENT *sec-label-comp-name*

ドロップするセキュリティ・ラベル・コンポーネントを指定します。*sec-label-comp-name* は、カタログに記述されているセキュリティ・ラベル・コンポーネントを識別するものでなければなりません (SQLSTATE 42704)。

RESTRICT

このオプションはデフォルトであり、以下のいずれかの従属関係が存在する場合に、セキュリティ・ラベル・コンポーネントがドロップされないようにします (SQLSTATE 42893)。

- セキュリティ・ラベル・コンポーネントを含んだセキュリティ・ポリシーが現在 1 つ以上定義されている場合

SECURITY POLICY *security-policy-name*

ドロップするセキュリティ・ポリシーを指定します。*security-policy-name* は、現在のサーバー上に存在するセキュリティ・ポリシーを識別するものでなければなりません (SQLSTATE 42704)。

RESTRICT

このオプションはデフォルトであり、以下のいずれかの従属関係が存在する場合に、セキュリティ・ポリシーがドロップされないようにします (SQLSTATE 42893)。

- 1 つ以上の表がこのセキュリティ・ポリシーに関連付けられている場合
- 1 つ以上の許可 ID が、このセキュリティ・ポリシー内のいずれかの規則に関する免除を保持している場合
- このセキュリティ・ポリシーに関して 1 つ以上のセキュリティ・ラベルが定義されている場合

SEQUENCE *sequence-name*

ドロップする特定のシーケンスを識別します。暗黙的または明示的スキーマ名を含む *sequence-name* は、現在のサーバーに存在するシーケンスを固有に識別し

ていなければなりません。この名前によるシーケンスが、明示的または暗黙的に指定されたスキーマに存在しない場合、エラーが戻されます (SQLSTATE 42704)。

RESTRICT

このオプションはデフォルトであり、以下のいずれかの従属関係が存在する場合に、シーケンスがドロップされないようにします。

- トリガーの NEXT VALUE または PREVIOUS VALUE 式がこのシーケンスを指定するようなトリガーが存在する (SQLSTATE 42893)。
- ルーチン本体の NEXT VALUE 式がこのシーケンスを指定するような SQL 関数または SQL メソッドが存在する (SQLSTATE 42893)。

SERVER *server-name*

カタログから定義をドロップするデータ・ソースを指定します。 *server-name* に指定するサーバー名は、カタログに記述されているデータ・ソースを識別するものでなければなりません (SQLSTATE 42704)。そのデータ・ソースの定義は削除されます。

データ・ソースに常駐する表およびビューのニックネームはすべてドロップされます。また、これらのニックネームに従属する SPECIFICATION ONLY 指定の索引もすべてドロップされます。ドロップされたサーバー定義に従属するユーザー定義関数マッピング、ユーザー定義タイプ・マッピング、およびユーザー・マッピングもすべてドロップされます。ドロップされたサーバー定義、関数マッピング、ニックネーム、および SPECIFICATION ONLY 指定の索引に依存するパッケージはすべて無効になります。サーバー定義に依存するすべてのフェデレーテッド・プロシージャもドロップされます。

SERVICE CLASS *service-class-name*

ドロップするサービス・クラスを識別します。 *service-class-name* は、カタログに記述されているサービス・クラスを識別するものでなければなりません (SQLSTATE 42704)。サービス・サブクラスをドロップするには、UNDER 節を使って *service-superclass-name* を指定する必要があります。

UNDER *service-superclass-name*

サービス・サブクラスをドロップする場合に、そのサービス・スーパークラスを指定します。 *service-superclass-name* は、カタログに記述されているサービス・スーパークラスを識別するものでなければなりません (SQLSTATE 42704)。

RESTRICT

このキーワードを指定すると、以下のいずれかの従属関係が存在する場合にサービス・クラスをドロップしないという規則が適用されます。

- サービス・クラスがサービス・スーパークラスであり、このサービス・クラスの下にユーザー定義のサービス・サブクラスが存在する場合 (SQLSTATE 5U031)。まず、サービス・サブクラスをドロップする必要があります。
- サービス・クラスがサービス・スーパークラスであり、このサービス・クラスに対する作業アクション・セットのマッピングが存在する場合 (SQLSTATE 5U031)。まず、作業アクション・セットをドロップする必要があります。

DROP

- サービス・クラスがサービス・サブクラスであり、このサービス・クラスに対する作業アクションのマッピングが存在する場合 (SQLSTATE 5U031)。まず、作業アクションをドロップする必要があります。
- サービス・クラスにワークロード・マッピングが存在する場合 (SQLSTATE 5U031)。まず、ワークロード・マッピングを除去する必要があります。ワークロード・マッピングを除去するには、ワークロードをドロップするか、サービス・クラスをマップしないようワークロードを変更します。
- サービス・クラスにしきい値が関連付けられている場合 (SQLSTATE 5U031)。まず、しきい値をドロップする必要があります。
- サービス・クラスを使用不可にする必要があります (SQLSTATE 5U031)。

RESTRICT は、デフォルトの動作です。

TABLE *table-name*

ドロップする基本表または宣言済み一時表を識別します。 *table-name* に指定する表名は、カタログに記述されている表が、宣言済み一時表を指定する場合は、スキーマ名 SESSION によって修飾され、アプリケーションに存在する一時表の名前でなければなりません (SQLSTATE 42704)。型付き表の副表は、それぞれスーパー表に従属しています。したがって、スーパー表をドロップする前には、副表をすべてドロップする必要があります (SQLSTATE 42893)。指定された表はデータベースから削除されます。

その表を参照するすべての索引、主キー、外部キー、チェック制約、マテリアライズ照会表、およびステージング表はドロップされます。表を参照するすべてのビューおよびトリガーは、作動不能になります。(これには、CREATE TRIGGER ステートメントの ON 節で参照されている表と、トリガー SQL ステートメントで参照されているすべての表が含まれます。)ドロップされたオブジェクトまたは作動不能としてマークされたオブジェクトに従属するすべてのパッケージは無効になります。これには、副表よりも上位の階層であるスーパー表に従属するパッケージが含まれます。参照列の中で、ドロップされた表を参照の有効範囲として定義したものがあれば、参照範囲は無効になります。

宣言済み一時表にパッケージに従属することはありません。したがって、宣言済み一時表がドロップされてもパッケージが無効になることはありません。

フェデレーテッド・システムでは、透過性 DDL を使用して作成されたりリモート表はドロップできます。リモート表をドロップすると、その表に関連したニックネームもドロップされ、そのニックネームに従属するパッケージが無効化されます。

表階層から副表をドロップすると、その副表に関連した列はアクセスできなくなります(ただし、列の数や行のサイズの制限に関しては考慮されます)。副表をドロップすると、スーパー表から副表の行がすべて削除されてしまいます。その結果、スーパー表に定義したトリガーや参照整合性制約が活動化することがあります。

宣言済み一時表が、現在の作業単位またはセーブポイントがアクティブになる前に作成されたものである場合は、その一時表をドロップすると機能上で表がドロップされてしまうため、アプリケーションからその一時表にアクセスすることができなくなります。しかし、表スペースでは、作業単位がコミットされるまで、

あるいはセーブポイントが終了するまで、依然としてこの表が予約された状態にあるため、USER TEMPORARY 表スペースをドロップしたり、USER TEMPORARY 表スペースのデータベース・パーティション・グループを再配分することはできません。宣言済み一時表がドロップされると、DROP がコミットされたかロールバックされたかにかかわらず、表に含まれていたデータはすべて破棄されます。

表に RESTRICT ON DROP 属性があると、その表はドロップできません。

新しくデタッチされた表は、初期状態ではアクセス不能になります。したがって、SET INTEGRITY ステートメントを実行して、MQT の増分リフレッシュを行うか、外部キー制約の処理を完了するまで、表の読み取り、変更、ドロップは不可能になります。すべての従属表に対して SET INTEGRITY ステートメントを実行すると、その表は完全にアクセス可能になり、そのデタッチされた属性はリセットされ、ドロップできる状態になります。

TABLE HIERARCHY *root-table-name*

ドロップする型付き表階層を指定します。 *root-table-name* で指定する型付き表は、型付き表階層のルート表でなければなりません (SQLSTATE 428DR)。 *root-table-name* で指定する型付き表とその表のすべての副表が、データベースから削除されます。

ドロップされた表を参照するすべての索引、マテリアライズ照会表、ステージング表、主キー、外部キー、およびチェック制約はドロップされます。ドロップされた表を参照するすべてのビューおよびトリガーは、作動不能になります。ドロップされたオブジェクトまたは作動不能としてマークされたオブジェクトに従属するすべてのパッケージは無効になります。参照列の中で、ドロップされた表を参照の有効範囲として定義したものがあれば、参照範囲は無効になります。

単一の副表をドロップする場合とは違い、表階層をドロップしても、階層内にある任意の表の削除トリガーが活動化したり、削除された行が記録されたりすることはありません。

TABLESPACE または TABLESPACES *tablespace-name*

ドロップする表スペースを指定します。 *tablespace-name* (表スペース名) は、カタログに記述されている表スペースを指定していなければなりません (SQLSTATE 42704)。これは、1 部構成の名前です。

表の一部がドロップされる表スペースに保管され、1 つかそれ以上の部分がドロップされない別の表スペースに保管されている場合、この表スペースはドロップされません (このような表は前もってドロップする必要があります)。また、その表スペースに存在する表に RESTRICT ON DROP 属性がある場合も、この表スペースはドロップされません (SQLSTATE 55024)。

オブジェクト名に「SYS」という接頭部が付いている場合、そのオブジェクトはシステム定義オブジェクトであり、削除できません (ただし、SYSTOOLSPACE 表スペースと SYSTOOLSTMPSPACE 表スペースは例外です) (SQLSTATE 42832)。

データベースに TEMPORARY 表スペースが 1 つしか存在しない場合は、SYSTEM TEMPORARY 表スペースをドロップすることはできません (SQLSTATE 55026)。宣言済み一時表が作成されている USER TEMPORARY 表スペースはドロップできません (SQLSTATE 55039)。USER TEMPORARY 表

DROP

スペースでは、宣言済み一時表がドロップされても、`DROP TABLE` ステートメントを含む作業単位がコミットされるまでは、その表スペースは使用中と見なされます。

表スペースをドロップすると、その表スペースに定義されているオブジェクトはすべてドロップされます。パッケージや参照制約などのその表スペースに付属する既存のすべてのデータベース・オブジェクトはドロップされるか、または無効になり、従属しているビューやトリガーは作動不能になります。

ユーザーによって作成されたコンテナは削除されません。`CREATE TABLESPACE` の実行時にデータベース・マネージャーによって作成されたコンテナ名のパスに含まれているディレクトリーは、いずれも削除されます。データベース・ディレクトリーの下にあるすべてのコンテナは削除されます。`DROP TABLESPACE` ステートメントがコミットされると、可能なら、指定された表スペースの `DMS` ファイル・コンテナや `SMS` コンテナが削除されます。コンテナが削除できない場合 (たとえば、別のエージェントによってオープンされたままになっている場合など) は、ファイルが長さ 0 に切り捨てられます。これらの長さ 0 のファイルは、すべての接続が終了するか、`DEACTIVATE DATABASE` コマンドが発行されたときに削除されます。

THRESHOLD *threshold-name*

ドロップするしきい値を指定します。*threshold-name* は現行のサーバーに存在するしきい値を識別するものでなければなりません (SQLSTATE 42704)。これは、1 部構成の名前です。キューで使用されるしきい値 (例えば `TOTALSCPARTITIONCONNECTIONS` や `CONCURRENTDBCOORDACTIVITIES`) をドロップするには、その前にまず使用不可にする必要があります (SQLSTATE 5U025)。指定されたしきい値はカタログから削除されます。

TRIGGER *trigger-name*

ドロップするトリガーを指定します。*trigger-name* (トリガー名) は、カタログに記述されているトリガーを指定していなければなりません (SQLSTATE 42704)。指定したトリガーは削除されます。

トリガーをドロップすると、特定のパッケージが無効としてマークされます。

trigger-name がビューに対して `INSTEAD OF` トリガーを指定する場合、そのビューに対する更新を行うことにより、他のトリガーはそのトリガーに従属できません。

TRANSFORM ALL FOR *type-name*

ユーザー定義データ・タイプ *type-name* に定義されたすべてのトランスフォーム・グループがドロップされることを示します。これらのグループで参照されるトランスフォーム関数はドロップされません。動的 SQL ステートメントでは、`CURRENT SCHEMA` 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、`QUALIFIER` プリコンパイル/ `BIND` オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。*type-name* に指定されるタイプ名は、カタログに記述されているユーザー定義タイプを識別するものでなければなりません (SQLSTATE 42704)。

type-name に定義されているトランスフォームが存在しない場合は、エラーが戻されます (SQLSTATE 42740)。

DROP TRANSFORM は、CREATE TRANSFORM の逆の処理を行います。

DROP TRANSFORM は、指定されたデータ・タイプで特定のグループに関連付けられたトランスフォーム関数を未定義の状態にします。これらのグループに関連付けられていた関数は引き続き存在しており、明示的に呼び出すことができますが、これらの関数にはもはやトランスフォーム・プロパティは含まれていないので、ホスト言語環境で値を交換するためにこれらの関数が暗黙的に呼び出されることはありません。

トランスフォーム・グループの中に SQL 以外の言語で書かれたユーザー定義関数 (またはメソッド) があり、その関数が、ユーザー定義タイプ *type-name* に定義されたそのグループのトランスフォーム関数のいずれかに従属している場合、そのトランスフォーム・グループはドロップされません (SQLSTATE 42893)。このようなユーザー定義関数が従属しているトランスフォーム関数は、*type-name* で定義された参照先のトランスフォーム・グループに関連付けられています。そのため、パッケージが属しているトランスフォーム関数が、指定されたトランスフォーム・グループと関連付けられていると、そのパッケージは作動不能としてマークされてしまいます。

TRANSFORMS *group-name* **FOR** *type-name*

ユーザーが定義したデータ・タイプ *type-name* から、指定したトランスフォーム・グループがドロップされることを示します。このグループで参照されるトランスフォーム関数はドロップされません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *type-name* に指定されるタイプ名は、カタログに記述されているユーザー定義タイプを識別するものでなければなりません (SQLSTATE 42704)。また、*group-name* には、 *type-name* に存在しているトランスフォーム・グループを指定しなければなりません。

TRIGGER *trigger-name*

ドロップするトリガーを指定します。 *trigger-name* (トリガー名) は、カタログに記述されているトリガーを指定していなければなりません (SQLSTATE 42704)。指定したトリガーは削除されます。

トリガーをドロップすると、特定のパッケージが無効としてマークされます。

trigger-name がビューに対して INSTEAD OF トリガーを指定する場合、そのビューに対する更新を行うことにより、他のトリガーはそのトリガーに従属できません。

TRUSTED CONTEXT *context-name*

ドロップするトラステッド・コンテキストを指定します。 *context-name* は現行のサーバーに存在するトラステッド・コンテキストを識別するものでなければなりません (SQLSTATE 42704)。トラステッド・コンテキストのトラステッド接続がアクティブである間にこのコンテキストがドロップされた場合、接続が終了するか、次の再使用が試行されるまで、これらの接続はトラステッド状態のままになります。これらのトラステッド接続でのユーザー切り替えが試行された場合、エラーが戻されます (SQLSTATE 42517)。指定されたトラステッド・コンテキストはカタログから削除されます。

TYPE *type-name*

ドロップするユーザー定義タイプを指定します。動的 SQL ステートメントで

は、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションにより、修飾子のないオブジェクト名の修飾子が暗黙指定されます。構造化タイプでは、関連した参照タイプもドロップされます。*type-name* (タイプ名) は、カタログに記述されているユーザー定義タイプを指定していなければなりません。

RESTRICT

以下の場合、このタイプはドロップされません (SQLSTATE 42893)。

- 表またはビューの列のタイプとして使用されるタイプである。
- サブタイプが含まれている。
- 型付き表または型付きビューのデータ・タイプとして使用されている構造化タイプである。
- 他の構造化タイプの属性として使用されるタイプである。
- 表の列のタイプに *type-name* のインスタンスが含まれている可能性がある。これには、列のタイプが *type-name* である場合や、列に関連付けられたタイプ階層以外のロケーションで *type-name* が使用される場合などがあります。もっと典型的な例としては、どのタイプ (T) であれ、表の列のタイプで *type-name* が直接または間接的に使用されている場合には、T をドロップすることはできません。
- タイプが、表またはビューの参照タイプ列のターゲット・タイプ、または別の構造化タイプの参照タイプ属性である。
- このタイプ、あるいはこのタイプを参照する値が、関数やメソッドのパラメーター・タイプまたは戻り値タイプである。
- タイプがパラメーター・タイプであるか、SQL プロシーチャーの本体で使用されている。
- このタイプ、またはこのタイプを参照する値が SQL 関数やメソッドの本体で使用されているが、パラメーター・タイプや戻り値タイプではない。
- このタイプがチェック制約、トリガー、ビュー定義、または索引の拡張で使用されている。

RESTRICT が指定されていない場合の振る舞いは、そのタイプを使用する関数およびメソッドの場合を除いて RESTRICT の振る舞いと同じです。

ドロップされるタイプを使用する関数の場合、ユーザー定義タイプがドロップ可能であると、ドロップするそのタイプ (またはドロップするタイプを参照するもの) のパラメーターまたは戻り値が含まれているすべての関数 (F) (特定名は SF) に、以下の DROP FUNCTION ステートメントが実行されることとなります。

DROP SPECIFIC FUNCTION SF

このステートメントがカスケードして、従属する関数もドロップされる可能性があります。ユーザー定義タイプへの従属関係に基づいて、それらの関数もすべてドロップ・リストに含まれている場合には、ユーザー定義タイプのドロップは正常に処理されます (そうでない場合、SQLSTATE 42893 のエラーになります)。

ドロップされるタイプを使用するメソッドの場合、ユーザー定義タイプがドロップ可能であると、ドロップするそのタイプ (またはドロップするタイプを参照す

るもの) のパラメーターまたは戻り値が含まれているタイプ T1 のメソッド (M) (特定名は SM) に、以下のステートメントが実行されることとなります。

```
DROP SPECIFIC METHOD SM
ALTER TYPE T1 DROP SPECIFIC METHOD SM
```

これらのメソッドに従属しているオブジェクトがあると、DROP TYPE 操作が失敗する場合があります。

ドロップするタイプのスーパータイプで定義されるメソッドに従属し、オーバーライドに適したパッケージはすべて、無効になります。

TYPE MAPPING *type-mapping-name*

ドロップするユーザー定義のデータ・タイプ・マッピングを指定します。

type-mapping-name (タイプ・マッピング名) は、カタログに記述されているデータ・タイプ・マッピングを指定していなければなりません (SQLSTATE 42704)。指定したデータ・タイプ・マッピングがデータベースから削除されません。

その他にドロップされるオブジェクトはありません。

USER MAPPING FOR *authorization-name* | **USER SERVER** *server-name*

ドロップするユーザー・マッピングを指定します。このマッピングは、フェデレーテッド・データベースにアクセスするために使う許可名を、データ・ソースにアクセスするために使う許可名に関連付けます。これら 2 つのうち最初の許可名は、*authorization-name* で指定されるか、または特殊レジスター USER によって参照されます。*server-name* は、アクセスするのに 2 番目の許可名を使用するデータ・ソースを指定します。

authorization-name は、カタログにリストされていなければなりません (SQLSTATE 42704)。*server-name* に指定するサーバー名は、カタログに記述されているデータ・ソースを識別するものでなければなりません (SQLSTATE 42704)。ユーザー・マッピングが削除されます。

その他にドロップされるオブジェクトはありません。

VARIABLE *variable-name*

ドロップするグローバル変数を指定します。*variable-name* は、現在のサーバーに存在するグローバル変数を識別するものでなければなりません (SQLSTATE 42704)。

RESTRICT

このキーワードを使用すれば、関数、メソッド、トリガー、またはビューの中で参照されているグローバル変数をドロップできないようにする規則が適用されます (SQLSTATE 42893) RESTRICT は、デフォルトの動作です。

VIEW *view-name*

ドロップするビューを指定します。*view-name* (ビュー名) は、カタログに記述されているビューを指定していなければなりません (SQLSTATE 42704)。型付きビューのサブビューは、それぞれスーパービューに従属しています。したがって、スーパービューをドロップする前に、サブビューをすべてドロップする必要があります (SQLSTATE 42893)。

指定したビューは削除されます。直接的または間接的にそのビューに従属するビューまたはトリガーの定義は、作動不能としてマークされます。作動不能というマークが付いた表に従属するマテリアライズ照会表またはステージング表はすべて

てドロップされます。ドロップされたビューまたは作動不能としてマークされたビューに從属するパッケージはいずれも無効になります。これには、サブビューよりも上位の階層であるスーパービューに從属するパッケージが含まれます。参照列の中で、ドロップされたビューを参照の有効範囲として定義したものがあれば、参照範囲は無効になります。

VIEW HIERARCHY *root-view-name*

ドロップする型付きビュー階層を指定します。 *root-view-name* で指定する型付きビューは、型付きビュー階層のルート・ビューでなければなりません (SQLSTATE 428DR)。 *root-view-name* で指定する型付きビューとそのビューのすべてのサブビューが、データベースから削除されます。

直接的または間接的にドロップされたビューに從属するビューまたはトリガーの定義は、作動不能としてマークされます。ドロップされたビューやトリガー、または作動不能としてマークされたビューやトリガーに從属するパッケージはいずれも、無効になります。参照列の中で、ドロップされたビューや作動不能とマークされたビューを参照の有効範囲として定義したものがあれば、参照範囲は無効になります。

WORK ACTION SET *work-action-set-name*

ドロップする作業アクション・セットを指定します。 *work-action-set-name* には、現行のサーバー上の既存の作業アクション・セットを指定する必要があります (SQLSTATE 42704)。 *work-action-set-name* に含まれるすべての作業アクションもまた、ドロップされます。

WORK CLASS SET *work-class-set-name*

ドロップする作業クラス・セットを指定します。 *work-class-set-name* には、現行のサーバー上に既存の作業クラス・セット名を指定する必要があります (SQLSTATE 42704)。 *work-class-set-name* に含まれるすべての作業クラスもまた、ドロップされます。

RESTRICT

このキーワードを使用すれば、いずれかの作業アクション・セットに関連付けられている作業クラス・セットをドロップしないという規則が適用されません (SQLSTATE 42893)。 **RESTRICT** は、デフォルトの動作です。

WORKLOAD *workload-name*

ドロップするワークロードを指定します。これは、1 部構成の名前です。 *workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。 **SYSDEFAULTUSERWORKLOAD** または **SYSDEFAULTADMWORKLOAD** はドロップできません (SQLSTATE 42832)。ワークロードをドロップする前に、そのワークロードをまず使用不可にして、それに関連したアクティブ・ワークロードが発生していないことを確認する必要があります (SQLSTATE 5U023)。指定されたワークロードはカタログから削除されます。

WRAPPER *wrapper-name*

ドロップするラッパーを指定します。 *wrapper-name* (ラッパー名) は、カタログに記述されているラッパーを指定していなければなりません (SQLSTATE 42704)。そのラッパーは削除されます。

そのラッパーに從属するすべてのサーバー定義、ユーザー定義関数マッピング、およびユーザー定義データ・タイプ・マッピングはドロップされます。ドロップ

プされたサーバー定義に従属するユーザー定義関数マッピング、ニックネーム、ユーザー定義データ・タイプ・マッピング、およびユーザー・マッピングもすべてドロップされます。ドロップされたニックネームに従属する SPECIFICATION ONLY 指定の索引はすべてドロップされ、こうしたニックネームに従属するビューはすべて、作動不能としてマークが付けられます。ドロップされたオブジェクトと作動不能ビューに従属するすべてのパッケージは無効になります。ドロップされたサーバー定義に依存するすべてのフェデレーテッド・プロシージャーもドロップされます。

XSROBJECT *xsobject-name*

ドロップする XSR オブジェクトを指定します。*xsobject-name* は、カタログに記述されている XSR オブジェクトを識別するものでなければなりません (SQLSTATE 42704)。

XSR オブジェクトを参照するチェック制約はドロップされます。XSR オブジェクトを参照するすべてのトリガーおよびビューは、作動不能とマークされます。ドロップされた XSR オブジェクトに依存しているパッケージは、無効になります。

規則

従属関係: 728 ページの表 28 は、オブジェクト相互間従属関係を示します。カタログには明示的に記録されない従属関係があります。たとえば、パッケージが従属している制約の記録はありません。このリストには、以下の 4 つの異なるタイプの従属関係が示されています。

- R** 制限 (Restrict) を意味します。従属オブジェクトが存在する限り、その基礎となるオブジェクトはドロップできません。
- C** カスケード (Cascade) を意味します。基礎となるオブジェクトをドロップすると、その従属オブジェクトも同時にドロップされます。ただし、その従属オブジェクトにさらに他のオブジェクトに対する制限 (R) 従属関係があり、それによってその従属オブジェクトをドロップできない場合には、基礎となるオブジェクトのドロップは失敗します。
- X** 作動不能 (Inoperative) を意味します。基礎となるオブジェクトをドロップすると、その従属オブジェクトは作動不能になります。ユーザーが何らかの明示的な処置を取るまで、それは作動不能のままになります。
- A** 自動無効化 / 再有効化 (Automatic Invalidation/Revalidation) を意味します。基礎となるオブジェクトをドロップすると、従属オブジェクトは無効になります。データベース・マネージャーは、無効になったオブジェクトを再度有効にしようとします。

関数またはメソッドによって使用されるか、関数またはメソッドによって直接あるいは間接に呼び出されるプロシージャーによって使用されるパッケージは、ルーチンが MODIFIES SQL DATA として定義される場合のみ、自動的に再度有効にされます。ルーチンが MODIFIES SQL DATA でなければ、エラーが戻されます (SQLSTATE 56098)。

DROP ステートメントのパラメーターおよびオブジェクトには、結果的にブランク行または列になるため、728 ページの表 28 に示されていないものもあります。

- 4 あるユーザー定義タイプ **T** を別のユーザー定義タイプ **B** に従属させるには、**T** が以下の条件を満たしていなければなりません。
- 属性のデータ・タイプとして **B** を指定している
 - REF(**B**) の属性を持っている
 - スーパータイプとして **B** を持っている
- 5 ユーザー定義タイプをドロップすると、その効果がカスケードして、パラメーターや結果タイプとしてそのタイプを使用する関数やメソッド、あるいは関数本体またはメソッド本体でそのタイプを使用する関数やメソッドもドロップされることとなります。ユーザー定義タイプが構造化タイプである場合、そのタイプに関連したメソッドもすべてドロップされます。そのタイプと関数またはメソッドが互いに依存している場合でも、これらの関数やメソッドがドロップ不能になるわけではありません。
- 6 表スペースまたは表スペースのリストをドロップすると、指定した表スペース内に完全に含まれているすべての表やリストがドロップされることとなります。ただし、表が複数の表スペース (異なる表スペース内の索引、長形式列、またはデータ・パーティション) にわたり、そうした表スペースがドロップされるリストにない場合、これらの表スペースは表が存在する限りはドロップできません。
- 7 従属関数が **SOURCE** 節内の基本関数の名前である場合、その関数は別の特定の関数に従属します。また、従属のルーチンが **SQL** で書かれており、その本体で基本のルーチンを使用する場合も、関数やメソッドは別の特定の関数やメソッドに従属することができます。加えて、構造化タイプのパラメーターや戻りタイプをもつ外部のメソッドや関数も、1 つまたは複数のトランスフォーム関数に従属することができます。
- 8 マテリアライズ照会表がドロップされたり、ビューが作動不能になるのは、**SELECT** 特権がない場合だけです。作動不能にされたビューが型付きビュー階層に含まれていれば、そのサブビューもすべて作動不能になります。
- 9 パッケージに、表 **T** に影響を与える **INSERT**、**UPDATE**、または **DELETE** ステートメントが含まれている場合、そのパッケージは **T** に対して挿入、更新、または削除の操作を行うこととなります。**UPDATE** の場合、パッケージは **UPDATE** によって修正される **T** の各列ごとに更新操作を行います。
- 型付き表に対して操作を行うステートメントがパッケージに含まれている場合、同じ表階層内で表を作成したりドロップしたりすると、パッケージが無効になります。
- 10 列に対する特権を個々に取り消すことはできないので、列レベルでの従属関係は存在しません。
- パッケージ、トリガー、またはビューの **FROM** 節で **OUTER(Z)** が使用されている場合、**Z** のすべての副表またはサブビューで **SELECT** 特権に対する従属関係が存在します。同じように、パッケージ、トリガー、またはビューで **DEREF(Y)** が使用されていて、**Y** が **Z** という表またはビューをターゲットとする参照タイプである場合、**Z** のすべての副表またはサブビューで **SELECT** 特権に対する従属関係が存在します。

- 11 マテリアライズ照会表は、基礎表、あるいは表定義の全選択で指定されたニックネームに従属しています。
- カスケードのセマンティクスが、従属するマテリアライズ照会表に適用されます。
- 副表はスーパー表に従属しており、この従属関係はルート表にまで及びます。従属するすべての副表がドロップされるまで、スーパー表はドロップできません。
- 12 `TYPE` 述部またはサブタイプ処理の式 (`TREAT expression AS data-type`) を使用した結果、パッケージは構造化タイプに従属することができます。パッケージは、`TYPE` 述部の右辺、または `TREAT` 式の右辺で指定した各構造化タイプのサブタイプすべてと従属関係にあります。構造化タイプをドロップしたり作成したりして、パッケージと従属関係にあるサブタイプを変更すると、ステートメントが無効になる場合があります。
- ドロップするタイプのスーパータイプで定義されるメソッドに従属し、オーバーライドに適したパッケージはすべて、無効になります。
- 13 あるタイプがチェック制約またはトリガーで使用されている場合、チェック制約またはトリガーはこのタイプに従属する関係にあります。チェック制約またはトリガーの `TYPE` 述部で使用される、構造化タイプのサブタイプに従属しません。
- 14 あるタイプがビュー定義で使用されている場合、ビューはこのタイプに従属する関係にあります (型付きビューのタイプも含まれます)。ビュー定義内の `TYPE` 述部で使用される、構造化タイプのサブタイプに従属しません。
- 15 サブビューはスーパービューに従属しており、この従属関係はルート・ビューにまで及びます。従属するすべてのサブビューがドロップされるまで、スーパービューはドロップできません。ビューの従属関係の詳細については、¹⁶ を参照してください。
- 16 トリガーまたはビューは間接参照操作または `DEREF` 関数のターゲット表やターゲット・ビューにも従属しています。 `FROM` 節のトリガーまたはビューで `OUTER(Z)` を含むものは、トリガーまたはビューが作成された時点で存在した `Z` の副表またはサブビューすべてに対して従属関係にあります。
- 17 型付きビューはユニーク索引が存在しているかどうか依存していることがあり、それによってオブジェクト ID 列がユニークなものにすることができます。
- 18 表はユーザー定義データ・タイプ (特殊タイプまたは構造化タイプ) に従属している場合があります、それには以下の理由があります。
- そのタイプが列のタイプとして使用されている
 - そのタイプが表のタイプとして使用されている
 - そのタイプが表のタイプの属性として使用されている
 - そのタイプが、表の列タイプまたは表のタイプの属性を表す、参照タイプのターゲット・タイプとして使用されている
 - そのタイプが、表の列のタイプによって直接または間接的に使用されている

DROP

- 19 サーバーをドロップすると、カスケード的に、そのネーム・サーバーに作成した関数マッピングとタイプ・マッピングがドロップされます。
- 20 複数パーティションのデータベース・パーティション・グループにある表に対して分散キーが定義されている場合、この分散キーは必須です。
- 21 従属している OLE DB 表関数に "R" 従属オブジェクト (DROP FUNCTION を参照) が含まれている場合は、サーバーをドロップできません。
- 22 SQL 関数またはメソッドは、その本体によって参照されるオブジェクトに従属することができます。
- 23 *type-name* T のタイプ TA の属性 A がドロップされると、以下の DROP ステートメントが実際に実行されます。
- ```
Mutator method: DROP METHOD A (TA) FOR T
Observer method: DROP METHOD A () FOR T
ALTER TYPE T
 DROP METHOD A(TA)
 DROP METHOD A()
```
- 24 次のような場合に、表はユーザー定義による構造化データ・タイプの属性に従属することがあります。
1. 表が、*type-name* またはそのサブタイプのいずれかに基づく型付き表である。
  2. 表に、*type-name* を直接または間接的に参照するタイプの列が含まれている。
- 25 定義された関数またはメソッド本体に SELECT 特権がなくなると、SQL 関数の本体またはメソッド本体で使用される表またはビューに対する SELECT 特権の REVOKE により、特権を失った関数またはメソッド本体のドロップが試行されます。これらの関数またはメソッド本体がビュー、トリガー、関数、またはメソッド本体で使用されている場合は、これをドロップすることはできないので、結果として REVOKE が制約されます。それ以外の場合は、REVOKE がカスケードしてそれらの関数はドロップされません。
- 26 トリガーは、INSTEAD OF トリガーが定義されるビューを変更して、INSTEAD OF トリガーが実行される場合、INSTEAD OF トリガーに従属します。
- 27 他のメソッドによってオーバーライドされた元のメソッドのメソッド宣言は、ドロップすることができません (SQLSTATE -2)。
- 28 作成されるメソッド本体のメソッドが、別のメソッドをオーバーライドするものと宣言される場合、オーバーライドされるメソッド (および、作成されるメソッドのスーパータイプでこのメソッドをオーバーライドするメソッド) に従属したパッケージはすべて無効になります。
- 29 既存のタイプの新しいサブタイプが作成されると、作成されるタイプのスーパータイプで定義されるメソッド (および、オーバーライドに適しているメソッド (たとえば、no mutator や observer)) に従属するパッケージはすべて無効になります。
- 30 ドロップされるメソッド本体の特定メソッドが、別のメソッドをオーバーライドするものと宣言される場合、オーバーライドされるメソッド (および、

ドロップされる特定メソッドのスーパータイプでこのメソッドをオーバーライドするメソッド) に従属したパッケージはすべて無効になります。

- 31 キャッシュに入れられた動的 SQL には、パッケージと同じセマンティクスがあります。
- 32 DROP TABLE ステートメントを使ってリモート基本表をドロップするときには、ニックネームとリモート基本表の両方がドロップされます。
- 33 外部キーが参照していない主キーまたはユニーク・キーは、ニックネームのローカル名やローカル・タイプの変更を制限しません。
- 34 XSROBJECT は、分解対象の XML スキーマに関連付けられている表を変更した結果として、分解操作が機能しなくなることがあります。分解操作に影響を与える変更としては、表のドロップ、表の列のドロップ、表の列の変更があります。ALTER XSROBJECT ステートメントを実行すれば、XML スキーマの分解状況をリセットして、XML スキーマの分解を使用可能/使用不可にすることができます。
- 35
- サービス・クラスに対して何らかのしきい値がマップされている場合、そのサービス・クラスはドロップできません (SQLSTATE 5U031)。
  - サービス・クラスに対して何らかのワークロードがマップされている場合、そのサービス・クラスはドロップできません (SQLSTATE 5U031)。
  - サービス・スーパークラスのユーザー定義サービス・サブクラスがすべてドロップされるまでは、そのサービス・スーパークラスはドロップできません (SQLSTATE 5U031)。
  - サービス・スーパークラスに対して作業アクション・セットがマップされている場合、そのサービス・スーパークラスはドロップできません (SQLSTATE 5U031)。
  - サービス・サブクラスに対して作業アクションがマップされている場合、そのサービス・サブクラスはドロップできません (SQLSTATE 5U031)。
- 36 作業クラス・セットに対して定義されている作業アクション・セットがドロップされるまでは、その作業クラス・セットをドロップできません。

## 注

- ユーザー定義関数を使用中に、そのユーザー定義関数をドロップすることは有効です。また、ユーザー定義関数への参照を含むステートメントでカーソルがオープンされているようにすることができます。そのカーソルがオープンされている間に、カーソルのフェッチがエラーになることなくその関数をドロップすることができます。
- ユーザー定義関数に従属しているパッケージが実行されている場合、そのパッケージが現行の作業単位を完了するまで、別の許可 ID からその関数をドロップすることはできません。その時点で、関数はドロップされ、パッケージは作動不能になります。このパッケージの次の要求はエラーになり、パッケージの明示再バインドが必要であることが示されます。
- 関数本体を必要とするアプリケーションが実行されている時に、関数本体が除去される場合があります (これは関数のドロップとは異なります)。ステートメント

## DROP

の代わりにデータベース・マネージャーが関数本体をストレージにロードする必要があるかどうかに応じて、ステートメントはエラーになる場合もあれば、エラーにならない場合もあります。

- 明示的に指定された UDF に記録されている従属関係に加えて、トランスフォームが暗黙的に必要な場合には以下の従属関係が記録されます。
  1. 構造化タイプのパラメーターや関数またはメソッドの結果にトランスフォームが必要な場合は、その関数またはメソッドに、`TO SQL` か `FROM SQL` の必要なトランスフォーム関数に対する従属関係が記録されます。
  2. パッケージに含まれている SQL ステートメントでトランスフォーム関数が必要になる場合は、そのパッケージに、`TO SQL` か `FROM SQL` の指定されたトランスフォーム関数に対する従属関係が記録されます。

上記の部分では、トランスフォームを暗黙的に呼び出すことによって従属関係が記録される場合のみを扱っているため、関数、メソッド、あるいはパッケージ以外のオブジェクトが、暗黙的に呼び出されたトランスフォーム関数に従属することはありません。一方、トランスフォーム関数を明示的に呼び出した場合（たとえば、ビューやトリガーなどで）は、これらの他のタイプのオブジェクトが通常どおりトランスフォーム関数に従属します。したがって、トランスフォームに対するこれらの「明示的な」タイプの従属がドロップされることによって、`DROP TRANSFORM` が失敗する場合があります (SQLSTATE 42893)。

- 従属関係カタログでは、暗黙的なトランスフォームによる関数への従属と明示的に関数を呼び出すことによって生じる従属とを区別していません。したがって、トランスフォーム関数に対する明示的な呼び出しは書かないよう勧められています。このようなインスタンスでは、単に SQL の式に明示的な呼び出しが含まれているという理由で、関数上のトランスフォーム・プロパティがドロップされなかったり、パッケージが作動不能としてマークされてしまいます。
- ID 列のシーケンスを作成したシステムを、`DROP SEQUENCE` ステートメントでドロップすることはできません。
- シーケンスがドロップされると、シーケンスに関する特権もすべてドロップされ、そのシーケンスを参照するパッケージはすべて無効になります。
- リレーショナル・ニックネームの場合、所定の作業単位 (UOW) 内の `DROP NICKNAME` ステートメントは、以下のいずれかの条件の下では処理できません (SQLSTATE 55007)。
  - このステートメントで参照されているニックネームには、同じ UOW 内でオープンされているカーソルがある。
  - このステートメントで参照されているニックネームに対して、同じ UOW 内ですでに `INSERT`、`DELETE`、または `UPDATE` ステートメントのいずれかが出されている。
- 非リレーショナル・ニックネームの場合、所定の作業単位 (UOW) 内の `DROP NICKNAME` ステートメントは、以下のいずれかの条件の下では処理できません (SQLSTATE 55007)。
  - このステートメントで参照されているニックネームには、同じ UOW 内でオープンされているカーソルがある。
  - このステートメントで参照されているニックネームは、同じ UOW 内の `SELECT` ステートメントですでに参照されている。

- このステートメントで参照されているニックネームに対して、同じ UOW 内ですでに INSERT、DELETE、または UPDATE ステートメントのいずれかが出されている。
- 所定の作業単位 (UOW) 内の DROP SERVER ステートメント (SQLSTATE 55006)、または DROP FUNCTION MAPPING あるいは DROP TYPE MAPPING ステートメント (SQLSTATE 55007) は、以下のいずれかの条件の下では処理できません。
  - ステートメントが 1 つのデータ・ソースを参照していて、次のいずれかがすでに UOW に含まれている。
    - このデータ・ソース内の表またはビューのニックネームを参照する SELECT ステートメント。
    - このデータ・ソース内の表またはビューのニックネーム上のオープン・カーソル。
    - このデータ・ソース内の表またはビューのニックネームに対して発行された INSERT、DELETE、または UPDATE ステートメント。
  - ステートメントがデータ・ソースのカテゴリ (たとえば、特定のタイプおよびバージョンのすべてのデータ・ソースなど) を参照しており、次のいずれかがすでに UOW に含まれている。
    - それらのデータ・ソースのいずれかの中の表またはビューのニックネームを参照する SELECT ステートメント。
    - それらのデータ・ソースのいずれかの中の表またはビューのニックネーム上のオープン・カーソル。
    - それらのデータ・ソースのいずれかの中の表またはビューのニックネームに対して発行された INSERT、DELETE、または UPDATE ステートメント。
- DROP WORKLOAD ステートメントは、コミットされるまでは有効になりません。これは、ステートメントを発行する接続でも同じです。
- これらのステートメントはアプリケーションによって一度に 1 つのみ発行され、1 つの作業単位内で 1 つのみ許可されます。これらのステートメントのいずれかを次に発行する前に、各ステートメントの後に COMMIT または ROLLBACK ステートメントを使用する必要があります (SQLSTATE 5U021)。
  - CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、または DROP (HISTOGRAM TEMPLATE)
  - CREATE SERVICE CLASS、ALTER SERVICE CLASS、または DROP (SERVICE CLASS)
  - CREATE THRESHOLD、ALTER THRESHOLD、または DROP (THRESHOLD)
  - CREATE WORK ACTION、ALTER WORK ACTION、または DROP (WORK ACTION)
  - CREATE WORK CLASS、ALTER WORK CLASS、または DROP (WORK CLASS)
  - CREATE WORKLOAD、ALTER WORKLOAD、または DROP (WORKLOAD)
  - GRANT (ワークロード特権) または REVOKE (ワークロード特権)
- **互換性**
  - 以前のバージョンの DB2 との互換性:

## DROP

- DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。
- TYPE *type-name* の代わりに DISTINCT TYPE *type-name* を指定できます。
- TYPE *type-name* の代わりに DATA TYPE *type-name* を指定できます。
- DB2 for z/OS との互換性:
  - ALIAS の代わりに SYNONYM を指定できます。
  - PACKAGE の代わりに PROGRAM を指定できます。

### 例

例 1: 表 TDEPT をドロップします。

```
DROP TABLE TDEPT
```

例 2: ビュー VDEPT をドロップします。

```
DROP VIEW VDEPT
```

例 3: 許可 ID HEDGES が別名のドロップを試みます。

```
DROP ALIAS A1
```

別名 HEDGES.A1 がカタログから除去されます。

例 4: Hedges は別名のドロップを試みますが、既存の表の名前である (別名でない) T1 を別名として指定しています。

```
DROP ALIAS T1
```

このステートメントはエラーになります (SQLSTATE 42809)。

例 5:

BUSINESS\_OPS データベース・パーティション・グループをドロップします。このデータベース・パーティション・グループをドロップするには、まずデータベース・パーティション・グループ内の表スペース (ACCOUNTING と PLANS) をドロップする必要があります。

```
DROP TABLESPACE ACCOUNTING
DROP TABLESPACE PLANS
DROP DATABASE PARTITION GROUP BUSINESS_OPS
```

例 6: Pellow は CENTRE 関数をドロップします。この関数は、ドロップする関数インスタンスであることを示すためにシグニチャーを使用して、PELLOW スキーマに作成したものです。

```
DROP FUNCTION CENTRE (INT,FLOAT)
```

例 7: McBride は FOCUS92 関数をドロップします。この関数は、ドロップする関数インスタンスであることを示すために特定名を使用して、PELLOW スキーマに作成したものです。

```
DROP SPECIFIC FUNCTION PELLOW.FOCUS92
```

例 8: CHEM スキーマから関数 ATOMIC\_WEIGHT をドロップします。このスキーマには、この名前関数は 1 つしかないことが分かっています。

```
DROP FUNCTION CHEM.ATOMIC_WEIGHT
```

例 9: トリガー SALARY\_BONUS をドロップします。このトリガーにより、従業員は指定の条件の下で給与に加えてボーナスを受け取ります。

```
DROP TRIGGER SALARY_BONUS
```

例 10: 現在使用していない SHOESIZE という名前の特殊データ・タイプをドロップします。

```
DROP TYPE SHOESIZE
```

例 11: SMITHPAY イベント・モニターをドロップします。

```
DROP EVENT MONITOR SMITHPAY
```

例 12: CREATE SCHEMA の例 2 で RESTRICT を使用して作成したスキーマをドロップします。PART という名前の表をまずドロップする必要があることに注意してください。

```
DROP TABLE PART
DROP SCHEMA INVENTORY RESTRICT
```

例 13: Macdonald は DESTROY プロシージャをドロップします。このプロシージャは、ドロップするプロシージャ・インスタンスであることを示すために特定名を使用して、EIGLER スキーマに作成したものです。

```
DROP SPECIFIC PROCEDURE EIGLER.DESTROY
```

例 14: BIOLOGY スキーマからプロシージャ OSMOSIS をドロップします。このスキーマには、この名前のプロシージャは 1 つしかないことが分かっています。

```
DROP PROCEDURE BIOLOGY.OSMOSIS
```

例 15: ユーザー SHAWN は、フェデレーテッド・データベースにアクセスするときと、ORACLE1 という Oracle データ・ソースのデータベースにアクセスするときでは、異なる許可 ID を使用しました。2 つの許可でマッピングが作成されましたが、SHAWN がそのデータ・ソースにアクセスする必要はなくなりました。マッピングをドロップします。

```
DROP USER MAPPING FOR SHAWN SERVER ORACLE1
```

例 16: ニックネームが参照するデータ・ソース表の索引が削除されました。オプティマイザーにこの索引を認識させるために作成した SPECIFICATION ONLY 指定の索引をドロップします。

```
DROP INDEX INDEXSPEC
```

例 17: トランスフォーム・グループ MYSTRUCT1 をドロップします。

```
DROP TRANSFORM MYSTRUCT1 FOR POLYGON
```

例 18: PERSONNEL スキーマで EMP データ・タイプからメソッド BONUS をドロップします。

```
DROP METHOD BONUS (SALARY DECIMAL(10,2)) FOR PERSONNEL.EMP
```

例 19: 制限を使用して ORG\_SEQ からシーケンスをドロップします。

```
DROP SEQUENCE ORG_SEQ
```

## DROP

例 20: リモート表 EMPLOYEE が、フェデレーテッド・システムに透過性 DDL を使用して作成されました。この表へのアクセスは、もう必要ありません。リモート表 EMPLOYEE をドロップします。

```
DROP TABLE EMPLOYEE
```

例 21: 関数マッピング BONUS\_CALC をドロップし、デフォルトの関数マッピングがあればそれを復元します。

```
DROP FUNCTION MAPPING BONUS_CALC
```

例 22: セキュリティー・ラベル・コンポーネント LEVEL をドロップします。

```
DROP SECURITY LABEL COMPONENT LEVEL
```

例 23: セキュリティー・ポリシー DATA\_ACCESS のセキュリティー・ラベル EMPLOYEESECLABEL をドロップします。

```
DROP SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
```

例 24: セキュリティー・ポリシー DATA\_ACCESS をドロップします。

```
DROP SECURITY POLICY DATA_ACCESS
```

例 25: セキュリティー・ラベル・コンポーネント GROUPS をドロップします。

```
DROP SECURITY LABEL COMPONENT GROUPS
```

例 26: SQL スキーマ HR にある XML スキーマ EMPLOYEE をドロップします。

```
DROP XSROBJECT HR.EMPLOYEE
```

例 27: サービス・スーパークラス PETSALLES の下にあるサービス・サブクラス DOGSALES をドロップします。

```
DROP SERVICE CLASS DOGSALES UNDER PETSALLES
```

例 28: ユーザー定義サービス・サブクラスを持たないサービス・スーパークラス PETSALLES をドロップします。サービス・クラス PETSALLES のデフォルト・サブクラスは自動的にドロップされます。

```
DROP SERVICE CLASS PETSALLES
```

---

## END DECLARE SECTION

END DECLARE SECTION ステートメントは、ホスト変数宣言セクションの終わりを示します。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、実行可能ステートメントではありません。また、REXX に指定することはできません。

### 許可

必要ありません。

## 構文

▶—END DECLARE SECTION—▶

## 説明

END DECLARE SECTION ステートメントは、ホスト言語の規則に従って宣言を指定できる個所であれば、アプリケーション・プログラムのどこにでもコーディングすることができます。これは、ホスト変数の宣言セクションの終了を示します。ホスト変数セクションは、BEGIN DECLARE SECTION ステートメントで開始されます。

BEGIN DECLARE SECTION と END DECLARE SECTION ステートメントは、対にして使用する必要があります、ネストすることはできません。

ホスト変数の宣言は、SQL INCLUDE ステートメントを使用して指定することができます。それ以外の場合、ホスト変数の宣言セクションに、ホスト変数の宣言以外のステートメントを含めることはできません。

REXX 以外のホスト言語では、SQL ステートメントで参照されるホスト変数をホスト変数宣言セクションで宣言しなければなりません。また、各変数の宣言は、その変数を最初に参照する個所よりも前にある必要があります。

宣言セクションの外部で宣言される変数の名前を、宣言セクションで宣言されている変数と同じ名前にすることはできません。

---

## EXECUTE IMMEDIATE

EXECUTE IMMEDIATE ステートメントは、以下のことを行います。

- 文字ストリング形式の SQL ステートメントから、実行可能形式の SQL ステートメントを準備します。
- その SQL ステートメントを実行します。

EXECUTE IMMEDIATE の機能は、PREPARE ステートメントと EXECUTE ステートメントの基本的な機能の組み合わせです。このステートメントは、ホスト変数もパラメーター・マーカーも含まれていない SQL ステートメントを準備し実行する場合に使用することができます。

## 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

## 許可

指定された SQL ステートメントに定義されているのと同じ許可規則が適用されます。

ステートメントの許可 ID は、DYNAMICRULES BIND オプションの影響を受けることがあります。

## 構文

▶▶—EXECUTE IMMEDIATE—*host-variable*————▶▶

## 説明

### *host-variable*

ホスト変数の指定は必須であり、文字ストリング変数の宣言規則に従ってプログラムに記述されたホスト変数を指定していなければなりません。これは、最大のステートメント・サイズの 2097152 バイトより小さい文字ストリング変数でなければなりません。CLOB(2097152)には最大のステートメント・サイズを含めることができますが、VARCHARには含めることができませんので注意してください。指定するホスト変数の値は、ステートメント・ストリングと呼ばれます。

ステートメント・ストリングは、以下のいずれかの SQL ステートメントでなければなりません。

- ALTER
- CALL
- COMMENT
- COMMIT
- CREATE
- DECLARE GLOBAL TEMPORARY TABLE
- DELETE
- DROP
- GRANT
- INSERT
- LOCK TABLE
- REFRESH TABLE
- RELEASE SAVEPOINT
- RENAME TABLE
- RENAME TABLESPACE
- REVOKE
- ROLLBACK
- SAVEPOINT
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ENCRYPTION PASSWORD

- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- UPDATE

ステートメント・ストリングには、パラメーター・マーカ―やホスト変数への参照を含めてはなりません。また EXEC SQL で始まってはなりません。ステートメント終止符を含めることはできません。ただし、CREATE TRIGGER および CREATE PROCEDURE ステートメントは例外です。CREATE TRIGGER ステートメントには、トリガーによって実行される SQL ステートメントを区切るために、セミコロン (;) を含めることができます。CREATE PROCEDURE ステートメントには、SQL プロシージャの本体で SQL ステートメントを区切るためにセミコロンを含めることができます。CALL ステートメントで指定されるプロシージャには、OUT または INOUT パラメーターがあってはなりません (SQLSTATE 07007)。

EXECUTE IMMEDIATE ステートメントを実行すると、指定したステートメント・ストリングの構文解析が行われ、エラーの有無が検査されます。その SQL ステートメントが無効である場合それは実行されず、実行を妨げているエラー条件が SQLCA に報告されます。SQL ステートメントが有効で、その実行の過程でエラーが発生した場合、エラー条件が SQLCA に報告されます。

## 注

- ステートメントのキャッシュは、EXECUTE IMMEDIATE ステートメントの動作に影響を与えます。

## 例

C プログラム・ステートメントを使用して SQL ステートメントをホスト変数 qstring (char[80]) に入れ、そのホスト変数 qstring に入れられた SQL ステートメントを作成および実行します。

```
if (strcmp(accounts,"BIG") == 0)
 strcpy (qstring,"INSERT INTO WORK TABLE SELECT *
 FROM EMP_ACT WHERE ACTNO < 100");
else
 strcpy (qstring,"INSERT INTO WORK TABLE SELECT *
 FROM EMP_ACT WHERE ACTNO >= 100");
.
.
EXEC SQL EXECUTE IMMEDIATE :qstring;
```

---

## EXECUTE

EXECUTE ステートメントは、準備済み SQL ステートメントを実行します。

## 呼び出し

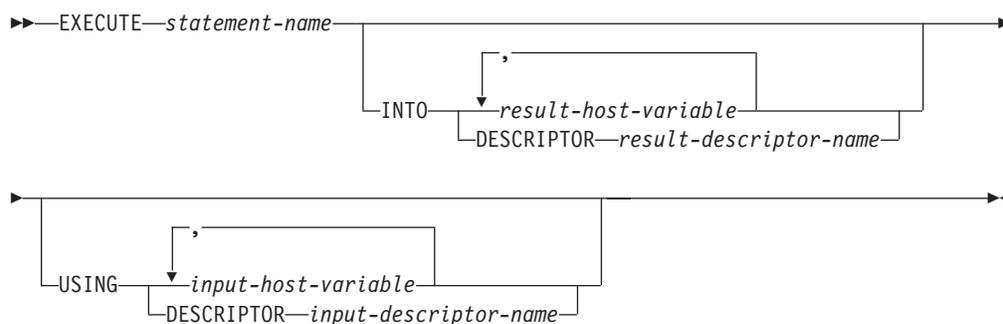
このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

## 許可

ステートメントの実行時に許可検査が行われるステートメント (DDL、GRANT、および REVOKE ステートメント) の場合、このステートメントの許可 ID の特権には、PREPARE ステートメントで指定されている SQL ステートメントを実行するための特権が含まれていなければなりません。ステートメントの許可 ID は、DYNAMICRULES BIND オプションの影響を受けることがあります。

許可検査がステートメントの準備の時点で行われるステートメント (DML) の場合、このステートメントを使用するために必要な権限はありません。

## 構文



## 説明

### *statement-name*

実行する準備済みのステートメントを指定します。 *statement-name* (ステートメント名) はすでに準備済みのステートメントを指定していなければならず、またそのステートメントが SELECT ステートメントであってはなりません。

### INTO

この後に、準備済みステートメントの出力パラメーター・マーカ (疑問符 (?)) から値を受け取るために使用される、ホスト変数のリストを指定します。

動的 CALL ステートメントの場合は、プロシージャに対する OUT および INOUT 引数に使用されるパラメーター・マーカは、出力パラメーター・マーカです。ステートメントに出力パラメーター・マーカを使用する場合は、INTO 節を指定する必要があります (SQLSTATE 07007)。

### *result-host-variable, ...*

ホスト変数の宣言規則に従って、該当プログラムで宣言されているホスト変数を指定します。変数の数は、準備されるステートメントの出力パラメーター・マーカの数と同じでなければなりません。 *n* 番目の変数は、準備済みステートメントの *n* 番目のパラメーター・マーカに対応します。場合によっては、ロケータ変数とファイル参照変数も、パラメーター・マーカの宛先として指定できます。

**DESCRIPTOR** *result-descriptor-name*

出力 **SQLDA** を指定します。その内容は、ホスト変数についての有効な記述でなければなりません。

**EXECUTE** ステートメントが処理される前に、ユーザーは、入力 **SQLDA** の以下のフィールドを設定する必要があります。

- **SQLDA** に用意する **SQLVAR** のエレメント数を示す **SQLN**
- **SQLDA** に割り振るストレージのバイト数を示す **SQLDABC**
- ステートメントの処理時にその **SQLDA** の使用される変数の数を示す **SQLD**
- 変数の属性を示す **SQLVAR** のオカレンス

**SQLDA** には、すべての **SQLVAR** オカレンスが入るだけの十分なストレージがなければなりません。

**LOB** または構造化データ・タイプの実出力データを入れる必要がある場合には、各パラメーター・マーカごとに 2 つの **SQLVAR** 項目が必要になります。

**SQLD** に設定する値は、ゼロ以上で **SQLN** 以下でなければなりません。

**USING**

この後に、準備済みステートメントの入力パラメーター・マーカ (疑問符 (?)) に置き換わる値を含むホスト変数のリストを指定します。

動的 **CALL** ステートメントの場合、プロシージャに対する **IN** および **INOUT** 引数に使用されるパラメーター・マーカは、入力パラメーター・マーカです。その他のすべての動的ステートメントの場合、すべてのパラメーター・マーカは入力パラメーター・マーカです。ステートメントに出力パラメーター・マーカを使用する場合は、**USING** 節を指定する必要があります (**SQLSTATE 07004**)。

*input-host-variable, ...*

ホスト変数の宣言規則に従って、該当プログラムで宣言されているホスト変数を指定します。変数の数は、準備されるステートメントの入力パラメーター・マーカの数と同じでなければなりません。 *n* 番目の変数は、準備済みステートメントの *n* 番目のパラメーター・マーカに対応します。場合によっては、ロケータ変数とファイル参照変数も、パラメーター・マーカの値のソースとして指定できます。

**DESCRIPTOR** *input-descriptor-name*

入力 **SQLDA** を指定します。その内容は、ホスト変数についての有効な記述でなければなりません。

**EXECUTE** ステートメントが処理される前に、ユーザーは、入力 **SQLDA** の以下のフィールドを設定する必要があります。

- **SQLDA** に用意する **SQLVAR** のエレメント数を示す **SQLN**
- **SQLDA** に割り振るストレージのバイト数を示す **SQLDABC**
- ステートメントの処理時にその **SQLDA** の使用される変数の数を示す **SQLD**
- 変数の属性を示す **SQLVAR** のオカレンス

SQLDA には、すべての SQLVAR オカレンスが入るだけの十分なストレージがなければなりません。したがって、SQLDABC の値は  $16 + \text{SQLN} * (\text{N})$  以上でなければなりません (N は 1 つの SQLVAR オカレンスの長さ)。

LOB または構造化データ・タイプの入力データを入れる必要がある場合には、各パラメーター・マーカごとに 2 つの SQLVAR 項目が必要になります。

SQLD に設定する値は、ゼロ以上で SQLN 以下でなければなりません。

## 注

- 準備済みステートメントを実行する前に、各入力パラメーター・マーカはそれに対応するホスト変数の値によって置き換えられます。型付きパラメーター・マーカの場合、ターゲット変数の属性は CAST 指定によって指定されます。タイプなしパラメーター・マーカの場合、ターゲット変数の属性はパラメーター・マーカのコンテキストに従って決定されます。

V は、パラメーター・マーカ P に対応する入力ホスト変数を表します。V の値は、列への値の割り振り規則に従って、P のターゲット変数に割り当てられます。したがって、

- V はターゲットと互換でなければなりません。
- V がストリングの場合、その長さはターゲットの長さ属性を超えることはできません。
- V が数値の場合、V の整数部分の絶対値はターゲットの整数部分の絶対値の最大を超えることはできません。
- V の属性がターゲットの属性と同一でない場合、その値はターゲットの属性に合うように変換されます。

準備済みステートメントを実行すると、P の代わりに使用される値は P のターゲット変数になります。たとえば、V が CHAR(6) でターゲットが CHAR(8) の場合、P の代わりに使用される値は V の値にブランクを 2 個付加したものになります。

- 動的 CALL ステートメントの場合は、準備済みステートメントの実行後は、OUT および INOUT の各引数の戻り値は、引数に使用された出力パラメーター・マーカに対応するホスト変数に割り当てられます。型付きパラメーター・マーカの場合、ターゲット変数の属性は CAST 指定によって指定されます。タイプなしパラメーター・マーカの場合、ターゲット変数の属性は、プロシージャのパラメーターの定義によって指定されます。

V は、パラメーター・マーカ P に対応する出力ホスト変数を表し、プロシージャの引数 A に使用されます。A の値は、列から値を検索するための規則に従って V に割り当てられます。したがって、

- V は A と互換でなければなりません。
- V がストリングの場合、その長さは A の長さより短いものであってはなりません。そうでないと A の値は切り捨てられます。
- V が数値の場合、V の整数部分の最大絶対値は A の整数部分の絶対値より小さいものであってはなりません。

- V の属性が A の属性と同一でない場合、A の値は V の属性に合うように変換されます。
- **動的 SQL ステートメント・キャッシング:** 動的および静的 SQL ステートメントの実行に必要な情報は、静的 SQL ステートメントが最初に参照された時点、または動的 SQL ステートメントが最初に準備された時点で、データベース・パッケージ・キャッシュに入れられます。この情報は、無効になるか、キャッシュ・スペースが他のステートメントで必要になるか、またはデータベースがシャットダウンされるまでは、パッケージ・キャッシュに存続します。

SQL ステートメントが実行または準備される場合に、要求を出したアプリケーションに関連するパッケージ情報が、システム・カタログからパッケージ・キャッシュにロードされます。個々の SQL ステートメントの実際の実行可能セクションもキャッシュに入れられます。静的 SQL セクションは、該当のステートメントが最初に参照された時点で、システム・カタログから読み取られてパッケージ・キャッシュに入れられ、動的 SQL セクションは作成後にキャッシュに直接入れられます。動的 SQL セクションは、PREPARE や EXECUTE IMMEDIATE などの明示的なステートメントによって作成されます。一度作成された動的 SQL ステートメントのセクションが、スペース管理のために削除された場合や、環境の変化によって無効になった場合に、システムによるステートメントの暗黙的な準備によって、再作成されることがあります。

各 SQL ステートメントは、データベース・レベルでキャッシュされ、アプリケーション間で共有できます。静的 SQL ステートメントは、同じパッケージを使用してアプリケーション間で共有されます。動的 SQL ステートメントは、同じコンパイル環境と、厳密に同じステートメント・テキストを使用してアプリケーション間で共有されます。アプリケーションによって発行される各 SQL ステートメントのテキストは、アプリケーションにローカルにキャッシュされ、暗黙的な準備が必要な場合に使用されます。アプリケーション・プログラム中の各 PREPARE ステートメントは、1 つのステートメントをキャッシュできます。アプリケーション・プログラム中のすべての EXECUTE IMMEDIATE ステートメントは、同じスペースを共用し、これらの EXECUTE IMMEDIATE ステートメントに対しては、キャッシュされるステートメントは同時に 1 つしか存在しません。それぞれ異なる SQL ステートメントに対して、同じ PREPARE またはいずれかの EXECUTE IMMEDIATE ステートメントが何度も発行される場合は、最後のステートメントだけがキャッシュに入れられ、再使用の対象になります。キャッシュの使用を最適化するには、アプリケーションの開始時に多くの異なる PREPARE ステートメントを一度に発行し、その後必要に応じて EXECUTE または OPEN ステートメントを発行することです。

動的 SQL ステートメントのキャッシングを使用すると、ステートメントを一度作成すれば、ステートメントを再度準備しなくても複数の作業単位にわたってステートメントを再使用できます。環境が変わった場合には、必要に応じてシステムはステートメントを再コンパイルします。

以下のイベントは、次の PREPARE、EXECUTE、EXECUTE IMMEDIATE、または OPEN の要求時に、キャッシュされた動的ステートメントが暗黙的に準備される原因となる環境またはデータ・オブジェクトの変更の例です。

- ALTER FUNCTION
- ALTER METHOD

## EXECUTE

- ALTER NICKNAME
- ALTER PROCEDURE
- ALTER SERVER
- ALTER TABLE
- ALTER TABLESPACE
- ALTER TYPE
- CREATE FUNCTION
- CREATE FUNCTION MAPPING
- CREATE INDEX
- CREATE METHOD
- CREATE PROCEDURE
- CREATE TABLE
- CREATE TEMPORARY TABLESPACE
- CREATE TRIGGER
- CREATE TYPE
- DROP (すべてのオブジェクト)
- 表または索引の RUNSTATS
- ビューが作動不能になる原因となるすべてのアクション
- システム・カタログ表の統計の UPDATE
- SET CURRENT DEGREE
- SET PATH
- SET QUERY OPTIMIZATION
- SET SCHEMA
- SET SERVER OPTION

キャッシュに入れられる動的 SQL ステートメントから予想される動作の概略は、以下のようになります。

- *PREPARE* 要求: 以後同じステートメントの準備に、セクションが有効であればステートメントのコンパイルのコストがかかりません。現在キャッシュに入れているセクションのコストとカーディナリティーの見積もりが戻されます。それらの値は、同じ SQL ステートメントに対するそれより前の *PREPARE* から戻される値とは違う場合があります。 *COMMIT* または *ROLLBACK* ステートメントの後に *PREPARE* ステートメントを発行する必要はありません。
- *EXECUTE* 要求: 元の *PREPARE* 以後にステートメントが無効になった場合に、ステートメントを暗黙的に準備するコストが *EXECUTE* ステートメントにかかる場合があります。セクションが暗黙的に準備される場合、当初の *PREPARE* ステートメントの環境でなく、現行の環境が使用されます。
- *EXECUTE IMMEDIATE* 要求: 以後同じステートメントに対して *EXECUTE IMMEDIATE* ステートメントを出す際に、セクションが有効であればステートメントのコンパイルのコストがかかりません。
- *OPEN* 要求: 当初の *PREPARE* ステートメント以後にステートメントが無効になった場合、ステートメントを暗黙的に準備するコストが動的に定義されたカ



## EXECUTE

```
.
/* Check for successful execution and put values into
 :employee, :dept, and :bonus */
.
EXEC SQL EXECUTE DO_BONUS INTO :cheque_no, :bonus
 USING :employee, :dept, :bonus;
.
/* Check for successful execution and process the
 values returned in :cheque_no and :bonus */
```

## EXPLAIN

EXPLAIN ステートメントは、指定された EXPLAIN 可能ステートメントに関して選択されたアクセス・プランについての情報をキャプチャーするとともに、この情報を Explain 表に入れます。

EXPLAIN 可能ステートメント は、有効な XQuery ステートメント、または SQL ステートメント CALL、コンパウンド SQL (動的)、DELETE、INSERT、MERGE、REFRESH、SELECT、SELECT INTO、SET INTEGRITY、UPDATE、VALUES、VALUES INTO のいずれかです。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

Explain 情報を取り込むステートメントは実行されません。

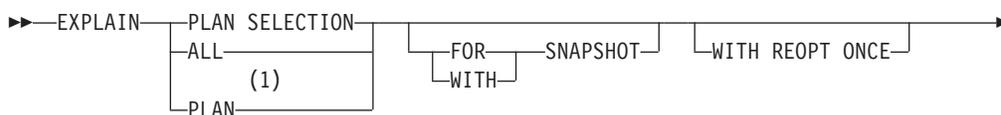
### 許可

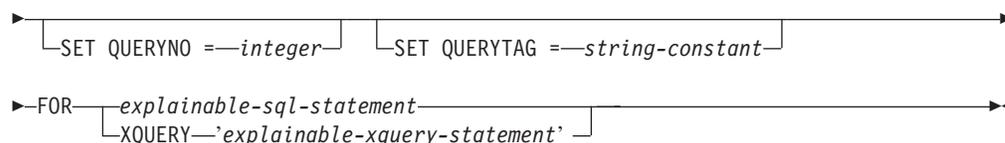
EXPLAIN ステートメントに指定された照会ステートメントに定義されているのと同じ許可規則が適用されます。例えば、DELETE ステートメントが EXPLAIN 可能ステートメントとして使用される場合、DELETE ステートメントの Explain 情報の取得時には DELETE ステートメントに関する許可規則が適用されます。

静的 EXPLAIN ステートメントの場合の許可規則は、EXPLAIN 可能ステートメントとして渡されるステートメントの静的バージョンに適用されるものと同じです。動的に準備された EXPLAIN ステートメントに関しては、EXPLAIN 可能ステートメントとして渡されるステートメントの動的準備の際に適用された許可規則が使用されます。

現行の許可 ID には、Explain 表に対する INSERT 特権が必要になります。

### 構文





注:

- 1 PLAN オプションは、DB2 for z/OS の既存の EXPLAIN ステートメントの構文を許容する目的でのみサポートされます。PLAN 表は存在しません。PLAN を指定することは、PLAN SELECTION を指定するのと同様です。

## 説明

### PLAN SELECTION

照会コンパイルのプラン選択フェーズからの情報を Explain 表に挿入することを示します。

### ALL

ALL を指定することは、PLAN SELECTION を指定するのと同様です。

### PLAN

PLAN オプションの指定によって、他のシステムからの既存データベース・アプリケーションの構文の相違を許容します。PLAN を指定することは、PLAN SELECTION を指定するのと同様です。

### FOR SNAPSHOT

この節は、Explain スナップショットだけを取り、それを EXPLAIN\_STATEMENT 表の SNAPSHOT 列に入れることを示します。EXPLAIN\_INSTANCE および EXPLAIN\_STATEMENT 表に存在するもの以外の Explain 情報はキャプチャーされません。

Explain スナップショット情報は、Visual Explain での使用を意図しています。

### WITH SNAPSHOT

この節は、通常の Explain 情報に加えて、Explain スナップショットも取ることを示します。

デフォルトでは、EXPLAIN ステートメントは通常の Explain 情報だけを収集し、Explain スナップショットは取りません。

Explain スナップショット情報は、Visual Explain での使用を意図しています。

### デフォルト (FOR SNAPSHOT も WITH SNAPSHOT も指定しない場合)

Explain 情報を Explain 表に入れます。Visual Explain での使用を目的としたスナップショットは取られません。

### WITH REOPT ONCE

この節は、WITH REOPT ONCE で EXPLAIN 可能ステートメントを再最適化するのに以前使用したホスト変数、パラメーター・マーカー、特殊レジスター、またはグローバル変数の値を使用して、このステートメントを再最適化することを指示します。Explain 表には、新しいアクセス・プランが取り込まれます。ユーザーが DBADM 権限を持っているか、データベース・レジストリー変数 DB2\_VIEW\_REOPT\_VALUES が YES に設定されている場合は、EXPLAIN\_PREDICATE 表にも値が取り込まれます (それらの値を使ってステートメントを再最適化する場合)。

**SET QUERYNO = integer**

*integer* (整数) を、EXPLAIN\_STATEMENT 表の QUERYNO 列を介して EXPLAIN 可能ステートメントに関連付けます。指定する整数値は、正の値でなければなりません。

動的 EXPLAIN ステートメントにこの節を指定しなかった場合は、デフォルト値 (1) が割り当てられます。静的 EXPLAIN ステートメントの場合には、プリコンパイラーによって割り当てられるステートメント番号がデフォルト値として割り当てられます。

**SET QUERYTAG = string-constant**

*string-constant* を、EXPLAIN\_STATEMENT 表の QUERYTAG 列を介して EXPLAIN 可能ステートメントに関連付けます。 *string-constant* には、長さ 20 バイトまでの任意の文字ストリングを指定できます。指定された値が 20 バイトに満たない場合は必要な長さに達するまで右側に空白が埋め込まれます。

EXPLAIN ステートメントにこの節を指定しなかった場合はデフォルト値として空白が使用されます。

**FOR explainable-sql-statement**

Explain 情報を取り出す SQL ステートメントを指定します。このステートメントは、有効な SQL ステートメント CALL、コンパウンド SQL (動的)、DELETE、INSERT、MERGE、REFRESH、SELECT、SELECT INTO、SET INTEGRITY、UPDATE、VALUES、VALUES INTO のいずれかです。EXPLAIN ステートメントがプログラムに組み込まれている場合には、*explainable-sql-statement* にホスト変数に対する参照を含めることができます (ただし、これらのホスト変数がプログラム内で定義されている必要があります)。同様に、EXPLAIN が動的に準備される場合には、*explainable-sql-statement* にパラメーター・マーカを含めることができます。

*explainable-sql-statement* には、EXPLAIN ステートメントによってそれぞれ個別に準備および実行された有効な SQL ステートメントを指定する必要があります。ステートメント名やホスト変数を指定することはできません。CLP を使用して定義されたカーソルを参照する SQL ステートメントを、このステートメントで使用することはできません。

アプリケーション内の動的 SQL に関する Explain 情報を取り出すためには、EXPLAIN ステートメント全体を動的に準備する必要があります。

**FOR XQUERY 'explainable-xquery-statement'**

Explain 情報を取り出す XQUERY ステートメントを指定します。このステートメントとしては、任意の有効な XQUERY ステートメントを指定できます。

EXPLAIN ステートメントをプログラムに埋め込む場合は、*'explainable-xquery-statement'* にホスト変数の参照を組み込むことができます。ただし、そのホスト変数を最上位の XQUERY ステートメントで使用するのはなく、XMLQUERY 関数の XMLEXISTS 述部によって、または XMLTABLE 関数によって渡すことが必要です。ホスト変数は、プログラムの中で定義しなければなりません。

同様に、EXPLAIN を動的に準備する場合は、*'explainable-xquery-statement'* にパラメーター・マーカを組み込むことができます。ただし、ホスト変数を渡す場合と同じ制約事項を守ることが必要です。

さらに、DB2 XQUERY 関数 `db2-fn:sqlquery` を使用して、ホスト変数やパラメーター・マーカの参照を指定した SQL ステートメントを埋め込むことも可能です。

'*explainable-xquery-statement*' には、EXPLAIN ステートメントによってそれぞれ個別に準備および実行された有効な XQUERY ステートメントを指定する必要があります。CLP を使用して定義されたカーソルを参照する照会ステートメントを、このステートメントで使用することはできません。

## 注

Explain 機能は、データの取り込み先である Explain 表を修飾するとき、スキーマとして以下の ID を使用します。

- 動的 SQL のセッション許可 ID
- 静的 SQL のステートメント許可 ID

そのスキーマは、一連の Explain 表に関連付けられている場合もあれば、別のスキーマの下で一連の Explain 表を参照する別名に関連付けられている場合もあります。そのスキーマの下で Explain 表が検出されなかった場合、Explain 機能は、SYSTOOLS スキーマの下に Explain 表があるかどうかをチェックし、その表を使用しようとしています。

次の表は、スナップショット・キーワードと Explain 情報の相互の関連を示しています。

| 指定したキーワード     | Explain 情報をキャプチャーするか? | Visual Explain 用にスナップショットを取るか? |
|---------------|-----------------------|--------------------------------|
| なし            | はい                    | いいえ                            |
| FOR SNAPSHOT  | いいえ                   | はい                             |
| WITH SNAPSHOT | はい                    | はい                             |

FOR SNAPSHOT 節と WITH SNAPSHOT 節のどちらも指定しなかった場合は、Explain スナップショットは取られません。

EXPLAIN ステートメントを呼び出す前に、Explain 表を作成しておく必要があります。このステートメントが生成した情報は、ステートメントをコンパイルした時点で指定されたスキーマにあるそれぞれの Explain 表に保管されます。

指定した EXPLAIN 可能ステートメントのコンパイル中に何らかのエラーが発生すると、Explain 表に情報は取り込まれません。

EXPLAIN 可能ステートメントについて生成されたアクセス・プランは保管されません。したがって、後から呼び出すことはできません。EXPLAIN 可能ステートメントについての Explain 情報が挿入されるのは、EXPLAIN ステートメント自体のコンパイルが正常に完了した場合です。

静的 EXPLAIN 照会ステートメントの場合、情報はバインド時および明示的な再バインド時に Explain 表に挿入されます。プリコンパイル中、静的 EXPLAIN ステートメントはコメント化され、修正済みのアプリケーション・ソース・ファイルに書き込まれます。バインド時に、EXPLAIN ステートメントは

## EXPLAIN

SYSCAT.STATEMENTS カタログに保管されます。パッケージが実行される時は、EXPLAIN ステートメントは実行されません。アプリケーション内にあるすべてのステートメントのセクション番号は連続した順序に並べられます。その中には EXPLAIN ステートメントも含まれることに注意してください。静的 EXPLAIN ステートメントを使用する代わりに、EXPLAIN と EXPLSNAP BIND または PREP オプションを組み合わせることもできます。静的 EXPLAIN ステートメントを使用することにより、数多くある静的照会ステートメントの中からただ 1 つだけ静的照会ステートメントを選び出し、そのステートメントに関する情報を Explain 表に入れることもできます。そのことは、適切な EXPLAIN ステートメント構文を指定したターゲット・ステートメントに簡単な接頭部を付け、Explain の BIND または PREP オプションのどちらも使用せずにアプリケーションをバインドすることによって行えます。実際の Explain の呼び出し時に QUERYNO または QUERYTAG フィールドを設定することが有利な場合にも、EXPLAIN ステートメントを使用することができます。

SQL プロシージャ内の静的 EXPLAIN ステートメントは、プロシージャのコンパイル時に評価されます。

追加バインド EXPLAIN 照会ステートメントの場合、Explain 表に情報が入れるのは、EXPLAIN ステートメントのコンパイルがサブミットされる時です。パッケージが実行される時は、EXPLAIN ステートメントは実行されません (ただし、ステートメントは正常終了します)。Explain 表にデータを取り込む際、Explain 表の修飾子と許可 ID には、パッケージ所有者の修飾子と許可 ID が使用されます。実際の Explain の呼び出し時に QUERYNO または QUERYTAG フィールドを設定することが有利な場合にも、EXPLAIN ステートメントを使用することができます。

動的 EXPLAIN ステートメントの場合、Explain 表に情報が入れるのは、EXPLAIN ステートメントのコンパイルがサブミットされる時です。PREPARE ステートメントを指定して EXPLAIN ステートメントを準備することもできますが、そのようにして実行しても処理は行われません (ステートメントは正常終了します)。動的 EXPLAIN ステートメントを発行する代わりに、CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターを組み合わせる使用することによっても、動的照会ステートメントの Explain 情報を取り出すことができます。実際の Explain の呼び出し時に QUERYNO または QUERYTAG フィールドを設定することが有利な場合には、EXPLAIN ステートメントを使用してください。

REOPT BIND オプションが ONCE に設定されていて、CURRENT EXPLAIN MODE または CURRENT EXPLAIN SNAPSHOT 特殊レジスターのいずれかが REOPT に設定されている場合、ホスト変数、特殊レジスター、パラメーター・マーカー、またはグローバル変数を含む静的および動的照会ステートメントを実行すると、ステートメントが再最適化される時のみ、ステートメントの Explain 情報がキャプチャーされます。一方、REOPT BIND オプションが ALWAYS に設定されている場合は、それらのステートメントが実行されるたびに Explain 情報がキャプチャーされます。

**例**

例 1: 単純な SELECT ステートメントの Explain 情報を取り出し、QUERYNO = 13 のタグを付けます。

```
EXPLAIN PLAN SET QUERYNO = 13
FOR SELECT C1
FROM T1
```

例 2: 単純な SELECT ステートメントの Explain 情報を取り出し、QUERYTAG = 'TEST13' のタグを付けます。

```
EXPLAIN PLAN SELECTION SET QUERYTAG = 'TEST13'
FOR SELECT C1
FROM T1
```

例 3: 単純な SELECT ステートメントの Explain 情報を取り出し、QUERYNO = 13 および QUERYTAG = 'TEST13' のタグを付けます。

```
EXPLAIN PLAN SELECTION SET QUERYNO = 13 SET QUERYTAG = 'TEST13'
FOR SELECT C1
FROM T1
```

例 4: Explain 表が存在しない場合に、Explain 情報の入手を試みます。

```
EXPLAIN ALL FOR SELECT C1
FROM T1
```

このステートメントは失敗します。Explain 表が定義されていないからです (SQLSTATE 42704)。

例 5: 以下のステートメントがパッケージ・キャッシュ内に見出され、REOPT ONCE を使ってすでにコンパイルされている場合に、ステートメントは成功します。

```
EXPLAIN ALL WITH REOPT ONCE FOR SELECT C1
FROM T1
WHERE C1 = :<host variable>
```

例 6: 以下の例では、db2-fn:xmlcolumn 関数を使用します。この関数は、XML 列の大文字/小文字の区別のある名前を引数として取り、XML 列の値の連結である XML シーケンスを戻します。

BUSINESS.CUSTOMER という表に、INFO という XML 列があるとします。INFO 列からすべての文書を戻す簡単な XQuery は、以下のとおりです。

```
EXPLAIN PLAN SELECTION
FOR XQUERY 'db2-fn:xmlcolumn ("BUSINESS.CUSTOMER.INFO")'
```

列の値が NULL の場合、その行の結果として生成される戻りシーケンスは空になります。

---

**FETCH**

FETCH ステートメントは、カーソルの位置を結果表の次の行に移し、その行の値をホスト変数に割り当てます。

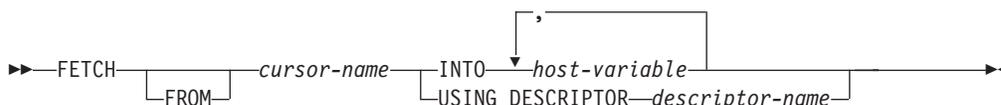
## 呼び出し

対話式 SQL 機能には外見上対話式的の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

## 許可

カーソルの使用に必要な許可については、『DECLARE CURSOR』を参照してください。

## 構文



## 説明

### *cursor-name*

フェッチ操作で使用するカーソルを指定します。『DECLARE CURSOR』で説明されているように、*cursor-name* は、宣言済みカーソルを指定しなければなりません。ソース・プログラムにおいて、FETCH ステートメントより前に DECLARE CURSOR ステートメントがなければなりません。FETCH ステートメントを実行する場合、該当のカーソルはオープン状態でなければなりません。

そのカーソルの位置が、現在その結果表の最終行またはそれ以降にある場合、

- SQLCODE は +100 に設定され、SQLSTATE は '02000' に設定されます。
- カーソルは最終行の後に位置づけられます。
- ホスト変数に値は割り当てられません。

ある行より前に現在カーソルが位置している場合、カーソルはその行に再配置され、INTO または USING で指定されたホスト変数に値が割り当てられます。

最終行以外の行に現在カーソルが位置している場合、カーソルは次の行に再配置され、その行の値は INTO または USING で指定されたホスト変数に割り当てられます。

### INTO *host-variable*, ...

1 つまたは複数のホスト変数を指定します。そのホスト変数は、ホスト変数の宣言規則に従って記述されていなければなりません。結果行の最初の値はリスト中の最初のホスト変数、その次の値は 2 番目のホスト変数、以下同様に割り当てられます。選択リストの LOB 値は、正規のホスト変数 (十分な大きさの場合)、ロケータ変数、またはファイル参照変数に割り当てることができます。

### USING DESCRIPTOR *descriptor-name*

ゼロ個以上のホスト変数の有効な記述を含む SQLDA を識別します。

FETCH ステートメントが処理される前に、ユーザーは次に示す SQLDA 内のフィールドを設定する必要があります。

- SQLDA に用意する SQLVAR のエレメント数を示す SQLN

- SQLDA に割り振るストレージのバイト数を示す SQLDABC
- ステートメントの処理時にその SQLDA の使用される変数の数を示す SQLD
- 変数の属性を示す SQLVAR のオカレンス

SQLDA には、すべての SQLVAR オカレンスが入るだけの十分なストレージがなければなりません。したがって、SQLDABC の値は  $16 + \text{SQLN} * (\text{N})$  以上でなければなりません (N は 1 つの SQLVAR オカレンスの長さ)。

LOB または構造化タイプの結果列を入れるには、各選択リスト項目 (または結果表の列) ごとに 2 つの SQLVAR 項目が必要です。

SQLD に設定する値は、ゼロ以上で SQLN 以下でなければなりません。

INTO 節で指定されるか、または SQLDA に記述される  $n$  番目の変数は、カーソルの結果表の  $n$  番目の列に対応します。各変数のデータ・タイプは、それに対応する列と互換性がなければなりません。

各変数には、特定の規則に従って値が割り当てられます。変数の数とその行の値の数よりも少ない場合、SQLDA の SQLWARN3 フィールドが 'W' に設定されます。変数の数が結果表の列の数よりも多い場合、警告は出されません。割り当てエラーが発生すると、値は変数に割り当てられず、値はそれ以上変数に割り当てられません。それまでにすでに変数に割り当てられていた値はそのままになります。

## 注

- オープン・カーソルの位置として、3 つの位置が考えられます。
  - 行の前
  - 行の上
  - 最終行のあと
- カーソル位置が行にある場合、その行はカーソルの現在行と呼ばれます。UPDATE ステートメントまたは DELETE ステートメントでカーソルを参照する場合、そのカーソル位置は行でなければなりません。カーソルの現在行となるのは、FETCH ステートメントの結果としての行だけです。
- 複数の FETCH を通じてロケータを維持する必要がない場合、LOB ロケータへの取り出しを行う場合には、ロケータ・リソースの限度を考慮して、その次の FETCH ステートメントを発行する前に FREE LOCATOR ステートメントを発行しておくといでしょう。
- エラーが発生したことによって、カーソルの状態が予測できないものになることがあります。
- 警告が FETCH に戻されなかったり、以前取り出された行に対する警告が戻されたりする場合があります。これらの問題は、システム一時表やプッシュダウン演算子を使用するような最適化によって生じる場合があります。
- ステートメントのキャッシュは、EXECUTE IMMEDIATE ステートメントの動作に影響を与えます。
- DB2 CLI は追加の取り出し機能をサポートしています。たとえば、カーソルの結果表が読み取り専用の場合に、SQLFetchScroll() 関数を使用してその結果表の中の任意のスポットにカーソルを位置づけることができます。
- 更新可能カーソルの場合は、行が取り出される時に行でロックが取得されます。

## FETCH

- カーソル定義に SQL データ変更ステートメントが含まれている場合や、SQL データを変更するルーチンの呼び出しが関係している場合は、フェッチ操作中にエラーが発生し、エラーによってカーソルがクローズされることがあっても、変更された行がロールバックされることはありません。

### 例

例 1: この C の例では、FETCH ステートメントは SELECT ステートメントの結果を取り出して、プログラム変数 `dnum`、`dname`、および `mnum` に入れます。取り出す行がなくなった場合、見つからないことを示す状態が戻されます。

```
EXEC SQL DECLARE C1 CURSOR FOR
 SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
 WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

while (SQLCODE==0) {
 EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;
}

EXEC SQL CLOSE C1;
```

例 2: この FETCH ステートメントは SQLDA を使用しています。

```
FETCH CURS USING DESCRIPTOR :sqlda3
```

---

## FLUSH EVENT MONITOR

FLUSH EVENT MONITOR ステートメントは、イベント・モニター *event-monitor-name* に関連付けられたすべてのアクティブ・モニター・タイプの現行のデータベース・モニター値を、イベント・モニターの I/O ターゲットに書き込みます。このため、レコードの生成頻度が低いイベント・モニター (データベース・イベント・モニターなど) で、いつでも部分イベント・レコードを使用することができます。こうしたレコードには、イベント・モニターのログで、部分レコード ID が付けられます。

イベント・モニターがフラッシュされると、そのモニターのアクティブな内部バッファが、イベント・モニターの出力オブジェクトに書き込まれます。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

## 構文

```
▶▶—FLUSH—EVENT—MONITOR—event-monitor-name—┬───┬───▶
 | |
 └─┬─┘
 |
 BUFFER
```

## 説明

*event-monitor-name*

イベント・モニターの名前。これは、1 部構成の名前です。これは、通常 ID です。

### BUFFER

イベント・モニターのバッファを書き出すことを示します。BUFFER を指定すると、部分レコードは生成されません。イベント・モニターのバッファにすでに入っているデータだけが書き出されます。

## 注

- イベント・モニターをフラッシュアウトしても、イベント・モニター値はリセットされません。これはつまり、フラッシュが実行されない場合に生成されていたイベント・モニターのレコードが、通常のモニター・イベントが起動される時にもやはり生成されるということです。

---

## FLUSH OPTIMIZATION PROFILE CACHE

同じ最適化プロファイルを使用して、複数のステートメントをコンパイルすることができます。最適化プロファイル処理をより効率的にするため、初めて最適化プロファイルを使ってステートメントを最適化するときに最適化プロファイルの処理が行われ、出力が最適化プロファイル・キャッシュに保管されます。最適化プロファイルへの後続の参照では、最適化プロファイル・キャッシュ内にある処理済みのバージョンが使用されます。

SYSTOOLS.OPT\_PROFILE に保管されているバージョンが更新されたときには、最適化プロファイル・キャッシュから最適化プロファイルを除去する必要があります。古いバージョンがキャッシュから除去されると、最適化プロファイルを使用する後続のステートメントの最適化の際に新規のバージョンが使用されます。

## 呼び出し

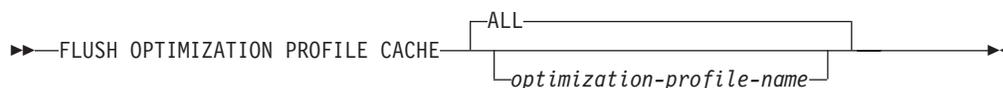
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM のいずれかの権限が含まれている必要があります (SQLSTATE 42502)。

## 構文

## FLUSH OPTIMIZATION PROFILE CACHE



### 説明

#### *optimization-profile-name*

最適化プロファイル・キャッシュからフラッシュされる最適化プロファイルの名前を指定します。指定した名前が修飾されていない場合、CURRENT DEFAULT SCHEMA レジスターの値が暗黙的な修飾子として使用されます。

#### ALL

すべてのアクティブなデータベース・パーティションのすべてのプロファイルが最適化プロファイル・キャッシュからフラッシュされることを指定します。

### 注

- FLUSH OPTIMIZATION PROFILE CACHE ステートメントは、最適化プロファイル・キャッシュからすべての最適化プロファイルまたは単一の最適化プロファイルを除去します。さらに、その最適化プロファイルを使用して準備されて、キャッシュに入れられたすべての動的 SQL ステートメントを論理的に無効化します。
- 無効にされた動的プランの新規アクセス・プランは、同じ SQL ステートメントが次回要求されるときに再生成されます。
- このステートメントによって最適化プロファイル・キャッシュから除去された最適化プロファイルを参照するパッケージは、新規アクセス・プランを生成できるよう、明示的に再度バインドする必要があります。

### 例

例 1: 最適化プロファイル "Rick"."Foo" が最適化プロファイル・キャッシュからフラッシュされます。

```
SET CURRENT SCHEMA = 'Rick'
FLUSH OPTIMIZATION PROFILE CACHE "Foo"
```

例 2: 最適化プロファイル JOHN.ALL が最適化プロファイル・キャッシュから除去されます。

```
SET CURRENT SCHEMA = 'Rick'
FLUSH OPTIMIZATION PROFILE CACHE JOHN.ALL
```

### メッセージ

- 最適化プロファイル・キャッシュが空である場合、または指定した最適化プロファイル (明示的または暗黙的に指定されているかにかかわらず) が最適化プロファイル・キャッシュに存在しない場合、エラーは発行されません。

---

## FLUSH PACKAGE CACHE

FLUSH PACKAGE CACHE ステートメントは、現在パッケージ・キャッシュ内に存在する、キャッシュされたすべての動的 SQL を除去します。このステートメントは、キャッシュされたすべての動的 SQL ステートメントを論理的に無効化し、同じ SQL ステートメントに対する次の要求が強制的に DB2 によって暗黙的にコンパイルされるようにします。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

### 構文

▶—FLUSH PACKAGE CACHE—DYNAMIC—▶

### 注

- このステートメントは、すべてのアクティブ・データベース・パーティション上のパッケージ・キャッシュ内の、キャッシュされたすべての動的 SQL 項目に影響を与えます。
- キャッシュされた動的 SQL ステートメントが無効にされると、キャッシュされた項目に使用されたパッケージ・キャッシュ・メモリーは、FLUSH PACKAGE CACHE ステートメントの実行時に項目が使用中でなければ解放されます。
- 現在使用中の、キャッシュされた動的 SQL ステートメントは、現在のユーザーが必要としなくなるまでパッケージ・キャッシュ内に存在することができます。同じステートメントの次の新しいユーザーは、DB2 によるステートメントの暗黙作成を強制するので、そのユーザーはキャッシュされた動的 SQL ステートメントの新しいバージョンを実行することになります。

---

## FOR

FOR ステートメントは、表の行ごとに、ステートメントまたはステートメントのグループを実行します。

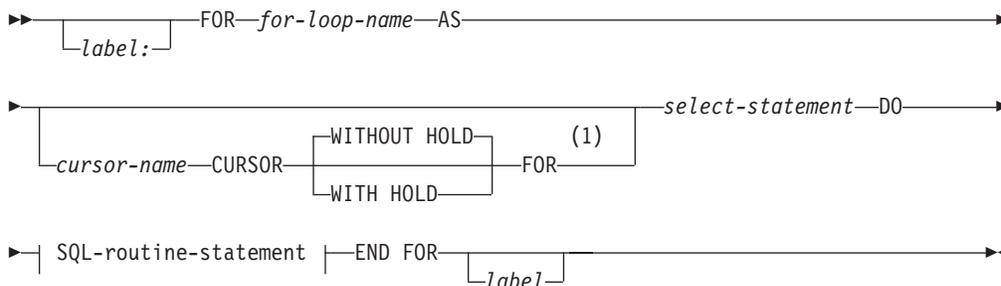
### 呼び出し

このステートメントは、SQL プロシージャまたは動的コンパウンド・ステートメントに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

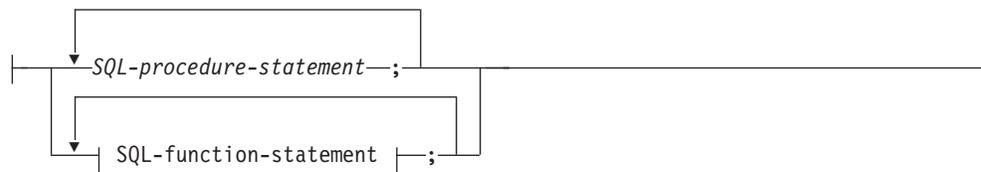
## 許可

FOR ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID には、FOR ステートメントに組み込まれている SQL ステートメントを呼び出すために必要な特権がなければなりません。カーソルの使用に必要な許可については、『DECLARE CURSOR』を参照してください。

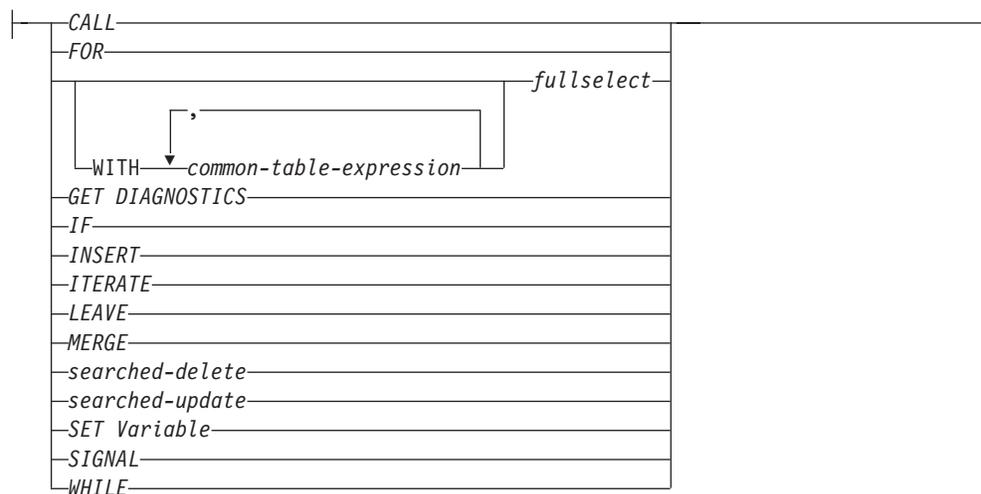
## 構文



### SQL-routine-statement:



### SQL-function-statement:



### 注:

- このオプションは、SQL プロシージャのコンテキスト内でのみ使用できません。

## 説明

### *label*

FOR ステートメントのラベルを指定します。開始ラベルが指定された場合、そのラベルは LEAVE および ITERATE ステートメントで使用できます。終了ラベルを指定する場合、そのラベルは開始ラベルと同じでなければなりません。

### *for-loop-name*

FOR ステートメントをインプリメントするために生成された暗黙的コンパウンド・ステートメントのラベルを指定します。FOR ステートメント内の ITERATE および LEAVE ステートメントで使用できないことを除いては、コンパウンド・ステートメントのラベルの規則に従います。*for-loop-name* は、指定された *select-statement* によって返された列名を修飾するために使用します。

### *cursor-name*

SELECT ステートメントの結果表から行を選択するために使用されるカーソルを指定します。指定しない場合は、DB2 がユニークなカーソル名を生成します。WITHOUT HOLD または WITH HOLD の説明については、「DECLARE CURSOR」を参照してください。

### *select-statement*

カーソルの SELECT ステートメントを指定します。選択リスト内の列にはすべて名前がなければならず、同じ名前の列が 2 つあってはいけません。

トリガー、関数、メソッド、または動的コンパウンド・ステートメントでは、*select-statement* はオプションで共通表式を持つ *fullselect* のみから構成されていなければなりません。

### *SQL-procedure-statement*

表の各行に対して呼び出すステートメントを 1 つ以上指定します。

*SQL-procedure-statement* は、SQL プロシーチャーのコンテキスト内でのみ使用できます。『コンパウンド SQL (プロシーチャー)』で、*SQL-procedure-statement* を参照してください。

### *SQL-function-statement*

表の各行に対して呼び出すステートメントを 1 つ以上指定します。ニックネームに対する searched-update (検索済み更新)、searched-delete (検索済み削除)、または INSERT 操作はサポートされていません。*SQL-function-statement* は、SQL 関数または SQL メソッドのコンテキスト内でのみ使用できます。

## 規則

- 選択リストはユニークな列名から構成されていることが必要で、選択リストで指定された表はプロシーチャーが作成されたときには存在していなければなりません。そうでなければ、これは前の SQL プロシーチャー・ステートメントで作成された表でなければなりません。
- *for-statement* で指定されたカーソルは、*for-statement* の外側では参照できず、OPEN、FETCH、または CLOSE ステートメントでは指定できません。

## 例

下の例では、*for-statement* は *employee* 表全体を繰り返すために使用されています。表の中の行ごとに、SQL 変数 *fullname* は、従業員のラストネーム (姓)、コン

## FOR

マ、ファーストネーム (名)、ブランク・スペース、そしてミドルネームのイニシャルという順序で設定されます。 `fullname` の各値は、表 `tnames` に挿入されます。

```
BEGIN ATOMIC
 DECLARE fullname CHAR(40);
 FOR v1 AS
 SELECT firstnme, midinit, lastname FROM employee
 DO
 SET fullname = lastname CONCAT ' '
 CONCAT firstnme CONCAT ' ' CONCAT midinit;
 INSERT INTO tnames VALUES (fullname);
 END FOR;
END
```

---

## FREE LOCATOR

`FREE LOCATOR` ステートメントは、ロケータ変数とその値との間の関連を除去します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文

```
▶▶ FREE LOCATOR variable-name ▶▶
```

### 説明

**LOCATOR** *variable-name*, ...

1 つまたは複数のロケータ変数を指定します。それらは、ロケータ変数の宣言規則に従って宣言されていなければなりません。

ロケータ変数には、現在ロケータが割り当てられていなければなりません。つまり、ロケータはこの作業単位で割り当てられている (`CALL`、`FETCH`、`SELECT INTO`、または `VALUES INTO` ステートメントによって) 必要があり、またその後解放されて (`FREE LOCATOR` ステートメントによって) いないことが必要です。これに違反する場合には、エラーが戻されます (`SQLSTATE 0F001`)。

複数のロケータを指定すると、リスト中の他のロケータにエラーがあるか否かには関係なく、解放可能なすべてのロケータは解放されることになります。

### 例

COBOL プログラムで、`BLOB` ロケータ変数 `TKN-VIDEO` と `TKN-BUF`、および `CLOB` ロケータ変数 `LIFE-STORY-LOCATOR` を解放します。



## GET DIAGNOSTICS

### condition-information

以前に実行した SQL ステートメントのエラー情報または警告情報が戻されることを指定します。エラーについての情報が必要であれば、GET DIAGNOSTICS ステートメントは、エラーを処理するハンドラーに指定された最初のステートメントでなければなりません。警告についての情報が必要で、ハンドラーが警告条件を制御している場合は、GET DIAGNOSTICS ステートメントは、そのハンドラーに指定された最初のステートメントでなければなりません。ハンドラーが警告条件を制御していない場合、GET DIAGNOSTICS ステートメントは、実行される次のステートメントでなければなりません。このオプションは、SQL プロシージャのコンテキスト内でのみ指定できます (SQLSTATE 42601)。

### MESSAGE\_TEXT

以前に実行された SQL ステートメントから戻されたエラーまたは警告メッセージ・テキストを示します。このメッセージ・テキストは、そのステートメントが処理されたデータベース・サーバーの言語で戻されます。ステートメントがゼロの SQLCODE で完了した場合、VARCHAR 変数には空ストリングが戻され、CHAR 変数にはブランクが戻されます。

### DB2\_TOKEN\_STRING

以前に実行された SQL ステートメントから戻されたエラーまたは警告メッセージ・トークンを示します。ステートメントがゼロの SQLCODE で完了したか、SQLCODE にトークンがない場合、VARCHAR 変数には空ストリングが戻され、CHAR 変数にはブランクが戻されます。

## 注

- GET DIAGNOSTICS ステートメントは、診断域の内容の変更は行いません (SQLCA)。SQLSTATE または SQLCODE 特殊変数が SQL プロシージャで宣言されている場合、GET DIAGNOSTICS ステートメントの発行によって戻される SQLSTATE または SQLCODE に設定されます。
- **互換性**
  - 以前のバージョンの DB2 との互換性:
    - DB2\_RETURN\_STATUS の代わりに RETURN\_STATUS を指定できます。

## 例

SQL プロシージャで GET DIAGNOSTICS ステートメントを実行し、更新された行数を判別します。

```
CREATE PROCEDURE sqlprocg (IN deptnbr VARCHAR(3))
LANGUAGE SQL
BEGIN
 DECLARE SQLSTATE CHAR(5);
 DECLARE rcount INTEGER;
 UPDATE CORPDATA.PROJECT
 SET PRSTAFF = PRSTAFF + 1.5
 WHERE DEPTNO = deptnbr;
 GET DIAGNOSTICS rcount = ROW COUNT;
 -- この時点で、rcount には更新済みの行数が入っています。
 ...
END
```

SQL プロシージャ内で TRYIT というプロシージャの呼び出しから戻される状況値を処理します。この呼び出しでは、ユーザー障害を示す正の値が明示的に戻さ

れるか、あるいは SQL エラーが発生して負の戻り状況値が戻されます。プロシージャが成功すると、ゼロの値が戻されます。

```
CREATE PROCEDURE TESTIT ()
LANGUAGE SQL
A1:BEGIN
 DECLARE RETVAL INTEGER DEFAULT 0;
 ...
 CALL TRYIT;
 GET DIAGNOSTICS RETVAL = DB2_RETURN_STATUS;
 IF RETVAL <> 0 THEN
 ...
 LEAVE A1;
 ELSE
 ...
 END IF;
END A1
```

---

## GOTO

GOTO ステートメントは、SQL プロシージャ内のユーザー定義ラベルに分岐させます。

### 呼び出し

このステートメントは、SQL プロシージャに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

### 許可

必要ありません。

### 構文

▶▶ GOTO *label* ◀◀

### 説明

*label*

処理を続行するラベル付きステートメントを指定します。このラベル付きステートメントと GOTO ステートメントの有効範囲は同じでなければなりません。

- GOTO ステートメントが FOR ステートメントで定義されている場合、*label* は同じ FOR ステートメントの内側で定義しなければなりません。ただし、ネストされている FOR ステートメントまたはネストされているコンパウンド・ステートメントは除きます。
- GOTO ステートメントがコンパウンド・ステートメントで定義されている場合、*label* は同じコンパウンド・ステートメントの内側で定義しなければなりません。ただし、ネストされている FOR ステートメントまたはネストされているコンパウンド・ステートメントは除きます。
- GOTO ステートメントがハンドラーで定義されている場合、*label* は他の有効範囲の規則に従って、同じハンドラーで定義しなければなりません。

## GOTO

- GOTO ステートメントがハンドラーの外側で定義されている場合、*label* をハンドラーの内側で定義してはなりません。

*label* が、GOTO ステートメントが到達できる有効範囲内で定義されていない場合、エラーが戻されます (SQLSTATE 42736)。

### 注

- GOTO ステートメントは使い過ぎないようにお勧めします。このステートメントは通常の処理シーケンスを妨げるので、ルーチンの読み取りおよび保守が困難になります。なるべく GOTO ステートメントを使用しなくて済むように、GOTO ステートメントを使用する前に、他のステートメント (IF や LEAVE など) を代わりに使用できるかどうか判別してください。

### 例

以下のコンパウンド・ステートメントでは、パラメーター *rating* および *v\_empno* がプロシージャに渡されます。そして、日付期間として出力パラメーター *return\_parm* が戻されます。従業員のその会社での就労期間が 6 カ月未満の場合、GOTO ステートメントは制御をプロシージャの最後に移動させ、*new\_salary* は未変更のままになります。

```
CREATE PROCEDURE adjust_salary
 (IN v_empno CHAR(6),
 IN rating INTEGER)
 OUT return_parm DECIMAL (8,2))
MODIFIES SQL DATA
LANGUAGE SQL
BEGIN
 DECLARE new_salary DECIMAL (9,2)
 DECLARE service DECIMAL (8,2)
 SELECT SALARY, CURRENT_DATE - HIREDATE
 INTO new_salary, service
 FROM EMPLOYEE
 WHERE EMPNO = v_empno
 IF service < 600
 THEN GOTO EXIT
 END IF
 IF rating = 1
 THEN SET new_salary = new_salary + (new_salary * .10)
 ELSE IF rating = 2
 THEN SET new_salary = new_salary + (new_salary * .05)
 END IF
 UPDATE EMPLOYEE
 SET SALARY = new_salary
 WHERE EMPNO = v_empno
 EXIT: SET return_parm = service
END
```

---

## GRANT (データベース権限)

この形式の GRANT ステートメントは、データベース全体に適用される権限 (データベース内の特定のオブジェクトに適用される特権ではなく) を付与します。

### 呼び出し

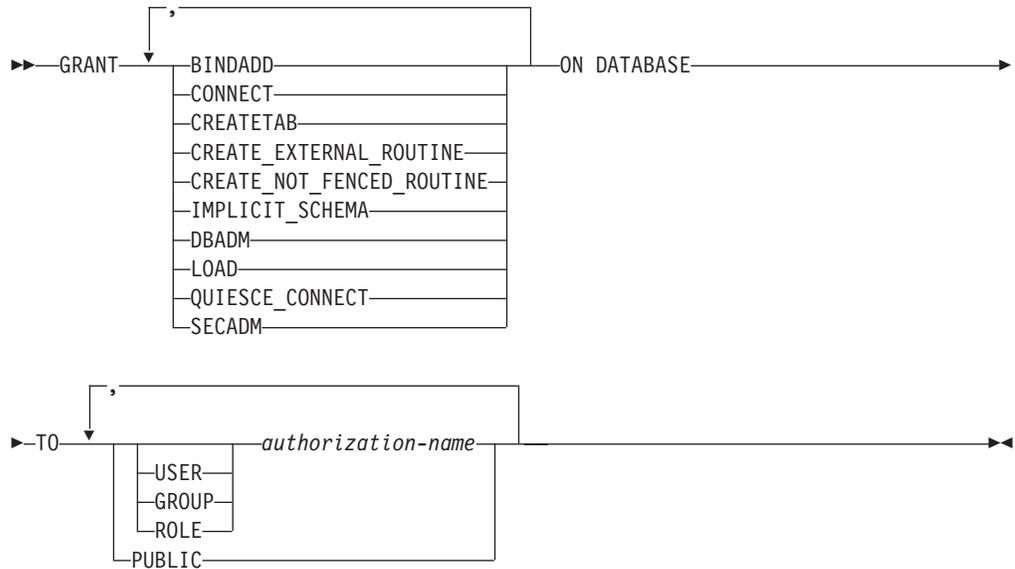
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の

実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

## 許可

DBADM 権限または SECADM 権限を付与するには、SYSADM が必要です。その他の権限を付与するには、DBADM 権限、または SYSADM 権限のいずれかが必要です。

## 構文



## 説明

### BINDADD

パッケージを作成する権限を付与します。パッケージの作成者には自動的にそのパッケージに対する CONTROL 特権が与えられ、後で BINDADD 権限が取り消されたとしてもその特権はそのまま保持されます。

### CONNECT

データベースにアクセスする権限を与えます。

### CREATETAB

基本表を作成する権限を付与します。基本表の作成者には、自動的にその表に対する CONTROL 特権が与えられます。後で CREATETAB 権限が取り消されたとしても、作成者はこの特権を保持したままになります。

ビュー作成に必要な明示的な権限は特にありません。ビューの作成に使用するステートメントの許可 ID に各ビューの基本表に対する CONTROL 特権または SELECT 特権のいずれかが与えられている場合には、いつでもビューを作成できます。

### CREATE\_EXTERNAL\_ROUTINE

外部ルーチンを登録する権限を付与します。そのようにして登録されたルーチン

## GRANT (データベース権限)

が不利な副次作用を引き起こすことがないように注意してください。(詳しくは、CREATE または ALTER ルーチン・ステートメント上の THREADSAFE 節の説明を参照してください。)

外部ルーチンを登録し終わると、後で CREATE\_EXTERNAL\_ROUTINE が取り消されても、そのまま保持されます。

### CREATE\_NOT\_FENCED\_ROUTINE

データベース・マネージャーの処理の中で実行するルーチンを登録する権限を与えます。そのようにして登録されたルーチンが不利な副次作用を引き起こすことがないように注意してください。(詳しくは、CREATE または ALTER ルーチン・ステートメント上の FENCED 節の説明を参照してください。)

ルーチンが非 fenced として登録された場合は、それ以降に CREATE\_NOT\_FENCED が取り消されたとしてもその方式での実行が続けられます。

CREATE\_NOT\_FENCED\_ROUTINE 権限を付与される *authorization-name* には、自動的に CREATE\_EXTERNAL\_ROUTINE が付与されます。

### IMPLICIT\_SCHEMA

スキーマを暗黙的に作成する権限を与えます。

### DBADM

データベース管理者権限や、他のすべてのデータベース権限を付与します (セキュリティ管理者権限 (SECADM) を除く)。データベース管理者は、データベース内のほとんどすべてのオブジェクトに対するほとんどすべての特権を保持します。唯一の例外は、セキュリティ管理者権限に含まれている特権です。

データベース管理者は、データベース管理者権限に含まれているすべての特権を他者に付与できます。

DBADM 権限が与えられている *authorization-name* には、SECADM 以外のすべてのデータベース権限が暗黙的かつ自動的に与えられます。

### LOAD

このデータベースでロードを実行する権限を付与します。この権限を付与されたユーザーは、このデータベースにおいて LOAD ユーティリティを使用する権利を持ちます。この権限は、デフォルトで SYSADM と DBADM にも付与されます。ただし、LOAD 権限しか付与されていないユーザー (SYSADM と DBADM 以外) の場合は、表レベルでの特権が別に必要になります。LOAD 特権に加えて、ユーザーは以下の特権を付与されていなければなりません。

- モード INSERT、TERMINATE (直前の LOAD INSERT を終了するため)、または RESTART (直前の LOAD INSERT を再び開始するため) で LOAD を実行する場合は、その表に対する INSERT 特権。
- モード REPLACE、TERMINATE (直前の LOAD REPLACE を終了するため)、または RESTART (直前の LOAD REPLACE を再び開始するため) で LOAD を実行する場合は、その表に対する INSERT および DELETE 特権。
- LOAD の一部として例外表を使用する場合は、その表に対する INSERT 特権。

### QUIESCE\_CONNECT

静止中のデータベースにアクセスする権限を与えます。

**SECADM**

セキュリティ管理者権限を与えます。SECADM 権限を付与できるのは、ユーザーに対してだけです。グループ、ロール (SQLSTATE 42521)、または PUBLIC (SQLSTATE 42508) に与えることはできません。この権限の保有者は、以下の操作を実行できます。

- セキュリティー・オブジェクト (監査ポリシー、ロール、セキュリティ・ラベル、セキュリティ・ラベル・コンポーネント、セキュリティ・ポリシー、トラステッド・コンテキストなど) の作成とドロップ
  - ロール、セキュリティ・ラベルおよび免除の付与と取り消し
  - SETSESSIONUSER 特権の付与および取り消し
  - 他者が所有しているオブジェクトに対する TRANSFER OWNERSHIP の実行
- 他のデータベース権限とは異なり、SYSADM 権限の保有者は上記の操作を実行できません。

**TO**

権限を誰に与えるかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループ名であることを指定します。

**ROLE**

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

**PUBLIC**

一連のユーザー (許可 ID) に権限を付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。DBADM は、PUBLIC に付与することはできません。

**規則**

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。

## GRANT (データベース権限)

- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティー・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。

### 注

- DBADM 権限を特殊グループ PUBLIC に付与することはできません。したがって、*role-name* が直接的または間接的に PUBLIC に与えられている場合には、ロール *role-name* への DBADM 権限の付与は失敗します (SQLSTATE 42508)。
  - 以下のステートメントが発行済みの場合、ロール *role-name* は PUBLIC に直接的に付与されます。

```
GRANT ROLE role-name TO PUBLIC
```

- 以下のステートメントが発行済みの場合、ロール *role-name* は間接的に PUBLIC に付与されます。

```
GRANT ROLE role-name TO ROLE role-name2
GRANT ROLE role-name2 TO PUBLIC
```

- 互換性

- 以前のバージョンの DB2 との互換性:
  - CREATE\_NOT\_FENCED\_ROUTINE の代わりに CREATE\_NOT\_FENCED を指定できます。

### 例

例 1: ユーザー WINKEN、BLINKEN、および NOD に、データベースに接続する権限を与えます。

```
GRANT CONNECT ON DATABASE TO USER WINKEN, USER BLINKEN, USER NOD
```

例 2: データベースに対する BINDADD 権限を D024 という名前のグループに与えます。システムには、D024 と呼ばれるグループとユーザーの両方が存在しています。

```
GRANT BINDADD ON DATABASE TO GROUP D024
```

GROUP キーワードの指定は必須であることに注意してください。この指定がない場合、D024 という名前のユーザーとグループが両方とも存在しているのでエラーになります。D024 グループのメンバーは、いずれもデータベースのパッケージをバインドできるようになります。しかし、D024 というユーザーにそれは許されません (ただし、このユーザーがグループ D024 のメンバーでもある場合、または以前に BINDADD 権限を与えられていた場合、または BINDADD 権限がユーザー D024 がメンバーとして属している別のグループに与えられていた場合を除きます)。

例 3: ユーザー Walid にセキュリティー管理者権限を与えます。

```
GRANT SECADM ON DATABASE TO USER Walid
```

## GRANT (免除)

この形式の GRANT ステートメントは、指定されたラベル・ベースのアクセス制御 (LBAC) セキュリティ・ポリシーに対して、アクセス規則の免除をユーザー、グループ、またはロールに付与します。免除を受けたユーザーが、そのセキュリティ・ポリシーによって保護されている表内のデータにアクセスした場合、そのユーザーがデータにアクセスできるかどうかを決定する際に、指示された規則は施行されません。

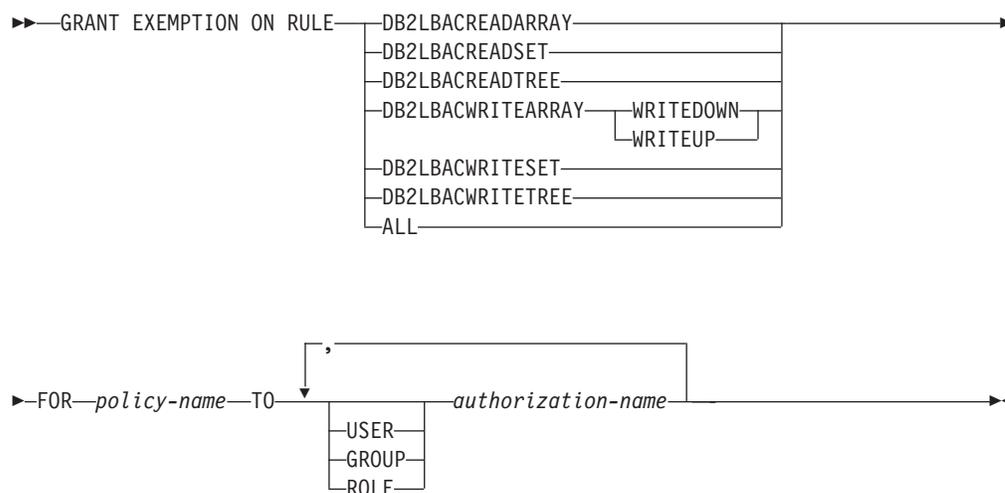
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

### 構文



### 説明

#### EXEMPTION ON RULE

アクセス規則に対する免除を付与します。

#### DB2LBACREADARRAY

事前定義された DB2LBACREADARRAY 規則に対する免除を付与します。

#### DB2LBACREADSET

事前定義された DB2LBACREADSET 規則に対する免除を付与します。

#### DB2LBACREADTREE

事前定義された DB2LBACREADTREE 規則に対する免除を付与します。

## GRANT (免除)

### DB2LBACWRITEARRAY

事前定義された DB2LBACWRITEARRAY 規則に対する免除を付与します。

### WRITEDOWN

免除が下方への書き込みにのみ適用されることを指定します。

### WRITEUP

免除が上方への書き込みにのみ適用されることを指定します。

### DB2LBACWRITESSET

事前定義された DB2LBACWRITESSET 規則に対する免除を付与します。

### DB2LBACWRITETREE

事前定義された DB2LBACWRITETREE 規則に対する免除を付与します。

### ALL

事前定義されたすべての規則に対する免除を付与します。

### FOR *policy-name*

免除の付与対象のセキュリティー・ポリシーを識別します。その免除が有効なのは、このセキュリティー・ポリシーで保護されている表に対してのみです。名前は、カタログにすでに記述されているセキュリティー・ポリシーを示すものでなければなりません (SQLSTATE 42704)。

### TO

免除を誰に付与するかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティー・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティー・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。

- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティー・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- セキュリティー・ポリシーがグループまたはロールを介してアクセスを考慮するように定義されていない場合、グループまたはロールに付与された免除はアクセスが試行されるときに無視されます。

## 注

- デフォルトでは、セキュリティー・ポリシーの作成時には、個々のユーザーに付与された免除のみが考慮されます。セキュリティー・ポリシーでグループまたはロールが考慮されるようにするには、ALTER SECURITY POLICY ステートメントを発行し、必要に応じて USE GROUP AUTHORIZATION または USE ROLE AUTHORIZATION を指定しなければなりません。

## 例

例 1: セキュリティー・ポリシー DATA\_ACCESS のアクセス規則 DB2LBACREADSET に対して、ユーザー WALID に免除を付与します。

```
GRANT EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS TO USER WALID
```

例 2: セキュリティー・ポリシー DATA\_ACCESS のアクセス規則 DB2LBACWRITEARRAY に対する免除を、WRITEDOWN オプションを指定して、ユーザー BOBBY に付与します。

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
FOR DATA_ACCESS TO USER BOBBY
```

例 3: セキュリティー・ポリシー DATA\_ACCESS のアクセス規則 DB2LBACWRITEARRAY に対する免除を、WRITEUP オプションを指定して、ユーザー BOBBY に付与します。

```
GRANT EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
FOR DATA_ACCESS TO USER BOBBY
```

---

## GRANT (グローバル変数特権)

この形式の GRANT ステートメントは、作成されたグローバル変数に 1 つ以上の特権を付与します。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

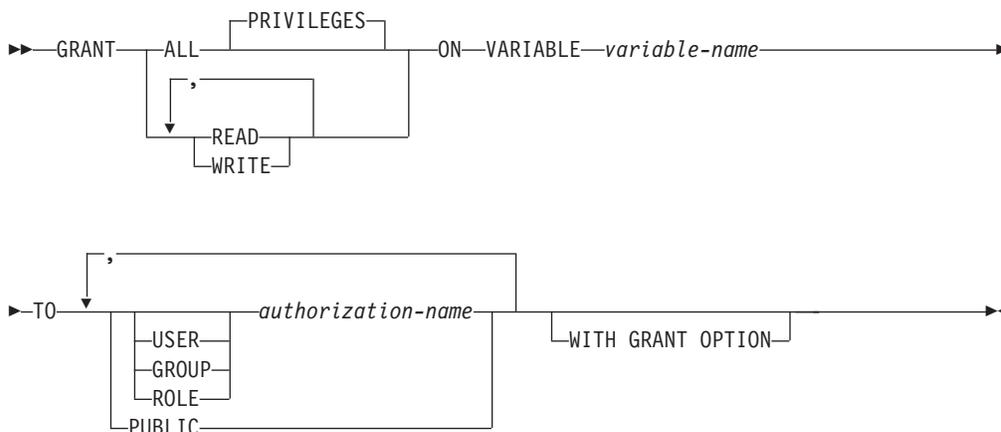
## GRANT (グローバル変数特権)

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- グローバル変数に対する指定された特権ごとに WITH GRANT OPTION
- SYSADM または DBADM 権限

### 構文



### 説明

#### ALL PRIVILEGES

指定したグローバル変数に対してすべての特権を付与します。

#### READ

指定したグローバル変数の値を読み取る特権を付与します。

#### WRITE

指定したグローバル変数に値を割り当てる特権を付与します。

#### ON VARIABLE *variable-name*

1 つ以上の特権が付与されるグローバル変数を指定します。暗黙修飾子または明示修飾子を含む *variable-name* は、現在のサーバーに存在するグローバル変数を識別するものでなければなりません (SQLSTATE 42704)。

#### TO

特権を誰に与えるかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループであることを指定します。

#### ROLE

*authorization-name* が現行サーバーにおける既存のロールを識別することを指定します (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定し

ます。この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

指定された特権をユーザー (許可 ID) の集合に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

### WITH GRANT OPTION

指定した *authorization-name* に対し、特権を他のユーザーに与えることを許可します。WITH GRANT OPTION 節を省略すると、指定した *authorization-name* は、他の何らかのソースからその権限を受け取らないかぎり、特権を他のユーザーに与えることはできません。

### 規則

- 指定した *authorization-name* ごとに、キーワード USER、GROUP、および ROLE のいずれも指定されていない場合には次のようになります。
  - インスタンスに対して有効なセキュリティー・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* が、データベースでは ROLE として定義されており、かつオペレーティング・システムでは GROUP または USER のいずれかとして定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティー・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
  - *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。

### 例

グローバル変数 MYSCHEMA.MYJOB\_PRINTER に対する READ および WRITE 特権を、ユーザー ZUBIRI に付与します。

```
GRANT READ, WRITE ON VARIABLE MYSCHEMA.MYJOB_PRINTER TO ZUBIRI
```

---

## GRANT (索引特権)

この形式の GRANT ステートメントは、索引に対する CONTROL 特権を付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の

## GRANT (索引特権)

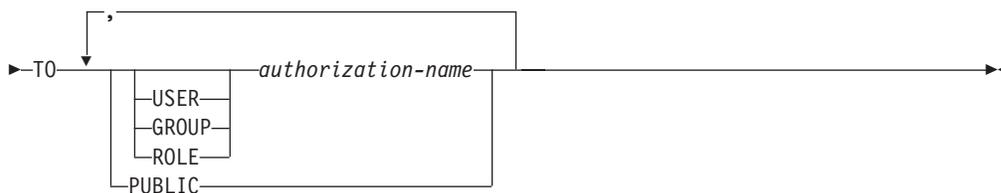
実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

### 構文

```
▶▶ GRANT CONTROL ON INDEX index-name ▶▶
```



### 説明

#### CONTROL

索引をドロップする特権を付与します。これは、索引の作成者に自動的に与えられるその索引に対する CONTROL 権限です。

#### ON INDEX *index-name*

CONTROL 特権を付与する対象となる索引の名前を指定します。

#### TO

特権を誰に与えるかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループ名であることを指定します。

#### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

#### *authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

#### PUBLIC

特権をユーザー (許可 ID) の集合に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
  - *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。

## 例

```
GRANT CONTROL ON INDEX DEPTIDX TO USER KIESLER
```

---

## GRANT (パッケージ特権)

この形式の GRANT ステートメントは、パッケージに対する特権を付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

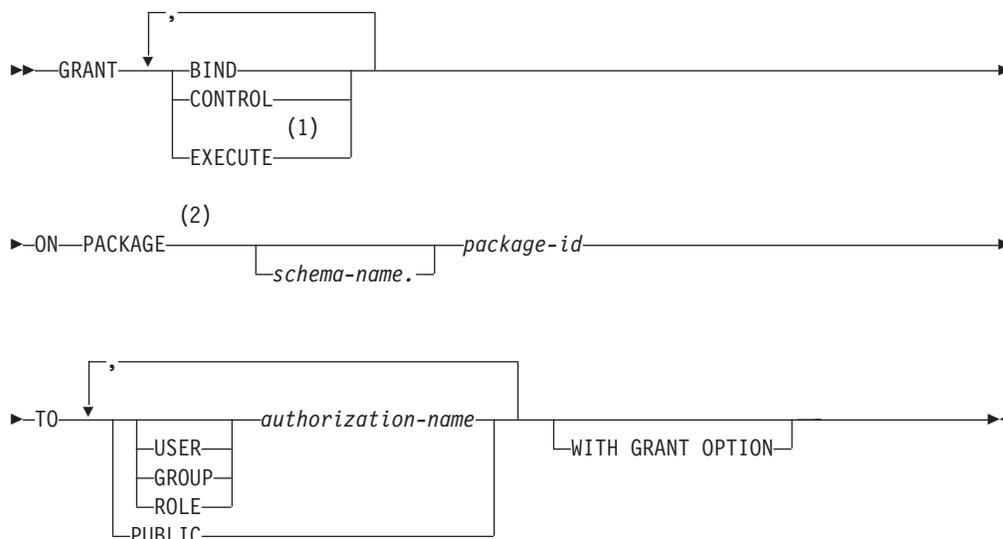
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 参照されるパッケージに対する CONTROL 特権
- *package-name* に対する指定された特権ごとに WITH GRANT OPTION
- SYSADM または DBADM 権限

SYSADM または DBADM 権限は、CONTROL 特権を付与するために必要です。

### 構文

## GRANT (パッケージ特権)



注:

- 1 EXECUTE の同義語として RUN を使用できます。
- 2 PACKAGE の同義語として PROGRAM を使用できます。

### 説明

#### BIND

パッケージをバインドする特権を付与します。BIND 特権を使用すると、ユーザーが BIND コマンドをパッケージに対して再発行したり、REBIND コマンドを発行したりできます。また、ユーザーが既存のパッケージの新しいバージョンを作成することもできます。

ユーザーには、BIND 特権に加えて、プログラムに含まれている静的 DML ステートメントによって参照される表ごとに必要な特権が与えられていなければなりません。これは、静的 DML ステートメントに対する許可がバインド時に検査されるので必要になります。

#### CONTROL

パッケージを再バインド、ドロップ、または実行するための特権、およびパッケージ特権を他のユーザーに与える特権を付与します。パッケージの作成者には、自動的にパッケージに対する CONTROL 特権が与えられます。パッケージ所有者は、パッケージ・バインド・プログラムか、またはバインド/プリコンパイル時に OWNER オプションを使って指定した ID です。

CONTROL 権限を付与される *authorization-name* には、自動的に BIND と EXECUTE が付与されます。

CONTROL は、他のユーザーに上記の特権 (CONTROL を除く) を付与する特権を付与します。

#### EXECUTE

パッケージを実行する特権を与えます。

#### ON PACKAGE *schema-name.package-id*

特権の対象となるパッケージの名前を指定します。スキーマ名が指定されていない場合、パッケージ ID は暗黙的にデフォルト・スキーマで修飾されます。パ

パッケージ特権の付与は、そのパッケージのすべてのバージョン (つまり、同一のパッケージ ID とパッケージ・スキーマを共有するすべてのパッケージ) に適用されます。

### TO

特権を誰に与えるかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

特権をユーザー (許可 ID) の集合に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

### WITH GRANT OPTION

指定した *authorization-name* に対し、特権を他のユーザーに与えることを許可します。

指定した特権に CONTROL が含まれる場合、WITH GRANT OPTION は CONTROL を除くすべての適用可能な特権に適用されます (SQLSTATE 01516)。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。

## GRANT (パッケージ特権)

- *authorization-name* がデータベースで **ROLE** としてのみ定義されている場合には、**ROLE** であると見なされます。

### 注

- パッケージ特権は、パッケージのすべてのバージョン (つまり、同一のパッケージ ID とパッケージ・スキーマを共有するすべてのパッケージ) に適用されます。アクセスを 1 つのバージョンだけに制約することはできません。  
CONTROL 特権はパッケージのバインド・プログラムに暗黙的に付与されるので、2 人のユーザーが 2 つのバージョンのパッケージをバインドすると、両者とも互いのパッケージに対するアクセス権を暗黙的に付与されます。

### 例

例 1: PACKAGE CORPDATA.PKGA に対する EXECUTE 特権を PUBLIC に与えます。

```
GRANT EXECUTE
ON PACKAGE CORPDATA.PKGA
TO PUBLIC
```

例 2: パッケージ CORPDATA.PKGA に対する EXECUTE 権限を EMPLOYEE という名前のユーザーに与えます。EMPLOYEE と呼ばれるグループもユーザーも存在していません。

```
GRANT EXECUTE ON PACKAGE
CORPDATA.PKGA TO EMPLOYEE
```

または

```
GRANT EXECUTE ON PACKAGE
CORPDATA.PKGA TO USER EMPLOYEE
```

---

## GRANT (ロール)

この形式の GRANT ステートメントはロールを、ユーザー、グループ、またはその他のロールに付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

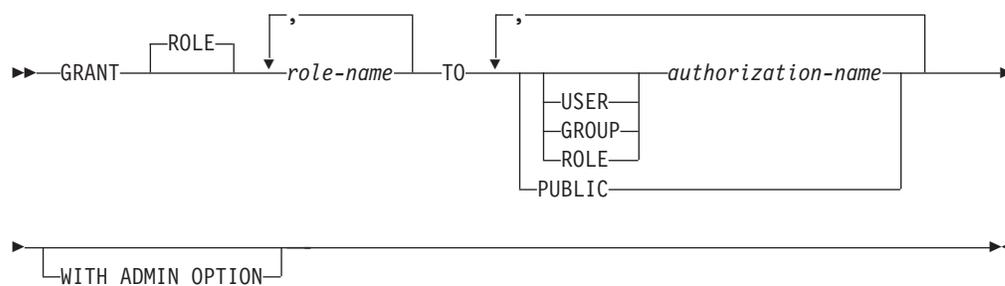
### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- ロールに対する WITH ADMIN OPTION
- SECADM 権限

SECADM 権限は、*authorization-name* に WITH ADMIN OPTION を付与するために必要です。

## 構文



## 説明

**ROLE** *role-name*,...

付与する 1 つ以上のロールを指定します。 *role-name* はそれぞれ、現行サーバーの既存のロールを識別するものでなければなりません (SQLSTATE 42704)。

**TO**

ロールを付与する対象のユーザーを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループであることを指定します。

**ROLE**

*authorization-name* が現行サーバーにおける既存のロールを識別することを指定します (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

**PUBLIC**

指定されたロールをユーザーのセット (許可 ID) に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

**WITH ADMIN OPTION**

指定された *authorization-name* が他のユーザーに対して *role-name* を付与したり、取り消したりできるようにします。また、コメントをロールに関連付けられるようにもします。指定された *authorization-name* によるロールのドロップは許可しません。

## 規則

- 指定した *authorization-name* ごとに、キーワード **USER**、**GROUP**、および **ROLE** のいずれも指定されていない場合には次のようになります。
  - インスタンスに対して有効なセキュリティー・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。

## GRANT (ロール)

- *authorization-name* が、データベースでは ROLE として定義されており、かつオペレーティング・システムでは GROUP または USER のいずれかとして定義されている場合、エラーが戻されます (SQLSTATE 56092)。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- 1 つのロールを別のロールに付与することによってロールの階層を構築できません。ただし、循環は使用できません (SQLSTATE 428GF)。例えば、ロール R1 を別のロール R2 に付与する場合はロール R2 (または R2 を含む他のロール Rn) を R1 に付与することはできません。それによって循環が生成される可能性があるからです。

### 注

- ロール R1 を別のロール R2 に付与すると、R2 に R1 が含まれることとなります。
- DBADM 権限を PUBLIC に付与することはできません。したがって、次のようになります。
  - R1 が DBADM 権限を直接または間接的に保持する場合、PUBLIC へのロール R1 の付与は失敗します (SQLSTATE 42508)。
    - 次のステートメントを発行済みの場合、ロール R1 は DBADM 権限を直接保持します。

```
GRANT DBADM ON DATABASE TO ROLE R1
```
    - 次のステートメントを発行済みの場合、ロール R1 は DBADM 権限を間接的に保持します。

```
GRANT DBADM ON DATABASE TO ROLE R2
```

```
GRANT ROLE R2 TO ROLE R1
```
  - ロール R2 が直接または間接的に PUBLIC に付与される場合、ロール R2 へのロール R1 (DBADM 権限を保持する) の付与は失敗します (SQLSTATE 42508)。
    - 次のステートメントを発行済みの場合、ロール R2 は PUBLIC に直接付与されます。

```
GRANT ROLE R2 TO PUBLIC
```
    - 次のステートメントを発行済みの場合、ロール R2 は PUBLIC に間接的に付与されます。

```
GRANT ROLE R2 TO ROLE R3
```

```
GRANT ROLE R3 TO PUBLIC
```

## 例

例 1: ロール INTERN をロール DOCTOR に、さらにロール DOCTOR をロール SPECIALIST に付与します。

```
GRANT ROLE INTERN TO ROLE DOCTOR
```

```
GRANT ROLE DOCTOR TO ROLE SPECIALIST
```

例 2: ロール INTERN を PUBLIC に付与します。

```
GRANT ROLE INTERN TO PUBLIC
```

例 3: ロール SPECIALIST をユーザー BOB とグループ TORONTO に付与します。

```
GRANT ROLE SPECIALIST TO USER BOB, GROUP TORONTO
```

## GRANT (ルーチン特権)

この形式の GRANT ステートメントは、ルーチン (関数、メソッド、またはプロシージャ) に対する特権を付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

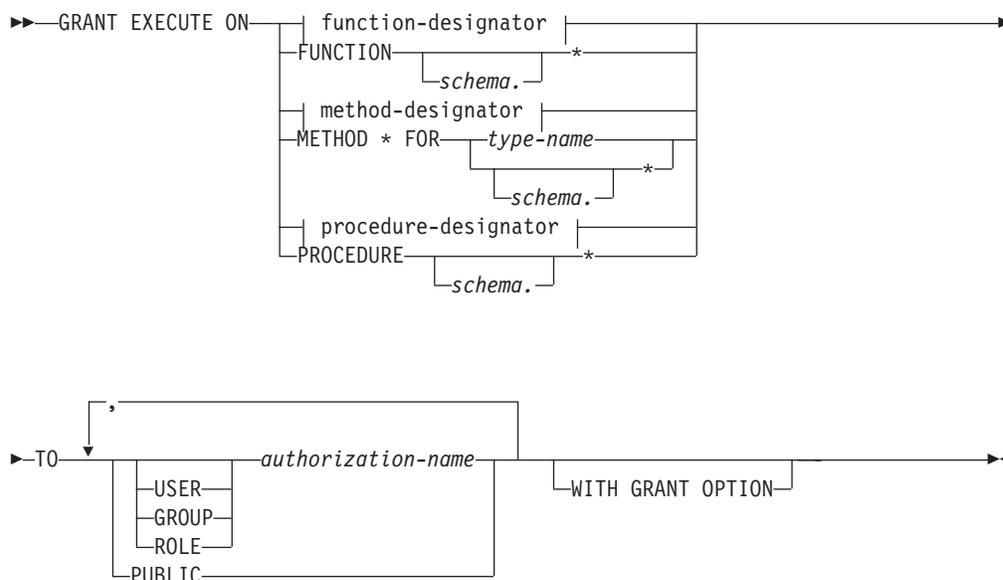
- ルーチンに対する EXECUTE の WITH GRANT OPTION
- SYSADM または DBADM 権限

特定のスキーマ中または特定のタイプのすべてのルーチン EXECUTE 特権を付与するには、ステートメントの許可 ID によって保持されている特権に少なくとも以下のいずれかが含まれていなければなりません。

- 指定されたスキーマ中の (指定されたタイプの)、すべての既存のルーチンと将来作成するルーチンに対する EXECUTE の WITH GRANT OPTION
- SYSADM または DBADM 権限

### 構文

## GRANT (ルーチン特権)



### 説明

#### EXECUTE

識別されたユーザー定義の関数、メソッド、またはプロシージャを実行する特権を付与します。

#### *function-designator*

関数を一意的に識別します。

#### **FUNCTION** *schema.\**

スキーマ中のすべての関数を識別します。将来作成される予定の関数も含まれません。動的 SQL ステートメント中でスキーマが指定されていない場合は、**CURRENT SCHEMA** 特殊レジスター中のスキーマが使用されます。静的 SQL ステートメント中でスキーマが指定されていない場合は、**QUALIFIER** プリコンパイル/**BIND** オプション中のスキーマが使用されます。

#### *method-designator*

メソッドを一意的に識別します。

#### **METHOD \***

タイプ *type-name* のすべてのメソッドを識別します。将来作成される予定のメソッドも含まれます。

#### **FOR** *type-name*

指定されたメソッドを検索する際のタイプを指定します。ここで指定される名前は、カタログにすでに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、**CURRENT SCHEMA** 特殊レジスターの値が、修飾子のないタイプ名の修飾子として使用されます。静的 SQL ステートメントでは、**QUALIFIER** プリコンパイル/**BIND** オプションによって、修飾子のないタイプ名に修飾子が暗黙指定されます。*type-name* の代わりにアスタリスク (\*) を使用して、スキーマ中のすべてのタイプを識別することもできます。これには将来作成される予定のタイプも含まれます。

*procedure-designator*

プロシージャを一意的に識別します。

**PROCEDURE** *schema.\**

スキーマ中のすべてのプロシージャを識別します。将来作成される予定のプロシージャも含まれます。動的 SQL ステートメント中でスキーマが指定されていない場合は、CURRENT SCHEMA 特殊レジスター中のスキーマが使用されます。静的 SQL ステートメント中でスキーマが指定されていない場合は、QUALIFIER プリコンパイル/ BIND オプション中のスキーマが使用されます。

**TO**

EXECUTE 特権を誰に付与するかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループ名であることを指定します。

**ROLE**

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

**PUBLIC**

一連のユーザー (許可 ID) に EXECUTE 特権を付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

**WITH GRANT OPTION**

指定した *authorization-name* に対し、EXECUTE 特権を他のユーザーに与えることを許可します。

WITH GRANT OPTION を省略すると、指定した *authorization-name* は以下のいずれかの場合にのみ、EXECUTE 特権を他のユーザーに与えることができます。

- SYSADM または DBADM 権限を持っている。
- 他のソースから EXECUTE 特権を与える許可を受けた。

**規則**

- スキーマ 'SYSIBM' または 'SYSFUN' を使って定義された関数やメソッドに対する EXECUTE 特権を付与することはできません (SQLSTATE 42832)。
- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。

## GRANT (ルーチン特権)

- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティー・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
  - *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- 一般に、GRANT ステートメントはステートメントの許可 ID が与えることを許されている特権の GRANT のみを処理し、1 つ以上の特権が与えられなかった場合は警告 (SQLSTATE 01007) を戻します。ステートメントの処理に使用されるパッケージが、LANGLEVEL を SQL92E または MIA に設定してプリコンパイルされていた場合、特権が付与されない場合には、警告が戻されます (SQLSTATE 01007)。付与者が GRANT 操作の対象に対して特権を持っていない場合、エラーが戻されます (SQLSTATE 42501)。

### 例

例 1: 関数 CALC\_SALARY に対する EXECUTE 特権をユーザー JONES に与えます。スキーマ中に CALC\_SALARY という名前の関数が 1 つだけ含まれていると想定しています。

```
GRANT EXECUTE ON FUNCTION CALC_SALARY TO JONES
```

例 2: プロシージャ VACATION\_ACCR に対する EXECUTE 特権を、現行サーバー上のすべてのユーザーに与えます。

```
GRANT EXECUTE ON PROCEDURE VACATION_ACCR TO PUBLIC
```

例 3: 関数 DEPT\_TOTALS に対する EXECUTE 特権を管理部門のアシスタントに与え、この関数に対する EXECUTE 特権を他者に付与する特権をこのアシスタントに与えます。この関数には DEPT85\_TOT という特定の名前があります。スキーマに DEPT\_TOTALS という名前の関数が複数あることを想定しています。

```
GRANT EXECUTE ON SPECIFIC FUNCTION DEPT85_TOT
TO ADMIN_A WITH GRANT OPTION
```

例 4: 関数 NEW\_DEPT\_HIRES に対する EXECUTE 特権を HR (Human Resources) に与えます。この関数には、2 つの入力パラメーターがあり、それぞれのパラメーターのタイプは INTEGER および CHAR(10) です。スキーマに NEW\_DEPT\_HIRES という名前の関数が複数あることを想定しています。

```
GRANT EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10)) TO HR
```

例 5: タイプ EMPLOYEE のメソッド SET\_SALARY に対する EXECUTE 特権をユーザー JONES に与えます。

```
GRANT EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE TO JONES
```

## GRANT (スキーマ特権)

この形式の GRANT ステートメントは、スキーマに対する特権を付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

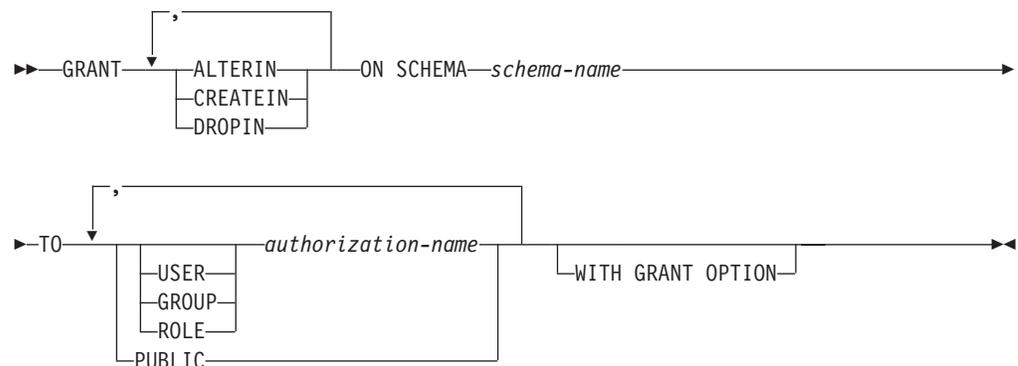
### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- *schema-name* に対する指定された特権ごとに WITH GRANT OPTION
- SYSADM または DBADM 権限

どのユーザーも、次のスキーマ名のいずれかに対する特権を付与することはできません。すなわち、SYSIBM、SYSCAT、SYSFUN、または SYSSTAT です (SQLSTATE 42501)。

### 構文



### 説明

#### ALTERIN

スキーマ内のすべてのオブジェクトの変更、またはコメント付けのための特権を与えます。明示的にスキーマを作成した所有者は、ALTERIN 特権が自動的に与えられます。

#### CREATEIN

スキーマにオブジェクトを作成する特権を与えます。オブジェクトの作成に必要なその他の権限または特権 (CREATETAB など) は、これを指定しても必要です。明示的に作成されたスキーマの所有者には、自動的に CREATEIN 特権が付与されます。暗黙的に作成されたスキーマの CREATEIN 特権は、PUBLIC に自動的に付与されます。

## GRANT (スキーマ特権)

### DROPIN

スキーマ内のオブジェクトをドロップする特権を与えます。明示的に作成されたスキーマの所有者は、DROPIN 特権を自動的に与えられます。

### ON SCHEMA *schema-name*

特権を与える対象となるスキーマを指定します。

### TO

特権を誰に与えるかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

特権をユーザー (許可 ID) の集合に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

### WITH GRANT OPTION

指定した *authorization-name* に対し、特権を他のユーザーに与えることを許可します。

WITH GRANT OPTION を省略すると、指定した *authorization-name* は以下のいずれかの場合にのみ、特権を他のユーザーに与えることができます。

- DBADM 権限を持っている
- 他のソースから特権を与える許可を受けた

### 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。

- 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- 一般に、GRANT ステートメントはステートメントの許可 ID が与えることを許されている特権の GRANT のみを処理し、1 つ以上の特権が与えられなかった場合は警告 (SQLSTATE 01007) を戻します。どのような特権も与えられなかった場合は、エラーが戻されます (SQLSTATE 42501)。(ステートメントの処理に使用されるパッケージが、LANGLEVEL を SQL92E または MIA に設定してプリコンパイルされていた場合、付与者が GRANT 操作の対象に対して特権を持っていない場合以外は警告が戻されます (SQLSTATE 01007)。)

### 例

例 1: スキーマ CORPDATA にオブジェクトを作成する特権を、ユーザー JSINGLETON に与えます。

```
GRANT CREATEIN ON SCHEMA CORPDATA TO JSINGLETON
```

例 2: スキーマ CORPDATA のオブジェクトを作成およびドロップする特権を、ユーザー IHAKES に与えます。

```
GRANT CREATEIN, DROPIN ON SCHEMA CORPDATA TO IHAKES
```

---

## GRANT (セキュリティ・ラベル)

この形式の GRANT ステートメントは、読み取りアクセス、書き込みアクセス、または読み取りアクセスと書き込みアクセスの両方に対する、ラベル・ベースのアクセス制御 (LBAC) セキュリティ・ラベルをユーザー、グループ、またはロールに認可します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

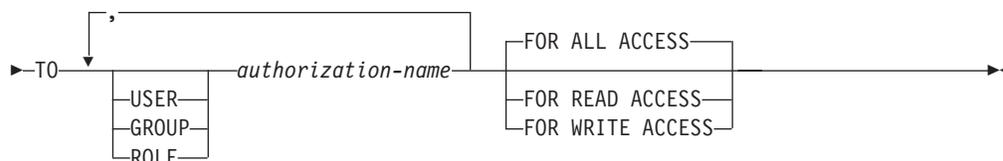
### 許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

### 構文

```
▶▶—GRANT SECURITY LABEL—security-label-name————▶▶
```

## GRANT (セキュリティー・ラベル)



### 説明

#### SECURITY LABEL *security-label-name*

セキュリティー・ラベル *security-label-name* を認可します。名前は、セキュリティー・ポリシーで修飾する必要があり (SQLSTATE 42704)、現在のサーバー上に存在するセキュリティー・ラベルを識別していなければなりません (SQLSTATE 42704)。

#### TO

指定されたセキュリティー・ラベルを付与する対象のユーザーを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループ名であることを指定します。

#### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

#### FOR ALL ACCESS

読み取りアクセスおよび書き込みアクセスの両方に対してセキュリティー・ラベルを認可することを指示します。

#### FOR READ ACCESS

読み取りアクセスに対してのみセキュリティー・ラベルを認可することを指示します。

#### FOR WRITE ACCESS

書き込みアクセスに対してのみセキュリティー・ラベルを認可することを指示します。

### 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティー・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティー・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。

- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティー・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- どのセキュリティー・ポリシーの場合でも、*authorization-name*に認可されるそのポリシーのセキュリティー・ラベルの数は、読み取りアクセスの場合は最大 1 つ、書き込みアクセスの場合は 1 つです。指示されたアクセス・タイプ (読み取りまたは書き込み) に対するセキュリティー・ラベルを被認可者がすでにもっている場合に、そのラベルが、*security-label-name* を修飾するセキュリティー・ポリシーの一部をなしていると、エラーが戻されます (SQLSTATE 428GR)。
- セキュリティー・ポリシーがグループまたはロールを介してアクセスを考慮するように定義されていない場合、グループまたはロールに付与されたセキュリティー・ラベルはアクセスが試行されるときに無視されます。
- 読み取りアクセスおよび書き込みアクセスに対してそれぞれ異なるセキュリティー・ラベルを *authorization-name* が保有している場合、それらのセキュリティー・ラベルは、以下の基準を満たす必要があります (SQLSTATE 428GQ)。
- セキュリティー・ラベル内のいずれかのコンポーネントがタイプ ARRAY である場合、そのコンポーネントの値は、両方のセキュリティー・ラベル内で同じでなければなりません。
- セキュリティー・ラベル内のいずれかのコンポーネントがタイプ SET である場合、書き込みセキュリティー・ラベル内のそのコンポーネントの値内の各エレメントは、読み取りセキュリティー・ラベル内のそのコンポーネントの値の一部でもある必要があります。
- セキュリティー・ラベル内のいずれかのコンポーネントがタイプ TREE である場合、書き込みセキュリティー・ラベル内のそのコンポーネントの値内の各エレメントは、読み取りセキュリティー・ラベル内のその同じコンポーネントの値内の同じエレメントであるか、またはエレメントのうちの 1 つの下層エレメントである必要があります。

## 注

- デフォルトでは、セキュリティー・ポリシーの作成時には、個々のユーザーに付与されたセキュリティー・ラベルのみが考慮されます。セキュリティー・ポリシーでグループまたはロールが考慮されるようにするには、ALTER SECURITY POLICY ステートメントを発行し、必要に応じて USE GROUP AUTHORIZATION または USE ROLE AUTHORIZATION を指定しなければなりません。

## 例

例 1: 以下のステートメントは、2 つのセキュリティー・ラベルをユーザー GUYLAINE に認可します。セキュリティー・ラベル EMPLOYEESECLABELREAD

## GRANT (セキュリティー・ラベル)

が読み取りアクセスに対して認可され、セキュリティー・ラベル EMPLOYEESECLABELWRITE が書き込みアクセスに対して認可されます。どちらのセキュリティー・ラベルも、セキュリティー・ポリシー DATA\_ACCESS に属します。

```
GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELREAD
TO USER GUYLAINE FOR READ ACCESS
```

```
GRANT SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABELWRITE
TO USER GUYLAINE FOR WRITE ACCESS
```

次に、同じユーザーに対して、読み取りアクセスと書き込みアクセスの両方に対するセキュリティー・ラベル BEGINNER が認可されます。これはエラーの原因にはなりません。BEGINNER はセキュリティー・ポリシー CLASSPOLICY の一部であり、すでに保有されているセキュリティー・ラベルはセキュリティー・ポリシー DATA\_ACCESS の一部であるためです。

```
GRANT SECURITY LABEL CLASSPOLICY.BEGINNER
TO USER GUYLAINE FOR ALL ACCESS
```

---

## GRANT (シーケンス特権)

この GRANT ステートメントのフォームは、シーケンスでの特権を付与します。

### 呼び出し

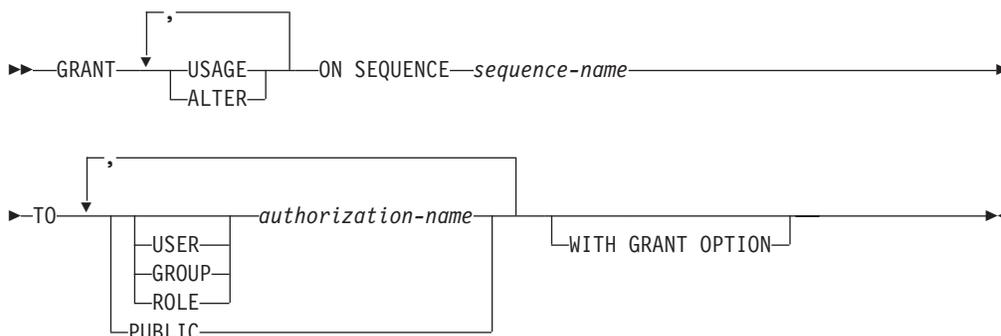
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- *sequence-name* に対する指定された特権ごとに WITH GRANT OPTION
- SYSADM または DBADM 権限

### 構文



**説明****USAGE**

*nextval-expression* または *prevval-expression* を使用してシーケンスを参照する特権を付与します。

**ALTER**

ALTER SEQUENCE ステートメントを使用してシーケンス・プロパティを変更する特権を付与します。

**ON SEQUENCE *sequence-name***

指定された特権が付与されるシーケンスを識別します。暗黙的または明示的スキーマ修飾子を含むシーケンス名は、現在のサーバーに存在するシーケンスを固有に識別していなければなりません。この名前によるシーケンスが存在しない場合、エラー (SQLSTATE 42704) が戻されます。

**TO**

指定された特権を誰に与えるかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループ名であることを指定します。

**ROLE**

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

**PUBLIC**

指定された特権をユーザー (許可 ID) の集合に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

**WITH GRANT OPTION**

指定した *authorization-name* に対して、指定した特権を他のユーザーに与えることを許可します。

WITH GRANT OPTION を省略すると、指定した *authorization-name* は以下のいずれかの場合のみ、指定された特権を他のユーザーに与えることができます。

- SYSADM または DBADM 権限を持っている。
- 他のソースから、指定された特権を与える許可を受けた。

**規則**

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。

## GRANT (シーケンス特権)

- *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
  - *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- 一般に、GRANT ステートメントはステートメントの許可 ID が与えることを許されている特権の GRANT のみを処理し、1 つまたは複数の特権が与えられていない場合は警告 (SQLSTATE 01007) を戻します。どの特権も与えられていない場合は、エラーが戻されます (SQLSTATE 42501)。ステートメントの処理に使用されるパッケージが、LANGLEVEL を SQL92E または MIA に設定してプリコンパイルされていた場合、付与者が GRANT 操作の対象に対して特権を持っていない場合以外は警告が戻されます (SQLSTATE 01007)。

### 例

例 1: シーケンス ORG\_SEQ での USAGE 特権をユーザーに付与します。

```
GRANT USAGE ON SEQUENCE ORG_SEQ TO PUBLIC
```

例 2: ユーザー BOBBY に、GENERATE\_ID というシーケンスを変更する許可と、この特権を他のユーザーに付与する許可を与えます。

```
GRANT ALTER ON SEQUENCE GENERATE_ID TO BOBBY WITH GRANT OPTION
```

---

## GRANT (サーバー特権)

この形式の GRANT ステートメントは、指定したデータ・ソースにパススルー・モードでアクセスおよび使用する特権を付与します。

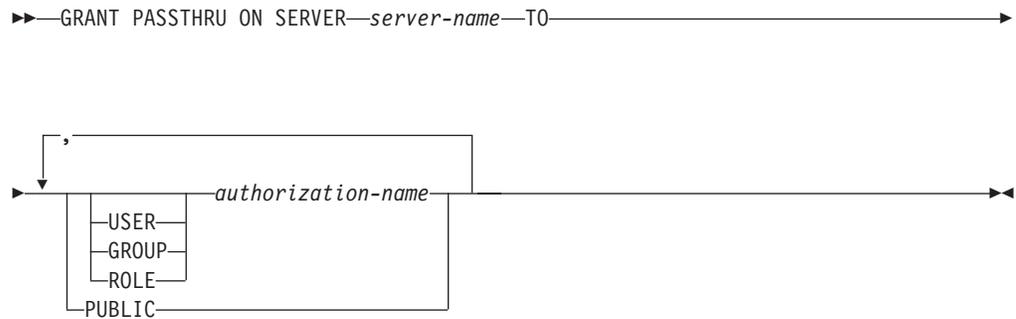
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

## 構文



## 説明

### *server-name*

パススルー・モードで使用する特権が与えられるデータ・ソースを指定します。  
*server-name* (サーバー名) は、カタログに記述されているデータ・ソースを指定していません。

### TO

特権を誰に付与するかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループ名であることを指定します。

#### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

### *authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

#### PUBLIC

*server-name* にパススルーする特権を一連のユーザー (許可 ID) に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。

## GRANT (サーバー特権)

- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
- 有効になっているセキュリティー・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティー・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。

### 例

例 1: R. Smith および J. Jones に、データ・ソース SERVALL にパススルーする特権を付与します。この 2 人の許可 ID は RSMITH および JJONES です。

```
GRANT PASSTHRU ON SERVER SERVALL
TO USER RSMITH,
USER JJONES
```

例 2: データ・ソース EASTWING にパススルーする特権を、許可 ID が D024 のグループに付与します。許可 ID が D024 であるユーザーも存在しています。

```
GRANT PASSTHRU ON SERVER EASTWING TO GROUP D024
```

GROUP キーワードの指定は必須です。この指定がない場合、D024 という名前のユーザーとグループが両方とも存在しているため、エラーになります (SQLSTATE 56092)。グループ D024 のメンバーはすべて、EASTWING にパススルーすることができます。また、ユーザー D024 がこのグループに所属する場合、このユーザーは EASTWING にパススルーすることができます。

---

## GRANT (SETSESSIONUSER 特権)

この形式の GRANT ステートメントは、1 つ以上の許可 ID に対する SETSESSIONUSER 特権を付与します。この特権の場合、その保有者は、SET SESSION AUTHORIZATION ステートメントを使用して、指定された一連の許可 ID のうちの 1 つに対してセッション許可を設定することができます。

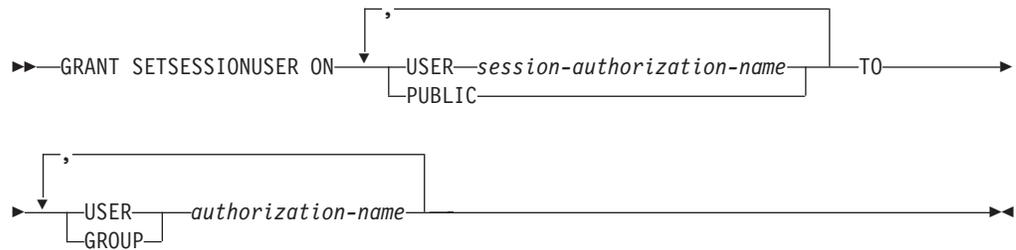
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

## 構文



## 説明

**SETSESSIONUSER ON**

新しい許可 ID を担うための特権を付与します。

**USER** *session-authorization-name*

SET SESSION AUTHORIZATION ステートメントによって *authorization-name* が担うことのできる許可 ID を指定します。*session-authorization-name* は、グループではなく、ユーザーを指定する必要があります。

**PUBLIC**

SET SESSION AUTHORIZATION ステートメントによって被認可者が任意の許可 ID を担えることを指定します。

**TO**

特権を誰に付与するかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループであることを指定します。

*authorization-name,...*

1 人または複数のユーザーまたはグループの許可 ID をリストします。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

## 規則

- 指定した *authorization-name* ごとに、USER または GROUP のどちらも指定されない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。





## GRANT (表スペース特権)

### 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
  - *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。

### 例

例 1: ユーザー BOBBY に、表スペース PLANS に表を作成する許可と、この特権を他のユーザーに付与する許可を与えます。

```
GRANT USE OF TABLESPACE PLANS TO BOBBY WITH GRANT OPTION
```

---

## GRANT (表、ビュー、またはニックネーム特権)

この形式の GRANT ステートメントは、表、ビュー、またはニックネームに対する特権を付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

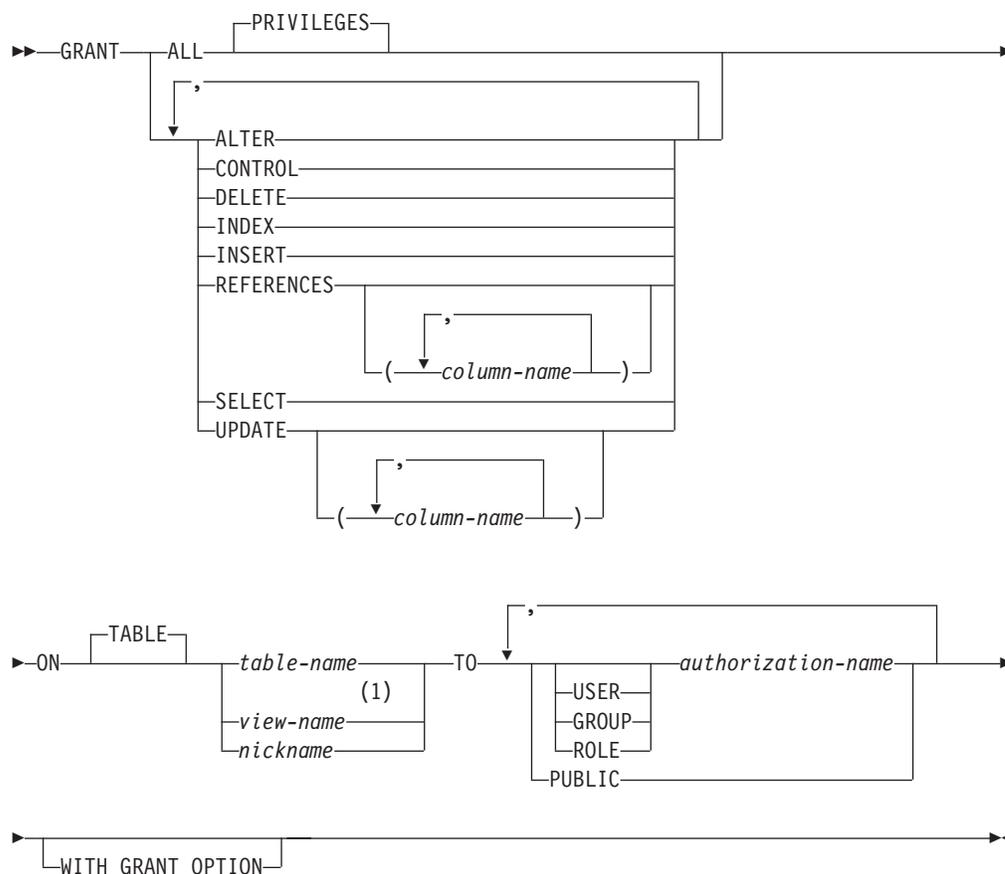
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 参照されている表、ビュー、またはニックネームに対する CONTROL 特権
- 指定したそれぞれの特権に対する WITH GRANT OPTION。ALL を指定する場合、許可 ID は指定した表、ビュー、またはニックネームに対して何らかの付与可能な特権を持っている必要があります。
- SYSADM または DBADM 権限

## GRANT (表、ビュー、またはニックネーム特権)

DBADM または SYSADM 権限は、CONTROL 特権を付与するため、またはカタログ表およびビューに対する特権を付与するために必要です。

### 構文



### 注:

- 1 ALTER、INDEX、および REFERENCES 特権は、ビューには適用されません。

### 説明

#### ALL または ALL PRIVILEGES

ON 節で指定する基本表、ビュー、またはニックネームについて、該当するすべての特権 (CONTROL を除く) を付与します。

ステートメントの許可 ID が表、ビュー、またはニックネームに対して CONTROL 特権を持っている場合、あるいは DBADM 権限または SYSADM 権限を持っている場合には、オブジェクトに適用できる特権のすべて (CONTROL を除く) が与えられます。そうでない場合、与えられる特権は、ステートメントの許可 ID が指定の表、ビュー、またはニックネームに対して持っているすべての付与可能な特権です。

ALL の指定がない場合、特権のリストに示されているキーワードの 1 つまたは複数指定する必要があります。

## GRANT (表、ビュー、またはニックネーム特権)

### ALTER

以下のことを行うための特権を付与します。

- 基本表の定義に列を追加する。
- 基本表の主キー制約またはユニーク制約を作成またはドロップする。
- 基本表の外部キーを作成またはドロップする。

親表のそれぞれの列に対する REFERENCES 特権も必要です。

- 基本表のチェック制約を作成またはドロップする。
- 基本表のトリガーを作成する。
- ニックネームの列オプションを追加、リセット、またはドロップする。
- ニックネームの列名またはデータ・タイプを変更する。
- 基本表、またはニックネームのコメントを追加または変更する。

### CONTROL

以下を付与します。

- リストに示されているすべての特権。すなわち、
  - 基本表に対する ALTER、CONTROL、DELETE、INSERT、INDEX、REFERENCES、SELECT、および UPDATE
  - ビューに対する CONTROL、DELETE、INSERT、SELECT、および UPDATE
  - ニックネームに対する ALTER、CONTROL、INDEX、および REFERENCES
- 他のユーザーに上記の特権 (CONTROL を除く) を付与する特権。
- 基本表、ビュー、またはニックネームをドロップする特権。

CONTROL 特権があっても、この特権を他のユーザーに拡張することはできません。拡張するための唯一の方法は、CONTROL 特権を付与することであり、それは SYSADM または DBADM の権限を持つユーザーのみが行うことができます。

- 表と索引に対して RUNSTATS ユーティリティーを実行する特権。
- 表に対して REORG ユーティリティーを実行する特権。
- 基本表、マテリアライズ照会表、またはステージング表に対して SET INTEGRITY ステートメントを発行する特権。

基本表、マテリアライズ照会表、ステージング表、またはニックネームの定義者には、自動的に CONTROL 特権が付与されます。

ビューの定義者に全選択で指定されているすべての表、ビュー、およびニックネームに対する CONTROL 特権が与えられている場合、その定義者には自動的に CONTROL 特権が付与されます。

### DELETE

表または更新可能なビューから行を削除する特権を付与します。

### INDEX

表の索引、またはニックネームの SPECIFICATION ONLY 指定の索引を作成する特権を付与します。この特権は、ビューに対して付与することはできません。索引または SPECIFICATION ONLY 指定の索引の作成者には、その索引ま

## GRANT (表、ビュー、またはニックネーム特権)

たは SPECIFICATION ONLY 指定の索引に対する CONTROL 特権が自動的に与えられます (これにより、作成者は索引または SPECIFICATION ONLY 指定の索引をドロップできます)。さらに、INDEX 特権が取り消されても、作成者は CONTROL 特権をそのまま保持します。

### INSERT

表または更新可能なビューに行を挿入し、IMPORT ユーティリティを実行する特権を与えます。

### REFERENCES

親表を参照するための外部キーの作成やドロップを行う特権を付与します。

ステートメントの許可 ID が以下のいずれかを持っている場合、

- DBADM 権限または SYSADM 権限
- 表に対する CONTROL 特権
- 表に対する REFERENCES WITH GRANT OPTION

特権を与えられたユーザーは、表のすべての列を親キーとして使用して参照制約を作成できます (ALTER TABLE ステートメントを使用して後で追加された列であっても)。そうでない場合、付与される特権はステートメントの許可 ID が指定の表に対して持っているすべての列の付与可能な REFERENCE 特権です。

この特権はニックネームに付与することができますが、ニックネームを参照するために外部キーは定義できません。

### REFERENCES (*column-name*,...)

列のリストに指定された列のみを親キーとして使用して外部キーを作成およびドロップする特権を与えます。各 *column-name* (列名) は、ON 節で指定される表の列を指定する非修飾名でなければなりません。型付き表、型付きビュー、またはニックネームに対する列レベルの REFERENCES 特権は付与できません (SQLSTATE 42997)。

### SELECT

以下のことを行うための特権を付与します。

- 表またはビューから行を検索する特権。
- 表にビューを作成する特権。
- 表またはビューに対して EXPORT ユーティリティを実行する特権。

### UPDATE

ON 節で指定される表または更新可能なビューに対して UPDATE ステートメントを使用する特権を付与します。

ステートメントの許可 ID が以下のいずれかを持っている場合、

- DBADM 権限または SYSADM 権限
- 表またはビューに対する CONTROL 特権
- その表またはビューに対する UPDATE WITH GRANT OPTION

特権を与えられたユーザーは、付与者が GRANT 特権を持っている表またはビューのすべての更新可能な列を更新できます (ALTER TABLE ステートメントを使用して後で追加された列であっても)。そうでない場合、与えられる特権はステートメントの許可 ID が指定の表またはビューに対して持っているすべての列の付与可能な UPDATE 特権です。

## GRANT (表、ビュー、またはニックネーム特権)

### UPDATE (*column-name*,...)

列のリストに指定した列だけを、UPDATE ステートメントを使用して更新する特権を付与します。各 *column-name* は、ON 節で指定される表またはビューの列を指定する非修飾名でなければなりません。型付き表、型付きビュー、またはニックネームに対する列レベルの UPDATE 特権は付与できません (SQLSTATE 42997)。

### ON TABLE *table-name* または *view-name* または *nickname*

特権を付与する表、ビュー、またはニックネームを指定します。

作動不能なビューまたは作動不能なマテリアライズ照会表に対する特権を付与することはできません (SQLSTATE 51024)。宣言済み一時表に対する特権を付与することはできません (SQLSTATE 42995)。

### TO

特権を誰に与えるかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

### *authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。ステートメントを発行しているユーザーの許可 ID への付与に関する、以前の制約はなくなりました。

グループに付与された特権は、次のような許可検査では使用されません。

- パッケージ内の静的 DML ステートメントに対する許可検査
- CREATE VIEW ステートメントの処理過程での基本表に対する許可検査
- マテリアライズ照会表の CREATE TABLE ステートメントの処理過程での基本表に対する許可検査

DB2 Database for Linux, UNIX, and Windows の場合、グループに付与される表特権は、動的に準備されるステートメントにのみ適用されます。例えば、PROJECT 表に対する INSERT 特権がグループ D204 に与えられ、UBIQUITY (D204 のメンバー) には与えられていない場合、UBIQUITY は以下のステートメントを出すことができます。

```
EXEC SQL EXECUTE IMMEDIATE :INSERT_STRING;
```

ここで、ストリングの内容は次のとおりです。

```
INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

ただし、以下のステートメントを含むプログラムをプリコンパイルまたはバインドすることはできません。

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP)
VALUES ('AD3114', 'TOOL PROGRAMMING', 'D21', '000260');
```

### PUBLIC

特権をユーザー (許可 ID) の集合に付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。静的 SQL ステートメントおよび CREATE VIEW ステートメントに対して PUBLIC に与えられた特権の使用に関する、以前の制約は除かれました。

### WITH GRANT OPTION

指定した *authorization-name* に対し、特権を他のユーザーに与えることを許可します。

指定した特権に CONTROL が含まれる場合、WITH GRANT OPTION は CONTROL を除くすべての適用可能な特権に適用されます (SQLSTATE 01516)。

### 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - インスタンスに対して有効なセキュリティ・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。
  - *authorization-name* がデータベースでは ROLE として定義され、有効なセキュリティ・プラグインでは GROUP または USER のいずれかとして定義されている場合には、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
  - 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
  - *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。
- 一般に、GRANT ステートメントはステートメントの許可 ID が与えることを許されている特権の GRANT のみを処理し、1 つ以上の特権が与えられなかった場合は警告 (SQLSTATE 01007) を戻します。どのような特権も与えられなかった場合は、エラーが戻されます (SQLSTATE 42501)。ステートメントの処理に使用されるパッケージが、LANGLEVEL を SQL92E または MIA に設定してプリコンパイルされていた場合、付与者が GRANT 操作の対象に対して特権を持っていない場合以外は警告が戻されます (SQLSTATE 01007)。CONTROL 特権を指定する場合、特権が与えられるのは、ステートメントの許可 ID に SYSADM または DBADM 権限を持っているときだけです (SQLSTATE 42501)。

### 注

- 特権は表階層のどのレベルにも個別に付与できます。スーパー表に対する特権を持つユーザーは、その副表にも影響を及ぼす場合があります。例えば、スーパー表 *T* に対する UPDATE 特権は持っているものの、そのスーパー表の副表である *S* に対する UPDATE 特権は持っていないユーザーが *T* を指定して更新を要求す

## GRANT (表、ビュー、またはニックネーム特権)

ると、*T* の副表 *S* 内にある行に対して変更を要求したかのような場合があります。ユーザーが副表を直接操作できるのは、その副表に対して必要な特権を持っている場合だけです。

- ニックネーム特権を付与しても、データ・ソース・オブジェクト (表またはビュー) の特権に与える影響はありません。通常、データ・ソースの特権は、データの検索を試行する際、ニックネームが参照する表またはビューで必要とされます。
- **互換性**
  - DB2 for z/OS との互換性:
    - 以下の構文は許容されますが、無視されます。
      - PUBLIC AT ALL LOCATIONS

### 例

例 1: 表 WESTERN\_CR に対するすべての特権を PUBLIC に与えます。

```
GRANT ALL ON WESTERN_CR
TO PUBLIC
```

例 2: ユーザー PHIL と CLAIRE が CALENDAR 表を読み取り、また新しい項目を挿入することができるように、CALENDAR 表に対する適切な特権を付与します。既存の項目の変更や削除を行うことは許可しません。

```
GRANT SELECT, INSERT ON CALENDAR
TO USER PHIL, USER CLAIRE
```

例 3: COUNCIL 表に対するすべての特権と、その特権を他のユーザーに適用する特権をユーザー FRANK に付与します。

```
GRANT ALL ON COUNCIL
TO USER FRANK WITH GRANT OPTION
```

例 4: 表 CORPDATA.EMPLOYEE に対する SELECT 特権を JOHN という名前のユーザーに付与します。JOHN と呼ばれるユーザーは存在していますが、JOHN と呼ばれるグループは存在していません。

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

または

```
GRANT SELECT
ON CORPDATA.EMPLOYEE TO USER JOHN
```

例 5: 表 CORPDATA.EMPLOYEE に対する SELECT 特権を JOHN という名前のグループに付与します。JOHN と呼ばれるグループは存在していますが、JOHN と呼ばれるユーザーは存在していません。

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO JOHN
```

または

```
GRANT SELECT ON CORPDATA.EMPLOYEE TO GROUP JOHN
```

例 6: D024 という名前のグループと、D024 という名前のユーザーの両方に、表 T1 に対する INSERT および SELECT 特権を付与します。

```
GRANT INSERT, SELECT ON TABLE T1
TO GROUP D024, USER D024
```

## GRANT (表、ビュー、またはニックネーム特権)

この場合、D024 グループのメンバーとユーザー D024 はいずれも、表 T1 に対する INSERT と SELECT を使用できるようになります。また、SYSCAT.TABAUTH カタログ・ビューには 2 つの行が追加されることとなります。

例 7: ユーザー FRANK に、CALENDAR 表に対する INSERT、SELECT、および CONTROL 特権を付与します。FRANK は特権を他のユーザーに渡すことが可能である必要があります。

```
GRANT CONTROL ON TABLE CALENDAR
TO FRANK WITH GRANT OPTION
```

このステートメントの結果、CONTROL に WITH GRANT OPTION が与えられなかったことを示す警告 (SQLSTATE 01516) が出されます。Frank は、INSERT と SELECT を含む CALENDAR に対する特権を必要に応じて付与することが可能になります。FRANK は、SYSADM 権限または DBADM 権限を持っていない限り、他のユーザーに CALENDAR に対する CONTROL 特権を付与することはできません。

例 8: ユーザー JON が、索引のない Oracle 表のニックネームを作成しました。ニックネームは ORAREM1 です。その後、Oracle DBA がこの表の索引を定義しました。そのため、ユーザー SHAWN は、さらに効率よく表にアクセスするための戦略をオプティマイザーが立てられるようにするため、この索引の存在を DB2 に認識させたいと思っています。SHAWN は、ORAREM1 の SPECIFICATION ONLY 指定の索引を作成することにより、索引を DB2 に認識させることができます。SHAWN が SPECIFICATION ONLY 指定の索引を作成できるようにするため、このニックネームに対する索引特権を SHAWN に与えます。

```
GRANT INDEX ON NICKNAME ORAREM1
TO USER SHAWN
```

---

## GRANT (ワークロード特権)

この形式の GRANT ステートメントは、ワークロードに対する USAGE 特権を付与します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

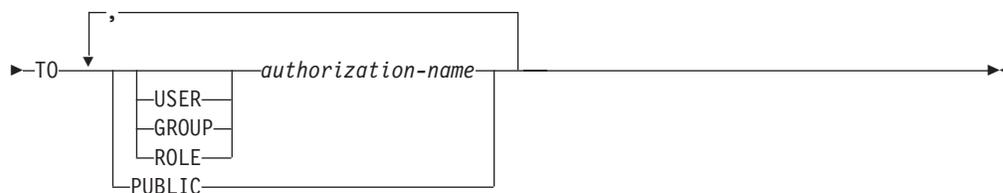
### 許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

### 構文

```
►►—GRANT—USAGE—ON—WORKLOAD—workload-name—►►
```

## GRANT (ワークロード特権)



### 説明

#### USAGE

ワークロードを使用する特権を付与します。ユーザーによってサブミットされる作業単位は、ユーザーが USAGE 特権を持っているワークロードにのみマップされます。SYSADM または DBADM 権限を持っているユーザーは、自動的に、現行のサーバーに存在するすべてのワークロードでの USAGE 特権を持ちます。

#### ON WORKLOAD *workload-name*

どのワークロードに対する USAGE 特権を付与するかを指定します。これは、1部構成の名前です。 *workload-name* には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。名前を 'SYSDEFAULTADMWORKLOAD' にすることはできません (SQLSTATE 42832)。

#### TO

USAGE 特権を誰に付与するかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループであることを指定します。

#### ROLE

*authorization-name* が現行サーバーにおける既存のロールを識別することを指定します (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

#### PUBLIC

一連のユーザー (許可 ID) に USAGE 特権を付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

### 規則

- 指定した *authorization-name* ごとに、キーワード USER、GROUP、および ROLE のいずれも指定されていない場合には次のようになります。
  - インスタンスに対して有効なセキュリティー・プラグインによって *authorization-name* の状況を判別できなければ、エラーが戻されます (SQLSTATE 56092)。

- *authorization-name* が、データベースでは ROLE として定義されており、かつオペレーティング・システムでは GROUP または USER のいずれかとして定義されている場合、エラーが戻されます (SQLSTATE 56092)。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER と GROUP の両方として定義されている場合、エラーが戻されます (SQLSTATE 56092)。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が USER としてのみ定義されている場合、または未定義の場合、USER であると見なされます。
- 有効になっているセキュリティ・プラグインに従って *authorization-name* が GROUP としてのみ定義されている場合、GROUP であると見なされます。
- *authorization-name* がデータベースで ROLE としてのみ定義されている場合には、ROLE であると見なされます。

### 注

- GRANT ステートメントは、コミットされるまでは有効になりません。これは、ステートメントを発行する接続でも同じです。
- データベースが RESTRICT オプション付きで作成される場合、デフォルト・ユーザー・ワークロード SYSDEFAULTUSERWORKLOAD の USAGE 特権は、DBADM 権限を持つユーザーによって明示的に付与される必要があります。データベースが RESTRICT オプションなしで作成されている場合、SYSDEFAULTUSERWORKLOAD の USAGE 特権は、データベースの作成時に PUBLIC に付与されます。

### 例

ユーザー LISA に、ワークロード CAMPAIGN を使用できるよう特権を付与します。

```
GRANT USAGE ON WORKLOAD CAMPAIGN TO USER LISA
```

---

## GRANT (XSR オブジェクト特権)

この形式の GRANT ステートメントは、XSR オブジェクトに対する USAGE 特権を付与します。

### 呼び出し

GRANT ステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

以下のいずれかの権限が必要です。

- SYSADM または DBADM
- XSR オブジェクトの所有権

## GRANT (XSR オブジェクト特権)

### 構文

```
▶▶ GRANT USAGE ON XSROBJECT xsobject-name TO PUBLIC ◀◀
```

### 説明

#### ON XSROBJECT *xsobject-name*

この名前で、USAGE 特権を付与される XSR オブジェクトを示します。  
*xsobject-name* (暗黙的または明示的スキーマ修飾子を含む) は、現行のサーバーに存在する XSR オブジェクトを固有に識別しなければなりません。この名前による XSR オブジェクトが存在しない場合、エラーが戻されます (SQLSTATE 42704)。

#### TO PUBLIC

一連のユーザー (許可 ID) に USAGE 特権を付与します。詳細は、『許可、特権、およびオブジェクト所有権』を参照してください。

### 例

XML スキーマ MYSCHEMA に対する USAGE 特権を、すべてのユーザーに付与します。

```
GRANT USAGE ON XSROBJECT MYSCHEMA TO PUBLIC
```

---

## IF

IF ステートメントは、条件の評価に基づいて実行パスを選択します。

### 呼び出し

このステートメントは、SQL プロシージャまたは動的コンパウンド・ステートメントに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

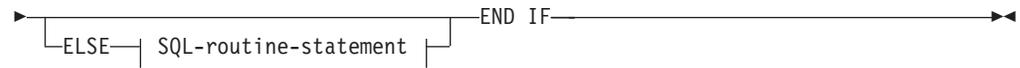
### 許可

IF ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID には、IF ステートメントに組み込まれている SQL ステートメントおよび検索条件を呼び出すために必要な特権がなければなりません。

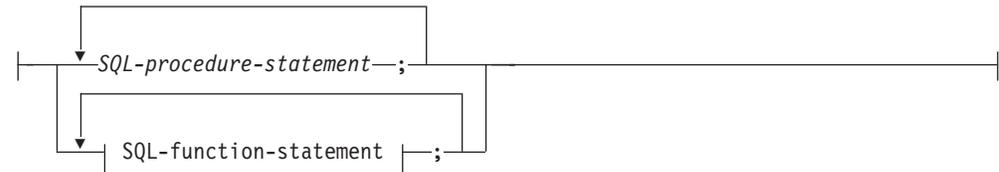
### 構文

```
▶▶ IF search-condition THEN | SQL-routine-statement | ◀◀
```

```
┌──┐
↓
▶▶ ┌──┐
└─ ELSEIF search-condition THEN | SQL-routine-statement | ┘ ◀◀
```



### SQL-routine-statement:



## 説明

### *search-condition*

SQL ステートメントを呼び出す条件を指定します。条件が不明または偽の場合、条件が真になるか、または処理が ELSE 節に到達するまで、処理は次の検索条件に継続されます。

### *SQL-procedure-statement*

前の *search-condition* が真の場合に呼び出されるステートメントを指定します。*SQL-procedure-statement* は、SQL プロシージャのコンテキスト内でのみ使用できます。『コンパウンド SQL (プロシージャ)』で、*SQL-procedure-statement* を参照してください。

### *SQL-function-statement*

前の *search-condition* が真の場合に呼び出されるステートメントを指定します。*SQL-function-statement* は、SQL 関数または SQL メソッドのコンテキスト内でのみ使用できます。『FOR』で、*SQL-function-statement* を参照してください。

## 例

以下の SQL プロシージャでは、2 つの IN パラメーター (従業員番号 *employee\_number* および従業員評定 *rating*) を使用します。 *rating* の値によっては、employee 表の salary および bonus 列が、新しい値に更新されます。

```
CREATE PROCEDURE UPDATE_SALARY_IF
 (IN employee_number CHAR(6), INOUT rating SMALLINT)
LANGUAGE SQL
BEGIN
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE EXIT HANDLER FOR not_found
 SET rating = -1;
 IF rating = 1
 THEN UPDATE employee
 SET salary = salary * 1.10, bonus = 1000
 WHERE empno = employee_number;
 ELSEIF rating = 2
 THEN UPDATE employee
 SET salary = salary * 1.05, bonus = 500
 WHERE empno = employee_number;
 ELSE UPDATE employee
 SET salary = salary * 1.03, bonus = 0
 WHERE empno = employee_number;
 END IF;
END
```

## INCLUDE

INCLUDE ステートメントは、宣言をソース・プログラムに挿入します。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、実行可能ステートメントではありません。

### 許可

必要ありません。

### 構文

```

▶▶ INCLUDE SQLCA
 SQLDA
 name
▶▶

```

### 説明

#### SQLCA

SQL 連絡域 (SQLCA) の記述を組み込むことを指定します。

#### SQLDA

SQL 記述子域 (SQLDA) の記述を組み込むことを指定します。

#### *name*

プリコンパイルするソース・プログラムに組み込むテキストが入っている外部ファイルを指定します。ファイル名拡張子のない SQL ID、または単一引用符で囲んだ ( ' ) リテラルを指定することができます。SQL ID は、そのファイル名拡張子として、プリコンパイルするソース・ファイルのファイル名拡張子が想定されます。引用符で囲んだリテラルにファイル名拡張子の指定がない場合には、拡張子はないものと想定されます。

### 注

- プログラムをプリコンパイルすると、INCLUDE ステートメントはソース・ステートメントで置き換えられます。したがって、ソース・プログラム中での INCLUDE ステートメントの位置は、展開結果のソース・ステートメントがコンパイラに受け入れられる位置でなければなりません。
- 外部ソース・ファイルは、*name* に指定されているホスト言語で作成しなければなりません。名前が 18 バイトを超える場合、または SQL ID としては使用できない文字が含まれている場合は、単一引用符で囲む必要があります。INCLUDE *name* ステートメントは、ネスト可能ですが、循環が発生してはなりません (たとえば、A と B というモジュールがあり、A の中に INCLUDE *name* ステートメントが含まれている場合、A が B を呼び出し、その B が A を呼び出すようにするのは有効ではありません)。
- LANGLEVEL プリコンパイル・オプションに SQL92E 値が指定されている場合、INCLUDE SQLCA を指定してはなりません。SQLSTATE と SQLCODE 変数は、ホスト変数宣言セクションで定義できます。

**例**

C プログラムに SQLCA を組み込みます。

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL DECLARE C1 CURSOR FOR
 SELECT DEPTNO, DEPTNAME, MGRNO FROM TDEPT
 WHERE ADMRDEPT = 'A00';

EXEC SQL OPEN C1;

while (SQLCODE==0) {
 EXEC SQL FETCH C1 INTO :dnum, :dname, :mnum;

 (Print results)
}

EXEC SQL CLOSE C1;
```

**INSERT**

INSERT ステートメントは、表、ニックネーム、またはビュー、あるいは指定された全選択の基礎になる表、ニックネーム、またはビューに、行を挿入します。行をニックネームに挿入することは、その行をそのニックネームが参照するデータ・ソース・オブジェクトに挿入することでもあります。このビューに対する挿入操作作用に INSTEAD OF トリガーが定義されていない場合、行をビューに挿入することは、その行をそのビューの基本となる表に挿入することでもあります。このようなトリガーが定義されている場合は、トリガーが代わりに実行されます。

**呼び出し**

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

**許可**

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- ターゲット表、ビュー、またはニックネームに対する INSERT 特権
- ターゲット表、ビュー、またはニックネームに対する CONTROL 特権
- SYSADM または DBADM 権限

さらに、ステートメントの許可 ID には、INSERT ステートメントで使用する全選択で参照される表、ビュー、またはニックネームのそれぞれに対して、以下の特権の少なくとも 1 つが含まれている必要があります。

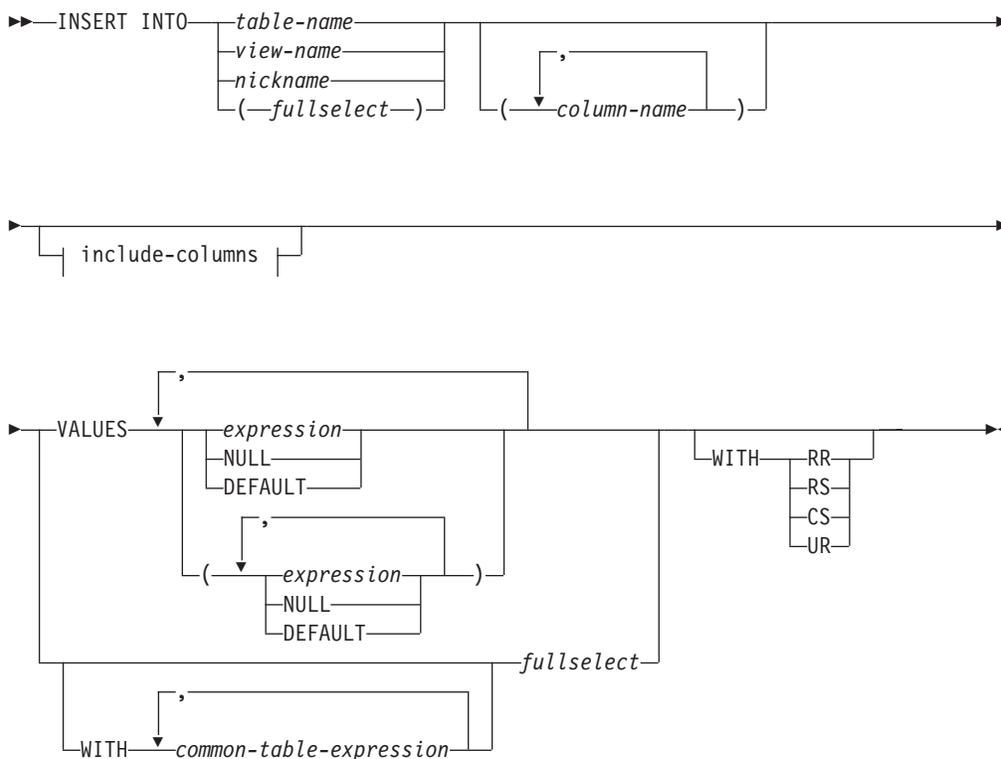
- SELECT 特権
- CONTROL 特権
- SYSADM または DBADM 権限

静的 INSERT ステートメントの場合、GROUP 特権はチェックされません。

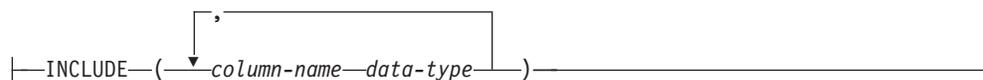
## INSERT

挿入操作の対象がニックネームの場合は、データ・ソースでステートメントが実行されないうちは、そのデータ・ソース上のオブジェクトに対する特権は考慮されません。この時点で、データ・ソースに接続するために使用される許可 ID は、データ・ソースのオブジェクトに対して操作を行うのに必要な特権を持っている必要があります。ステートメントの許可 ID は、データ・ソースの別の許可 ID へマップできます。

### 構文



#### include-columns:



### 説明

#### INTO *table-name*, *view-name*, *nickname*, または (*fullselect*)

挿入操作の対象のオブジェクトを指定します。 *table-name* (表名)、 *view-name* (ビュー名)、または *nickname* (ニックネーム) は、それぞれアプリケーション・サーバーに存在する表、ビュー、またはニックネームを指定していなければならず、カタログ表、システムで保守されているマテリアライズ照会表、カタログ表のビュー、または読み取り専用のビュー (対象となるビューに対する挿入操作に `INSTEAD OF` トリガーが定義されていない場合) は指定できません。行をニックネームに挿入することは、その行をそのニックネームが参照するデータ・ソース・オブジェクトに挿入することでもあります。

挿入操作のオブジェクトが全選択である場合、CREATE VIEW ステートメントの説明の『注』にある『挿入可能ビュー』という項目で定義されているように、全選択が挿入可能になっている必要があります。

このビューに対する挿入操作用に INSTEAD OF トリガーがない場合、以下のようなビューの列には、値を挿入することはできません。

- 定数、式、またはスカラー関数から得られる列。
- そのビューの他の列と同じ基本表の列から得られる列。

挿入操作の対象となるビューにこのような列がある場合は、列名のリストを指定する必要があります、そのリストに上記の列を指定してはなりません。

行が基礎となる基本表のうち 1 つだけのチェック制約を満たしている場合、UNION ALL を使用して定義されているビューまたは全選択にその行を挿入できます。行が複数の表のチェック制約を満たしている場合や、どの表のチェック制約も満たしていない場合は、エラーが戻されます (SQLSTATE 23513)。

*(column-name,...)*

挿入する値の対象となる列を、各 *column-name* に指定します。それぞれの名前は、指定された表、ビュー、またはニックネームの列、あるいは全選択の列を指定しなければなりません。同じ列を複数回指定することはできません。挿入値を受け入れることのできない列 (例えば、式を基にした列) を指定することはできません。

列のリストを省略すると、(暗黙的に隠されていない) 表またはビューのすべての列、あるいは全選択の選択リストのすべての項目を左から右に並べたリストが暗黙に指定されます。このリストはステートメントが準備される時点で確立されます。したがって、ステートメントの準備後に表に追加された列は含まれません。

*include-columns*

全選択の FROM 節にネストされているとき、*table-name* や *view-name* などの列と一緒に INSERT ステートメントの中間結果表に組み込まれている列セットを指定します。*include-columns* は、*table-name* や *view-name* で指定されている列のリストの最後に付加されます。

**INCLUDE**

INSERT ステートメントの中間結果表に組み込まれる列のリストを指定します。この節は、INSERT ステートメントが全選択の FROM 節にネストされている場合にのみ指定できます。

*column-name*

INSERT ステートメントの中間結果表の列を指定します。名前は、他の組み込み列や、*table-name* または *view-name* の列と同じ名前であってはなりません (SQLSTATE 42711)。

*data-type*

組み込み列のデータ・タイプを指定します。データ・タイプは、CREATE TABLE ステートメントでサポートされているものでなければなりません。

**VALUES**

挿入する 1 つ以上の行の値を、この後に指定します。

ホスト変数を指定する場合、それらのホスト変数は、ホスト変数の宣言規則に従ってそのプログラムで記述されていなければなりません。

## INSERT

各行の値の数は、暗黙的または明示的な列のリスト、および INCLUDE 節で識別された列にある名前の数と一致していなければなりません。最初の値はリストの最初の列に挿入され、2 番目の値は 2 番目の列に挿入されます。以下同様です。

### expression

『Expressions』で定義されている *expression* (式) を使用できます。

### NULL

NULL 値を指定します。これは NULL 可能な列に対してのみ指定できます。

### DEFAULT

デフォルト値を使用することを指定します。DEFAULT を指定したときに使用される値は、該当の列がどのように定義されているかによって決まります。次のとおりです。

- 式に基づいて生成される列として列が定義されている場合は、その式に基づいた列の値がシステムによって生成されます。
- IDENTITY 節が使用されている場合は、データベース・マネージャーによって値が生成されます。
- ROW CHANGE TIMESTAMP 節を使用すると、データベース・マネージャーによって、データベース・パーティション内の表パーティションごとの固有のタイム・スタンプ値が、挿入される各行に生成されます。
- WITH DEFAULT 節が使用されている場合は、その列に対して定義された値が挿入されます (『CREATE TABLE』の *default-clause* を参照してください)。
- NOT NULL 節が使用されているが GENERATED 節は使用されていない場合、または WITH DEFAULT 節は使用されていないか DEFAULT NULL が使用されている場合は、その列に対して DEFAULT キーワードを指定することができません (SQLSTATE 23502)。
- ニックネームに挿入する場合、データ・ソースが照会言語構文中の DEFAULT キーワードをサポートしている場合に限り、DEFAULT キーワードはそのデータ・ソースに対して INSERT ステートメントをパススルーします。

### WITH *common-table-expression*

後続の *fullselect* で使用する共通表式を定義します。

### *fullselect*

新しい行の集合を、全選択の結果表の形式で指定します。行の数は、1 つか、複数か、またはゼロのいずれかです。結果表が空の場合、SQLCODE は +100 に設定され、SQLSTATE は '02000' に設定されます。

INSERT の基本オブジェクトおよび全選択の基本オブジェクトまたは全選択の副照会のいずれかが同一の表である場合、行挿入の前に、全選択が完全に評価されます。

結果表の列の数は、列リストの名前の数と同じでなければなりません。結果の最初の列の値はリストの最初の列に挿入され、2 番目の値は 2 番目の列に挿入されます。以下同様です。

**WITH**

`fullselect` が実行される分離レベルを指定します。

**RR**

反復可能読み取り

**RS**

読み取り固定

**CS**

カーソル固定

**UR**

非コミット読み取り

ステートメントのデフォルト分離レベルは、ステートメントがバインドされているパッケージの分離レベルです。**WITH** 節はニックネームには影響を与えません。ニックネームは常にステートメントのデフォルトの分離レベルを使用します。

**規則**

- **トリガー:** INSERT ステートメントによってトリガーの実行が引き起こされる場合があります。トリガーが他のステートメントの実行を引き起こす場合や、挿入値に起因するエラーが発生する場合があります。ビューに挿入操作を行うと **INSTEAD OF** トリガーが起動する場合は、そのトリガーによって実行される更新に対して妥当性、参照整合性、および制約がチェックされます。トリガーを起動させたビューやその基礎表に対するチェックは行われません。
- **デフォルト値:** 列リストにない列に挿入される値は、列のデフォルト値または NULL 値のいずれかになります。NULL 可能でない列で **NOT NULL WITH DEFAULT** として定義されていない列は、列リストに含める必要があります。同様に、ビューへの挿入の場合、基本表の列で、ビューにはない列に挿入される値は、その列のデフォルト値か、または NULL 値のいずれかになります。したがって、基本表に存在し、ビューにはない列はすべて、デフォルト値があるか、または NULL 可能であるかのいずれかでなければなりません。生成される列が **GENERATED ALWAYS** 節で定義されている場合は、**DEFAULT** 以外の値を挿入することはできません (SQLSTATE 428C9)。
- **長さ:** 列の挿入値が数値の場合、列はその数の整数部分を入れる容量を持つ数値列でなければなりません。列の挿入値がストリングの場合、列は、長さ属性がそのストリングの長さ以上である列であるか、またはストリングが日付、時刻、またはタイム・スタンプを表す場合は、日付/時刻列でなければなりません。
- **割り当て:** 挿入値は、特定の割り当ての規則に従って列に割り当てられます。
- **妥当性:** 指定された表または指定されたビューの基本表に 1 つまたは複数のユニーク索引がある場合、表に挿入される各行は、それらの索引の制約に適合していなければなりません。その定義に **WITH CHECK OPTION** を伴うビューが指定された場合、そのビューに挿入する各行は、そのビューの定義に適合していなければなりません。この状況に関連する規則については、『**CREATE VIEW**』を参照してください。
- **参照整合性:** 表に対して定義されている制約ごとに、外部キーの挿入値のうち NULL 以外の値は、それぞれ親表の主キーの値に等しくなければなりません。

## INSERT

- **チェック制約:** 挿入値は、表に定義されているチェック制約のチェック条件を満たしていなければなりません。チェック制約が定義されている表に対する INSERT では、挿入される各行ごとに一度、制約条件が評価されます。
- **XML 値:** XML 列に挿入する値は、整形 XML 文書でなければなりません (SQLSTATE 2200M)。
- **セキュリティー・ポリシー:** 指定された表または指定されたビューの基本表がセキュリティー・ポリシーで保護されている場合、セッション許可 ID は、以下を許可するラベル・ベースのアクセス制御 (LBAC) クレデンシャルを持っている必要があります。
  - データ値が明示的に提供される、保護されたすべての列に対する書き込みアクセス (SQLSTATE 42512)
  - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL オプションを使って生成されたセキュリティー・ポリシーに関して DB2SECURITYLABEL 列に明示的に与えられる値に対する書き込みアクセス (SQLSTATE 23523)

さらに、DB2SECURITYLABEL 列に暗黙的な値が使用される場合には、セキュリティー・ポリシーの書き込みアクセスに関するセキュリティー・ラベルもまた、セッション許可 ID に付与されている必要があります (SQLSTATE 23523)。このような暗黙的な値は、以下の場合に使用される可能性があります。

- DB2SECURITYLABEL 列の値が明示的に提供されていない
- DB2SECURITYLABEL 列の値が明示的に提供されているが、セッション許可 ID がその値に対する書き込みアクセスを持たず、OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL オプションを使ってセキュリティー・ポリシーが生成されている

### 注

- INSERT ステートメントの実行後、SQLCA の SQLERRD の 3 番目の変数 (SQLERRD(3)) の値は、挿入操作に渡された行の数を示します。SQL プロシージャ・ステートメントでは、値は GET DIAGNOSTICS ステートメントの ROW\_COUNT 変数を使用して検索できます。SQLERRD(5) には、トリガーによって実行された挿入、更新、および削除操作の数が入られます。
- 適切な既存のロックが存在しない場合、1 つ以上の排他ロックが正常な INSERT ステートメントの実行時に獲得されます。それらのロックが解放されるまで、挿入された行は以下によってのみアクセス可能です。
  - その挿入を行ったアプリケーション・プロセス
  - 読み取り専用カーソル、SELECT INTO ステートメント、または副照会で使用されている副選択を介して分離レベル UR を使用する他のアプリケーション・プロセス
- ロッキングについての詳細は、COMMIT、ROLLBACK、および LOCK TABLE のステートメントの説明を参照してください。
- パーティション・データベースに対してアプリケーションが実行されており、INSERT BUF オプションを指定してアプリケーションがバインドされている場合、EXECUTE IMMEDIATE を使用して処理されない VALUES を伴う INSERT はバッファーに入れられます。DB2 は、このような INSERT ステートメントがアプリケーションの論理においてループ中で処理されるものと想定します。ステートメントをその完了まで実行する代わりに、DB2 は新しい行の値を 1 つまた

は複数のバッファに入れるを試みます。その結果として、表に対する行の実際の挿入は後で行われ、アプリケーションの INSERT の論理とは非同期になります。この非同期の挿入が原因で、アプリケーションでその INSERT に続く他の SQL ステートメントに INSERT が戻されることに関連してエラーが生じる場合がある点に注意してください。

この方法は、INSERT のパフォーマンスを大幅に向上させる可能性を持っていますが、エラー処理が非同期であるために、クリーン・データに対して使用するのが最適です。

- ID 列が含まれている表に行が挿入されると、DB2 は ID 列の値を生成します。
  - GENERATED ALWAYS の ID 列に対しては、常に DB2 が値を生成します。
  - GENERATED BY DEFAULT 列に対しては、値が (VALUES 節や副選択によって) 明示的に指定されていない場合に、DB2 が値を生成します。

DB2 は、その ID 列に対して START WITH で指定された値を最初の値として生成します。

- ユーザー定義特殊タイプの ID 列に値が挿入される時は、まずすべての計算がソース・タイプで行われます。そして計算された値は、値が列に実際に割り当てられる前に、ソース・タイプから定義された特殊タイプにキャストされます。(計算に先立って、元の値がソース・タイプにキャストされることはありません。)
- GENERATED ALWAYS の ID 列に挿入するときは、常に DB2 がその列の値を生成します。挿入の際にユーザーが値を指定することはできません。列のリストに GENERATED ALWAYS という ID 列がリストされている INSERT ステートメントで、VALUES 節に DEFAULT 以外の値が指定された場合は、エラーが発生します (SQLSTATE 428C9)。

例えば、EMPID という列が GENERATED ALWAYS の ID 列として定義されているとします。そこで、次のコマンドを入力します。

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
VALUES (:hv_valid_emp_id, :hv_name, :hv_addr)
```

すると、エラーが戻されます。

- GENERATED ALWAYS ROW CHANGE TIMESTAMP 列への挿入時には、列の値は常に DB2 によって生成されます。挿入時にユーザーが値を指定することはできません (SQLSTATE 428C9)。DB2 によって生成される値は、データベース・パーティションに挿入される各行に固有です。
- GENERATED BY DEFAULT 列に挿入するときは、DB2 では VALUES 節で、または副選択からその列に実際の値を指定することができます。ただし、VALUES 節に値を指定するとき、DB2 は指定された値を一切検査しません。IDENTITY 列の値を固有にするには、ID 列に対するユニーク索引を作成する必要があります。

列のリストを指定せずに、GENERATED BY DEFAULT の ID 列のある表に挿入するときは、ID 列の値を表す DEFAULT キーワードを VALUES 節で指定することができます。DB2 は、指定された値を ID 列に生成します。

```
INSERT INTO T2 (EMPID, EMPNAME, EMPADDR)
VALUES (DEFAULT, :hv_name, :hv_addr)
```

## INSERT

この例では、EMPID が ID 列として定義され、この列に挿入される値は DB2 によって生成されます。

- 副選択を使用して ID 列に値を挿入する場合も、VALUES 節を使用する場合と同様の規則が適用されます。ID 列に値を指定できるのは、ID 列が GENERATED BY DEFAULT として定義されている場合だけです。

例えば、同じ定義を持つ、T1 と T2 という 2 つの表があるとします。これらの表にはいずれも列 *intcol1* および *identcol2* (これらはどちらもタイプ INTEGER の列で、2 番目の列には識別属性がある) が含まれています。次のような挿入について考慮します。

```
INSERT INTO T2
 SELECT *
 FROM T1
```

この例は、論理的には以下と同じ意味になります。

```
INSERT INTO T2 (intcol1,identcol2)
 SELECT intcol1, identcol2
 FROM T1
```

このどちらの場合においても、INSERT ステートメントには T2 の ID 列を表す明示的な値が指定されています。このように明示的な値を指定した場合は、ID 列の値を指定することができます。しかしこれは、T2 の ID 列が GENERATED BY DEFAULT として定義されている場合に限られます。それ以外の場合は、ID 列に値を指定するとエラーが戻されます (SQLSTATE 428C9)。

表に GENERATED ALWAYS の ID 列として定義された列がある場合でも、同じ定義を持つ表から、他のすべての列に伝搬することができます。例えば、先に取り上げた例の表 T1 と T2 であれば、T1 と T2 に含まれている *intcol1* の値を以下の SQL で伝搬することができます。

```
INSERT INTO T2 (intcol1)
 SELECT intcol1
 FROM T1
```

なお、*identcol2* は列のリストで指定されていないため、この列にはデフォルトの (生成) 値が使用されます。

- GENERATED ALWAYS の ID 列または ROW CHANGE TIMESTAMP 列として定義された単一系列の表に行を挿入するときは、VALUES 節に DEFAULT キーワードを指定することができます。この場合、表の値はアプリケーションによって提供されません。ID 列または ROW CHANGE TIMESTAMP 列の値は DB2 によって生成されます。

```
INSERT INTO IDTABLE
 VALUES(DEFAULT)
```

識別属性をもつ列が含まれているこの同じ単一系列の表に、1 つの INSERT ステートメントを使用して複数の行を挿入するとします。その場合は、次のような INSERT ステートメントを使用できます。

```
INSERT INTO IDTABLE
 VALUES (DEFAULT), (DEFAULT), (DEFAULT), (DEFAULT)
```

- DB2 によって生成される ID 列の値は消費され、次に値が必要な時には、DB2 によってまた新しい値が生成されます。これは、ID 列に関連した INSERT ステートメントが失敗した場合やロールバックされた場合も同様です。

例えば、ID 列にユニーク索引が作成されていると想定します。ID 列に対する値の生成で重複キーの違反が検出されると、エラーが戻され (SQLSTATE 23505)、その ID 列に対して生成される値は破棄されることとなります。このエラーが生じる可能性があるのは、ID 列が GENERATED BY DEFAULT として定義されており、システムが新しい値を生成しようとしたものの、ユーザーが以前の INSERT ステートメントで ID 列に明示的な値を指定していた場合です。このような場合は、同じ INSERT ステートメントをもう一度発行すればうまくいきます。DB2 は ID 列に対して次の値を生成します。次に生成された値が重複していなければ、INSERT ステートメントは正常に完了します。

- ID 列に対して生成される値が ID 列の最大値 (降順で値が生成される場合は最小値) を超えると、エラーが発生します (SQLSTATE 23522)。この場合、ユーザーは、表を DROP して、より広い範囲を持つ ID 列 (より広い値の範囲で、列のデータ・タイプを変更したり、値を増分したりできるようにするため) を指定して、新しい表の CREATE を実行する必要があります。

例えば、データ・タイプ SMALLINT で定義されている ID 列があり、この列で割り当てられる値を使い切ってしまったとします。ID の列を INTEGER として再定義するには、データをアンロードし、表をドロップし、新しい定義の列で表を再作成して、それからデータを再ロードしなければなりません。そして、表を再定義する際は、DB2 によって生成される次の値が元の順序で次の値になるように、ID 列の START WITH 値を指定しなければなりません。最後の値を確認するには、データをアンロードする前に、ID 列の MAX (昇順で値を生成している場合) または MIN (降順で値を生成している場合) を使用して照会を発行します。

## 例

例 1: DEPARTMENT 表に、以下の指定で新しい部門を挿入します。

- 部門番号 (DEPTNO) は 'E31'
- 部門名 (DEPTNAME) は 'ARCHITECTURE'
- その管理者の社員番号 (MGRNO) は '00390'
- 報告先の部門 (ADMRDEPT) は 'E01'

```
INSERT INTO DEPARTMENT
VALUES ('E31', 'ARCHITECTURE', '00390', 'E01')
```

例 2: 例 1 と同様に DEPARTMENT 表に新しい部門を挿入しますが、新しい部門に管理者は割り当てません。

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
VALUES ('E31', 'ARCHITECTURE', 'E01')
```

例 3: 例 2 と同様の DEPARTMENT 表に 2 つの新しい部門を 1 つのステートメントを使用して挿入しますが、新しい部門に管理者は割り当てません。

```
INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
VALUES ('B11', 'PURCHASING', 'B01'),
('E41', 'DATABASE ADMINISTRATION', 'E01')
```

例 4: EMP\_ACT 表と同じ列を持つ一時表 MA\_EMP\_ACT を作成します。EMP\_ACT 表から、'MA' で始まるプロジェクト番号 (PROJNO) を持つ行を MA\_EMP\_ACT 表にロードします。

## INSERT

```
CREATE TABLE MA_EMP_ACT
(EMPNO CHAR(6) NOT NULL,
 PROJNO CHAR(6) NOT NULL,
 ACTNO SMALLINT NOT NULL,
 EMPTIME DEC(5,2),
 EMSTDATE DATE,
 EMENDATE DATE)
INSERT INTO MA_EMP_ACT
SELECT * FROM EMP_ACT
WHERE SUBSTR(PROJNO, 1, 2) = 'MA'
```

例 5: C プログラムのステートメントを使用して、PROJECT 表にスケルトン・プロジェクトを追加します。プロジェクト番号 (PROJNO)、プロジェクト名 (PROJNAME)、部門番号 (DEPTNO)、および責任者 (RESPEMP) は、ホスト変数から入手します。プロジェクトの開始日 (PRSTDATE) として、現在の日付を使用します。表のその他の列には、NULL (NULL) 値を割り当てます。

```
EXEC SQL INSERT INTO PROJECT (PROJNO, PROJNAME, DEPTNO, RESPEMP, PRSTDATE)
VALUES (:PRJNO, :PRJNM, :DPTNO, :REMP, CURRENT DATE);
```

例 6: SELECT ステートメントで、INSERT ステートメントを *data-change-table-reference* として指定します。VALUE 節で値が指定されている組み込み列を別に定義し、それを、挿入される行の配列用の列として使用します。

```
SELECT inorder.ordernum
FROM (INSERT INTO orders(custno)INCLUDE (insertnum integer)
VALUES (:cnum1, 1), (:cnum2, 2)) InsertedOrders
ORDER BY insertnum;
```

例 7: C プログラムのステートメントを使用して、DOCUMENTS 表に文書を追加します。SQL TYPE IS XML AS BLOB\_FILE にバインドするホスト変数から文書 ID (DOCID) 列と文書データ (XMLDOC) 列の値を取得します。

```
EXEC SQL INSERT INTO DOCUMENTS
(DOCID, XMLDOC) VALUES (:docid, :xmldoc)
```

例 8: この例の INSERT ステートメントでは、表 SALARY\_INFO に 3 つの列が定義されて、最後の列は ROW CHANGE TIMESTAMP の暗黙的な隠し列であると想定します。以下のステートメントでは、暗黙的な隠し列が列リストで明示的に参照され、その値が VALUES 節で提供されます。

```
INSERT INTO SALARY_INFO (LEVEL, SALARY, UPDATE_TIME)
VALUES (2, 30000, CURRENT_TIMESTAMP)
```

以下の INSERT ステートメントは暗黙的な列リストを使用しています。暗黙的な列リストには暗黙的な隠し列が含まれないため、他の 2 つの列の値だけが VALUES 節に含まれています。

```
INSERT INTO SALARY_INFO VALUES (2, 30000)
```

この場合、UPDATE\_TIME 列はデフォルト値を持つように定義される必要があり、挿入される行にはそのデフォルト値が使用されます。

---

## ITERATE

ITERATE ステートメントを使用すると、制御のフローがラベル付きループの最初に戻ります。

## 呼び出し

このステートメントは、SQL プロシージャまたは動的コンパウンド・ステートメントに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

## 許可

必要ありません。

## 構文

```
▶▶—ITERATE—label—▶▶
```

## 説明

*label*

DB2 が制御のフローを渡す先の FOR、LOOP、REPEAT、または WHILE ステートメントのラベルを指定します。

## 例

この例では、カーソルを使用して新しい部門の情報を戻します。 *not\_found* 条件処理ルーチンが呼び出されると、制御のフローがループの外側に渡されます。 *v\_dept* の値が 'D11' の場合、ITERATE ステートメントは制御のフローを LOOP ステートメントの先頭に戻します。それ以外の場合は、新しい行が DEPARTMENT 表に挿入されます。

```
CREATE PROCEDURE ITERATOR()
LANGUAGE SQL
BEGIN
 DECLARE v_dept CHAR(3);
 DECLARE v_deptname VARCHAR(29);
 DECLARE v_admdept CHAR(3);
 DECLARE at_end INTEGER DEFAULT 0;
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE c1 CURSOR FOR
 SELECT deptno, deptname, admrdept
 FROM department
 ORDER BY deptno;
 DECLARE CONTINUE HANDLER FOR not_found
 SET at_end = 1;
 OPEN c1;
 ins_loop:
 LOOP
 FETCH c1 INTO v_dept, v_deptname, v_admdept;
 IF at_end = 1 THEN
 LEAVE ins_loop;
 ELSEIF v_dept = 'D11' THEN
 ITERATE ins_loop;
 END IF;
 INSERT INTO department (deptno, deptname, admrdept)
 VALUES ('NEW', v_deptname, v_admdept);
 END LOOP;
 CLOSE c1;
END
```

---

## LEAVE

LEAVE ステートメントは、プログラム制御をループまたはコンパウンド・ステートメントの外側に移動させます。

### 呼び出し

このステートメントは、SQL プロシージャまたは動的コンパウンド・ステートメントに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

### 許可

必要ありません。

### 構文

▶▶—LEAVE—*label*—————▶▶

### 説明

*label*

終了するコンパウンド、FOR、LOOP、REPEAT、または WHILE ステートメントのラベルを指定します。

### 注

- LEAVE ステートメントがコンパウンド・ステートメントの外側に制御を移動すると、そのコンパウンド・ステートメント内のすべてのオープン・カーソル（結果セットを戻すのに使用されているカーソルを除く）がクローズされます。

### 例

以下の例には、カーソル *c1* のデータを取り出すループが含まれています。SQL 変数 *at\_end* の値がゼロでなければ、LEAVE ステートメントは制御をループの外側に移動させます。

```
CREATE PROCEDURE LEAVE_LOOP(OUT counter INTEGER)
LANGUAGE SQL
BEGIN
 DECLARE v_counter INTEGER;
 DECLARE v_firstnme VARCHAR(12);
 DECLARE v_midinit CHAR(1);
 DECLARE v_lastname VARCHAR(15);
 DECLARE at_end SMALLINT DEFAULT 0;
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE c1 CURSOR FOR
 SELECT firstnme, midinit, lastname
 FROM employee;
 DECLARE CONTINUE HANDLER for not_found
 SET at_end = 1;
 SET v_counter = 0;
 OPEN c1;
 fetch_loop:
 LOOP
 FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
 IF at_end <> 0 THEN LEAVE fetch_loop;
 END IF;
```

```

 SET v_counter = v_counter + 1;
 END LOOP fetch_loop;
 SET counter = v_counter;
 CLOSE c1;
END

```

## LOCK TABLE

LOCK TABLE ステートメントを使用すると、並行アプリケーション・プロセスが表を変更したり表を使用したりできなくなります。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- その表に対する SELECT 特権
- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

### 構文

```

▶▶ LOCK TABLE {table-name | nickname} IN {SHARE | EXCLUSIVE} MODE

```

### 説明

#### *table-name* or *nickname*

該当の表またはニックネームを指定します。 *table-name* は、アプリケーション・サーバーに存在する表を指定していなければなりません。カタログ表または宣言済み一時表は指定できません (SQLSTATE 42995)。 *table-name* が型付き表である場合、その表は表階層のルート表でなければなりません (SQLSTATE 428DR)。ニックネームを指定すると、DB2 は、そのニックネームが参照するデータ・ソースの基礎オブジェクト (つまり表かビュー) をロックします。

#### IN SHARE MODE

複数の並行するアプリケーション・プロセスが、その表に対して読み取り専用以外の操作を実行するのを防止します。

#### IN EXCLUSIVE MODE

複数の並行するアプリケーション・プロセスが、その表に対してどのような操作も実行できないようにします。ただし、EXCLUSIVE MODE は、非コミット読み取り分離レベル (UR) で実行している並行アプリケーション・プロセスが、その表に対して読み取り専用操作を実行することは妨げない点に注意してください。

### 注

- ロッキングは、複数の操作が並行して行われるのを防止するのに使用されます。すでに適切なロックが存在している場合には、LOCK TABLE ステートメントを実行しても、必ずしもロックが獲得されるとは限りません。並行操作を防止するロックは、少なくともその作業単位の終了まで保持されます。
- パーティション・データベースでは、表ロックはデータベース・パーティション・グループ内の最初のデータベース・パーティション (最も番号の小さいデータベース・パーティション) で最初に獲得され、その後他のデータベース・パーティションで獲得されます。LOCK TABLE ステートメントが割り込まれると、表は一部のデータベース・パーティションではロックされ、その他ではロックされないことになります。このような場合、他の LOCK TABLE ステートメントを出してすべてのデータベース・パーティションに対してロッキングを完了するか、COMMIT または ROLLBACK ステートメントを出して現在のロックを解放します。
- このステートメントは、データベース・パーティション・グループ内のすべてのデータベース・パーティションに影響を与えます。
- パーティション表の場合、LOCK TABLE ステートメントで獲得されるロックは、表レベルのロックだけであり、データ・パーティションのロックは獲得されません。

### 例

表 EMP に対するロックを入手します。他のプログラムは、その表の読み取りや更新を行うことができなくなります。

```
LOCK TABLE EMP IN EXCLUSIVE MODE
```

## LOOP

LOOP ステートメントは、ステートメント、またはステートメントのグループの実行を繰り返します。

### 呼び出し

このステートメントは、SQL プロシージャに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

### 許可

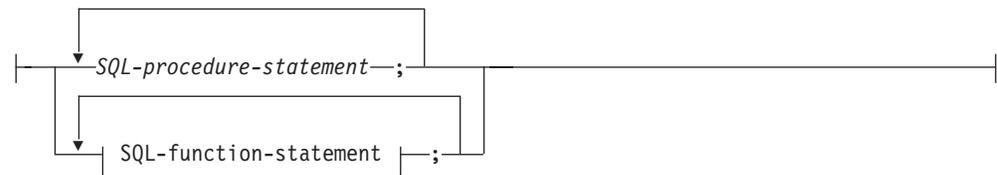
LOOP ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID には、LOOP ステートメントに組み込まれている SQL ステートメントを呼び出すために必要な特権がなければなりません。

### 構文

```

┌──┴──┐
└─┬──┘ ┌──┴──┐
 label: LOOP SQL-routine-statement END LOOP label

```

**SQL-routine-statement:****説明***label*

LOOP ステートメントのラベルを指定します。開始ラベルを指定した場合、そのラベルを LEAVE および ITERATE ステートメントで指定することができます。終了ラベルを指定する場合、一致する開始ラベルを指定しなければなりません。

*SQL-procedure-statement*

ループ内で呼び出される SQL ステートメントを指定します。

*SQL-procedure-statement* は、SQL プロシージャのコンテキスト内でのみ使用できます。『コンパウンド SQL (プロシージャ)』で、*SQL-procedure-statement* を参照してください。

*SQL-function-statement*

ループ内で呼び出される SQL ステートメントを指定します。

*SQL-function-statement* は、SQL 関数または SQL メソッドのコンテキスト内でのみ使用できます。『FOR』で、*SQL-function-statement* を参照してください。

**例**

以下のプロシージャでは、LOOP ステートメントを使用して、employee 表から値を取り出します。ループが繰り返されるたびに、OUT パラメーター *counter* が増加し、*v\_midinit* が検査されて、値が単一スペース ( ' ') でないことを確認します。*v\_midinit* が単一スペースの場合、LEAVE ステートメントは制御のフローをループの外側に渡します。

```
CREATE PROCEDURE LOOP_UNTIL_SPACE(OUT counter INTEGER)
LANGUAGE SQL
BEGIN
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE v_firstname VARCHAR(12);
 DECLARE v_midinit CHAR(1);
 DECLARE v_lastname VARCHAR(15);
 DECLARE c1 CURSOR FOR
 SELECT firstname, midinit, lastname
 FROM employee;
 DECLARE CONTINUE HANDLER FOR NOT FOUND
 SET counter = -1;
 OPEN c1;
 fetch_loop:
 LOOP
 FETCH c1 INTO v_firstname, v_midinit, v_lastname;
 IF v_midinit = ' ' THEN
 LEAVE fetch_loop;
 END IF;
 SET v_counter = v_counter + 1;
 END LOOP;
END;
```

## LOOP

```
END LOOP fetch_loop;
SET counter = v_counter;
CLOSE c1;
END
```

---

## MERGE

MERGE ステートメントは、ソース (表参照の結果) からのデータを使ってターゲット (表またはビュー、あるいは全選択の基礎表またはビュー) を更新します。ターゲット内にあるソースと一致する行を削除または更新するよう指定でき、ターゲットに存在しない行を挿入することができます。ビュー内の行を更新、削除、または挿入すると、ビューの元になっている表の行が更新、削除、または挿入されます。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 挿入操作が指定されている場合、表またはビューに対する INSERT 特権。削除挿入操作が指定されている場合、表またはビューに対する DELETE 特権。更新操作が指定されている場合には、以下のいずれか。
  - 表またはビューに対する UPDATE 特権
  - 更新されるそれぞれの列に対する UPDATE 特権
- 表に対する CONTROL 特権
- SYSADM または DBADM 権限

このステートメントの許可 ID が持つ特権には、少なくとも次のいずれかも含まれている必要があります。

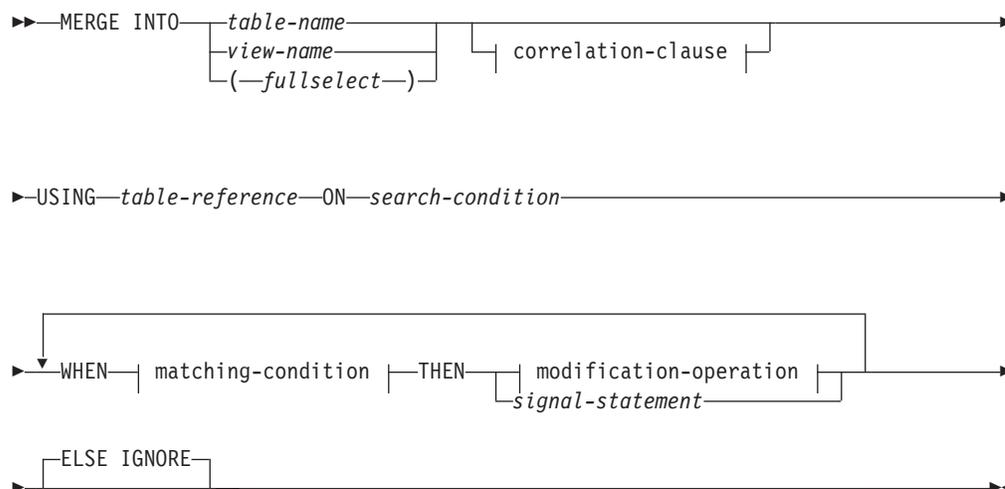
- *table-reference* で指定されたすべての表またはビューに対する SELECT 特権
- *table-reference* で指定された表またはビューに対する CONTROL 特権
- SYSADM または DBADM 権限

*search-condition*、*insert-operation*、または *assignment-clause* が副照会を持つ場合、このステートメントの許可 ID が持つ特権には、少なくとも次のいずれかも含まれている必要があります。

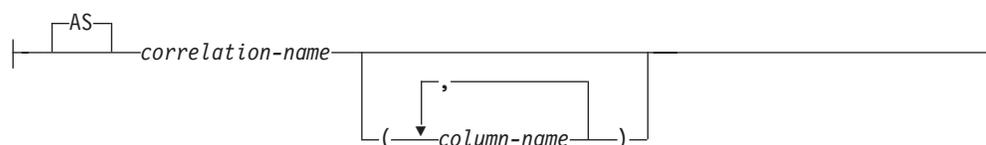
- 副照会で指定されたすべての表またはビューに対する SELECT 特権
- 副照会で指定された表またはビューに対する CONTROL 特権
- SYSADM または DBADM 権限

関数を参照する式が指定されている場合、特権セットにはその関数を実行するのに必要な権限が含まれていなければなりません。

## 構文



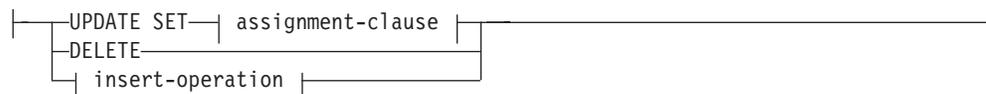
## correlation-clause:



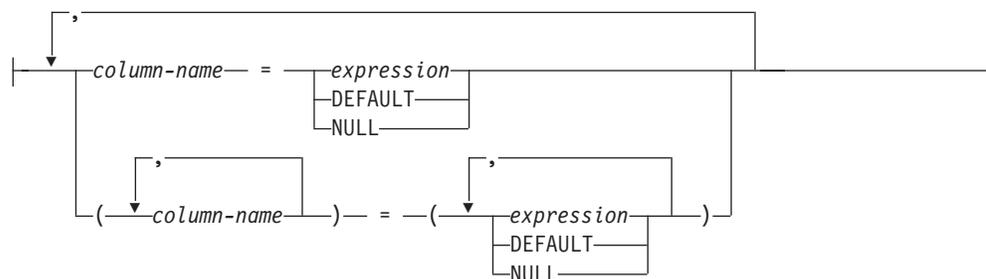
## matching-condition:

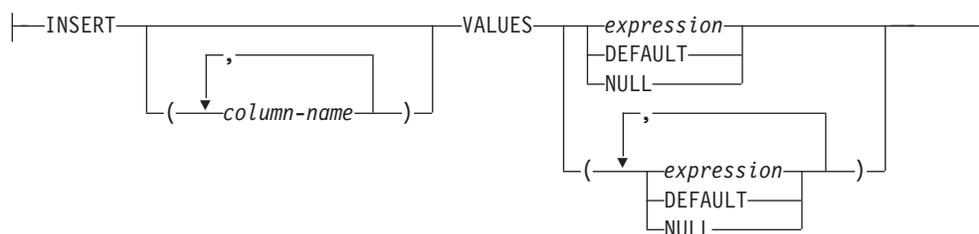


## modification-operation:



## assignment-clause:



**insert-operation:****説明****table-name、view-name、または (fullselect)**

マージの更新、削除、または挿入操作のターゲットを識別します。この名前は、現在のサーバーに存在する表またはビューを識別する必要があります。ただし、カタログ表、システムで保守されているマテリアライズ照会表、カタログ表のビュー、読み取り専用のビューを参照することはできません。さらに、NOT DETERMINISTIC または EXTERNAL ACTION を使って定義されている副照会またはルーチンを参照する WHERE 節を直接的または間接的に含むようなビューを参照することもできません (SQLSTATE 42807)。

マージ操作のターゲットが全選択である場合、全選択は、CREATE VIEW ステートメントの説明の『注』にある、『更新可能ビュー』、『削除可能ビュー』、または『挿入可能ビュー』という項目で定義されているように、更新可能、削除可能、または挿入可能になっている必要があります。

**correlation-clause**

*search-condition* 内や *assignment-clause* の右側で使用して、表、ビュー、または全選択を指定できます。 *correlation-clause* についての説明は、『副選択』の説明にある『table-reference』を参照してください。

**USING table-reference**

ターゲットにマージされる結果表として、行のセットを指定します。結果表が空の場合、警告が戻されます (SQLSTATE 02000)。

**ON search-condition**

マージの更新または削除操作のために *table-reference* のどの行が使用されるか、およびマージの挿入操作のためにどの行が使用されるかを指定します。 *search-condition* は、ターゲット表および *table-reference* の結果表の各行に適用されます。 *table-reference* の結果表の中で、 *search-condition* が真である行に対して、指定された更新または削除操作が実行されます。 *table-reference* の結果表の中で、 *search-condition* が真でない行に対しては、指定された挿入操作が実行されます。

*search-condition* には、以下の制限があります (SQLSTATE 42972)。

- 副照会 (スカラーなど) を入れることはできない
- 参照値がオブジェクト ID 列以外の場合、間接参照操作または Deref 関数を組み込むことはできない
- SQL 関数を組み入れることはできない
- XMLQUERY または XMLEXISTS 式を組み入れることはできない

- *search-condition* の式で参照されるどの列も、ターゲット表、ビュー、または *table-reference* の列でなければならない。
- 全外部結合の *join-condition* の式で参照される関数は、決定論的なものでなければならず、外部アクションがあってはならない。

**WHEN *matching-condition***

*modification-operation* または *signal-statement* が実行される条件を指定します。それぞれの *matching-condition* は、指定された順序で評価されます。*matching-condition* が真と評価された行は、後続の一致条件では無視されます。

**MATCHED**

ON 検索条件が真である行に対して実行される操作を示します。THEN の後には、UPDATE、DELETE、または *signal-statement* のみを指定できます。

**AND *search-condition***

ON 検索条件に一致する行に対して THEN 後の操作を実行するための、さらに適用される追加の検索条件を指定します。

**NOT MATCHED**

ON 検索条件が偽または不明である行に対して実行される操作を示します。THEN の後には、INSERT または *signal-statement* のみを指定できます。

**AND *search-condition***

ON 検索条件に一致しなかった行に対して THEN 後の操作を実行するための、さらに適用される追加の検索条件を指定します。

**THEN *modification-operation***

*matching-condition* が真と評価された場合に実行される操作を指定します。

**UPDATE SET**

*matching-condition* が真と評価された行に対して実行される更新操作を指定します。

***assignment-clause***

列更新のリストを指定します。

***column-name***

更新する列を指定します。 *column-name* は指定された表またはビューの列を識別する必要がありますが、スカラー関数、定数、または式から得られたビュー列を識別することはできません。同じ列を複数回指定することはできません (SQLSTATE 42701)。

1 つのビュー列から得られた 2 つのビュー列を更新するとき、両方の列を 1 つの MERGE ステートメントで更新することはできません (SQLSTATE 42701)。

***expression***

列の新しい値を指定します。 *expression* には列関数を含めることはできません (SQLSTATE 42903)。

*expression* には、 *table-name* または *view-name* の列への参照を含めることができます。更新対象の行ごとに、式の中のそのような列の値は、行の更新前のその行の列の値になります。

### DEFAULT

列に割り当てられるデフォルト値。デフォルト値を持つ列に関してのみ、DEFAULT を指定できます。データ・タイプのデフォルト値については、CREATE TABLE ステートメントの DEFAULT 節に関する説明を参照してください。

GENERATED ALWAYS として定義された列に関しては、DEFAULT を指定する必要があります。GENERATED BY DEFAULT として定義された列に関しては、有効な値を指定することができます。

### NULL

列の新しい値として NULL 値を指定します。NULL は、NULL 可能列にのみ指定できます (SQLSTATE 23502)。

### DELETE

*matching-condition* が真と評価された行に対して実行される削除操作を指定します。

#### *insert-operation*

*matching-condition* が真と評価された行に対して実行される挿入操作を指定します。

### INSERT

挿入操作に使われる、列名と行値の式からなるリストを指定します。

行値の式における行の数、挿入列リストにおける名前数と同じでなければなりません。最初の値はリストの最初の列に挿入され、2 番目の値は 2 番目の列に挿入されます。以下同様です。

(*column-name*,...)

挿入値が提供される列を指定します。それぞれの名前は、表またはビューの列を識別する必要があります。同じ列を複数回指定することはできません (SQLSTATE 42701)。挿入値を受け入れることのできないビューの列を指定することはできません。以下のようなビュー列には、値を挿入できません。

- 定数、式、またはスカラー関数から得られる列。
- そのビューの他の列と同じ基本表の列から得られる列。

操作の対象となるビューにこのような列がある場合は、列名のリストを指定する必要があり、そのリストに上記の列を指定してはなりません。

列のリストを省略すると、(暗黙的な非表示として定義されていない) 表またはビューのすべての列を左から右に指定したリストが暗黙に指定されます。このリストはステートメントが準備される時点で確立されます。したがって、ステートメントの準備後に表に追加された列は含まれません。

### VALUES

挿入する 1 つ以上の行の値を、この後に指定します。

*expression*

列名を含まない任意の式 (SQLSTATE 42703)。

**DEFAULT**

列に割り当てられるデフォルト値。デフォルト値を持つ列に関してのみ、**DEFAULT** を指定できます。データ・タイプのデフォルト値については、**CREATE TABLE** ステートメントの **DEFAULT** 節に関する説明を参照してください。

**GENERATED ALWAYS** として定義された列に関しては、**DEFAULT** を指定する必要があります。 **GENERATED BY DEFAULT** として定義された列に関しては、有効な値を指定することができます。

**NULL**

列の値として **NULL** 値を指定します。 **NULL** は、**NULL** 可能列にのみ指定できます (**SQLSTATE 23502**)。

*signal-statement*

*matching-condition* が真と評価された場合にエラーを戻すために実行される **SIGNAL** ステートメントを指定します。

**ELSE IGNORE**

どの *matching-condition* も真と評価されない場合に、行に対してアクションが実行されないことを指定します。

**規則**

- 複数の *modification-operation* (**UPDATE SET**、**DELETE**、または *insert-operation*) あるいは *signal-statement* を、単一の **MERGE** ステートメントの中で指定できます。
- ターゲット内の各行は、一度だけ操作できます。ターゲット内の各行は、*table-reference* の結果表のただ 1 つの行とのみ **MATCHED** として識別されます (**SQLSTATE 21506**)。ネストした SQL 操作 (**RI**、または **INSTEAD OF** トリガーを除くトリガー) では、ターゲット表 (または同じ表階層内の表) を **UPDATE**、**DELETE**、**INSERT**、または **MERGE** ステートメントのターゲットとして指定することはできません (**SQLSTATE 27000**)。
- **セキュリティー・ポリシー**: 識別されたターゲット表または識別されたターゲット・ビューの基本表がセキュリティー・ポリシーによって保護されている場合、**SESSION** 許可 ID は以下のタイプのアクセスを許可するラベル・ベースのアクセス制御 (**LBAC**) 信任状を持つ必要があります。
  - 更新操作の場合:
    - 更新対象となる保護されたすべての列に対する書き込みアクセス (**SQLSTATE 42512**)
    - **RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL** オプションを使って生成されたセキュリティー・ポリシーに関して **DB2SECURITYLABEL** 列に明示的に与えられる値に対する書き込みアクセス (**SQLSTATE 23523**)
    - 更新対象となるすべての行に対する読み取りおよび書き込みアクセス (**SQLSTATE 42519**)

さらに、**DB2SECURITYLABEL** 列に暗黙的な値が使用される場合には、セキュリティー・ポリシーの書き込みアクセスに関するセキュリティー・ラベルも

## MERGE

また、セッション許可 ID に付与されている必要があります (SQLSTATE 23523)。このような暗黙的な値は、以下の場合に使用される可能性があります。

- DB2SECURITYLABEL 列が更新される列のリストに含まれていない (そのため、SESSION 許可 ID の書き込みアクセスのセキュリティー・ラベルに暗黙的に更新される)
- DB2SECURITYLABEL 列の値が明示的に提供されているが、セッション許可 ID がその値に対する書き込みアクセスを持たず、OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL オプションを使ってセキュリティー・ポリシーが生成されている
- 削除操作の場合:
  - すべての保護された列に対する書き込みアクセス (SQLSTATE 42512)
  - 削除のために選択されたすべての行に対する読み取りおよび書き込みアクセス (SQLSTATE 42519)
- 挿入操作の場合:
  - データ値が明示的に提供される、保護されたすべての列に対する書き込みアクセス (SQLSTATE 42512)
  - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL オプションを使って生成されたセキュリティー・ポリシーに関して DB2SECURITYLABEL 列に明示的に与えられる値に対する書き込みアクセス (SQLSTATE 23523)

さらに、DB2SECURITYLABEL 列に暗黙的な値が使用される場合には、セキュリティー・ポリシーの書き込みアクセスに関するセキュリティー・ラベルもまた、セッション許可 ID に付与されている必要があります (SQLSTATE 23523)。このような暗黙的な値は、以下の場合に使用される可能性があります。

- DB2SECURITYLABEL 列の値が明示的に提供されていない
- DB2SECURITYLABEL 列の値が明示的に提供されているが、セッション許可 ID がその値に対する書き込みアクセスを持たず、OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL オプションを使ってセキュリティー・ポリシーが生成されている

MERGE ステートメントにおける更新、挿入、または削除操作に関連した他の規則については、該当するステートメントの『規則』セクションを参照してください。

### 注

- 処理順序
  1. ソースからターゲットにかけて処理される行のセットを判別します。このステートメントで CURRENT TIMESTAMP が使用される場合、ステートメント全体でただ一度だけクロックが読み取られます。
  2. ON 節を使用して、これらの行が MATCHED または NOT MATCHED のいずれであるかを分類します。
  3. WHEN 節内に *matching-condition* があれば評価します。
  4. *assignment-clause* および *insert-operation* 内に *expression* があれば評価します。

5. それぞれの *signal-statement* を実行します。
  6. 指定された順序に従って、それぞれの *modification-operation* を該当する行に適用します。それぞれの *modification-operation* によって活動化される制約およびトリガーが、その *modification-operation* に関して実行されます。ステートメント・レベルのトリガーは、*modification-operation* を満たす行が存在しない場合でも活動化されます。それぞれの *modification-operation* は、後続の各 *modification-operation* のトリガーや参照制約に影響する可能性があります。
- **ステートメント・レベルの原子性:** MERGE ステートメントの実行中にエラーが発生した場合、ステートメント全体がロールバックされます。
  - **更新される行数:** MERGE ステートメントの実行が完了すると、SQLCA の GET DIAGNOSTICS および SQLERRD(3) の ROW\_COUNT 項目の値は、MERGE ステートメントによって処理された行数になります (ELSE IGNORE 節によって識別された行を除く)。SQLERRD(3) の値には、制約またはトリガーの結果として処理された行数は含まれません。SQLERRD(5) の値には、このような行の数が含まれます。
  - **挿入された行を更新することはできない:** ターゲット内で、MERGE ステートメントの実行前に存在しなかった行の更新操作は一切行われません。つまり、MERGE ステートメントによって挿入された行は更新されません。
  - **INSTEAD OF トリガー:** MERGE ステートメントのターゲットとしてビューが指定される場合、そのビューに対して、まったく INSTEAD OF トリガーを定義しないか、更新、削除、挿入の各 INSTEAD OF トリガーを定義する必要があります (SQLSTATE 428FZ)。

## 例

例 1: 記述 (description) が変更されたアクティビティーに関して、アーカイブ表内の記述を更新します。新しいアクティビティーについては、アーカイブ表に挿入します。アーカイブ表とアクティビティー表にはどちらも、主キーとしてアクティビティーが含まれます。

```
MERGE INTO archive ar
USING (SELECT activity, description FROM activities) ac
ON (ar.activity = ac.activity)
WHEN MATCHED THEN
 UPDATE SET
 description = ac.description
WHEN NOT MATCHED THEN
 INSERT
 (activity, description)
 VALUES (ac.activity, ac.description)
```

例 2: 出荷 (shipment) 表を使って、在庫 (inventory) 表に行をマージします。その際、出荷表のマッチした行の部品カウント (part count) ごとに数量を増分します。そうでない場合は、新しい *partno* を在庫表に挿入します。

```
MERGE INTO inventory AS in
USING (SELECT partno, description, count FROM shipment
WHERE shipment.partno IS NOT NULL) AS sh
ON (in.partno = sh.partno)
WHEN MATCHED THEN
 UPDATE SET
 description = sh.description,
 quantity = in.quantity + sh.count
```

## MERGE

```
WHEN NOT MATCHED THEN
 INSERT
 (partno, description, quantity)
 VALUES (sh.partno, sh.description, sh.count)
```

例 3: トランザクション (transaction) 表を使用して、アカウント (account) 表に行をマージします。その際、いくつかのトランザクション・セットのアカウント ID に対するバランスを更新し、アカウントがまだ存在しない場合には、統合トランザクションからアカウントを新しく挿入します。

```
MERGE INTO account AS a
USING (SELECT id, sum(amount) sum_amount FROM transaction
 GROUP BY id) AS t
ON a.id = t.id
WHEN MATCHED THEN
 UPDATE SET
 balance = a.balance + t.sum_amount
WHEN NOT MATCHED THEN
 INSERT
 (id, balance)
 VALUES (t.id, t.sum_amount)
```

例 4: トランザクション・ログ (transaction\_log) 表を使って、employee\_file (従業員ファイル) 表に行をマージします。その際、トランザクション時間に基づいて、最新のトランザクション・ログ (transaction\_log) 行の内容で電話 (phone) および部署 (office) を更新し、まだ存在しない場合には、最新の新しい従業員ファイル (employee\_file) 行を挿入します。

```
MERGE INTO employee_file AS e
USING (SELECT empid, phone, office
 FROM (SELECT empid, phone, office,
 ROW_NUMBER() OVER (PARTITION BY empid
 ORDER BY transaction_time DESC) rn
 FROM transaction_log) AS nt
 WHERE rn = 1) AS t
ON e.empid = t.empid
WHEN MATCHED THEN
 UPDATE SET
 (phone, office) =
 (t.phone, t.office)
WHEN NOT MATCHED THEN
 INSERT
 (empid, phone, office)
 VALUES (t.empid, t.phone, t.office)
```

例 5: 従業員 (employee) 行に動的に提供される値を使って、既存の従業員に該当するデータの場合はマスター従業員 (employee) 表を更新します。データが新しい従業員に関するものである場合は、行を挿入します。次の例は、C プログラムのコードの断片です。

```
hvl =
"MERGE INTO employee AS t
USING TABLE(VALUES(CAST (? AS CHAR(6)), CAST (? AS VARCHAR(12)),
 CAST (? AS CHAR(1)), CAST (? AS VARCHAR(15)),
 CAST (? AS SMALLINT), CAST (? AS INTEGER)))
 s(empno, firstme, midinit, lastname, edlevel, salary)
ON t.empno = s.empno
WHEN MATCHED THEN
 UPDATE SET
 salary = s.salary
WHEN NOT MATCHED THEN
 INSERT
 (empno, firstme, midinit, lastname, edlevel, salary)
```

```
VALUES (s.empno, s.firstnme, s.midinit, s.lastname, s.edlevel,
 s.salary)";
EXEC SQL PREPARE s1 FROM :hv1;
EXEC SQL EXECUTE s1 USING '000420', 'SERGE', 'K', 'FIELDING', 18, 39580;
```

例 6: Group A によって編成されたアクティビティのリストをアーカイブ表内で更新します。期限切れのアクティビティをすべて削除し、アーカイブ表のアクティビティ情報 (日付と記述) が変更されていれば、それを更新します。新規の着信アクティビティについては、アーカイブ表に挿入します。アクティビティの日付が不明の場合、エラーを発生します。アーカイブ表内のアクティビティ日付の指定は必須です。アクティビティ表は、それぞれのグループごとに存在します。例えば、activities\_groupA にはこのグループが編成したすべてのアクティビティが含まれ、アーカイブ表には企業のさまざまなグループによって編成された将来のアクティビティがすべて含まれます。アーカイブ表には主キーとして (group, activity) が含まれ、日付を NULL にすることはできません。すべてのアクティビティ表には、主キーとして activity が含まれます。アーカイブ内の最終更新 (last\_modified) 列は、デフォルト値として CURRENT\_TIMESTAMP を使って定義されます。

```
MERGE INTO archive ar
USING (SELECT activity, description, date, last_modified
 FROM activities_groupA) ac
ON (ar.activity = ac.activity) AND ar.group = 'A'
WHEN MATCHED AND ac.date IS NULL THEN
 SIGNAL SQLSTATE '70001'
 SET MESSAGE_TEXT =
 ac.activity CONCAT ' cannot be modified. Reason: Date is not known'
WHEN MATCHED AND ac.date < CURRENT_DATE THEN
 DELETE
WHEN MATCHED AND ar.last_modified < ac.last_modified THEN
 UPDATE SET
 (description, date, last_modified) = (ac.description, ac.date, DEFAULT)
WHEN NOT MATCHED AND ac.date IS NULL THEN
 SIGNAL SQLSTATE '70002'
 SET MESSAGE_TEXT =
 ac.activity CONCAT ' cannot be inserted. Reason: Date is not known'
WHEN NOT MATCHED AND ac.date >= CURRENT_DATE THEN
 INSERT
 (group, activity, description, date)
 VALUES ('A', ac.activity, ac.description, ac.date)
ELSE IGNORE
```

---

## OPEN

OPEN ステートメントは、カーソルをオープンして、そのカーソルを結果表からの行の取り出しに使用できるようにします。

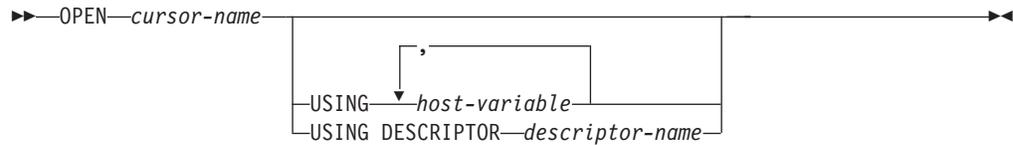
### 呼び出し

対話式 SQL 機能には外見上対話式の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

### 許可

カーソルの使用に必要な許可については、『DECLARE CURSOR』を参照してください。

## 構文



## 説明

### *cursor-name*

プログラムのそれ以前の個所にある DECLARE CURSOR ステートメントで定義されているカーソルの名前を指定します。この OPEN ステートメントが実行される時点で、そのカーソルはクローズ状態でなければなりません。

DECLARE CURSOR ステートメントは、次のいずれかの方法で、SELECT ステートメントを指定しなければなりません。

- その DECLARE CURSOR ステートメントに SELECT ステートメントを組み込む。
- 準備済み SELECT ステートメントを指定する *statement-name* を組み込む。

カーソルの結果表は、その SELECT ステートメントを評価することによって得られます。評価の際には、その SELECT ステートメントで指定されている特殊レジスター、グローバル変数、または PREVIOUS VALUE 式の現行値と、SELECT ステートメントまたは OPEN ステートメントの USING 節で指定されたホスト変数の現行値が使用されます。結果表の行は、OPEN ステートメントの実行中に得られ、それらを入れる一時表が作成されるか、あるいは後続の FETCH ステートメントの実行中に得られます。いずれの場合でも、カーソルはオープン状態になり、その位置はその結果表の最初の行の前になります。表が空の場合、カーソルの状態は事実上「最終行の後」になります。

### USING

この後に、準備済みステートメントのパラメーター・マーカ (疑問符 (?)) に代入する値が入っているホスト変数のリストを指定します。DECLARE CURSOR ステートメントでパラメーター・マーカを含む準備済みステートメントを指定した場合、USING の使用は必須です。準備済みステートメントにパラメーター・マーカが含まれていない場合、USING は無視されます。

### *host-variable*

ホスト変数の宣言規則に従って、そのプログラムで記述されている変数を指定します。変数の数は、準備済みステートメントのパラメーター・マーカの数と同じでなければなりません。n 番目の変数は、準備済みステートメントの n 番目のパラメーター・マーカに対応します。場合によっては、ロケータ変数とファイル参照変数も、パラメーター・マーカの値のソースとして指定できます。

### DESCRIPTOR *descriptor-name*

ホスト変数の有効な記述の入った SQLDA を指定します。

OPEN ステートメントが処理される前に、ユーザーは次に示す SQLDA 内のフィールドを設定する必要があります。

- SQLDA に用意する SQLVAR のエレメント数を示す SQLN

- SQLDA に割り振るストレージのバイト数を示す SQLDABC
- ステートメントの処理時にその SQLDA の使用される変数の数を示す SQLD
- 変数の属性を示す SQLVAR のオカレンス

SQLDA には、すべての SQLVAR オカレンスが入るだけの十分なストレージがなければなりません。したがって、SQLDABC の値は  $16 + \text{SQLN} * (\text{N})$  以上でなければなりません (N は 1 つの SQLVAR オカレンスの長さ)。

LOB の結果列を入れるには、各選択リスト項目 (または結果表の列) ごとに 2 つの SQLVAR 項目が必要です。

SQLD に設定する値は、ゼロ以上で SQLN 以下でなければなりません。

## 規則

- カーソルの SELECT ステートメントが評価される場合に、そのステートメント中の各パラメーター・マーカーは、対応するホスト変数によって置き換えられます。型付きパラメーター・マーカーの場合、ターゲット変数の属性は CAST 指定によって指定されます。タイプなしパラメーター・マーカーの場合、ターゲット変数の属性はパラメーター・マーカーのコンテキストに従って決定されます。
- V は、パラメーター・マーカー P に対応するホスト変数を表します。V の値は、列への値の割り振り規則に従って、P のターゲット変数に割り当てられます。したがって、
  - V はターゲットと互換でなければなりません。
  - V がストリングの場合、その長さ (ただし、LONG ストリングでないストリングの末尾ブランクは含まない) はターゲットの長さ属性を超えることはできません。
  - V が数値の場合、V の整数部分の絶対値はターゲットの整数部分の絶対値の最大を超えることはできません。
  - V の属性がターゲットの属性と同一でない場合、その値はターゲットの属性に合うように変換されます。

カーソルの SELECT ステートメントが評価されると、P の代わりに使用される値は P のターゲット変数になります。例えば、V が CHAR(6) でターゲットが CHAR(8) の場合、P の代わりに使用される値は V の値にブランクを 2 個付加したのになります。

- USING 節は、パラメーター・マーカーを含む準備済み SELECT ステートメントのために用意されています。ただし、これは、カーソルの SELECT ステートメントが DECLARE CURSOR ステートメントの一部である場合にも使用できます。このような場合、OPEN ステートメントは、あたかも SELECT ステートメントの各ホスト変数がパラメーター・マーカーであるかのように実行されます。ただし、ターゲット変数の属性は SELECT ステートメントのホスト変数と同じになります。その結果、USING 節に指定するホスト変数の値によって、カーソルの SELECT ステートメントの中のホスト変数の値がオーバーライドされることになります。
- カーソル定義に組み込まれている SQL データを変更する SQL データ変更ステートメントとルーチンは完全に実行され、結果セットは、カーソルのオープン時に一時表に保管されます。ステートメントの実行が正常に完了すると、

SQLERRD(3) フィールドには、挿入、更新、および削除操作が可能な行の数の合計が入ります。全選択内にデータ変更ステートメントを含むカーソルが関係する OPEN ステートメントの実行中にエラーが発生した場合は、そのデータ変更ステートメントがロールバックされます。

OPEN ステートメントの明示的なロールバック、つまり OPEN ステートメント以前のセーブポイントまでのロールバックにより、カーソルはクローズします。カーソルの定義で、全選択の FROM 節内にデータ変更ステートメントが含まれている場合、データ変更ステートメントの結果はロールバックされます。

SELECT ステートメントや SELECT INTO ステートメントにネストされていたデータ変更ステートメントで、ターゲット表の行に対して行われた変更は、カーソルがオープンされるときに処理されるため、カーソルに対するフェッチ操作の途中でエラーが発生しても、変更が元に戻ることはありません。

## 注

- **クローズ状態のカーソル:** プログラムが開始された時点、およびプログラムが ROLLBACK ステートメントを開始した時点では、そのプログラム中のすべてのカーソルはクローズ状態になります。

WITH HOLD として宣言されたオープン・カーソル以外のすべてのカーソルは、プログラムが COMMIT ステートメントを発行する際にクローズ状態になります。

また、CLOSE ステートメントを実行した場合、またはカーソル位置が予期できなくなるようなエラーが検出された場合にも、カーソルはクローズ状態になることがあります。

- カーソルの結果表から行を取り出すには、カーソルがオープンされている時に FETCH ステートメントを実行します。カーソルの状態をクローズからオープンに変更する唯一の方法は、OPEN ステートメントを実行することです。
- **一時表の効果:** 場合によっては、FETCH ステートメントの実行の過程でカーソルの結果表が得られます。また、一時表メソッドが使用される場合もあります。このメソッドでは、結果表全体が OPEN ステートメントの実行中に一時表に転送されます。一時表が使用される場合、プログラムの結果は、以下の点で異なる可能性があります。
  - 以後の FETCH ステートメントまでは起こることのないエラーが、OPEN の過程で起こる可能性があります。
  - カーソルがオープン状態の間、同じトランザクションで実行された INSERT、UPDATE、および DELETE ステートメントは結果表に影響を与えません。
  - OPEN の実行中に、結果表の行ごとに SELECT ステートメント中の NEXT VALUE 式が評価されます。

逆に、一時表を使用しない場合、カーソルがオープン状態の間に実行される INSERT、UPDATE、および DELETE ステートメントが、同じ作業単位から発行される場合には結果表に影響を与えることがあり、個々の行が取り出されるたびに SELECT ステートメント中の NEXT VALUE 式が評価されます。この結果表は、同じ作業単位で実行される操作による影響を受けることがあり、そのような操作の影響は、必ずしも予測可能であるとは限りません。例えば、カーソル C

の位置が `SELECT * FROM T` と定義された結果表の 1 つの行である場合に、`T` に新しい行を挿入すると、行の順序が整っていないために、その挿入が結果表に与える影響は予測できません。したがって、後続する `FETCH C` で `T` の新しい行が取り出される場合もあれば、取り出されない場合もあります。

- ステートメントのキャッシュは、`OPEN` ステートメントによってオープンと宣言されているカーソルに影響を与えます。

## 例

例 1: COBOL プログラムで、以下を行う組み込みステートメントを作成します。

1. カーソル `C1` を定義します。このカーソルは、`DEPARTMENT` 表から管理部門 (`ADMRDEPT`) 'A00' によって管理される部門の行すべてを検索するために使用します。
2. 最初に取り出す行の前に、カーソル `C1` を置きます。

```
EXEC SQL DECLARE C1 CURSOR FOR
 SELECT DEPTNO, DEPTNAME, MGRNO
 FROM DEPARTMENT
 WHERE ADMRDEPT = 'A00'
END-EXEC.
```

```
EXEC SQL OPEN C1
END-EXEC.
```

例 2: C プログラムで動的に定義される選択ステートメントにカーソル `DYN_CURSOR` を関連付ける `OPEN` ステートメントをコーディングします。選択ステートメントの述部には 2 つのパラメーター・マーカーが使用されており、2 つのホスト参照変数をその `OPEN` ステートメントに指定して、アプリケーションとデータベースとの間で整数と `VARCHAR(64)` の値を渡すために使用します。(関連するホスト変数の定義、`PREPARE` ステートメント、および `DECLARE CURSOR` ステートメントも以下の例に示しています。)

```
EXEC SQL BEGIN DECLARE SECTION;
static short hv_int;
char hv_vchar64[65];
char stmt1_str[200];
EXEC SQL END DECLARE SECTION;

EXEC SQL PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL OPEN DYN_CURSOR USING :hv_int, :hv_vchar64;
```

例 3: 例 2 と同様に `OPEN` ステートメントをコーディングしますが、この例では `WHERE` 節のパラメーター・マーカーの数とデータ・タイプは不明です。

```
EXEC SQL BEGIN DECLARE SECTION;
char stmt1_str[200];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLDA;

EXEC SQL PREPARE STMT1_NAME FROM :stmt1_str;
EXEC SQL DECLARE DYN_CURSOR CURSOR FOR STMT1_NAME;

EXEC SQL OPEN DYN_CURSOR USING DESCRIPTOR :sqlda;
```

## PREPARE

PREPARE ステートメントは、SQL ステートメントの動的な実行を準備するために、アプリケーション・プログラムによって使用されます。PREPARE ステートメントは、ステートメント・ストリングと呼ばれる文字ストリング形式のステートメントから、準備済みステートメントと呼ばれる実行可能な SQL ステートメントを作成します。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

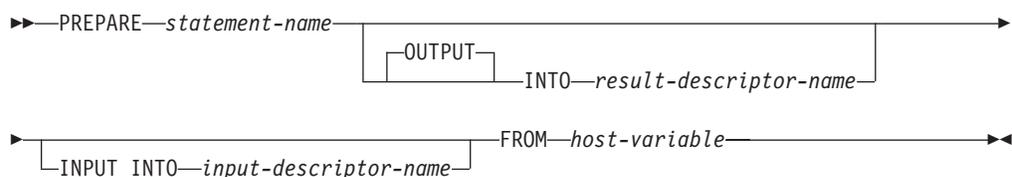
### 許可

ステートメントの準備時に許可検査が行われるステートメント (DML) の場合、ステートメントの許可 ID の特権には、PREPARE ステートメントで指定されている SQL ステートメントの実行に必要な特権が含まれていなければなりません。ステートメントの許可 ID は、DYNAMICRULES BIND オプションの影響を受けることがあります。

ステートメントの実行時に許可検査が行われるステートメント (DDL、GRANT、および REVOKE ステートメント) の場合、このステートメントを使用するために必要な許可は特にありません。ただし、準備済みステートメントの実行時にその許可が検査されます。

セキュリティー・ポリシーで保護された表が関係するステートメントの場合、そのセキュリティー・ポリシーに関連した規則は、常にステートメントの実行時に評価されます。

### 構文



### 説明

#### *statement-name*

準備済みステートメントの名前を指定します。名前として既存の準備済みのステートメントを指定した場合、前もって準備されたそのステートメントは破棄されます。名前として、オープン・カーソルの SELECT ステートメントである準備済みステートメントを指定することはできません。

#### OUTPUT INTO

OUTPUT INTO を使用すると、PREPARE ステートメントを正常に実行した場合に、準備済みステートメント中の出力パラメーター・マーカーについての情報が、*result-descriptor-name* で指定する SQLDA に入れられます。



## PREPARE

- RENAME TABLESPACE
- REVOKE
- ROLLBACK
- SAVEPOINT
- select-statement
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET INTEGRITY
- SET PASSTHRU
- SET PATH
- SET SCHEMA
- SET SERVER OPTION
- SET 変数
- UPDATE

### 注

- **パラメーター・マーカー:** ステートメント・ストリングにホスト変数への参照を組み込むことはできませんが、パラメーター・マーカーを組み込むことはできます。準備済みステートメントの実行時に、パラメーター・マーカーはホスト変数の値に置き換えることができます。CALL ステートメントの場合、プロシージャに対する OUT 引数や INOUT 引数に、パラメーター・マーカーを使用することもできます。CALL の実行後に、引数の戻り値は、パラメーター・マーカーに対応するホスト変数に割り当てられます。

パラメーター・マーカーは疑問符 (?) で表されます。ステートメント・ストリングが静的 SQL ステートメントであれば、パラメーター・マーカーは、ホスト変数を使用できる場所に使用できます。パラメーター・マーカーがどのように値で置き換えられるかについては、『OPEN』と『EXECUTE』を参照してください。

パラメーター・マーカーには、以下の 2 つのタイプがあります。

#### 型付きパラメーター・マーカー

ターゲットのデータ・タイプと一緒に指定するパラメーター・マーカー。一般的な形式は、次のとおりです。

```
CAST(? AS data-type)
```

この表記は関数呼び出しではなく、実行時のパラメーター・タイプが指定のデータ・タイプであること、または指定のデータ・タイプに変換できるデータ・タイプであることを「保証」するものです。たとえば、以下の例で、

```
UPDATE EMPLOYEE
SET LASTNAME = TRANSLATE(CAST(? AS VARCHAR(12)))
WHERE EMPNO = ?
```

TRANSLATE 関数の引数の値は、実行時に与えられます。その値のデータ・タイプは、VARCHAR(12)、または VARCHAR(12) に変換可能なタイプになるはずで

#### 非型付きパラメーター・マーカ

ターゲットのデータ・タイプを指定しないで指定するパラメーター・マーカ。これは、1 つの疑問符の形式です。非型付きパラメーター・マーカのデータ・タイプは、そのコンテキストによって決まります。たとえば、上記の UPDATE ステートメントの述部にある非型付きパラメーター・マーカは、EMPNO 列のデータ・タイプと同じになります。

型付きパラメーター・マーカは、動的 SQL ステートメントで、ホスト変数がサポートされている場所であれば、どこにでも使用でき、そのデータ・タイプは CAST 関数で行った保証に基づきます。

非型付きパラメーター・マーカは、動的 SQL ステートメントで、ホスト変数がサポートされている位置の中から選択された位置で使用できます。それらの位置と結果データ・タイプを以下の表に示しています。位置は、式、述部、組み込み関数、およびユーザー定義ルーチンに類別されており、非型付きパラメーター・マーカの適用度を容易に判別することができます。非修飾関数名の関数(算術演算子、CONCAT、および日付/時刻演算子を含む)に非型付きパラメーター・マーカを使用する場合、関数解決の目的で、その修飾子は 'SYSIBM' に設定されます。

表 29. 式 (選択リスト、CASE、VALUES を含む) での非型付きパラメーター・マーカの使

| 非型付きパラメーター・マーカの位置                                           | データ・タイプ            |
|-------------------------------------------------------------|--------------------|
| 選択リストに単独で                                                   | エラー                |
| 演算子優先順位と演算順序規則の分析後に、<br>単一算術演算子のオペランドの両方となる位置               | エラー                |
| 例:<br>$? + ? + 10$                                          |                    |
| 日付/時刻の式以外の算術式の単一演算子のオペランドのいずれか一方                            | もう一方のオペランドのデータ・タイプ |
| 例:<br>$? + ? * 10$                                          |                    |
| 日付/時刻の式のラベル付き期間 (ラベル付き期間の単位のタイプを示す部分には、パラメーター・マーカを使用できません。) | DECIMAL(15,0)      |

表 29. 式 (選択リスト、CASE、VALUES を含む) での非型付きパラメーター・マーカの  
使用法 (続き)

| 非型付きパラメーター・マーカの位置                                                                        | データ・タイプ                                                                                                                                                                   |
|------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 日付/時刻の式のその他のオペランド ('timecol<br>+ ?' または '? - datecol' など)                                | エラー                                                                                                                                                                       |
| CONCAT 演算子の 2 つのオペランド                                                                    | エラー                                                                                                                                                                       |
| CONCAT 演算子の一方のオペランド (もう一<br>方のオペランドが CLOB 以外の文字データ・<br>タイプである場合)                         | 一方のオペランドが CHAR( <i>n</i> ) または<br>VARCHAR( <i>n</i> ) ( <i>n</i> は 128 より小さい) の場<br>合、もう一方のオペランドは VARCHAR(254<br>- <i>n</i> )であり、他のすべての場合のデータ・タイ<br>プは VARCHAR(254)       |
| CONCAT 演算子の一方のオペランド (もう一<br>方のオペランドが DBCLOB 以外の<br>GRAPHIC データ・タイプである場合)                 | 一方のオペランドが GRAPHIC( <i>n</i> ) または<br>VARGRAPHIC( <i>n</i> ) ( <i>n</i> は 64 より小さい) の場<br>合、もう一方のオペランドは VARCHAR(127<br>- <i>n</i> ) であり、他のすべての場合のデータ・タ<br>イプは VARCHAR(127) |
| CONCAT 演算子の一方のオペランド (もう一<br>方のオペランドがラージ・オブジェクト・ス<br>tringである場合)                          | もう一方のオペランドと同じ                                                                                                                                                             |
| UPDATE ステートメントの SET 節の右側の<br>値                                                           | 列のデータ・タイプ。その列がユーザー定義<br>特殊タイプとして定義されている場合は、そ<br>のユーザー定義特殊タイプのソース・デー<br>タ・タイプ。その列がユーザー定義の構造化<br>タイプとして定義されている場合は、構造化<br>タイプ。これはトランスフォーム関数の戻り<br>タイプも示している。                 |
| 単純な CASE 式の CASE キーワードに続く<br>式                                                           | エラー                                                                                                                                                                       |
| 結果式の残りが非型付きパラメーター・マー<br>カまたは NULL のいずれかである CASE<br>式 (単純および検索) の結果式の少なくとも 1<br>つ         | エラー                                                                                                                                                                       |
| 単純 CASE 式の WHEN の後のいずれかまた<br>はすべての式                                                      | 非型付きパラメーター・マーカ以外の<br>CASE の後の式および WHEN の後の式に結<br>果データ・タイプに関する規則を適用した結<br>果                                                                                                |
| NULL でも非型付きパラメーター・マーカ<br>でもない結果式が少なくとも 1 つある CASE<br>式 (単純および検索) の結果式                    | NULL または非型付きパラメーター・マーカ<br>以外のすべての結果式に結果データ・タイ<br>プに関する規則を適用した結果                                                                                                           |
| INSERT ステートメント内にはない単一行<br>VALUES 節の列式として単独で                                              | エラー                                                                                                                                                                       |
| INSERT ステートメント内になく、他のすべ<br>ての行式での同じ位置にある列式が非型付き<br>パラメーター・マーカである複数行<br>VALUES 節の列式として単独で | エラー                                                                                                                                                                       |

表 29. 式 (選択リスト、CASE、VALUES を含む) での非型付きパラメーター・マーカの  
使用法 (続き)

| 非型付きパラメーター・マーカの位置                                                                             | データ・タイプ                                                                                                                           |
|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| INSERT ステートメント内になく、他の行式のうちの少なくとも 1 つで同じ位置にある式が非型付きパラメーター・マーカでも NULL でもない複数行 VALUES 節の列式として単独で | 非型付きパラメーター・マーカ以外のすべてのオペランドに結果データ・タイプに関する規則を適用した結果                                                                                 |
| INSERT ステートメント内にある単一行 VALUES 節の列式として単独で                                                       | 列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、そのユーザー定義特殊タイプのソース・データ・タイプ。その列がユーザー定義の構造化タイプとして定義されている場合は、構造化タイプ。これはトランスフォーム関数の戻りタイプも示している。 |
| INSERT ステートメント内にある複数行 VALUES 節の列式として単独で                                                       | 列のデータ・タイプ。その列がユーザー定義特殊タイプとして定義されている場合は、そのユーザー定義特殊タイプのソース・データ・タイプ。その列がユーザー定義の構造化タイプとして定義されている場合は、構造化タイプ。これはトランスフォーム関数の戻りタイプも示している。 |
| SET 特殊レジスター・ステートメントの右側にある値として                                                                 | 特殊レジスターのデータ・タイプ                                                                                                                   |

表 30. 述部での非型付きパラメーター・マーカの使用方法

| 非型付きパラメーター・マーカの位置                                                                               | データ・タイプ                                                                                |
|-------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| 比較演算子の両方のオペランド                                                                                  | エラー                                                                                    |
| 比較演算子の一方のオペランド (もう一方のオペランドは非型付きパラメーター・マーカ以外の場合)                                                 | もう一方のオペランドのデータ・タイプ                                                                     |
| BETWEEN 述部のすべてのオペランド                                                                            | エラー                                                                                    |
| 以下のいずれか<br><ul style="list-style-type: none"> <li>• 第 1 および第 2</li> <li>• 第 1 および第 3</li> </ul> | 唯一の非パラメーター・マーカと同じ                                                                      |
| BETWEEN 述部のオペランド                                                                                |                                                                                        |
| その他の BETWEEN の場合 (非型付きパラメーター・マーカが 1 個だけの場合)                                                     | 非型付きパラメーター・マーカ以外のすべてのオペランドに結果データ・タイプに関する規則を適用した結果                                      |
| IN 述部のすべてのオペランド                                                                                 | エラー                                                                                    |
| IN 述部の第 1 オペランド (右側が副選択でない場合: ? IN (?,A,B) や ? IN (A,?,B,?) など)                                 | IN リストのオペランド (IN キーワードの右側のオペランド) のうち、非型付きパラメーター・マーカ以外のすべてのオペランドに結果データ・タイプに関する規則を適用した結果 |
| IN 述部の第 1 オペランド (右側が全選択の場合)                                                                     | 選択した列のデータ・タイプ。                                                                         |

表 30. 述部での非型付きパラメーター・マーカの使用法 (続き)

| 非型付きパラメーター・マーカの使用位置                                      | データ・タイプ                                                                                                                                       |
|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| IN 述部の IN リストのいずれかまたはすべてのオペランド                           | IN 述部のオペランド (IN 述部の左右のオペランド) のうち、非型付きパラメーター・マーカ以外のすべてのオペランドに結果データ・タイプに関する規則を適用した結果                                                            |
| LIKE 述部の 3 つのオペランドすべて                                    | 一致式 (オペランド 1) とパターン式 (オペランド 2) は VARCHAR(32672) であり、エスケープ式 (オペランド 3) は VARCHAR(2)                                                             |
| LIKE 述部の一致式 (パターン式またはエスケープ式のいずれかが、非型付きパラメーター・マーカ以外である場合) | 第 1 オペランドのデータ・タイプ (非型付きパラメーター・マーカ以外) に応じて、VARCHAR(32672) または VARGRAPHIC(16336) のいずれか。                                                         |
| LIKE 述部のパターン式 (一致式またはエスケープ式のいずれかが、非型付きパラメーター・マーカ以外である場合) | 非型付きパラメーター・マーカではない第 1 オペランドのデータ・タイプに応じて、VARCHAR(32672) または VARGRAPHIC(16336) のいずれかになり、一致式のデータ・タイプが BLOB の場合、パターン式のデータ・タイプは BLOB(32672) と見なされる |
| LIKE 述部のエスケープ式 (一致式またはパターン式のいずれかが、非型付きパラメーター・マーカ以外である場合) | 非型付きパラメーター・マーカではない第 1 オペランドのデータ・タイプに応じて、VARCHAR(2) または VARGRAPHIC(1) のいずれかになり、一致式またはパターン式のデータ・タイプが BLOB の場合、エスケープ式のデータ・タイプは BLOB(1) と見なされる    |
| NULL 述部のオペランド                                            | エラー                                                                                                                                           |

表 31. 組み込み関数での非型付きパラメーター・マーカの使用法

| 非型付きパラメーター・マーカの使用位置                                          | データ・タイプ                                           |
|--------------------------------------------------------------|---------------------------------------------------|
| COALESCE (VALUE と呼ばれる) または NULLIF のすべてのオペランド                 | エラー                                               |
| 少なくとも最初オペランドが非型付きパラメーター・マーカ以外である COALESCE または NULLIF のオペランド。 | 非型付きパラメーター・マーカ以外のすべてのオペランドに結果データ・タイプに関する規則を適用した結果 |
| POSSTR (両方のオペランド)                                            | 両方のオペランドは VARCHAR(32672)                          |
| POSSTR の一方のオペランド (もう一方のオペランドが文字データ・タイプである場合)                 | VARCHAR(32672)                                    |
| POSSTR の一方のオペランド (もう一方のオペランドが GRAPHIC データ・タイプである場合)          | VARGRAPHIC(16336)                                 |
| POSSTR の検索ストリング・オペランド (もう一方のオペランドが BLOB である場合)               | BLOB(32672)                                       |
| SUBSTR (第 1 オペランド)                                           | VARCHAR(32672)                                    |
| SUBSTR (第 2 と第 3 のオペランド)                                     | INTEGER                                           |
| TRANSLATE スカラー関数の第 1 オペランド                                   | エラー                                               |

表 31. 組み込み関数での非型付きパラメーター・マーカの使用法 (続き)

| 非型付きパラメーター・マーカの使用位置              | データ・タイプ                                                                         |
|----------------------------------|---------------------------------------------------------------------------------|
| TRANSLATE スカラー関数の第 2 と第 3 のオペランド | 第 1 オペランドが文字タイプの場合は VARCHAR(32672)、第 1 オペランドが GRAPHIC タイプの場合は VARGRAPHIC(16336) |
| TRANSLATE スカラー関数の第 4 オペランド       | 第 1 オペランドが文字タイプの場合は VARCHAR(1)、第 1 オペランドが GRAPHIC タイプの場合は VARGRAPHIC(1)         |
| TIMESTAMP スカラー関数の第 2 オペランド       | TIME                                                                            |
| 単項マイナス                           | DOUBLE                                                                          |
| 単項プラス                            | DOUBLE                                                                          |
| VARCHAR_FORMAT 関数の第 1 オペランド      | TIMESTAMP                                                                       |
| TIMESTAMP_FORMAT 関数の第 1 オペランド    | VARCHAR (短ストリングの長さ)                                                             |
| XMLVALIDATE 関数の第 1 オペランド         | XML                                                                             |
| その他のすべてのスカラー関数のその他のすべてのオペランド     | エラー                                                                             |
| 列関数のオペランド                        | エラー                                                                             |

表 32. ユーザー定義ルーチンでの非型付きパラメーター・マーカの使用法

| 非型付きパラメーター・マーカの使用位置 | データ・タイプ                         |
|---------------------|---------------------------------|
| 関数の引数               | エラー                             |
| メソッドの引数             | エラー                             |
| プロシージャの引数           | プロシージャの作成時に定義された、パラメーターのデータ・タイプ |

- **PREPARE** ステートメントが実行される時点で、ステートメント・ストリングの構文解析が行われ、エラーの有無が検査されます。ステートメント・ストリングが無効な場合には、エラー条件が **SQLCA** に報告されます。エラーが訂正されない限り、そのステートメントを参照するそれ以降の **EXECUTE** または **OPEN** ステートメントも同じエラーになります (システムにより行われる暗黙の準備によって)。
- 準備済みステートメントは、以下の種類のステートメントで、示された制限付きで参照できます。

#### 場所 準備済みステートメント

##### **DESCRIBE**

任意のステートメント

##### **DECLARE CURSOR**

**SELECT** でなければならない

##### **EXECUTE**

**SELECT** であってはならない

## PREPARE

- 準備済みステートメントは、何回でも実行できます。実際に、準備済みステートメントが 1 回しか実行されず、しかもパラメーター・マーカが含まれていない場合には、PREPARE と EXECUTE ステートメントを使用するよりも、EXECUTE IMMEDIATE ステートメントを使用する方が効率が良いになります。
- ステートメントのキャッシュは、準備の繰り返しの影響します。

### 例

例 1: select ステートメント以外のステートメントを COBOL プログラムで準備して実行します。そのステートメントはホスト変数 HOLDER に含まれ、ユーザーによる何らかの指示に基づいて、プログラムはそのステートメント・ストリングをそのホスト変数に入れるものと想定します。準備するステートメントには、パラメーター・マーカは含まれていません。

```
EXEC SQL PREPARE STMT_NAME FROM :HOLDER
END-EXEC.
EXEC SQL EXECUTE STMT_NAME
END-EXEC.
```

例 2: 例 1 と同様に select ステートメント以外のステートメントを準備しますが、この例では、C プログラムにコーディングします。また、準備するステートメントには、いくつかのパラメーター・マーカが含まれていると想定します。

```
EXEC SQL PREPARE STMT_NAME FROM :holder;
EXEC SQL EXECUTE STMT_NAME USING DESCRIPTOR :insert_da;
```

以下のステートメントを準備するものと想定します。

```
INSERT INTO DEPT VALUES(?, ?, ?, ?)
```

DEPT 表の列は、以下のように定義されています。

```
DEPT_NO CHAR(3) NOT NULL, -- department number
DEPTNAME VARCHAR(29), -- department name
MGRNO CHAR(6), -- manager number
ADMRDEPT CHAR(3) -- admin department number
```

部門長が存在せず、部門 A00 に報告を行う COMPLAINTS という名前の部門番号 G01 を挿入するには、EXECUTE ステートメントを実行する前に、構造体 INSERT\_DA は表 33 中の値を持っていないければなりません。

表 33.

| SQLDA フィールド | 値              |
|-------------|----------------|
| SQLDAID     | SQLDA          |
| SQLDABC     | 192 (注 1 を参照。) |
| SQLN        | 4              |
| SQLD        | 4              |
| SQLTYPE     | 452            |
| SQLLEN      | 3              |
| SQLDATA     | G01 を指すポインタ    |
| SQLIND      | (注 2 を参照。)     |
| SQLNAME     |                |

表 33. (続き)

| SQLDA フィールド                                                                                                 | 値                  |
|-------------------------------------------------------------------------------------------------------------|--------------------|
| SQLTYPE                                                                                                     | 449                |
| SQLLEN                                                                                                      | 29                 |
| SQLDATA                                                                                                     | COMPLAINTS を指すポインタ |
| SQLIND                                                                                                      | 0 を指すポインタ          |
| SQLNAME                                                                                                     |                    |
| SQLTYPE                                                                                                     | 453                |
| SQLLEN                                                                                                      | 6                  |
| SQLDATA                                                                                                     | (注 3 を参照。)         |
| SQLIND                                                                                                      | -1 を指すポインタ         |
| SQLNAME                                                                                                     |                    |
| SQLTYPE                                                                                                     | 453                |
| SQLLEN                                                                                                      | 3                  |
| SQLDATA                                                                                                     | A00 を指すポインタ        |
| SQLIND                                                                                                      | 0 を指すポインタ          |
| SQLNAME                                                                                                     |                    |
| 注:                                                                                                          |                    |
| 1. この値は、32 ビット・アプリケーションで PREPARE が実行される場合を想定しています。64 ビット・アプリケーションで PREPARE が実行される場合は、SQLDABC の値は 240 になります。 |                    |
| 2. SQLTYPE は NULL 不可データ・タイプを識別するので、SQLIND 中のこの SQLVAR の値は無視されます。                                            |                    |
| 3. SQLIND の値は、SQLDATA 中のこの SQLVAR の値が NULL 値であることを識別するので、この値は無視されます。                                        |                    |

## REFRESH TABLE

REFRESH TABLE ステートメントは、マテリアライズ照会表内のデータをリフレッシュします。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

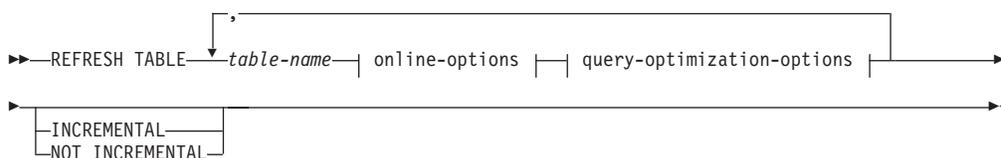
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 表に対する CONTROL 特権

## REFRESH TABLE

- SYSADM または DBADM 権限

### 構文



#### online-options:



#### query-optimization-options:



### 説明

#### table-name

リフレッシュする表を指定します。

名前 (暗黙的または明示的なスキーマ名を含む) は、現行サーバーにすでに存在する表を指定していなければなりません。表は、REFRESH TABLE ステートメントを許可していなければなりません (SQLSTATE 42809)。これには、次のステートメントで定義したマテリアライズ照会表が含まれます。

- REFRESH IMMEDIATE
- REFRESH DEFERRED

#### online-options

処理中の表のアクセス可能性を指定します。

#### ALLOW NO ACCESS

他のユーザーは、非コミット読み取り分離レベルを使用している場合を除き、更新中の表にアクセスできないことを指定します。

#### ALLOW READ ACCESS

他のユーザーは更新中の表に対して読み取り専用アクセスを持つことを指定します。

#### ALLOW WRITE ACCESS

他のユーザーは更新中の表に対して読み取り/書き込みアクセスを持つことを指定します。

ALLOW READ ACCESS オプションまたは ALLOW WRITE ACCESS オプションを使用する場合は、ロック・タイムアウトが原因でステートメント全体がロ

ールバックされる事態を避けるために、REFRESH TABLE ステートメントを実行する前に、SET CURRENT LOCK TIMEOUT ステートメントを (WAIT オプションを指定して) 実行することによって、後でその特殊レジスターを元の値にリセットすることをお勧めします。ただし、CURRENT LOCK TIMEOUT レジスターは、すべてのロック・タイプではなく、特定セットのロック・タイプだけに影響を与えます。

#### *query-optimization-options*

**REFRESH DEFERRED** マテリアライズ照会表のリフレッシュに関する照会最適化オプションを指定します。

#### **ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY**

CURRENT REFRESH AGE 特殊レジスターが「ANY」に設定されている場合に、*table-name* のリフレッシュで REFRESH DEFERRED マテリアライズ照会表を使用することによって、*table-name* のリフレッシュに使用する照会を最適化できるようにすることを指定します。*table-name* が REFRESH DEFERRED マテリアライズ照会表でない場合は、エラーが戻されます (SQLSTATE 428FH)。REFRESH IMMEDIATE マテリアライズ照会表は、常に照会の最適化のために考慮されます。

#### **INCREMENTAL**

基礎表のデルタ部分 (ある場合) か、関連したステージング表の内容 (この表があり、内容が一貫している場合) だけを考慮する方法での、表の増分リフレッシュを指定します。この要求が満たされない場合 (たとえば、システムがマテリアライズ照会表定義を完全に再計算する必要があると判断する場合)、エラー (SQLSTATE 55019) が戻されます。

#### **NOT INCREMENTAL**

マテリアライズ照会表の定義を再計算する方法での、表のフル・リフレッシュを指定します。

INCREMENTAL と NOT INCREMENTAL をどちらも指定しない場合、システムは増分処理が可能かどうかを判断します。それが可能でなければ、フル・リフレッシュが実行されます。リフレッシュ対象のマテリアライズ照会表に関するステージング表がある場合に、ステージング表がペンディング状態のため増分処理ができないと、エラーが戻されます (SQLSTATE 428A8)。ステージング表かマテリアライズ照会表が不整合な状態の場合は、フル・リフレッシュが実行されます。不整合でない場合は、ステージング表の内容を使用して増分処理が行われます。

#### **規則**

- 1 つ以上のニックネームを参照するマテリアライズ照会表に対して REFRESH TABLE を実行する場合は、データ・ソースの表から選択する権限がステートメントの許可 ID になければなりません (SQLSTATE 42501)。

#### **注**

- このステートメントを使用して、基礎表のロード、アタッチ、またはデタッチが行われた REFRESH IMMEDIATE マテリアライズ照会表をリフレッシュする場合には、基礎表のデルタ部分を使用してマテリアライズ照会表の増分リフレッシュを行うことをシステムが選択する場合があります。このステートメントを使用して、ステージング表をサポートしている REFRESH DEFERRED マテリアライズ

## REFRESH TABLE

照会表をリフレッシュする場合には、ステージング表にキャプチャーされた基礎表のデルタ部分を使用してマテリアライズ照会表の増分リフレッシュを行うことをシステムが選択する場合があります。ただし、データの保全性を保証するために、この最適化を実行できずにフル・リフレッシュ（つまり、マテリアライズ照会表の定義の再計算）を行う必要が生じる場合もあります。INCREMENTAL オプションを指定して増分保守を明示的に要求することもできます。この最適化を実行できない場合は、システムはエラーを戻します (SQLSTATE 55019)。

- ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY オプションを使用する場合は、REFRESH DEFERRED マテリアライズ照会表のリフレッシュの順序が正しいことを確認してください。たとえば、2 つのマテリアライズ照会表 MQT1 と MQT2 があり、それぞれのマテリアライズ照会が同じ基礎表を共有するとします。この場合、MQT2 のマテリアライズ照会は、基礎表ではなく MQT1 を使用して計算できます。この 2 つのマテリアライズ照会表をリフレッシュするために別々のステートメントを使用し、MQT2 を最初にリフレッシュする場合、システムは、MQT2 のリフレッシュのために、まだリフレッシュされていない MQT1 の内容を使用することを選択する可能性があります。その場合、MQT1 には現在のデータが入りますが、両方のリフレッシュをほとんど同時に実行したとしても、MQT2 には失効したデータが入る可能性があります。1 つではなく 2 つの REFRESH ステートメントを使用する場合は、MQT1 を最初にリフレッシュするのが正しい順序になります。
- マテリアライズ照会表にステージング表が関連付けられている場合、リフレッシュが正常に実行されるとそのステージング表は整理されます。

---

## RELEASE (接続)

RELEASE (接続) ステートメントは、1 つまたは複数の接続を解放ペンディング状態にします。

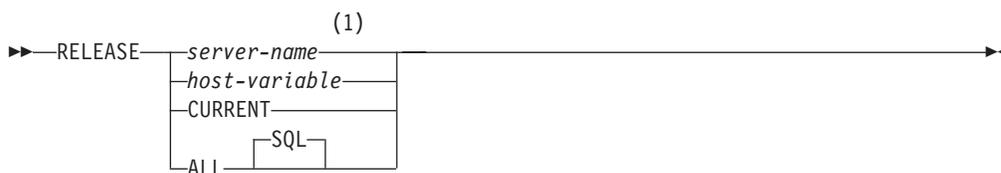
### 呼び出し

対話式 SQL 機能には外見上対話式の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

### 許可

必要ありません。

### 構文



注:

- 1 CURRENT または ALL という名前のアプリケーション・サーバーは、ホスト変数または区切り ID を使用してのみ指定することができます。

## 説明

*server-name* または *host-variable*

*server-name* (サーバー名) またはその *server-name* を含む *host-variable* (ホスト変数) によって、アプリケーション・サーバーを指定します。

*host-variable* (ホスト変数) を指定する場合、それは、長さ属性が 8 以下の文字ストリング変数でなければならず、標識変数を含めることはできません。その *host-variable* に入る *server-name* は、左寄せする必要がある、引用符で区切ることはできません。

*server-name* は、アプリケーション・サーバーを指定するデータベース別名である点に注意してください。この名前は、アプリケーション・リクエストのローカル・ディレクトリーにリストされている必要があります。

指定されたデータベース別名、またはホスト変数に含まれているデータベース別名は、そのアプリケーション・プロセスの既存の接続を指定するものでなければなりません。データベース別名が既存の接続を指定していない場合、エラー (SQLSTATE 08003) になります。

## CURRENT

アプリケーション・プロセスの現行接続を指定します。アプリケーション・プロセスは、接続された状態でなければなりません。接続されていない場合、エラー (SQLSTATE 08003) になります。

## ALL または ALL SQL

アプリケーション・プロセスの既存のすべての接続を指定します。この形式の RELEASE ステートメントの使用により、アプリケーション・プロセスの既存のすべての接続が解放ペンディング状態になります。そのような場合、すべての接続は、次回のコミット操作の過程で破棄されることになります。ステートメント実行時に接続が存在していない場合でも、エラーまたは警告のメッセージは出されません。

## 例

例 1: IBMSTHDB への SQL 接続は、アプリケーションではもはや必要でなくなりました。以下のステートメントを実行すると、次のコミット操作の過程でその接続が破棄されることになります。

```
EXEC SQL RELEASE IBMSTHDB;
```

例 2: 現行の接続は、アプリケーションでもはや必要でなくなりました。以下のステートメントを実行すると、次のコミット操作の過程でその接続が破棄されることになります。

```
EXEC SQL RELEASE CURRENT;
```

例 3: アプリケーションがコミット後にデータベースにアクセスする必要がなく、実行はしばらく継続する場合、不必要に接続を続けないようにした方が得策です。コミット時にすべての接続が破棄されるようにするために、コミット前に次のステートメントを実行できます。

```
EXEC SQL RELEASE ALL;
```

---

## RELEASE SAVEPOINT

RELEASE SAVEPOINT ステートメントは、指定されたセーブポイントの保持を、アプリケーションがもはや必要としなくなったことを指示するために使用されます。このステートメントが呼び出されると、そのセーブポイントまでロールバックすることはできなくなります。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文

```
▶▶—RELEASE—TO—SAVEPOINT—savepoint-name—▶▶
```

### 説明

*savepoint-name*

解放するセーブポイントを指定します。指名されたセーブポイント内でネストされているセーブポイントもすべて解放されます。そのセーブポイントおよびその内部でネストされているセーブポイントへのロールバックは不可能になります。現行のセーブポイント・レベルに名前付きセーブポイントがない場合 (SAVEPOINT ステートメントの『規則』の節を参照) は、エラーが戻されます (SQLSTATE 3B001)。 *savepoint-name* を指定する際に、'SYS' で始めることはできません (SQLSTATE 42939)。

### 注

- 同じセーブポイントの名前を指定した以前の SAVEPOINT ステートメントで UNIQUE キーワードが指定されたかどうかに関係なく、一度解放されたセーブポイントの名前は他の SAVEPOINT ステートメントでも再使用できるようになります。

### 例

例 1: SAVEPOINT1 という名前のセーブポイントを解放します。

```
RELEASE SAVEPOINT SAVEPOINT1
```

---

## RENAME TABLESPACE

RENAME TABLESPACE ステートメントは、既存の表スペースの名前を変更します。

## 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

## 許可

このステートメントの許可 ID が持つ特権には、SYSCTRL または SYSADM のいずれかの権限が含まれている必要があります。

## 構文

```
►►—RENAME—TABLESPACE—source-tablespace-name—TO—target-tablespace-name—◄◄
```

## 説明

### *source-tablespace-name*

1 つの部分からなる名前で、名前を変更する既存の表スペースを指定します。これは、SQL ID です (通常 ID または区切り ID)。表スペース名は、カタログ内にすでに存在している表スペースを識別するものでなければなりません (SQLSTATE 42704)。

### *target-tablespace-name*

表スペースに 1 つの部分からなる新しい名前を指定します。これは、SQL ID です (通常 ID または区切り ID)。新しく指定する表スペース名は、カタログ内にすでに存在する表スペースを識別するものであってはならず (SQLSTATE 42710)、また、'SYS' から始まる名前を指定することもできません (SQLSTATE 42939)。

## 規則

- SYSCATSPACE 表スペースの名前を変更することはできません (SQLSTATE 42832)。
- 「ロールフォワード・ペンディング」状態または「ロールフォワード進行中」状態にある表スペースの名前は変更できません (SQLSTATE 55039)。

## 注

- 表スペースの名前を変更すると、表スペースの最短のリカバリー時間が、名前変更の行われた時点に更新されます。これにより、表スペースのレベルでロールフォワードを実行すると、最低でもこの時点までロールフォワードされることになります。
- バックアップの作成後にバックアップ・イメージで名前の変更を行った場合は、バックアップ・イメージから表スペースをリストアするときに、新しい表スペース名を使用する必要があります。

## 例

表スペースの名前 USERSPACE1 を DATA2000 に変更します。

```
RENAME TABLESPACE USERSPACE1 TO DATA2000
```

## RENAME

RENAME ステートメントは、既存の表または索引の名前を変更します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 表または索引に対する CONTROL 特権
- スキーマに対する ALTERIN 特権
- SYSADM または DBADM 権限

### 構文



### 説明

#### TABLE *source-table-name*

名前を変更する既存の表を指定します。名前 (スキーマ名を含む) は、データベースにすでに存在する表を指定していなければなりません (SQLSTATE 42704)。この名前に、カタログ表 (SQLSTATE 42832)、マテリアライズ照会表、型付き表 (SQLSTATE 42997)、宣言されたグローバル一時表 (SQLSTATE 42995)、ニックネーム、または表や別名以外のオブジェクト (SQLSTATE 42809) を指定することはできません。TABLE キーワードはオプションです。

#### INDEX *source-index-name*

名前を変更する既存の索引を指定します。名前 (スキーマ名を含む) は、データベースにすでに存在する索引を指定していなければなりません (SQLSTATE 42704)。宣言されたグローバル一時表上の索引の名前は指定できません (SQLSTATE 42995)。スキーマ名は SYSIBM、SYSCAT、SYSFUN、または SYSSTAT であってはなりません (SQLSTATE 42832)。

#### *target-identifier*

表または索引の新しい名前をスキーマ名を付けずに指定します。ソース・オブジェクトのスキーマ名が、オブジェクトの新しい名前の修飾に使用されます。修飾された名前が、データベースにすでに存在する表、ビュー、別名、または索引を指定するものであってはなりません (SQLSTATE 42710)。

## 規則

表の名前を変更する場合、ソース表は以下に該当してはなりません。

- 既存のビュー定義またはマテリアライズ照会表定義で参照されている
- 既存のトリガーのトリガー SQL ステートメントで参照されているか、既存のトリガーの対象の表である
- SQL 関数で参照されている
- チェック制約がある
- ID 列以外に生成列がある
- 参照保全制約における親表または従属表である
- 既存の参照列の有効範囲内である
- 分解が可能になっている XSR オブジェクトによって参照されている

ソース表が上記の条件の 1 つまたは複数に違反している場合、エラー (SQLSTATE 42986) が戻されます。

索引の名前を変更する場合:

- 型付き表の基になっているインプリメンテーション表のシステム生成索引を、ソース索引にすることはできません (SQLSTATE 42858)。

## 注

- カタログ項目が更新され、新しい表名または索引名が反映されます。
- ソース表名または索引名に関連するすべての 許可は、新しい表名または索引名に転送 されます (許可カタログ表が適切に更新されます)。
- ソース表に対して定義された索引は、新しい表に転送 されます (索引カタログ表が適切に更新されます)。
- RENAME TABLE を行うと、ソース表に従属するパッケージはいずれも無効になります。 RENAME INDEX を行うと、ソース索引に従属するパッケージはいずれも無効になります。
- *source-table-name* として別名を使用する場合、その別名は表名に解決されなければなりません。表の名前は、その表のスキーマの中で変更されます。別名は RENAME ステートメントによって変更されず、従来の表名を引き続き指します。
- 主キー制約またはユニーク制約のある表の名前は、主キーまたはユニーク制約がいずれも外部キーによって参照されていない場合に変更できます。

## 例

EMP 表の名前を EMPLOYEE に変更します。

```
RENAME TABLE EMP TO EMPLOYEE
RENAME TABLE ABC.EMP TO EMPLOYEE
```

索引 NEW-IND の名前を IND に変更します。

```
RENAME INDEX NEW-IND TO IND
RENAME INDEX ABC.NEW-IND TO IND
```

## REPEAT

REPEAT ステートメントは、検索条件が真になるまでステートメントまたはステートメントのグループを実行します。

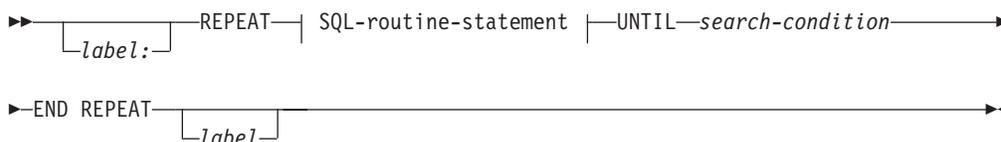
### 呼び出し

このステートメントは、SQL プロシージャに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

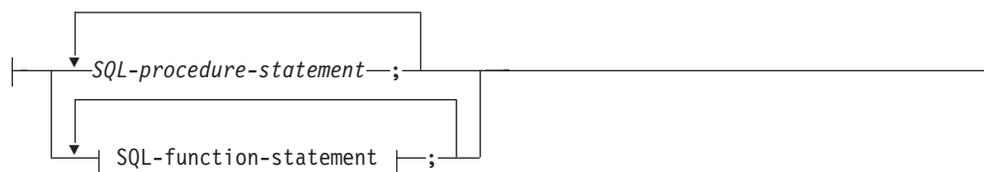
### 許可

REPEAT ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID には、REPEAT ステートメントに組み込まれている SQL ステートメントおよび検索条件を呼び出すために必要な特権がなければなりません。

### 構文



#### SQL-routine-statement:



### 説明

#### label

REPEAT ステートメントのラベルを指定します。開始ラベルを指定した場合、そのラベルを LEAVE および ITERATE ステートメントで指定することができます。終了ラベルを指定する場合、一致する開始ラベルも指定しなければなりません。

#### SQL-procedure-statement

ループ内で実行する SQL ステートメントを指定します。SQL-procedure-statement は、SQL プロシージャのコンテキスト内でのみ使用できます。『コンパウンド SQL (プロシージャ)』で、SQL-procedure-statement を参照してください。

#### SQL-function-statement

ループ内で実行する SQL ステートメントを指定します。SQL-function-statement は、SQL 関数または SQL メソッドのコンテキスト内でのみ使用できます。『FOR』で、SQL-function-statement を参照してください。

*search-condition*

*search-condition* は、毎回、REPEAT ループの実行後に評価されます。条件が真であれば、ループは終了します。条件が不明または偽であれば、ループは続行されます。

**例**

REPEAT ステートメントは、*not\_found* 条件処理ルーチンが呼び出されるまで、表から行を取り出します。

```
CREATE PROCEDURE REPEAT_STMT(OUT counter INTEGER)
LANGUAGE SQL
BEGIN
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE v_firstnme VARCHAR(12);
 DECLARE v_midinit CHAR(1);
 DECLARE v_lastname VARCHAR(15);
 DECLARE at_end SMALLINT DEFAULT 0;
 DECLARE not_found CONDITION FOR SQLSTATE '02000';
 DECLARE c1 CURSOR FOR
 SELECT firstnme, midinit, lastname
 FROM employee;
 DECLARE CONTINUE HANDLER FOR not_found
 SET at_end = 1;
 OPEN c1;
 fetch_loop:
 REPEAT
 FETCH c1 INTO v_firstnme, v_midinit, v_lastname;
 SET v_counter = v_counter + 1;
 UNTIL at_end > 0
 END REPEAT fetch_loop;
 SET counter = v_counter;
 CLOSE c1;
END
```

---

**RESIGNAL**

RESIGNAL ステートメントは、ハンドラーがアクティブ化された条件を再通知するか、または条件をより高いレベルで処理できるように代替条件を発生させるために条件処理ルーチン内で使用されます。このステートメントにより、オプションのメッセージ・テキストと共に返される例外、警告、または未検出条件が生成されません。

**呼び出し**

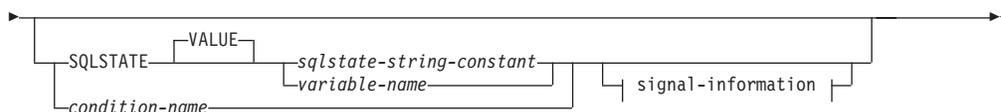
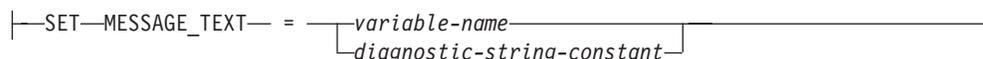
このステートメントは、SQL プロシージャーに組み込む方法でのみ使用可能です。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

**許可**

必要ありません。

**構文**

▶—RESIGNAL—▶

**signal-information:****説明****SQLSTATE VALUE** *sqlstate-string-constant*

指定された文字列定数が SQLSTATE を表します。この定数は、正確に 5 文字の文字列定数でなければならず、SQLSTATE の規則に従っていません。

- 各文字は、数字 ('0' から '9')、またはアクセントのない大文字の英字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は '00' にはできません。これは正常な完了を示します。

SQLSTATE がこれらの規則に従っていない場合には、エラーになります (SQLSTATE 428B3)。

**SQLSTATE VALUE** *variable-name*

指定する変数名はタイプ CHAR(5) でなければなりません。ステートメント実行時のこの変数の値が、*sqlstate-string-constant* に記述されているものと同じ規則に準拠する必要があります。SQLSTATE がこれらの規則に従っていない場合、エラーが戻されます (SQLSTATE 428B3)。

*condition-name*

条件の名前を指定します。

**SET MESSAGE\_TEXT =**

エラーまたは警告を記述する文字列を指定します。文字列は SQLCA の `sqlerrmc` フィールドに戻されます。実際の文字列が 70 バイトを超えている場合は、警告なしで切り捨てられます。

*variable-name*

コンパウンド・ステートメント内で宣言される必要のある SQL 変数を識別します。SQL 変数は CHAR または VARCHAR データ・タイプとして定義されていなければなりません。

*diagnostic-string-constant*

メッセージ・テキストを含む文字列定数を指定します。

**注**

- SQLSTATE 節または *condition-name* を使用せずに RESIGNAL ステートメントを指定すると、ハンドラーを呼び出したのと同じ条件が戻されます。この条件と関連付けられた SQLSTATE、SQLCODE および SQLCA は変更されません。

- RESIGNAL ステートメントが発行され、SQLSTATE または *condition-name* が指定された場合、以下のように、戻される SQLCODE は SQLSTATE 値に基づいています。
    - 指定した SQLSTATE クラスが '01' か '02' のいずれかである場合、警告か、見つからないことを示す条件が戻され、SQLCODE は +438 に設定されます。
    - それ以外の場合、例外条件が戻され、SQLCODE は -438 に設定されます。
- SQLCA の他のフィールドは、以下のように設定されます。
- sqlerrd フィールドはゼロに設定されます
  - sqlwarn フィールドはブランクに設定されます
  - sqlerrmc は MESSAGE\_TEXT の先頭の 70 バイトに設定されます
  - sqlerrml は sqlerrmc の長さか、SET MESSAGE\_TEXT 節が指定されていない場合にはゼロに設定されます
  - sqlerrp は ROUTINE に設定されます
- SQLSTATE 値の詳細は、『SIGNAL ステートメント』の『注』を参照してください。

## 例

以下の例では、ゼロ除算によるエラーを検出します。IF ステートメントは、SIGNAL ステートメントを使用して *overflow* 条件処理ルーチンを呼び出します。その条件処理ルーチンは、RESIGNAL ステートメントを使用して別の SQLSTATE 値をクライアント・アプリケーションに戻します。

```
CREATE PROCEDURE divide (IN numerator INTEGER,
 IN denominator INTEGER,
 OUT result INTEGER)

LANGUAGE SQL
BEGIN
 DECLARE overflow CONDITION FOR SQLSTATE '22003';
 DECLARE CONTINUE HANDLER FOR overflow
 RESIGNAL SQLSTATE '22375';
 IF denominator = 0 THEN
 SIGNAL overflow;
 ELSE
 SET result = numerator / denominator;
 END IF;
END
```

---

## RETURN

RETURN ステートメントはルーチンから戻るために使用されます。SQL 関数またはメソッドの場合、関数またはメソッドの結果を返します。SQL プロシージャの場合、オプションで整数状況値が戻されます。

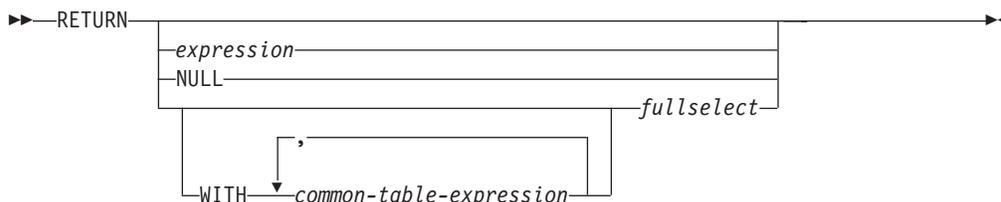
## 呼び出し

このステートメントは、SQL 関数、SQL メソッド、または SQL プロシージャに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

## 許可

RETURN ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID には、RETURN ステートメントに組み込まれている式または全選択を呼び出すために必要な特権がなければなりません。

## 構文



## 説明

### *expression*

ルーチンから戻される値を指定します。

- ルーチンが関数またはメソッドの場合は、*expression*、NULL または *fullselect* の指定が必要 (SQLSTATE 42631) であり、結果のデータ・タイプはルーチンの RETURNS タイプに割り当て可能でなければなりません (SQLSTATE 42866)。
- ルーチンが表関数の場合は、スカラー式 (スカラー *fullselect* 以外) は定義できません (SQLSTATE 428F1)。
- ルーチンがプロシージャの場合は、*expression* のデータ・タイプは INTEGER でなければなりません (SQLSTATE 428F2)。プロシージャは NULL または *fullselect* を返すことができません。

### NULL

関数またはメソッドが、RETURNS 節で定義されたデータ・タイプの NULL 値を返すことを指定します。NULL はプロシージャからの RETURN には指定できません。

### WITH *common-table-expression*

後続の *fullselect* で使用する共通表式を定義します。

### *fullselect*

関数に対して返される行を指定します。 *fullselect* 内の列数は、関数の結果の列数に一致していなければなりません (SQLSTATE 42811)。さらに、 *fullselect* の静的列タイプが、列に対する割り当て規則を使用して関数結果について宣言された列タイプに割り当てられていなければなりません (SQLSTATE 42866)。

*fullselect* はプロシージャからの RETURN には指定できません。

ルーチンがスカラー関数またはメソッドの場合、 *fullselect* は 1 つの列 (SQLSTATE 42823) と、 1 つの行 (SQLSTATE 21000) を返さなければなりません。

ルーチンが行関数の場合は、 1 つの行 (SQLSTATE 21505) を返さなければなりません。ただし、 1 つ以上の列が戻されることがあります。

ルーチンが表関数の場合は、1 つまたは複数の列を持つゼロ以上の行を返すことができます。

## 規則

- SQL 関数またはメソッドの実行は RETURN ステートメントで終わっていなければなりません (SQLSTATE 42632)。
- *dynamic-compound-statement* を使用する SQL 表または行関数では、RETURN ステートメントのみがコンパウンド・ステートメントの終わりで指定できます (SQLSTATE 429BD)。
- SQL プロシージャで、RETURN ステートメントは条件処理ルーチンの本体では許可されていません (SQLSTATE 42601)。

## 注

- プロシージャから値が返されると、呼び出し元は以下のようにして値にアクセスすることができます。
  - SQL プロシージャが他の SQL プロシージャから呼び出されたときに DB2\_RETURN\_STATUS を検索する GET DIAGNOSTICS ステートメントを使用する。
  - CLI アプリケーションでエスケープ節 CALL 構文 (?=CALL...) にある戻り値パラメーター・マーカーに結合されたパラメーターを使用する。
  - SQL プロシージャの CALL の処理後に SQLCA の sqlerrd[0] フィールドから直接。このフィールドは、SQLCODE がゼロまたは正の場合にのみ有効です (これ以外の場合は -1 の値と見なされます)。

## 例

RETURN ステートメントを使用して、SQL プロシージャから状況値を戻します。成功した場合は値ゼロが、失敗した場合は -200 が戻されます。

```
BEGIN
...
 GOTO FAIL
...
 SUCCESS: RETURN 0
 FAIL: RETURN -200
END
```

---

## REVOKE (データベース権限)

この形式の REVOKE ステートメントは、データベース全体に適用される権限を取り消します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

## REVOKE (データベース権限)

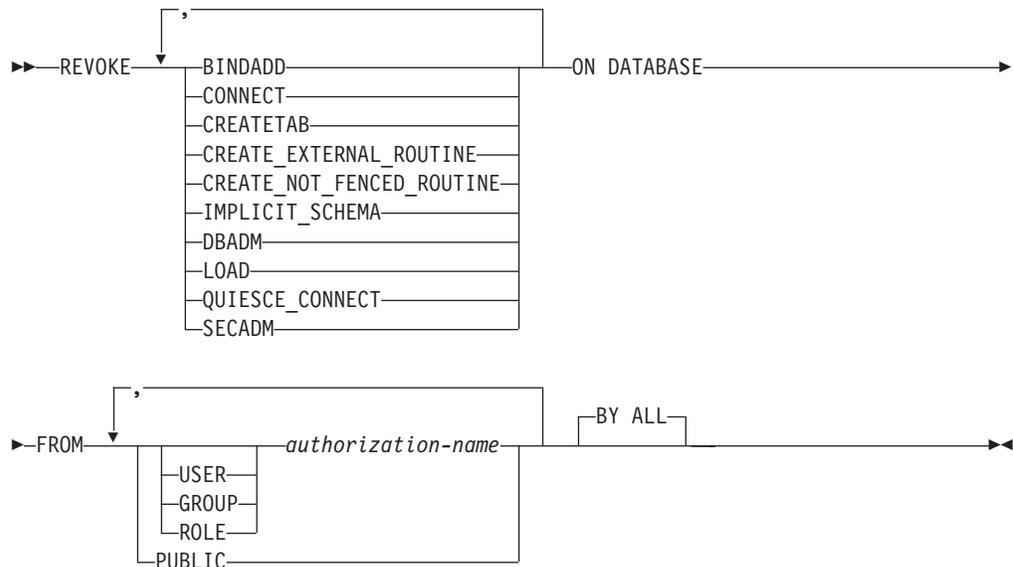
### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

以下の権限のいずれかを取り消すには、SYSADM 権限が必要です。

- DBADM
- SECADM

### 構文



### 説明

#### BINDADD

パッケージを作成する権限を取り消します。パッケージの作成者には自動的にそのパッケージに対する CONTROL 特権が与えられ、後でその BINDADD 権限が取り消されたとしてもその特権はそのまま保持されます。

DBADM 権限も取り消さない限り、DBADM 権限を与えられている *authorization-name* から BINDADD 権限を取り消すことはできません。

#### CONNECT

データベースにアクセスする権限を取り消します。

ユーザーから CONNECT 権限を取り消しても、そのユーザーに付与されていたデータベースのオブジェクトに対する特権には影響しません。後で再度そのユーザーに CONNECT 権限が付与された場合でも、以前に持っていた特権は、明示的に取り消さない限り、依然としてすべて有効です。

DBADM 権限も取り消さない限り、DBADM 権限を与えられている *authorization-name* から CONNECT 権限を取り消すことはできません (SQLSTATE 42504)。

**CREATETAB**

表を作成する権限を取り消します。表の作成者には自動的にその表に対する CONTROL 特権が与えられ、後で CREATETAB 権限が取り消されたとしても、その特権はそのまま保持します。

DBADM 権限も取り消さない限り、DBADM 権限を与えられている *authorization-name* から CREATETAB 権限を取り消すことはできません (SQLSTATE 42504)。

**CREATE\_EXTERNAL\_ROUTINE**

外部ルーチンを登録する権限を取り消します。外部ルーチンを登録し終わると、それ以降にそのルーチンを登録した許可 ID から CREATE\_EXTERNAL\_ROUTINE が取り消されても、そのまま保持されます。

DBADM または CREATE\_NOT\_FENCED\_ROUTINE 権限も取り消すのでない限り、DBADM または CREATE\_NOT\_FENCED\_ROUTINE 権限を与えられている *authorization-name* から CREATE\_EXTERNAL\_ROUTINE 権限を取り消すことはできません (SQLSTATE 42504)。

**CREATE\_NOT\_FENCED\_ROUTINE**

データベース・マネージャーの処理の中で実行するルーチンを登録する権限を取り消します。ルーチンが非 fenced としていったん登録されると、それ以降にそのルーチンを登録した許可 ID から CREATE\_NOT\_FENCED\_ROUTINE が取り消されるとしても、そのまま実行が続けられます。

DBADM 権限も取り消さない限り、DBADM 権限を与えられている *authorization-name* から CREATE\_NOT\_FENCED\_ROUTINE 権限を取り消すことはできません (SQLSTATE 42504)。

**IMPLICIT\_SCHEMA**

スキーマを暗黙的に作成する権限を取り消します。既存のスキーマにオブジェクトを作成する権限、または CREATE SCHEMA ステートメントを処理する権限には影響しません。

**DBADM**

DBADM 権限を取り消します。

DBADM 権限を PUBLIC から取り消すことはできません (PUBLIC に対して与えることができないので、当然取り消しもできません)。

**注意:**

DBADM 権限の取り消しによって、データベース内のオブジェクトに対して *authorization-name* が持っていた特権が自動的に取り消されることはなく、また元々 DBADM 権限が付与された際に暗黙的および自動的に付与された他のデータベース権限も取り消されることもありません。

**LOAD**

このデータベースで LOAD を実行する権限を取り消します。

**QUIESCE\_CONNECT**

静止中のデータベースにアクセスする権限を取り消します。

**SECADM**

セキュリティ管理者権限を取り消します。

## REVOKE (データベース権限)

### FROM

権限を誰から取り消すかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

PUBLIC から該当の権限を取り消します。

### BY ALL

指定された個々の特権を、その付与者にかかわらず、それらの特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者が *authorization-name* である SYSCAT.DBAUTH カタログ・ビュー内の指定されたオブジェクトのすべての行について、以下が該当します。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされます。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされます。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

## 注

- 特定の特権の取り消しにより、アクションを実行する権限が取り消されるとは限りません。PUBLIC、グループ、またはロールが他の特権を持っている場合、またはユーザーがより高いレベルの権限 (例えば DBADM) を持っている場合には、ユーザーは作業を続行できます。
- 互換性**

バージョン 8 より前のバージョンとの互換性を保つ場合は、オプション CREATE\_NOT\_FENCED を CREATE\_NOT\_FENCED\_ROUTINE に置き換えることができます。

**例**

例 1: USER6 はユーザーであり、グループではない場合に、ユーザー USER6 の表を作成する特権を取り消します。

```
REVOKE CREATETAB ON DATABASE FROM USER6
```

例 2: D024 という名前のグループのデータベースに対する BINDADD 権限を取り消します。SYSCAT.DBAUTH カタログ・ビューには、このグループの行として 2 つの行があります。その 1 つでは GRANTEETYPE が U、もう 1 つでは GRANTEETYPE が G になっています。

```
REVOKE BINDADD ON DATABASE FROM GROUP D024
```

この場合、GROUP キーワードの指定は必須です。指定しないとエラーになります (SQLSTATE 56092)。

例 3: ユーザー Walid のセキュリティー管理者権限を取り消します。

```
REVOKE SECADM ON DATABASE FROM USER Walid
```

**REVOKE (免除)**

この形式の REVOKE ステートメントは、ラベル・ベースのアクセス制御 (LBAC) のアクセス規則に対する免除を取り消します。

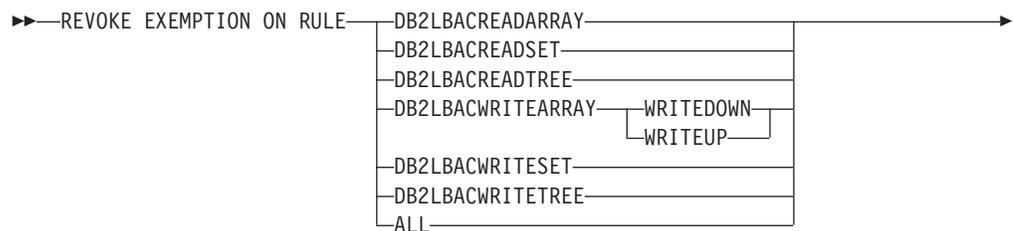
**呼び出し**

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

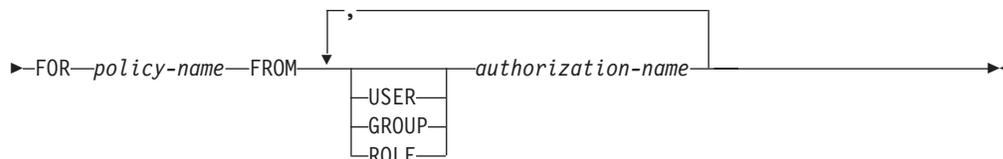
**許可**

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

**構文**



## REVOKE (免除)



### 説明

#### EXEMPTION ON RULE

アクセス規則に対する免除を取り消します。

#### DB2LBACREADARRAY

事前定義された DB2LBACREADARRAY 規則に対する免除を取り消します。

#### DB2LBACREADSET

事前定義された DB2LBACREADSET 規則に対する免除を取り消します。

#### DB2LBACREADTREE

事前定義された DB2LBACREADTREE 規則に対する免除を取り消します。

#### DB2LBACWRITEARRAY

事前定義された DB2LBACWRITEARRAY 規則に対する免除を取り消します。

#### WRITEDOWN

免除が下方への書き込みにのみ適用されることを指定します。

#### WRITEUP

免除が上方への書き込みにのみ適用されることを指定します。

#### DB2LBACWRITESET

事前定義された DB2LBACWRITESET 規則に対する免除を取り消します。

#### DB2LBACWRITETREE

事前定義された DB2LBACWRITETREE 規則に対する免除を取り消します。

#### ALL

事前定義されたすべての規則に対する免除を取り消します。

#### FOR *policy-name*

免除を取り消される対象のセキュリティー・ポリシーの名前を指定します。

#### FROM

免除を誰から取り消すかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループ名であることを指定します。

#### ROLE

*authorization-name* がロール名であることを指定します。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者を *authorization-name* とする  
SYSCAT.SECURITYPOLICYEXEMPTIONS カタログ・ビューで指定されたオブジェクトのすべての行について、次のことが言えます。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされず。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされず。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

## 例

例 1: セキュリティー・ポリシー DATA\_ACCESS のアクセス規則 DB2LBACREADSET に対するユーザー WALID の免除を取り消します。

```
REVOKE EXEMPTION ON RULE DB2LBACREADSET FOR DATA_ACCESS
FROM USER WALID
```

例 2: セキュリティー・ポリシー DATA\_ACCESS のアクセス規則 DB2LBACWRITEARRAY に対する免除を、WRITEDOWN オプションを指定して、ユーザー BOBBY から取り消します。

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEDOWN
FOR DATA_ACCESS FROM USER BOBBY
```

例 3: セキュリティー・ポリシー DATA\_ACCESS のアクセス規則 DB2LBACWRITEARRAY に対する免除を、WRITEUP オプションを指定して、ユーザー BOBBY から取り消します。

```
REVOKE EXEMPTION ON RULE DB2LBACWRITEARRAY WRITEUP
FOR DATA_ACCESS FROM USER BOBBY
```

---

## REVOKE (グローバル変数特権)

この形式の REVOKE ステートメントは、作成されたグローバル変数に 1 つ以上の特権を取り消します。

### 呼び出し

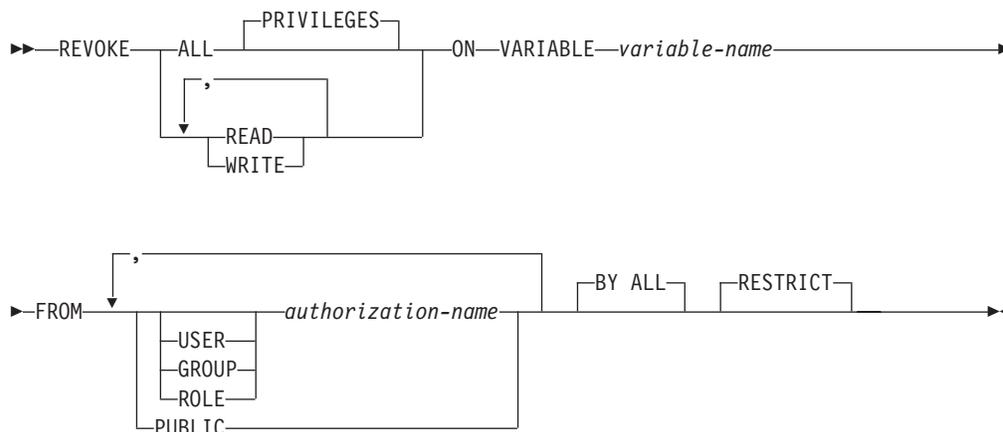
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

## REVOKE (グローバル変数特権)

### 構文



### 説明

#### ALL PRIVILEGES

指定されたグローバル変数に対して *authorization-name* に与えられている特権をすべて取り消します。ALL の指定がない場合、READ または WRITE を指定する必要があります。READ または WRITE は複数回指定することはできません。

#### READ

指定したグローバル変数の値を読み取る特権を取り消します。

#### WRITE

指定したグローバル変数に値を割り当てる特権を取り消します。

#### ON VARIABLE *variable-name*

1 つ以上の特権が取り消されるグローバル変数を指定します。 *variable-name* は、現在のサーバーに存在するグローバル変数を識別するものでなければなりません (SQLSTATE 42704)。

#### FROM

特権を誰から取り消すかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループであることを指定します。

#### ROLE

*authorization-name* が現行サーバーにおける既存のロールを識別することを指定します (SQLSTATE 42704)。

#### *authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

#### PUBLIC

PUBLIC から指定した特権を取り消します。

**BY ALL**

指定された各特権を、その付与者にかかわらず、それらの特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

**RESTRICT**

取り消される特権に依存するオブジェクトがある場合、ステートメントが失敗することを指定します。これがデフォルトの動作です。

**規則**

- 指定した *authorization-name* ごとに、キーワード **USER**、**GROUP**、および **ROLE** のいずれも指定されていない場合には、被認可者が *authorization-name* である、**SYSCAT.VARIABLEAUTH** カタログ・ビュー内の指定されたオブジェクトのすべての行において、次のようになります。
  - **GRANTEETYPE** が **U** の場合、**USER** であると見なされます。
  - **GRANTEETYPE** が **G** の場合、**GROUP** であると見なされます。
  - **GRANTEETYPE** が **R** の場合、**ROLE** であると見なされます。
  - **GRANTEETYPE** の値が同じでない場合、エラーが戻されます (**SQLSTATE 56092**)。
- グローバル変数が、取り消される特権に依存するいずれかの **SQL** 関数、**SQL** メソッド、プロシージャ、ビュー、トリガー、または別のグローバル変数に含まれている場合、その取り消し操作は失敗します (**SQLSTATE 42893**)。

**注**

- グローバル変数に対する **READ** 特権が取り消された場合、(例えば、**SET** ステートメントにより) グローバル変数の値を書き込むための従属関係を持つパッケージは影響を受けません。これは、グローバル変数への書き込みはそのグローバル変数に対する **WRITE** 特権により制御されているためです。
- グローバル変数に対する **WRITE** 特権が取り消された場合、グローバル変数の値を読み取るための従属関係を持つパッケージは影響を受けません。これは、グローバル変数からの読み取りはそのグローバル変数に対する **READ** 特権により制御されているためです。
- 特権を取り消したからといって、必ずしもアクションが実行できなくなるとは限りません。別のグループまたはロールのメンバーになっている場合、または **PUBLIC** によりユーザーが必要な特権を保持している場合には、操作を行うことができる場合があります。

**例**

グローバル変数 **MYSHEMA.MYJOB\_PRINTER** に対する **WRITE** 特権を、ユーザー **ZUBIRI** から取り消します。

```
REVOKE WRITE ON VARIABLE MYSCHEMA.MYJOB_PRINTER FROM ZUBIRI
```

**REVOKE (索引特権)**

この形式の **REVOKE** ステートメントは、索引に対する **CONTROL** 特権を取り消します。

## REVOKE (索引特権)

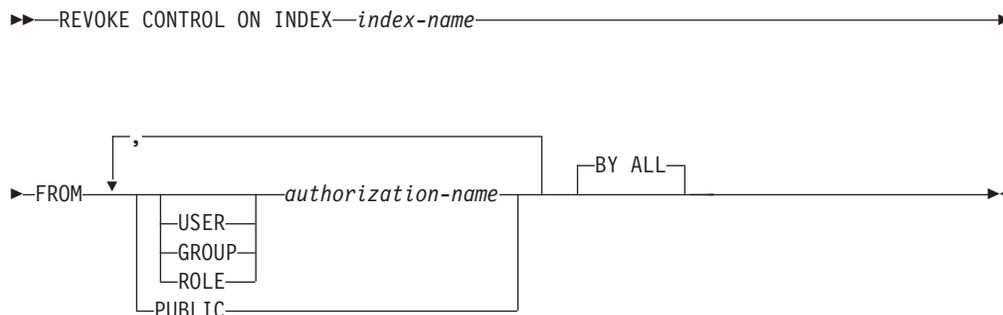
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

### 構文



### 説明

#### CONTROL

索引をドロップする特権を取り消します。これは、索引に対する CONTROL 特権であり、この特権は索引の作成者に自動的に付与されます。

#### ON INDEX *index-name*

その CONTROL 特権を取り消す索引の名前を指定します。

#### FROM

特権を誰から取り消すかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループ名であることを指定します。

#### ROLE

*authorization-name* がロール名であることを指定します。

#### *authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

#### PUBLIC

PUBLIC から特権を取り消します。

**BY ALL**

特権の付与者にかかわらず、その特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

**規則**

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者が *authorization-name* である SYSCAT.INDEXAUTH カタログ・ビュー内の指定されたオブジェクトのすべての行について、以下が該当します。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされます。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされます。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

**注**

- 特定の特権の取り消しにより、アクションを実行する権限が取り消されるとは限りません。PUBLIC、グループ、またはロールが他の特権を持っている場合、またはユーザーが索引のスキーマに対する ALTERIN などの権限を持っている場合には、ユーザーは作業を続行できます。

**例**

例 1: USER4 はユーザーであり、グループではない場合に、ユーザー USER4 から索引 DEPTIDX をドロップする特権を取り消します。

```
REVOKE CONTROL ON INDEX DEPTIDX FROM KIESLER
```

例 2: ユーザー CHEF とグループ WAITERS から、索引 LUNCHITEMS をドロップする特権を取り消します。

```
REVOKE CONTROL ON INDEX LUNCHITEMS
FROM USER CHEF, GROUP WAITERS
```

---

**REVOKE (パッケージ特権)**

この形式の REVOKE ステートメントは、パッケージに対する CONTROL、BIND、および EXECUTE 特権を取り消します。

**呼び出し**

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

## REVOKE (パッケージ特権)

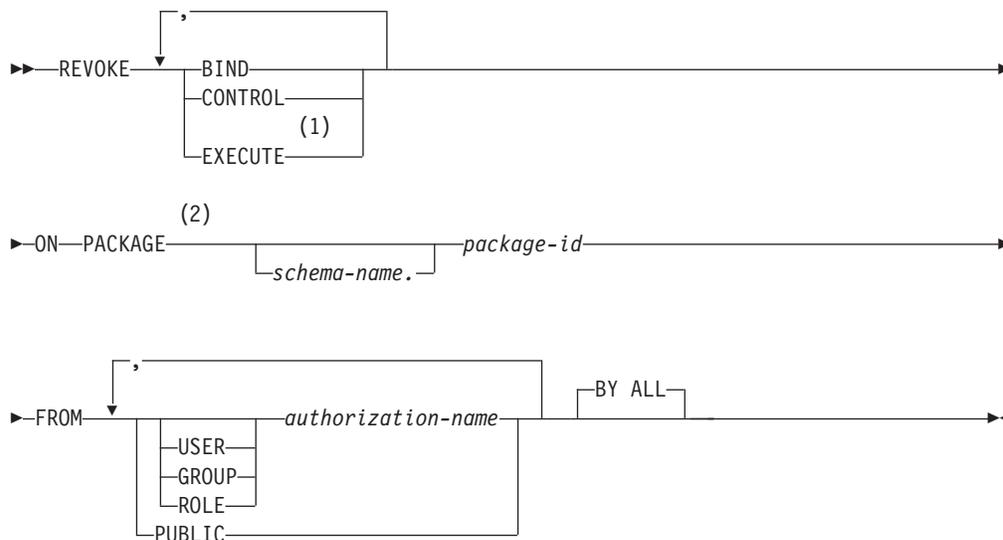
### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 参照されるパッケージに対する CONTROL 特権
- SYSADM または DBADM 権限

DBADM または SYSADM 権限は、CONTROL 特権を取り消すために必要です。

### 構文



#### 注:

- 1 EXECUTE の同義語として RUN を使用できます。
- 2 PACKAGE の同義語として PROGRAM を使用できます。

### 説明

#### BIND

指定されたパッケージに対する BIND または REBIND を実行する特権か、または指定されたパッケージの新しいバージョンを追加する特権を取り消します。

CONTROL 特権も取り消すのでない限り、パッケージに対する CONTROL 特権を与えられている *authorization-name* から BIND 特権を取り消すことはできません。

#### CONTROL

パッケージをドロップする特権、および他のユーザーに対してパッケージの特権を拡張する特権を取り消します。

CONTROL を取り消しても、他のパッケージ特権は取り消されません。

#### EXECUTE

パッケージを実行する特権を取り消します。

CONTROL 特権も取り消さない限り、パッケージに対する CONTROL 特権を与えられている *authorization-name* から EXECUTE 特権を取り消すことはできません。

**ON PACKAGE** *schema-name.package-id*

特権を取り消す対象のパッケージの名前を指定します。スキーマ名が指定されていない場合、パッケージ ID は暗黙的にデフォルト・スキーマで修飾されます。パッケージ特権の取り消しは、すべてのバージョンのパッケージに適用されます。

**FROM**

特権を誰から取り消すかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループ名であることを指定します。

**ROLE**

*authorization-name* がロール名であることを指定します。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

**PUBLIC**

PUBLIC から特権を取り消します。

**BY ALL**

指定された個々の特権を、その付与者にかかわらず、それらの特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

**規則**

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者が *authorization-name* である SYSCAT.PACKAGEAUTH カタログ・ビュー内の指定されたオブジェクトのすべての行について、以下が該当します。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされません。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされません。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

## REVOKE (パッケージ特権)

### 注

- 特定の特権の取り消しにより、アクションを実行する権限が取り消されるとは限りません。PUBLIC、グループ、またはロールが他の特権を持っている場合、またはユーザーがパッケージのスキーマに対する ALTERIN などの特権を持っている場合には、ユーザーは作業を続行できます。

### 例

例 1: PUBLIC から、パッケージ CORPDATA.PKGA に対する EXECUTE 権限を取り消します。

```
REVOKE EXECUTE
ON PACKAGE CORPDATA.PKGA
FROM PUBLIC
```

例 2: ユーザー FRANK および PUBLIC から、RRSP\_PKG パッケージに対する CONTROL 権限を取り消します。

```
REVOKE CONTROL
ON PACKAGE RRSP_PKG
FROM USER FRANK, PUBLIC
```

---

## REVOKE (ロール)

この形式の REVOKE ステートメントはロールを、ユーザー、グループ、またはその他のロールから取り消します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

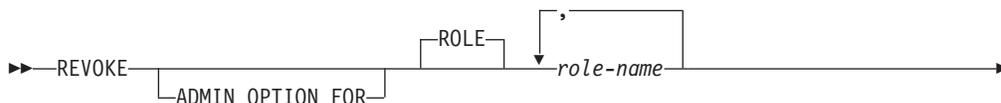
### 許可

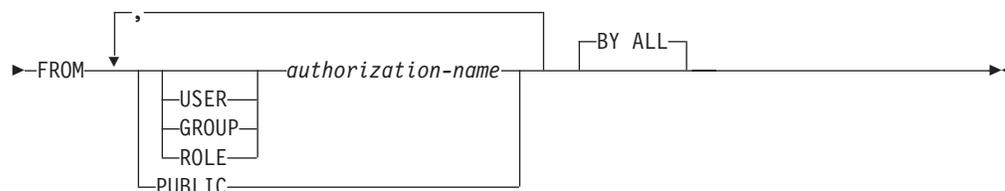
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- ロールに対する WITH ADMIN OPTION
- SECADM 権限

SECADM 権限は、*authorization-name* から ADMIN OPTION FOR *role-name*、またはそのロールに対する WITH ADMIN OPTION を持つ *authorization-name* から *role-name* を取り消すために必要です。

### 構文





## 説明

### ADMIN OPTION FOR

*role-name* に対する WITH ADMIN OPTION を取り消します。  
*authorization-name* または PUBLIC (PUBLIC が指定されている場合) は、  
*role-name* に対する WITH ADMIN OPTION を保持していなければなりません  
 (SQLSTATE 42504)。ADMIN OPTION FOR 節が指定される場合、ロールその  
 ものは取り消されず、ROLE *role-name* に対する WITH ADMIN OPTION だけ  
 が取り消されます。

### ROLE *role-name*

取り消すロールを指定します。 *role-name* は、*authorization-name* または  
 PUBLIC (PUBLIC が指定されている場合) (SQLSTATE 42504) に対してすでに  
 付与されている現行サーバー (SQLSTATE 42704) の既存のロールを識別するも  
 のでなければなりません。

### FROM

ロールを誰から取り消すかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループであることを指定します。

### ROLE

*authorization-name* が現行サーバーにおける既存のロールを識別することを  
 指定します (SQLSTATE 42704)。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定し  
 ます。この許可 ID のリストに、このステートメントを発行するユーザーの  
 許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

指定されたロールを PUBLIC から取り消します。

### BY ALL

ロールの付与者が誰であるかに関係なく、ロールが明示的に付与された指定の  
*authorization-name* からそれぞれ、*role-name* を取り消します。これがデフォルト  
 の動作です。

## 規則

- 指定されたそれぞれの *authorization-name* でキーワード USER、GROUP、または  
 ROLE のいずれも指定されない場合、被認可者を *authorization-name* とする  
 SYSCAT.ROLEAUTH カタログ・ビューで指定されたオブジェクトのすべての行  
 について、次のことが言えます。

## REVOKE (ロール)

- GRANTEETYPE が U の場合、USER であると見なされます。
- GRANTEETYPE が G の場合、GROUP であると見なされます。
- GRANTEETYPE が R の場合、ROLE であると見なされます。
- GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。
- あるロールがルーチンに対する EXECUTE 特権またはシーケンスに対する USAGE 特権を持っていて、パッケージ以外の SQL オブジェクトがそのルーチンまたはシーケンスに依存している場合には、*role-name* はそのロール、または *role-name* を含むそのようなロールを識別するものであってはなりません (SQLSTATE 42893)。 SQL オブジェクトの所有者は、*authorization-name* または *authorization-name* のメンバーである任意のユーザーのいずれかです (*authorization-name* はロール)。

### 注

- ロールが *authorization-name* または PUBLIC から取り消されると、そのロールが保持する特権はすべて、*authorization-name* またはそのロールを介した PUBLIC で使用できなくなります。
- ロールを取り消すことで必ずしも、そのロールに付与された特権による特定のアクションの実行が不可能になるわけではありません。 PUBLIC、ユーザーが所属するグループ、ユーザーに付与された別のロールに他の特権が与えられている場合、あるいは DBADM などのより上位の権限をユーザーが持っている場合には、ユーザーは作業を続行できるかもしれません。

### 例

例 1: ロール DOCTOR からロール INTERN を、さらにロール SPECIALIST からロール DOCTOR を取り消します。

```
REVOKE ROLE INTERN FROM ROLE DOCTOR

REVOKE ROLE DOCTOR FROM ROLE SPECIALIST
```

例 2: PUBLIC からロール INTERN を取り消します。

```
REVOKE ROLE INTERN FROM PUBLIC
```

例 3: ユーザー BOB とグループ TORONTO からロール SPECIALIST を取り消します。

```
REVOKE ROLE SPECIALIST FROM USER BOB, GROUP TORONTO BY ALL
```

---

## REVOKE (ルーチン特権)

この形式の REVOKE ステートメントは、ルーチン (関数、メソッド、またはプロシージャ) に対する特権を取り消します。

### 呼び出し

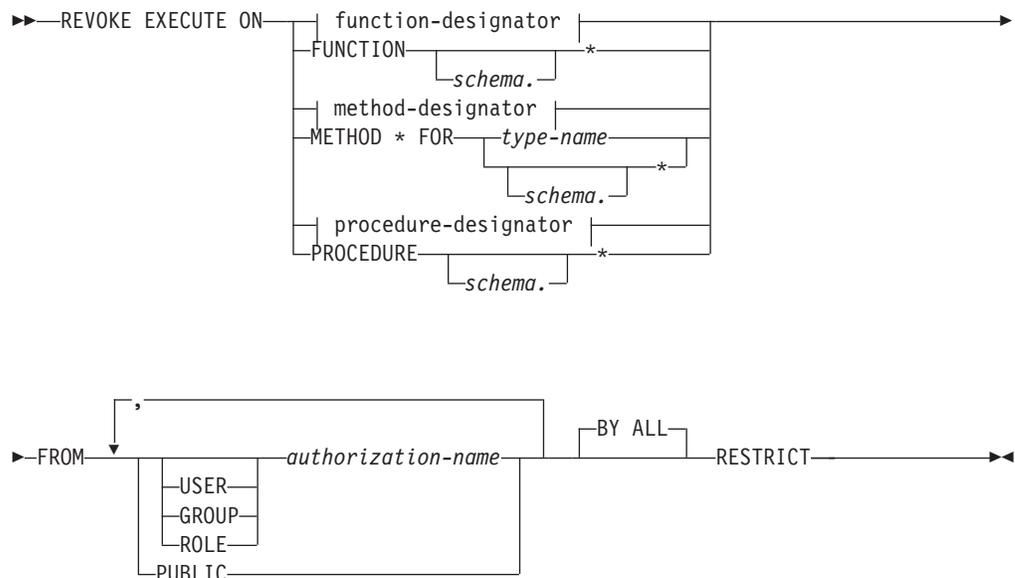
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の

実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

## 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

## 構文



## 説明

### EXECUTE

識別されたユーザー定義の関数、メソッド、またはプロシージャを実行する特権を取り消します。

#### *function-designator*

関数を一意的に識別します。

#### **FUNCTION** *schema.\**

スキーマ中の既存の関数と将来作成される関数に関する明示的な権限付与を識別します。 *schema.\** 特権を取り消しても、特定の関数に付与された特権は取り消されません。動的 SQL ステートメント中でスキーマが指定されていない場合は、CURRENT SCHEMA 特殊レジスタ中のスキーマが使用されます。静的 SQL ステートメント中でスキーマが指定されていない場合は、QUALIFIER プリコンパイル/ BIND オプション中のスキーマが使用されます。

#### *method-designator*

メソッドを一意的に識別します。

#### **METHOD \***

タイプ *type-name* の既存のメソッドと将来作成されるメソッドに関する明示的な権限付与を識別します。 \* 特権を取り消しても、特定のメソッドに付与された特権は取り消されません。

## REVOKE (ルーチン特権)

### **FOR** *type-name*

指定されたメソッドを検索する際のタイプを指定します。ここで指定される名前は、カタログにすでに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターの値が、修飾子のないタイプ名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/BIND オプションによって、修飾子のないタイプ名に修飾子が暗黙指定されます。 *type-name* の代わりにアスタリスク (\*) を使用して、スキーマ中のすべての既存のタイプと将来作成されるタイプの、すべての既存のメソッドと将来作成されるメソッドに対する明示的な権限付与を識別することもできます。アスタリスクを使用したメソッドおよび *type-name* に関する特権を取り消しても、特定のメソッドまたは特定のタイプのすべてのメソッドに付与された特権は取り消されません。

### *procedure-designator*

プロシーチャーを一意的に識別します。

### **PROCEDURE** *schema.\**

スキーマ中の既存のプロシーチャーと将来作成されるプロシーチャーに関する明示的な権限付与を識別します。 *schema.\** 特権を取り消しても、特定のプロシーチャーに付与された特権は取り消されません。動的 SQL ステートメント中でスキーマが指定されていない場合は、CURRENT SCHEMA 特殊レジスター中のスキーマが使用されます。静的 SQL ステートメント中でスキーマが指定されていない場合は、QUALIFIER プリコンパイル/BIND オプション中のスキーマが使用されます。

### **FROM**

EXECUTE 特権を誰から取り消すかを指定します。

### **USER**

*authorization-name* がユーザーであることを指定します。

### **GROUP**

*authorization-name* がグループ名であることを指定します。

### **ROLE**

*authorization-name* がロール名であることを指定します。

### *authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### **PUBLIC**

PUBLIC から EXECUTE 特権を取り消します。

### **BY ALL**

EXECUTE 特権の付与者にかかわらず、EXECUTE 特権を明示的に付与されたユーザーのうち、指定された人から取り消します。これがデフォルトの動作です。

**RESTRICT**

以下の両方が該当する場合に、EXECUTE 特権を取り消さないことを指定します (SQLSTATE 42893)。

- 指定されたルーチンがビュー、トリガー、制約、索引拡張、SQL 関数、SQL メソッド、またはトランスフォーム・グループ中で使用されているか、または指定されたルーチンがソース関数の SOURCE として参照されている。
- EXECUTE 特権がなくなると、ビュー、トリガー、制約、索引拡張、SQL 関数、SQL メソッド、トランスフォーム・グループ、またはソース関数の所有者が、指定されたルーチンを実行できなくなる。

**規則**

- スキーマ 'SYSIBM' または 'SYSFUN' を使って定義された関数やメソッドに対する EXECUTE 特権を取り消すことはできません (SQLSTATE 42832)。
- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者を *authorization-name* とする SYSCAT.ROUTINEAUTH カタログ・ビューで指定されたオブジェクトのすべての行について、次のことが言えます。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされます。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされます。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

**注**

- パッケージがルーチン (関数、メソッド、またはプロシージャ) に依存し、そのルーチンに対する EXECUTE 特権が PUBLIC、ユーザー、またはロールから取り消された場合、そのルーチンが関数またはメソッドである場合にはパッケージは作動不能になり、そのルーチンがプロシージャである場合にはパッケージは無効になります (パッケージ所有者がまだそのルーチンに対する EXECUTE 特権を保持している場合を除く)。パッケージ所有者は、次の場合に EXECUTE 特権を保持したままにできます。
  - パッケージ所有者に明示的に EXECUTE 特権が付与された場合
  - パッケージ所有者が EXECUTE 特権を保持するロールのメンバーである場合
  - EXECUTE 特権が PUBLIC に付与された場合

静的パッケージではグループ特権は考慮されないため、パッケージ所有者が属するグループが EXECUTE 特権を保持している場合でも、パッケージは作動不能 (関数またはメソッドである場合)、または無効 (プロシージャである場合) になります。

## REVOKE (ルーチン特権)

### 例

例 1: ユーザー JONES から、関数 CALC\_SALARY に対する EXECUTE 特権を取り消します。スキーマ中に CALC\_SALARY という名前の関数が 1 つだけ含まれていると想定しています。

```
REVOKE EXECUTE ON FUNCTION CALC_SALARY FROM JONES RESTRICT
```

例 2: 現行サーバー上のすべてのユーザーから、プロシージャ VACATION\_ACCR に対する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE ON PROCEDURE VACATION_ACCR FROM PUBLIC RESTRICT
```

例 3: HR (Human Resources) から、関数 NEW\_DEPT\_HIRES に対する EXECUTE 特権を取り消します。この関数には、2 つの入力パラメーターがあり、それぞれのパラメーターのタイプは INTEGER および CHAR(10) です。スキーマに NEW\_DEPT\_HIRES という名前の関数が複数あることを想定しています。

```
REVOKE EXECUTE ON FUNCTION NEW_DEPT_HIRES (INTEGER, CHAR(10))
FROM HR RESTRICT
```

例 4: ユーザー Jones から、タイプ EMPLOYEE のメソッド SET\_SALARY に対する EXECUTE 特権を取り消します。

```
REVOKE EXECUTE ON METHOD SET_SALARY FOR EMPLOYEE FROM JONES RESTRICT
```

---

## REVOKE (スキーマ特権)

この形式の REVOKE ステートメントは、スキーマに対する特権を取り消します。

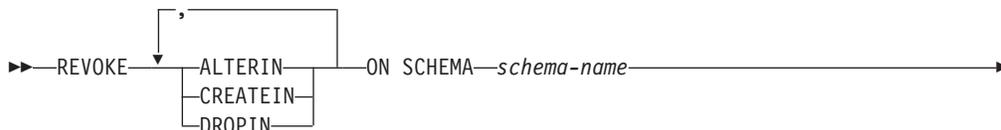
### 呼び出し

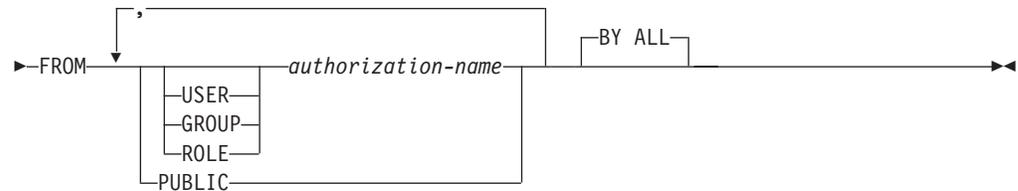
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

### 構文





## 説明

### ALTERIN

スキーマ中のオブジェクトの変更、またはコメント付けを行う特権を取り消します。

### CREATEIN

スキーマにオブジェクトを作成する特権を取り消します。

### DROPIN

スキーマのオブジェクトをドロップする特権を取り消します。

### ON SCHEMA *schema-name*

特権を取り消す対象のスキーマの名前を指定します。

### FROM

特権を誰から取り消すかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。

### *authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

PUBLIC から特権を取り消します。

### BY ALL

指定された個々の特権を、その付与者にかかわらず、それらの特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者が *authorization-name* である SYSCAT.SCHEMAAUTH カタログ・ビュー内の指定されたオブジェクトのすべての行について、以下が該当します。

## REVOKE (スキーマ特権)

- すべての行の GRANTEETYPE が「U」の場合、USER であると見なされ  
ます。
- すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされ  
ます。
- すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされま  
す。
- すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます  
(SQLSTATE 56092)。

### 注

- 特定の特権の取り消しにより、アクションを実行する権限が取り消されるとは限  
りません。PUBLIC、グループ、またはロールが他の特権を持っている場合、ま  
たはユーザーがより高いレベルの権限 (例えば DBADM) を持っている場合に  
は、ユーザーは作業を続行できます。

### 例

例 1: USER4 がユーザーで、グループではない場合に、ユーザー USER4 からスキ  
ーマ DEPTIDX にオブジェクトを作成する特権を取り消します。

```
REVOKE CREATEIN ON SCHEMA DEPTIDX FROM USER4
```

例 2: ユーザー CHEF とグループ WAITERS から、スキーマ LUNCH のオブジェ  
クトをドロップする特権を取り消します。

```
REVOKE DROPIN ON SCHEMA LUNCH
FROM USER CHEF, GROUP WAITERS
```

---

## REVOKE (セキュリティー・ラベル)

この形式の REVOKE ステートメントは、ラベル・ベースのアクセス制御 (LBAC)  
セキュリティー・ラベルを取り消します。

### 呼び出し

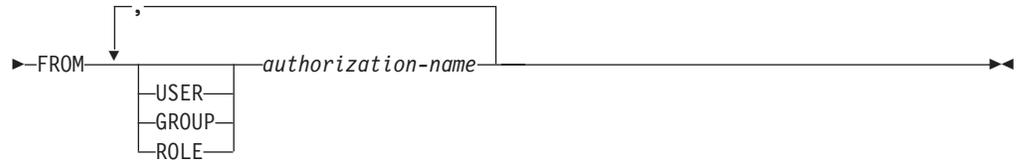
このステートメントはアプリケーション・プログラムに組み込むことができ、また  
動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の  
実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステ  
ートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要  
があります。

### 構文

```
▶▶—REVOKE SECURITY LABEL—security-label-name————▶▶
```



## 説明

### SECURITY LABEL *security-label-name*

セキュリティー・ラベル *security-label-name* を取り消します。名前は、セキュリティー・ポリシーを使って修飾する必要があり (SQLSTATE 42704)、現在のサーバー上に存在する (SQLSTATE 42704)、*authorization-name* によって保有されている (SQLSTATE 42504) セキュリティー・ラベルを識別するものでなければなりません。

### FROM

指定されたセキュリティー・ラベルを誰から取り消すかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。ロール名は、現行サーバーに存在するものでなければなりません (SQLSTATE 42704)。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者を *authorization-name* とする SYSCAT.SECURITYLABELACCESS カタログ・ビューで指定されたオブジェクトのすべての行について、次のことが言えます。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされます。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされます。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

## 例

例 1: セキュリティー・ポリシー DATA\_ACCESS の一部をなすセキュリティー・ラベル EMPLOYEESECLABEL をユーザー WALID から取り消します。

## REVOKE (セキュリティー・ラベル)

```
REVOKE SECURITY LABEL DATA_ACCESS.EMPLOYEESECLABEL
FROM USER WALID
```

## REVOKE (シーケンス特権)

この形式の REVOKE ステートメントは、シーケンスに対する特権を取り消します。

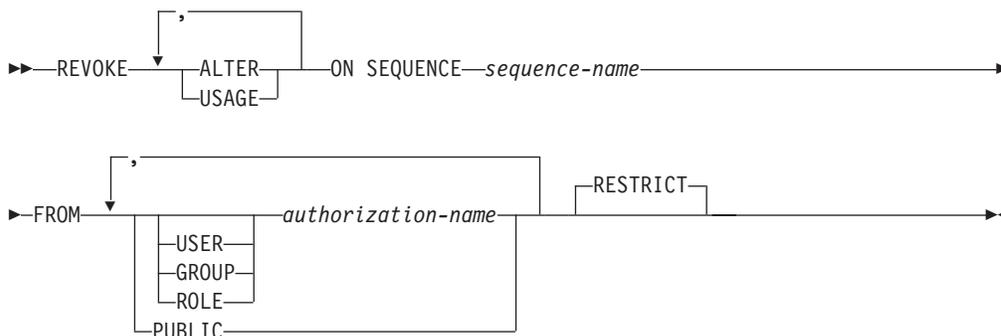
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。ただし、BIND オプション DYNAMICRULES BIND を適用する場合、ステートメントを動的に準備することはできません (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

### 構文



### 説明

#### ALTER

ALTER SEQUENCE ステートメントを使用して、シーケンスのプロパティを変更する特権、またはシーケンス番号の生成を再始動する特権を取り消します。

#### USAGE

*nextval-expression* または *prevval-expression* を使用してシーケンスを参照する特権を取り消します。

#### ON SEQUENCE *sequence-name*

指定された特権が取り消されるシーケンスを識別します。暗黙的または明示的スキーマ修飾子を含むシーケンス名は、現在のサーバーに存在するシーケンスを固有に識別していなければなりません。この名前によるシーケンスが存在しない場合、エラーが戻されます (SQLSTATE 42704)。

#### FROM

特権を誰から取り消すかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループ名であることを指定します。

**ROLE**

*authorization-name* がロール名であることを指定します。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

**PUBLIC**

PUBLIC から指定した特権を取り消します。

**RESTRICT**

このオプション・キーワードは、取り消される特権に依存するオブジェクトがある場合、ステートメントが失敗することを示します。

**規則**

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者を *authorization-name* とする SYSCAT.SEQUENCEAUTH カタログ・ビューで指定されたオブジェクトのすべての行について、次のことが言えます。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされます。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされます。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

**注**

- シーケンスに対する特権をパッケージのバインド時の許可 ID から取り消すと、許可 ID がそのシーケンスに対する特権を別の方法で (例えば、その特権を保持するロールのメンバーシップにより) 保持し続けるのでない限り、パッケージは無効になります。
- 特定の特権を取り消しても、アクションを実行する権限が必ずしも取り除かれるとは限りません。PUBLIC またはユーザーの属するグループに他の特権が与えられている場合、あるいは DBADM などのより上位の権限をユーザーが持っている場合には、ユーザーは作業を続行できます。

## REVOKE (シーケンス特権)

### 例

例 1: シーケンス GENERATE\_ID に対する USAGE 特権をユーザー ENGLES から取り消します。SYSCAT.SEQUENCEAUTH カタログ・ビューにはこのシーケンスとユーザーについての行が 1 つあり、その GRANTEETYPE の値は U です。

```
REVOKE USAGE ON SEQUENCE GENERATE_ID FROM ENGLES
```

例 2: 以前にすべてのローカル・ユーザーに与えられたシーケンス GENERATE\_ID に対する更新特権を取り消します。(特定のユーザーに対する特権付与は、影響を受けません。)

```
REVOKE ALTER ON SEQUENCE GENERATE_ID FROM PUBLIC
```

例 3: シーケンス GENERATE\_ID に対するすべての特権を、ユーザー PELLOW と MLI、およびグループ PLANNERS から取り消します。

```
REVOKE ALTER, USAGE ON SEQUENCE GENERATE_ID
FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

---

## REVOKE (サーバー特権)

この形式の REVOKE ステートメントは、指定したデータ・ソースにパススルー・モードでアクセスおよび使用する特権を取り消します。

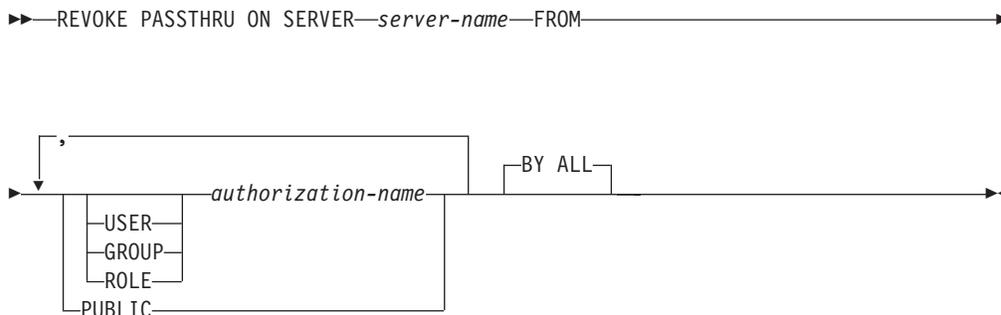
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

### 構文



## 説明

### SERVER *server-name*

パススルー・モードで使用する特権が取り消されるデータ・ソースを指定します。*server-name* (サーバー名) は、カタログに記述されているデータ・ソースを指定していなければなりません。

### FROM

特権を誰から取り消すかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。

*authorization-name,...*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

*server-name* にパススルーする特権を PUBLIC から取り消します。

### BY ALL

特権の付与者にかかわらず、その特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

## 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者が *authorization-name* である SYSCAT.PASSTHROUGH カタログ・ビュー内の指定されたオブジェクトのすべての行について、以下が該当します。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされます。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされます。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

## 例

例 1: USER6 が持っているデータ・ソース MOUNTAIN にパススルーする特権を取り消します。

```
REVOKE PASSTHRU ON SERVER MOUNTAIN FROM USER USER6
```

## REVOKE (サーバー特権)

例 2: グループ D024 が持っている、データ・ソース EASTWING にパススルーする特権を取り消します。

```
REVOKE PASSTHRU ON SERVER EASTWING FROM GROUP D024
```

グループ D024 のメンバーは、このグループ ID を使って EASTWING にパススルーすることはできなくなります。しかし、EASTWING にパススルーする特権をユーザー ID に持っているメンバーがいれば、それらのメンバーはこの特権を保持することができます。

---

## REVOKE (SETSESSIONUSER 特権)

この形式の REVOKE ステートメントは、1 つ以上の SETSESSIONUSER 特権を 1 つ以上の許可 ID から取り消します。

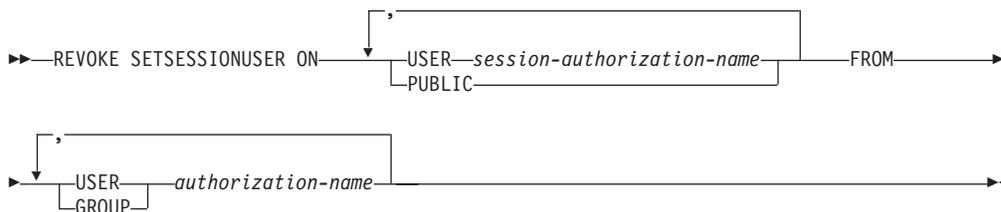
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、SECADM 権限が含まれている必要があります。

### 構文



### 説明

#### SETSESSIONUSER ON

新しい許可 ID を担うための特権を取り消します。

#### USER *session-authorization-name*

SET SESSION AUTHORIZATION ステートメントによって *authorization-name* が担うことのできる許可 ID を指定します。*session-authorization-name* は、グループではなく、*authorization-name* が担うことのできるユーザーを指定する必要があります (SQLSTATE 42504)。

#### PUBLIC

セッション許可を設定するすべての特権を取り消すことを指定します。

#### FROM

特権を誰から取り消すかを指定します。

**USER**

*authorization-name* がユーザーであることを指定します。

**GROUP**

*authorization-name* がグループ名であることを指定します。

*authorization-name,...*

1 人または複数のユーザーまたはグループの許可 ID をリストします。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

**例**

例 1: ユーザー PAUL は、セッション許可を WALID に対して設定する特権を保有しているので、ユーザー WALID として SQL ステートメントを実行する特権ももっています。以下のステートメントは、その特権を取り消します。

```
REVOKE SETSESSIONUSER ON USER WALID
FROM USER PAUL
```

例 2: ユーザー GUYLAINE は、セッション許可を BOBBY、RICK、または KEVIN に対して設定する特権を保有しているので、ユーザー BOBBY、RICK、または KEVIN として SQL ステートメントを実行する特権ももっています。以下のステートメントは、それらの許可 ID のうちの 2 つの使用特権を取り消します。このステートメントの実行後、GUYLAINE は KEVIN に対するセッション許可だけを設定できるようになります。

```
REVOKE SETSESSIONUSER ON USER BOBBY, USER RICK
FROM USER GUYLAINE
```

例 3: グループ ACCTG およびユーザー WALID は、任意の許可 ID に対してセッション許可を設定することができます。以下のステートメントは、ACCTG および WALID の両方の特権を取り消します。

```
REVOKE SETSESSIONUSER ON PUBLIC
FROM USER WALID, GROUP ACCTG
```

---

**REVOKE (表スペース特権)**

この形式の REVOKE ステートメントは、表スペースに対する USE 特権を取り消します。

**呼び出し**

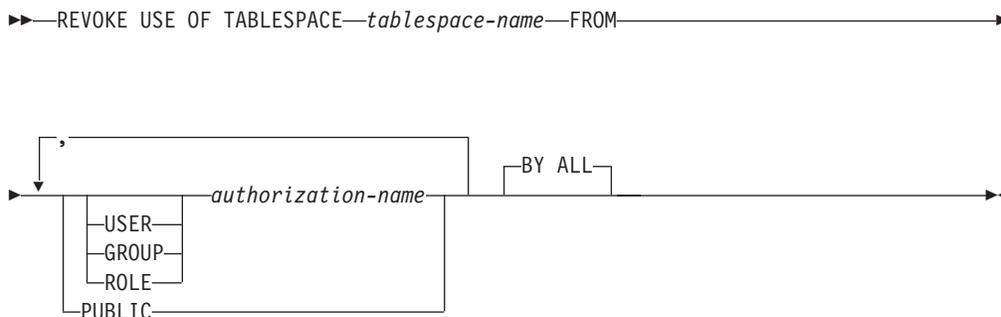
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

**許可**

このステートメントの許可 ID が持つ特権には、DBADM、SYSCTRL、または SYSADM 権限が含まれている必要があります。

## REVOKE (表スペース特権)

### 構文



### 説明

#### USE

表を作成する際に表スペースを指定したり、デフォルトの表スペースを使用したりするための特権を取り消します。

#### OF TABLESPACE *tablespace-name*

どの表スペースに対する `USE` 特権を取り消すかを指定します。ここで、`SYSCATSPACE` (SQLSTATE 42838) や `SYSTEM TEMPORARY` 表スペース (SQLSTATE 42809) を指定することはできません。

#### FROM

`USE` 特権を誰から取り消すかを指定します。

#### USER

*authorization-name* がユーザーであることを指定します。

#### GROUP

*authorization-name* がグループ名であることを指定します。

#### ROLE

*authorization-name* がロール名であることを指定します。

#### *authorization-name*

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

#### PUBLIC

`PUBLIC` から `USE` 特権を取り消します。

#### BY ALL

特権の付与者にかかわらず、その特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

### 規則

- 指定したそれぞれの *authorization-name* に関して、`USER`、`GROUP`、`ROLE` のいずれも指定されていない場合には、次のようになります。
  - 被認可者を *authorization-name* とする `SYSCAT.TBSPACEAUTH` カタログ・ビューで指定されたオブジェクトのすべての行について、次のことが言えます。

- すべての行の GRANTEETYPE が「U」の場合、USER であると見なされ  
ます。
- すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされ  
ます。
- すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされま  
す。
- すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます  
(SQLSTATE 56092)。

### 注

- USE 特権が取り消されたからといって、必ずしもその表スペースに表を作成する  
権限が取り消されるとは限りません。 PUBLIC またはグループが USE 特権を保  
持している場合、またはユーザーが DBADM などのより上位の権限を持ってい  
る場合は、引き続きその表スペースに表を作成することができます。

### 例

例 1: ユーザー BOBBY から、表スペース PLANS で表を作成する特権を取り消し  
ます。

```
REVOKE USE OF TABLESPACE PLANS FROM USER BOBBY
```

---

## REVOKE (表、ビュー、またはニックネーム特権)

この形式の REVOKE ステートメントは、表、ビュー、またはニックネームに対す  
る特権を取り消します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また  
動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の  
実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステ  
ートメントです (SQLSTATE 42509)。

### 許可

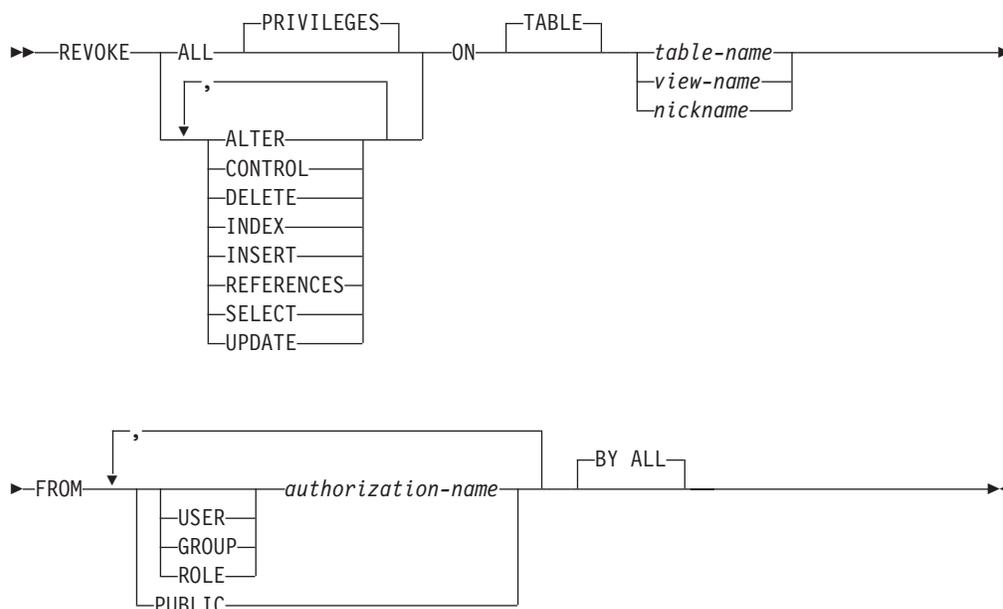
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のい  
ずれかが含まれていなければなりません。

- 参照されている表、ビュー、またはニックネームに対する CONTROL 特権
- SYSADM または DBADM 権限

DBADM または SYSADM 権限は、CONTROL 特権を取り消すため、またはカタログ  
表およびビューに対する特権を取り消すために必要です。

### 構文

## REVOKE (表、ビュー、またはニックネーム特権)



### 説明

#### ALL または ALL PRIVILEGES

指定された表、ビュー、またはニックネームに対して *authorization-name* に与えられている特権をすべて (CONTROL を除く) 取り消します。

ALL を指定しない場合は、以下に示すキーワードのうち 1 つまたは複数指定する必要があります。各キーワードは、それぞれ説明されている特権を取り消しますが、その取り消しは ON 節に指定する表、ビュー、またはニックネームに当てはまる場合にのみ行われます。同じキーワードを複数回指定することはできません。

#### ALTER

基本表の定義への列の追加、表の主キーまたはユニーク制約の作成またはドロップ、表の外部キーの作成またはドロップ、表、ビュー、またはニックネームに対するコメントの追加や変更、チェック制約の作成またはドロップ、トリガーの作成、ニックネームに対する列オプションの追加、リセット、ドロップ、またはニックネームの列名やデータ・タイプの変更を行うための特権を取り消します。

#### CONTROL

表、ビュー、またはニックネームをドロップする権限、および表または索引に対して RUNSTATS ユーティリティを実行する権限を取り消します。

*authorization-name* から CONTROL 特権を取り消しても、そのオブジェクトに対してそのユーザーに付与されているその他の特権は取り消されません。

#### DELETE

表、更新可能なビュー、またはニックネームから行を削除する特権を取り消します。

#### INDEX

表の索引、またはニックネームの SPECIFICATION ONLY 指定の索引を作成する特権を取り消します。索引または SPECIFICATION ONLY 指定の索引の作成者には、その索引または SPECIFICATION ONLY 指定の索引に対する

## REVOKE (表、ビュー、またはニックネーム特権)

CONTROL 特権が自動的に与えられます (これにより、作成者は索引または SPECIFICATION ONLY 指定の索引をドロップできます)。さらに、INDEX 特権が取り消されても、作成者は CONTROL 特権をそのまま保持します。

### INSERT

表、更新可能なビュー、またニックネームに行を挿入したり、IMPORT ユーティリティーを実行したりする特権を取り消します。

### REFERENCES

親として表を参照する外部キーの作成、またはドロップを行う特権を取り消します。列レベルの REFERENCES 特権もすべて取り消されます。

### SELECT

表またはビューからの行の検索、表に対するビューの作成、および表またはビューに対して EXPORT ユーティリティーを実行する特権を取り消します。

SELECT 特権を取り消すと、ビューによっては作動不能になるものがあります。(作動不能なビューについては、『CREATE VIEW』を参照してください。)

### UPDATE

表、更新可能なビュー、またはニックネームの行を更新する特権を取り消します。列レベルの UPDATE 特権もすべて取り消されます。

### ON TABLE *table-name* または *view-name* または *nickname*

特権を取り消す表、ビュー、またはニックネームを指定します。 *table-name* を宣言済み一時表にすることはできません (SQLSTATE 42995)。

### FROM

特権を誰から取り消すかを指定します。

### USER

*authorization-name* がユーザーであることを指定します。

### GROUP

*authorization-name* がグループ名であることを指定します。

### ROLE

*authorization-name* がロール名であることを指定します。

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。

この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### PUBLIC

PUBLIC から特権を取り消します。

### BY ALL

指定された個々の特権を、その付与者にかかわらず、それらの特権を明示的に付与されたユーザーのうち指定された人から取り消します。これがデフォルトの動作です。

## REVOKE (表、ビュー、またはニックネーム特権)

### 規則

- 指定したそれぞれの *authorization-name* に関して、USER、GROUP、ROLE のいずれも指定されていない場合には、次のようになります。
  - 被認可者が *authorization-name* である SYSCAT.TABAUTH および SYSCAT.COLAUTH カタログ・ビュー内の指定されたオブジェクトのすべての行について、以下が該当します。
    - すべての行の GRANTEETYPE が「U」の場合、USER であると見なされず。
    - すべての行の GRANTEETYPE が「G」の場合、GROUP であると見なされます。
    - すべての行の GRANTEETYPE が「R」の場合、ROLE であると見なされず。
    - すべての行の GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

### 注

- ビューの所有者である *authorization-name* (SYSCAT.VIEWS の OWNER 列に記録されている) から特権が取り消されると、従属するビューの特権も取り消されます。
- ビューの所有者が、そのビュー定義が従属しているオブジェクトに対する SELECT 特権を失った場合、またはそのビュー定義が従属するオブジェクトがドロップされるか、または別のビューのために作動不能になった場合、そのビューは作動不能になります。

ただし、DBADM または SYSADM が明示的に所有者からビューの特権すべてを取り消した場合、SYSCAT.TABAUTH にはその所有者についてのレコードが表示されませんが、ビューには何も影響がなく作動可能のままになります。

- 作動不能なビューに対する特権は取り消すことはできません。
- パッケージのバインド時の許可 ID が、パッケージの依存するオブジェクトに対する特権を失った場合、パッケージは無効になる可能性があります。特権は、次のいずれかの場合に失われる可能性があります。
  - 許可 ID から特権が取り消された場合
  - 許可 ID がメンバーとなっているロールから特権が取り消された場合
  - PUBLIC から特権が取り消された場合

そのようなパッケージは、そのアプリケーションでバインド操作または再バインド操作が正常に実行されるか、またはそのアプリケーションが実行され、そのアプリケーションを (カタログに保管されている情報を使用して) データベース・マネージャーが正常に再バインドするまで、無効のままです。取り消しによって無効としてマークされたパッケージは、追加の付与操作なしで正常に再バインドできます。

例えば、USER1 が所有するパッケージに表 T1 からの SELECT が含まれ、その表 T1 に対する SELECT 特権が USER1 から取り消された場合、パッケージは無効としてマークされます。SELECT 権限が再び付与された場合、またはそのユーザーに DBADM 権限が与えられている場合には、パッケージは実行時に正常に再バインドされます。

## REVOKE (表、ビュー、またはニックネーム特権)

別の例として、あるパッケージが、ロール R1 のメンバーである USER1 によって所有されているとします。パッケージには表 T1 からの SELECT が含まれ、その表 T1 に対する SELECT 特権がロール R1 から取り消されたとします。USER1 が表 T1 に対する SELECT 特権を他の方法で保持していない限り、パッケージは無効としてマークされます。

- FROM 節で OUTER(Z) を使用するパッケージ、トリガー、またはビューは、Z のすべての副表またはサブビューに対する SELECT 特権に依存しています。同じように、パッケージ、トリガー、またはビューで Deref(Y) が使用されていて、Y が表またはビュー Z をターゲットとする参照タイプである場合には、Z のすべての副表またはサブビューに対する SELECT 特権があるかどうかによって依存しています。パッケージのバインド時の許可 ID、またはトリガーやビューの所有者が SELECT 特権を失った場合には、そのようなパッケージが無効になり、そのようなトリガーやビューが作動不能になる可能性があります。SELECT 特権は、次のいずれかの場合に失われる可能性があります。
  - 許可 ID から SELECT 特権が取り消された場合
  - 許可 ID がメンバーとなっているロールから SELECT 特権が取り消された場合
  - PUBLIC から SELECT 特権が取り消された場合
- CONTROL 特権も取り消さない限り、そのオブジェクトに対する CONTROL が与えられている *authorization-name* から表、ビューまたはニックネームの特権を取り消すことはできません (SQLSTATE 42504)。
- 特定の特権の取り消しにより、アクションを実行する権限が取り消されるとは限りません。PUBLIC、グループ、またはロールが他の特権を持っている場合、あるいはユーザーが表またはビューのスキーマに対する ALTERIN などの特権を持っている場合には、ユーザーは作業を続行できます。
- マテリアライズ照会表の所有者が、マテリアライズ照会表定義が従属している表に対する SELECT 特権を失った場合 (またはマテリアライズ照会表定義が従属する表がドロップされる場合)、マテリアライズ照会表はドロップされます。

ただし、DBADM または SYSADM が明示的に所有者から表の特権すべてを取り消した場合には、SYSTABAUTH のその所有者についてのレコードは削除されませんが、マテリアライズ照会表には何も影響がなく作動可能のままになります。
- ニックネーム特権を取り消しても、データ・ソース・オブジェクト (表またはビュー) の特権に影響を与えることはありません。
- オブジェクトが従属しているためにドロップできない SQL 関数またはメソッド本体がある場合は、その SQL 関数またはメソッド本体で直接または間接的に参照される表やビューに対する SELECT 特権も取り消せない場合があります (SQLSTATE 42893)。
- SELECT 特権を取り消すと、以下のような場合に、SQL 関数またはメソッド本体がドロップされます。
  - SQL 関数またはメソッド本体の所有者が、SQL 関数またはメソッド本体の定義が依存しているオブジェクトに対する SELECT 特権を失った場合。PUBLIC からの取り消しまたは所有者がメンバーとなっているロールからの取り消しによって特権が失われる場合がある点に注意してください。
  - SQL 関数またはメソッド本体の定義が依存しているオブジェクトがドロップされた場合。

## REVOKE (表、ビュー、またはニックネーム特権)

ただし、別のオブジェクトが関数またはメソッドに依存している場合には、取り消しが失敗します (SQLSTATE 42893)。

### 例

例 1: ユーザー ENGLES から、表 EMPLOYEE に対する SELECT 特権を取り消します。SYSCAT.TABAUTH カタログ・ビューにはこの表とユーザーについての行が 1 行あり、その GRANTEETYPE の値は U です。

```
REVOKE SELECT
ON TABLE EMPLOYEE
FROM ENGLES
```

例 2: 以前にすべてのローカル・ユーザーに与えられた表 EMPLOYEE に対する更新特権を取り消します。特定のユーザーに対する特権付与には影響を与えない点に注意してください。

```
REVOKE UPDATE
ON EMPLOYEE
FROM PUBLIC
```

例 3: ユーザー PELLOW と MLI、およびグループ PLANNERS から、表 EMPLOYEE に対する特権をすべて取り消します。

```
REVOKE ALL
ON EMPLOYEE
FROM USER PELLOW, USER MLI, GROUP PLANNERS
```

例 4: JOHN という名前のユーザーから、表 CORPDATA.EMPLOYEE に対する SELECT 特権を取り消します。SYSCAT.TABAUTH カタログ・ビューにはこの表とユーザーについての行が 1 行あり、その GRANTEETYPE の値は U です。

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM JOHN
```

または

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM USER JOHN
```

GROUP JOHN には特権が与えられていないので、GROUP JOHN から特権を取り消そうとしてもエラーになります。

例 5: JOHN という名前のグループから、表 CORPDATA.EMPLOYEE に対する SELECT 特権を取り消します。SYSCAT.TABAUTH カタログ・ビューにはこの表とユーザーについての行が 1 行あり、その GRANTEETYPE の値は G です。

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM JOHN
```

または

```
REVOKE SELECT
ON CORPDATA.EMPLOYEE FROM GROUP JOHN
```

例 6: ユーザー SHAWN から、ニックネーム ORAREM1 の SPECIFICATION ONLY 指定の索引を作成する特権を取り消します。

```
REVOKE INDEX
ON ORAREM1 FROM USER SHAWN
```

## REVOKE (ワークロード特権)

この形式の REVOKE ステートメントは、ワークロードでの USAGE 特権を取り消します。

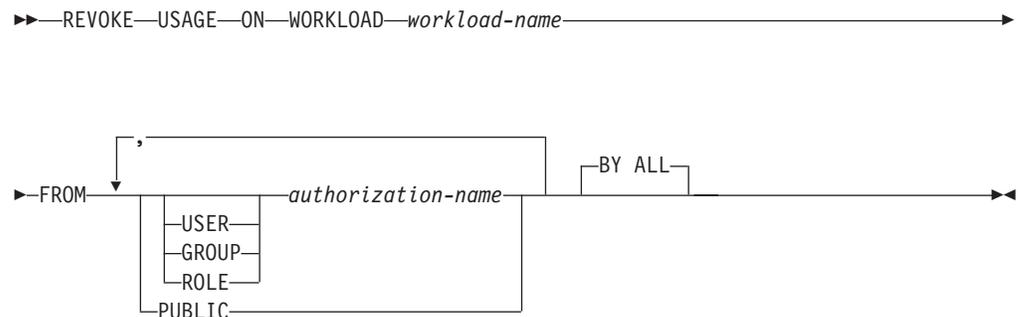
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

このステートメントの許可 ID が持つ特権には、SYSADM または DBADM 権限が含まれている必要があります。

### 構文



### 説明

#### USAGE

ワークロードを使用する特権を取り消します。

#### ON WORKLOAD workload-name

どのワークロードでの USAGE 特権を取り消すかを指定します。これは、1 部構成の名前です。workload-name には、現行のサーバー上の既存のワークロードを指定する必要があります (SQLSTATE 42704)。名前を 'SYSDEFAULTADMWORKLOAD' にすることはできません (SQLSTATE 42832)。

#### FROM

USAGE 特権を誰から取り消すかを指定します。

#### USER

authorization-name がユーザーであることを指定します。

#### GROUP

authorization-name がグループであることを指定します。

#### ROLE

authorization-name が現行サーバーにおける既存のロールを識別することを指定します (SQLSTATE 42704)。

## REVOKE (ワークロード特権)

*authorization-name*,...

1 つ以上のユーザー、グループ、またはロールの許可 ID のリストを指定します。この許可 ID のリストに、このステートメントを発行するユーザーの許可 ID を含めることはできません (SQLSTATE 42502)。

### **PUBLIC**

PUBLIC から USAGE 特権を取り消します。

### **BY ALL**

特権の付与者にかかわらず、特権を明示的に付与されたすべての名前付きユーザーから USAGE 特権を取り消します。これがデフォルトの動作です。

### **規則**

- 指定する各 *authorization-name* で、キーワード USER、GROUP、または ROLE がいずれも指定されなかった場合、被認可者が *authorization-name* の SYSCAT.WORKLOADAUTH カタログ・ビューで指定されたオブジェクトのすべての行で、次のようになります。
  - GRANTEETYPE が U の場合、USER であると見なされます。
  - GRANTEETYPE が G の場合、GROUP であると見なされます。
  - GRANTEETYPE が R の場合、ROLE であると見なされます。
  - GRANTEETYPE の値が同じでない場合、エラーが戻されます (SQLSTATE 56092)。

### **注**

- REVOKE ステートメントは、コミットされるまでは有効になりません。これは、ステートメントを発行する接続でも同じです。

### **例**

ユーザー LISA から、ワークロード CAMPAIGN を使用する特権を取り消します。

```
REVOKE USAGE ON WORKLOAD CAMPAIGN FROM USER LISA
```

---

## REVOKE (XSR オブジェクト特権)

この形式の REVOKE ステートメントは、XSR オブジェクトに対する USAGE 特権を取り消します。

### **呼び出し**

REVOKE ステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### **許可**

以下のいずれかの権限が必要です。

- SYSADM または DBADM

## 構文

```

▶▶ REVOKE USAGE ON XSROBJECT xsobject-name FROM PUBLIC BY ALL

```

## 説明

### ON XSROBJECT *xsobject-name*

この名前で、USAGE 特権を取り消される XSR オブジェクトを示します。

*xsobject-name* (暗黙的または明示的スキーマ修飾子を含む) は、現行のサーバーに存在する XSR オブジェクトを固有に識別しなければなりません。指定したスキーマにこの名前の XSR オブジェクトが存在しない場合は、エラーになります (SQLSTATE 42704)。

### FROM PUBLIC

PUBLIC から USAGE 特権を取り消します。

### BY ALL

指定された個々の特権を、その付与者にかかわらず、それらの特権を明示的に付与されたすべてのユーザーから取り消します。これがデフォルトの動作です。

## 例

XML スキーマ MYSCHEMA に対する PUBLIC の USAGE 特権を取り消します。

```
REVOKE USAGE ON XSROBJECT MYSCHEMA FROM PUBLIC
```

## ROLLBACK

ROLLBACK ステートメントは、作業単位またはセーブポイントにおいてデータベースに加えられた変更を撤回するために使用します。

## 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

必要ありません。

## 構文

```

▶▶ ROLLBACK WORK TO SAVEPOINT savepoint-name

```

### 説明

ROLLBACK ステートメントが実行される作業単位は終了し、新しい作業単位が開始されます。その作業単位の中でデータベースに対して行われた変更はすべて取り消されます。

ただし、以下のステートメントはトランザクションによって制御されず、これらのステートメントによって行われた変更は ROLLBACK ステートメントとは無関係です。

- SET CONNECTION
- SET CURRENT DEFAULT TRANSFORM GROUP
- SET CURRENT DEGREE
- SET CURRENT EXPLAIN MODE
- SET CURRENT EXPLAIN SNAPSHOT
- SET CURRENT LOCK TIMEOUT
- SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
- SET CURRENT PACKAGESET
- SET CURRENT QUERY OPTIMIZATION
- SET CURRENT REFRESH AGE
- SET ENCRYPTION PASSWORD
- SET EVENT MONITOR STATE
- SET PASSTHRU

注: SET PASSTHRU ステートメントはトランザクションによって制御されませんが、ステートメントによって開始されたパススルー・セッションはトランザクションにより制御されます。

- SET PATH
- SET SCHEMA
- SET SERVER OPTION

シーケンスと ID 値の生成は、トランザクションの制御下にはありません。

ROLLBACK ステートメントを発行しても、*nextval-expression* によって生成されて使用される値や、ID 列のある表に行を挿入することによって生成されて使用される値には影響を与えません。また、ROLLBACK ステートメントを発行しても、*prevval-expression* によって戻される値と IDENTITY\_VAL\_LOCAL 関数のどちらにも影響を与えません。

グローバル変数の値の変更は、トランザクションの制御下にはありません。

ROLLBACK ステートメントは、グローバル変数に割り当てられた値に影響を与えません。

### TO SAVEPOINT

部分的なロールバック (ROLLBACK TO SAVEPOINT) を実行することを指定します。現行のセーブポイント・レベルのアクティブなセーブポイントがない場合 (SAVEPOINT ステートメントの『規則』の節を参照) は、エラーが戻されます (SQLSTATE 3B502)。セーブポイントは、ロールバックが正常に完了した後もそのまま存続しますが、ネストされたセーブポイントはすべて解放され、存在しな

くなります。ネストされたセーブポイントがある場合、それらはロールバックされたものとみなされ、その後、現行セーブポイントへのロールバックの一部として解放されます。 *savepoint-name* が指定されない場合は、現行セーブポイント・レベルでの最新セットのセーブポイントへのロールバックが行われます。

この節を省略して ROLLBACK ステートメントを実行すると、トランザクション全体がロールバックされます。また、トランザクション内のセーブポイントは解放されます。

#### *savepoint-name*

ロールバック操作に使用されるセーブポイントを指定します。 *savepoint-name* を指定する際に、'SYS' で始めることはできません (SQLSTATE 42939)。ロールバックが正常に完了した後も、その名前のセーブポイントはそのまま継続します。指定された名前のセーブポイントが存在しない場合は、エラーが戻されます (SQLSTATE 3B001)。セーブポイントが設定された後に加えられたデータおよびスキーマの変更が取り消されます。

### 注

- ROLLBACK が実行された作業単位では、保持されていたロックがすべて解放されます。オープン・カーソルはすべてクローズされます。LOB ロケータはすべて解放されます。
- ROLLBACK ステートメントの実行により、特殊レジスタの値を変更する SET ステートメントまたは RELEASE ステートメントは影響を受けません。
- プログラムが異常終了した場合は、暗黙的にその作業単位がロールバックされます。
- ステートメントのキャッシュは、ロールバック操作の影響を受けます。
- ROLLBACK TO SAVEPOINT がカーソルに与える影響は、セーブポイントに含まれているステートメントによって異なります。
  - セーブポイントに DDL が含まれており、この DDL にカーソルが従属している場合、カーソルは無効としてマークされます。これらのカーソルを使おうとすると、エラーが戻されます (SQLSTATE 57007)。
  - それ以外の場合は、次のとおりです。
    - セーブポイントで参照されているカーソルは、オープンされたままになり、結果表の次の論理行の前に置かれます。(位置指定の UPDATE ステートメントまたは DELETE ステートメントが出される前に、FETCH を実行する必要があります。)
    - セーブポイントで参照されていないカーソルは、ROLLBACK TO SAVEPOINT の影響を受けません (元の位置でオープンされたままになります)。
- 動的に準備されたステートメントの名前は依然として有効ですが、セーブポイント内でロールバックされた DDL 操作の結果として、ステートメントが暗黙的に再び準備されることがあります。
- ROLLBACK TO SAVEPOINT 操作が行われると、セーブポイントの中で指定されていた宣言済み一時表はすべてドロップされます。宣言済み一時表をセーブポイントの中で変更していた場合は、すべての行が表から削除されます。
- すべてのロックは、ROLLBACK TO SAVEPOINT ステートメントの後にも保持されます。

## ROLLBACK

- すべての LOB ロケータは、ROLLBACK TO SAVEPOINT 操作の後にも保持されます。

### 例

最後のコミット・ポイントまたはロールバック以後に行われた変更を削除します。

**ROLLBACK WORK**

---

## SAVEPOINT

SAVEPOINT ステートメントを使用して、トランザクション内にセーブポイントを設定します。

### 呼び出し

このステートメントは、アプリケーション・プログラム (ストアド・プロシージャを含む) に組み込むこともでき、対話式に発行することもできます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文

```
▶▶ SAVEPOINT savepoint-name [UNIQUE] ON ROLLBACK RETAIN CURSORS
▶▶ [ON ROLLBACK RETAIN LOCKS]
```

### 説明

#### *savepoint-name*

セーブポイントの名前を指定します。 *savepoint-name* を指定する際に、'SYS' で始めることはできません (SQLSTATE 42939)。同じ名前のセーブポイントがこのセーブポイント・レベル内で UNIQUE としてすでに定義されている場合、エラーが戻されます (SQLSTATE 3B501)。

#### UNIQUE

セーブポイントが現行セーブポイント・レベル内でアクティブな間、このセーブポイントの名前がアプリケーションによって再使用されないことを指定します。 *savepoint-name* がこのセーブポイント・レベル内にすでに存在していると、エラーが戻されます (SQLSTATE 3B501)。

#### ON ROLLBACK RETAIN CURSORS

SAVEPOINT ステートメントの後に処理されるオープン・カーソルのステートメントに関して、このセーブポイントへのロールバックでのシステムの動作を指定します。この節は可能な限り、セーブポイントへのロールバックによる影響を

受けないことを示します。どのような場合にカーソルがセーブポイントへのロールバックから影響を受けるかについては、「ROLLBACK」を参照してください。

### ON ROLLBACK RETAIN LOCKS

セーブポイントの設定後にはかけられるロックに関して、このセーブポイントへのロールバックでのシステムの動作を指定します。このセーブポイント以降に獲得したロックは追跡されず、このセーブポイントへのロールバック時にはロールバック (解放) されません。

### 規則

- セーブポイント関連のステートメントをトリガー定義内で使用することはできません (SQLSTATE 42987)。
- 以下のいずれかの状態になると、新規のセーブポイント・レベルが開始します。
  - 新規の作業単位 (UOW) が開始する。
  - NEW SAVEPOINT LEVEL 節で定義されたプロシージャが呼び出される。
  - アトミック・コンパウンド SQL ステートメントが開始する。
- セーブポイント・レベルの作成の原因となったイベントが終了されるか削除されると、セーブポイント・レベルは終了します。セーブポイント・レベルが終了すると、その中に含まれるすべてのセーブポイントは解放されます。オープン・カーソル、DDL アクション、またはデータ変更すべてはその親セーブポイント・レベル (すなわち、今終了したセーブポイント・レベルがその内部で作成されたセーブポイント・レベル) によって継承され、親セーブポイント・レベルに対して出されたセーブポイント関連のステートメントが適用されます。
- セーブポイント・レベル内のアクションには、以下の規則が適用されます。
  - セーブポイントは、それが設定されているセーブポイント・レベル内でのみ参照可能です。現行のセーブポイント・レベルの外で設定されたセーブポイントを解放、破棄、またはロールバックすることはできません。
  - 現行のセーブポイント・レベル内で設定されているすべてのアクティブなセーブポイントは、セーブポイント・レベルが終了すると自動的に解放されます。
  - 現行のセーブポイント・レベル内でのみ、セーブポイント名の固有性が強制されます。他のセーブポイント・レベルでアクティブであるセーブポイントの名前に影響がなければ、そのセーブポイントの名前を現行のセーブポイント・レベルで再利用できます。

### 注

- SAVEPOINT ステートメントを発行し終わると、ニックネームに対する挿入、更新、または削除操作は行えなくなります。
- UNIQUE 節を省略した場合、別のセーブポイントが *savepoint-name* を同じ保管レベル内で再使用してもよいと指定したことになります。同じ名前のセーブポイントが保管レベル内にすでに存在しているときには、既存のセーブポイントが破棄され、新規のセーブポイントがその名前前で現在処理中のポイントに作成されます。この新規のセーブポイントが、アプリケーションによって最後に設定されたセーブポイントであるとみなされます。同じ名前の別のセーブポイントを再利用したことによって既存のセーブポイントが破棄されたとしても、それはそのセーブポイントだけが破棄されたのであり、破棄されたセーブポイント以降に設定されたセーブポイントが解放されることはないので注意してください。これら後で

設定されたセーブポイントは `RELEASE SAVEPOINT` ステートメントでのみ解放できます。このステートメントは、指名されたセーブポイントと、そのセーブポイント以降に設定されたすべてのセーブポイントを解放します。

- `UNIQUE` 節を指定した場合、`savepoint-name` は同じ名前の既存のセーブポイントを解放した後でのみ再利用できます。
- あるセーブポイントにおいて、処理の途中でユーティリティ、SQL ステートメント、または `DB2` コマンドが断続的にコミットを実行した場合、そのセーブポイントは暗黙的に解放されます。
- あるセーブポイントで `SET INTEGRITY` ステートメントがロールバックされた場合、動的に準備されたステートメントの名前は依然として有効ですが、そのステートメントが暗黙的に再び準備されることがあります。
- 挿入がバッファに入れられることになっている場合 (すなわち、アプリケーションが `INSERT BUF` オプションを指定してプリコンパイルされた場合)、バッファは、`SAVEPOINT`、`ROLLBACK`、または `RELEASE TO SAVEPOINT` ステートメントが出されるとフラッシュされます。

## 例

例 1: ネストされたセーブポイントに対してロールバック操作を実行します。まず、`DEPARTMENT` という名前の表を作成します。 `SAVEPOINT1` を開始する前に 1 行挿入し、 `SAVEPOINT2` を開始する前にもう 1 行挿入し、 `SAVEPOINT3` を開始する前にさらにもう 1 行挿入します。

```
CREATE TABLE DEPARTMENT (
 DEPTNO CHAR(6),
 DEPTNAME VARCHAR(20),
 MGRNO INTEGER)

INSERT INTO DEPARTMENT VALUES ('A20', 'MARKETING', 301)

SAVEPOINT SAVEPOINT1 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('B30', 'FINANCE', 520)

SAVEPOINT SAVEPOINT2 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('C40', 'IT SUPPORT', 430)

SAVEPOINT SAVEPOINT3 ON ROLLBACK RETAIN CURSORS

INSERT INTO DEPARTMENT VALUES ('R50', 'RESEARCH', 150)
```

この時点で、`DEPARTMENT` 表には `A20`、`B30`、`C40`、および `R50` という行があります。そして、次のステートメントを出すと、

```
ROLLBACK TO SAVEPOINT SAVEPOINT3
```

行 `R50` が `DEPARTMENT` 表からなくなります。そして、次のステートメントを出すと、

```
ROLLBACK TO SAVEPOINT SAVEPOINT1
```

`DEPARTMENT` 表は残っていますが、`SAVEPOINT1` を設定した後に挿入した行 (`B30` と `C40`) が表からなくなります。

## SELECT INTO

SELECT INTO ステートメントは、最大 1 行から成る結果表を作成し、その行の値をホスト変数に割り当てます。その表が空の場合、ステートメントは、SQLCODE に +100、SQLSTATE に '02000' を割り当て、ホスト変数には値を割り当てません。複数の行が検索条件を満たしている場合、ステートメントの処理は終了し、エラーが発生します (SQLSTATE 21000)。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

### 許可

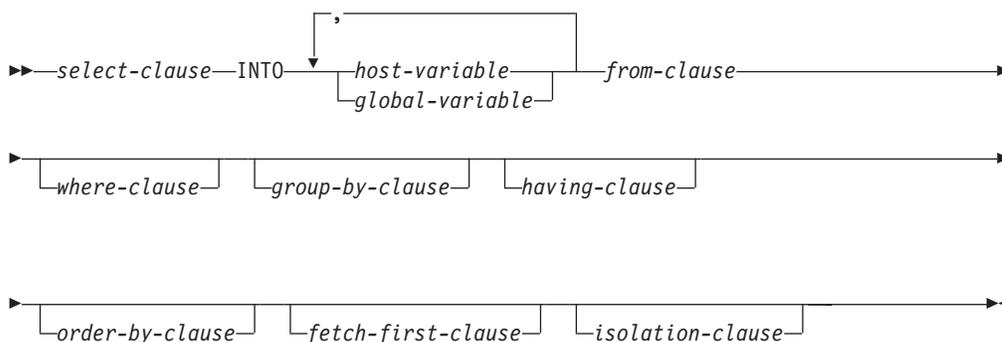
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 表、ビュー、またはニックネームに対する SELECT 特権
- 表、ビュー、またはニックネームに対する CONTROL 特権
- SYSADM または DBADM 権限

静的 SELECT INTO ステートメントの場合、GROUP 特権は検査されません。

SELECT INTO ステートメントの対象がニックネームの場合は、データ・ソースでステートメントが実行されないうちは、そのデータ・ソース上のオブジェクトに対する特権は考慮されません。この時点で、データ・ソースに接続するために使用される許可 ID は、データ・ソースのオブジェクトに対して操作を行うのに必要な特権を持っている必要があります。ステートメントの許可 ID は、データ・ソースの別の許可 ID へマップできます。

### 構文



### 説明

`select-clause`、`from-clause`、`where-clause`、`group-by-clause`、`having-clause`、`order-by-clause`、`fetch-first-clause`、および `isolation-clause` についての説明は、「SQL リファレンス 第 1 巻」の『照会』を参照してください。

**INTO**

この後にホスト変数またはグローバル変数のリストを指定します。

*host-variable*

ホスト変数の宣言規則に従ってプログラムに記述されている変数を指定します。

*global-variable*

現在のサーバーに存在するグローバル変数を指定します。

結果行の最初の値はリストの最初の変数、2番目の値は2番目の変数に割り当てられます。以下同様です。変数の数が列の値の数より少ない場合は、SQLCAのSQLWARN3フィールドに値'W'が割り当てられます。

変数への個々の割り当ては、リストに指定された順序で行われます。エラーが発生した場合、値は変数に割り当てられません。

**規則**

- トリガー、関数、メソッド、動的コンパウンド・ステートメントの内部で、あるいはこれらのいずれかのオブジェクトによって直接的または間接的に呼び出されるプロシージャの内部で、グローバル変数を割り当てることはできません (SQLSTATE 428GX)。

**例**

例 1: この C の例では、EMP 表における給与の最高額をホスト変数 MAXSALARY に割り当てています。

```
EXEC SQL SELECT MAX(SALARY)
 INTO :MAXSALARY
 FROM EMP;
```

例 2: この C の例では、EMP 表にある従業員 528671 の行をホスト変数に割り当てています。

```
EXEC SQL SELECT * INTO :h1, :h2, :h3, :h4
 FROM EMP
 WHERE EMPNO = '528671';
```

例 3: この SQLJ の例では、EMP 表にある従業員 528671 の行をホスト変数に割り当てています。その後、その行は検索更新を使用して更新されますが、照会の実行時にはロックされることになります。

```
#sql { SELECT * INTO :FIRSTNAME, :LASTNAME, :EMPNO, :SALARY
 FROM EMP
 WHERE EMPNO = '528671'
 WITH RS USE AND KEEP EXCLUSIVE LOCKS };
```

例 4: この C の例では、EMP 表における給与の最高額をグローバル変数 GV\_MAXSALARY に割り当てます。

```
EXEC SQL SELECT MAX(SALARY)
 INTO GV_MAXSALARY
 FROM EMP;
```

## SELECT

SELECT ステートメントは、照会の 1 つの形式です。これは、アプリケーション・プログラムに組み込むことも、または対話式に発行することも可能です。

## SET COMPILATION ENVIRONMENT

SET COMPILATION ENVIRONMENT ステートメントは、接続内の現行コンパイル環境を、イベント・モニターによって提供されるコンパイル環境に含まれている値と一致するように変更します。このステートメントは、1 つ以上の特殊レジスタの値を変更します。これらの変更は、後続の動的 SQL ステートメントのコンパイルに影響を与えます。

このステートメントは、トランザクションの制御下にありません。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文

```

▶▶ SET COMPILATION ENVIRONMENT [] host-variable ◀◀

```

### 説明

*host-variable*

タイプ BLOB の変数。イベント・モニターによって提供されるコンパイル環境です。NULL に設定することはできません。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。コンパイル環境のフォーマットが誤っている場合には、エラーが戻され、接続の設定は変更されません (SQLSTATE 51040)。

### 注

- コンパイル環境を元のデフォルト値にリセットするには、接続を終了してから再始動してください。このステートメントを SQL ルーチン内で発行することによっても同じ効果が得られるので、そのルーチンから戻るときに、特殊レジスタの変更が接続の中で反映されることはありません。
- コンパイル環境内に含まれる個々のエレメントを参照するには COMPILATION\_ENV 表関数を使用してください。

### 例

例 1: 現行セッションのコンパイル環境を、デッドロック・イベント・モニターによって以前にキャプチャーされたコンパイル環境内に含まれている値に設定しま

## SET COMPILATION ENVIRONMENT

す。WITH DETAILS HISTORY オプションを指定して作成されたデッドロック・イベント・モニターは、動的 SQL ステートメントのコンパイル環境をキャプチャーします。このキャプチャーされた環境が、ステートメントへの入力として受け入れられます。

```
SET COMPILATION ENVIRONMENT = :hv1
```

---

## SET CONNECTION

SET CONNECTION ステートメントは、接続の状態を休止状態から現行状態に変更して、指定された位置を現行サーバーにします。このステートメントは、トランザクションの制御下にはありません。

### 呼び出し

対話式 SQL 機能には外見上対話式的の実行に見えるインターフェースが用意されている場合がありますが、このステートメントはアプリケーション・プログラムに組み込むことだけが可能です。これは、動的に作成できない実行可能ステートメントです。

### 許可

必要ありません。

### 構文

```
▶▶ SET CONNECTION server-name | host-variable ◀◀
```

### 説明

*server-name* または *host-variable*

*server-name* (サーバー名) またはその *server-name* を含む *host-variable* (ホスト変数) によって、アプリケーション・サーバーを指定します。

*host-variable* (ホスト変数) を指定する場合、それは、長さ属性が 8 以下の文字ストリング変数でなければならない、標識変数を含めることはできません。その *host-variable* に入る *server-name* は、左寄せする必要がある、引用符で区切ることとはできません。

*server-name* は、アプリケーション・サーバーを指定するデータベース別名である点に注意してください。この名前は、アプリケーション・リクエストのローカル・ディレクトリーにリストされている必要があります。

*server-name* または *host-variable* は、アプリケーション・プロセスの既存の接続を指定していなければなりません。既存の接続を指定していない場合には、エラー (SQLSTATE 08003) になります。

現行接続に対する SET CONNECTION の場合、アプリケーション・プロセスのすべての接続の状態は変更されません。

#### 正常に接続された場合

SET CONNECTION ステートメントが正常に実行された場合、

- 作成される接続はありません。CURRENT SERVER 特殊レジスターは、指定した *server-name* で更新されます。
- それ以前の現行接続がある場合、それは休止状態になります (別の *server-name* を指定した場合)。
- CURRENT SERVER 特殊レジスターと SQLCA は、『CONNECT (タイプ 1)』で説明した方法と同じ方法で更新されます。

#### 正常に接続されなかった場合

SET CONNECTION ステートメントが失敗した場合、

- エラーの理由に関係なく、アプリケーション・プロセスの接続状態とその接続の状態は変更されません。
- エラーになったタイプ 1 の CONNECT の場合と同様に、SQLCA の SQLERRP フィールドは、エラーを検出したモジュール名に設定されます。

#### 注

- タイプ 1 CONNECT ステートメントの使用は、SET CONNECTION の使用を排除するわけではありませんが、休止状態の接続は存在し得ないので、SET CONNECTION ステートメントに現行接続を指定するのではない限り、このステートメントは常にエラーになります (SQLSTATE 08003)。
- SQLRULES(DB2) 接続オプション (『分散作業単位のセマンティクスを制御するオプション』を参照) を使用した場合、SET CONNECTION の使用を排除するわけではありませんが、タイプ 2 CONNECT ステートメントが使用できるので、このステートメントは不要です。
- 同じ作業単位で接続が使用され、休止状態になり、次に現行状態にリストアすると、ロック、カーソル、および準備済みステートメントの状況に関して、その接続はアプリケーション・プロセスでの最後の使用を反映したものになります。

#### 例

IBMSTHDB で SQL ステートメントを実行し、次に IBMTOKDB で SQL ステートメントを実行し、その後、IBMSTHDB で SQL ステートメントを実行します。

```
EXEC SQL CONNECT TO IBMSTHDB;
/* Execute statements referencing objects at IBMSTHDB */

EXEC SQL CONNECT TO IBMTOKDB;
/* Execute statements referencing objects at IBMTOKDB */

EXEC SQL SET CONNECTION IBMSTHDB;
/* Execute statements referencing objects at IBMSTHDB */
```

最初の CONNECT ステートメントでは IBMSTHDB の接続が作成され、2 番目の CONNECT ステートメントでその接続は休止状態になり、SET CONNECTION ステートメントによってその接続は現行状態に戻ります。

---

## SET CURRENT DECFLOAT ROUNDING MODE

SET CURRENT DECFLOAT ROUNDING MODE ステートメントは、指定の丸めモードが、現在 CURRENT DECFLOAT ROUNDING MODE 特殊レジスターに対して設定されている値であることを検査します。

## SET CURRENT DECFLOAT ROUNDING MODE

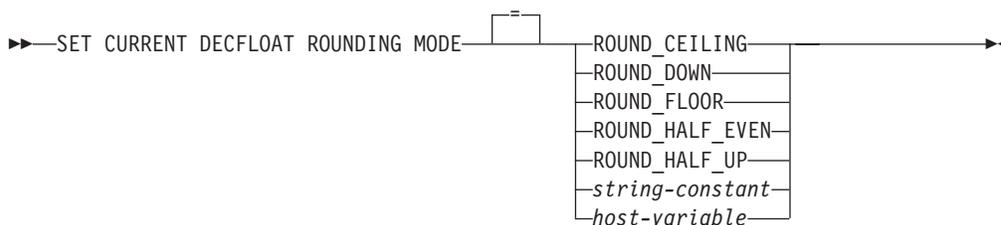
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

必要ありません。

### 構文



### 説明

#### ROUND\_CEILING

値を正の無限大の方向に丸めます。廃棄されたすべての桁がゼロであるか、符号が負の場合、(廃棄された桁の除去以外は) 結果は変わりません。そうでない場合、結果の係数は 1 だけ増分されます。

#### ROUND\_DOWN

値を 0 の方向に丸めます (切り捨て)。廃棄された桁は無視されます。

#### ROUND\_FLOOR

値を負の無限大の方向に丸めます。廃棄されたすべての桁がゼロであるか、符号が正の場合、(廃棄された桁の除去以外は) 結果は変わりません。そうでない場合、符号は負であり、結果の係数は 1 だけ増分されます。

#### ROUND\_HALF\_EVEN

値を最も近い値に丸めます。最も近い値がそれぞれ等距離の場合、最終の数字が偶数になるように丸めます。廃棄される数字が、左隣り桁の数の値の 2 分の 1 より大きい場合、結果の係数は 1 だけ増分されます。2 分の 1 より小さい場合、結果の係数は調整されません (つまり、廃棄される桁は無視されます)。そうでない場合、結果の係数は、その右端の数字が偶数の場合は変更されず、右端の数字が奇数の場合は 1 だけ増分されます (偶数にされる)。

#### ROUND\_HALF\_UP

値を最も近い値に丸めます。最も近い値がそれぞれ等距離の場合、値を切り上げます。廃棄される数字が、左隣り桁の数の値の 2 分の 1 より大きい場合、結果の係数は 1 だけ増分されます。そうでない場合、廃棄される桁は無視されます。

## SET CURRENT DECFLOAT ROUNDING MODE

### *string-constant*

末尾ブランクの除去後に 15 バイトを超えない文字ストリング定数。値は、5 つの丸めモード・キーワードの 1 つを指定する左揃えストリングでなければなりません (大/小文字を区別しない)。

### *host-variable*

タイプ CHAR または VARCHAR の変数です。ホスト変数の値は、5 つの丸めモード・キーワードの 1 つを指定する左揃えストリングでなければなりません (大/小文字を区別しない)。 *host-variable* の内容の実際の長さは、末尾ブランクの除去後に 15 バイトを超えてはなりません。固定長文字ホスト変数を使用する場合は、値の右側をブランクで埋め込まなければなりません。ホスト変数は NULL 値に設定することはできません。

## 規則

- 指定の丸めモード値は、CURRENT DECFLOAT ROUNDING MODE 特殊レジスタの値と同じでなければなりません (SQLSTATE 42815)。

## 注

- このステートメントは、DB2 for Linux, UNIX, and Windows サーバー上の CURRENT DECFLOAT ROUNDING MODE 特殊レジスタの値は変更しません。ただし、ステートメントが DB2 for z/OS サーバーまたは DB2 for i5/OS サーバーにより処理される場合には、そのサーバー上で CURRENT DECFLOAT ROUNDING MODE 特殊レジスタの値を変更するために使用できます。

## 例

例 1: 以下のステートメントは、クライアントの指定の丸めモード値が、現在サーバー上で設定されている丸めモード値と一致することを検査します。

```
SET CURRENT DECFLOAT ROUNDING MODE = ROUND_CEILING
```

---

## SET CURRENT DEFAULT TRANSFORM GROUP

SET CURRENT DEFAULT TRANSFORM GROUP ステートメントは、CURRENT DEFAULT TRANSFORM GROUP 特殊レジスタの値を変更します。このステートメントは、トランザクションの制御下にありません。

## 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

必要ありません。

## 構文

```
→ SET CURRENT DEFAULT TRANSFORM GROUP = group-name →
```

## SET CURRENT DEFAULT TRANSFORM GROUP

### 説明

*group-name*

トランスフォーム・グループを識別する名前を 1 部構成の名前で指定します。このグループ名はすべての構造化タイプに定義されます。ここで指定された名前は、このステートメントに続く他のステートメントでも (つまり、別の SET CURRENT DEFAULT TRANSFORM GROUP ステートメントによって特殊レジスタの値が再び変更されるまで) 参照することができます。

名前は、長さが 128 バイト以下の SQL ID でなければなりません (SQLSTATE 42815)。特殊レジスタが設定される際に、構造化タイプに定義されている *group-name* の妥当性が検査されることはありません。特定の構造化タイプを指定して参照するときのみ、指定されたトランスフォーム・グループの定義が妥当であるかどうか検査されます。

### 規則

- 指定された値が *group-name* の規則に準拠していない場合は、エラーが発生します (SQLSTATE 42815)。
- トランスフォーム・グループ *group-name* に定義されている TO SQL 関数と FROM SQL 関数は、ユーザー定義構造化タイプのデータをホスト・プログラムとの間で交換するために使用されます。

### 注

- CURRENT DEFAULT TRANSFORM GROUP 特殊レジスタの初期値は空ストリングです。

### 例

例 1: デフォルトのトランスフォーム・グループを MYSTRUCT1 に設定します。トランスフォーム・グループ MYSTRUCT1 に定義されている TO SQL 関数と FROM SQL 関数は、ユーザー定義構造化タイプの変数を現在のホスト・プログラムとの間で交換するために使用されます。

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

---

## SET CURRENT DEGREE

SET CURRENT DEGREE ステートメントは、CURRENT DEGREE 特殊レジスタに値を割り当てます。このステートメントは、トランザクションの制御下にありません。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

## 構文

```

SET CURRENT DEGREE
└── string-constant ──┘
└── host-variable ───┘

```

## 説明

CURRENT DEGREE の値は、ストリング定数またはホスト変数の値によって置き換えられます。値は 5 バイトを超えない文字ストリングでなければなりません。その値は、1 から 32 767 (両端を含む) の整数の文字ストリング表現、または 'ANY' でなければなりません。

SQL ステートメントが動的に準備される時点で、整数として表現される CURRENT DEGREE の値が 1 である場合には、そのステートメントの実行にパーティション内並列処理は使用されません。

SQL ステートメントが動的に準備される時点で、CURRENT DEGREE の値が 1 以外の数値である場合には、そのステートメントの実行には、指定した度合いのパーティション内並列処理を使用できます。

SQL ステートメントが動的に準備される時点で、CURRENT DEGREE の値が 'ANY' である場合、そのステートメントの実行には、データベース・マネージャーによって決定された度合いを用いたパーティション内並列処理を使用できます。

### *host-variable*

*host-variable* (ホスト変数) は、そのデータ・タイプが CHAR または VARCHAR で、5 文字を超えない長さでなければなりません。それより長いフィールドを指定すると、エラーになります (SQLSTATE 42815)。実際に指定する値が、指定した置換値より大きい場合は、入力の右側にブランクを入れる必要があります。先行ブランクは使用できません (SQLSTATE 42815)。すべての入力値は、大文字小文字を区別しないものとして処理されます。*host-variable* が標識変数を伴う場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

### *string-constant*

*string-constant* (ストリング定数) の長さは 5 を超えてはなりません。

## 注

静的 SQL ステートメントのパーティション内並列処理の度合いは、PREP または BIND コマンドの DEGREE オプションを使用して制御できます。

パーティション内並列処理の実際の実行時の度合いは、以下のものより小さい値になります。

- 最大照会度合 (**max\_querydegree**) 構成パラメーター
- アプリケーション実行時の多重度
- SQL ステートメントのコンパイルの度合い

パーティション内並列処理を使用するには、**intra\_parallel** データベース・マネージャー構成パラメーターをオンにする必要があります。オフに設定されている場

## SET CURRENT DEGREE

合、このレジスターの値は無視され、ステートメントは最適化にパーティション内並列処理を使用しません (SQLSTATE 01623)。

SQL ステートメントによっては、パーティション内並列処理を使用できません。

### 例

例 1: 以下のステートメントは、パーティション内並列処理を禁止する CURRENT DEGREE を設定します。

```
SET CURRENT DEGREE = '1'
```

例 2: 以下のステートメントは、パーティション内並列処理を許可する CURRENT DEGREE を設定します。

```
SET CURRENT DEGREE = 'ANY'
```

---

## SET CURRENT EXPLAIN MODE

SET CURRENT EXPLAIN MODE ステートメントは、CURRENT EXPLAIN MODE 特殊レジスターの値を変更します。このステートメントは、トランザクションの制御下にはありません。

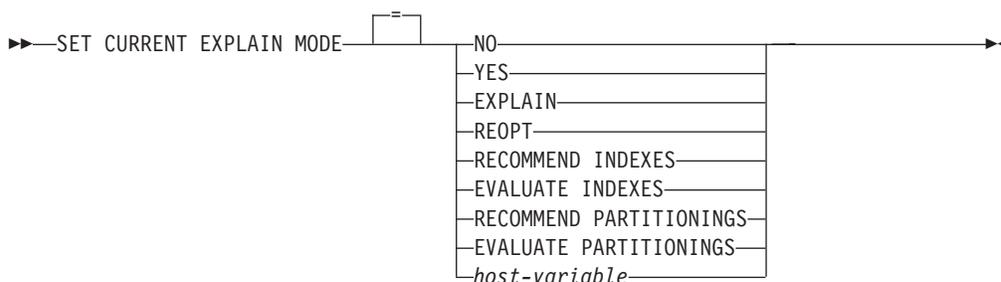
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文



### 説明

#### NO

Explain 機能を使用不可にします。Explain 情報はキャプチャーされません。NO は、特殊レジスターの初期値です。

#### YES

Explain 機能を使用可能にし、適格な動的 SQL ステートメントについての

Explain 情報を Explain 表に挿入します。すべての動的 SQL ステートメントが、通常どおりにコンパイルおよび実行されます。

#### EXPLAIN

Explain 機能を使用可能にし、準備される適切な動的 SQL ステートメントについての Explain 情報をキャプチャーします。ただし、動的ステートメントは実行されません。

#### REOPT

Explain 機能を使用可能にし、実行時のステートメント再最適化の際 (すなわち、ホスト変数、特殊レジスター、グローバル変数、またはパラメーター・マーカーの実際の値が使用可能になるとき) に、静的または動的 SQL ステートメントについての Explain 情報がキャプチャーされるようにします。

#### RECOMMEND INDEXES

SQL コンパイラーが索引を推奨できるようにします。この Explain モードで実行される照会はすべて、推奨された索引を ADVISE\_INDEX 表に埋め込みます。さらに、推奨された索引を使用する方法を示すため、Explain 表に Explain 情報がキャプチャーされますが、そのステートメントのコンパイルや実行は行われません。

#### EVALUATE INDEXES

SQL コンパイラーが索引を評価できるようにします。評価される索引は ADVISE\_INDEX 表から読み取られ、EVALUATE = Y というマークが付けられる必要があります。オプティマイザーは、カタログの値に基づく仮想索引を生成します。この Explain モードで実行される照会はすべて、仮想索引に基づいて見積もられた統計を使用してコンパイルされ、最適化されます。ステートメントは実行されません。

#### RECOMMEND PARTITIONINGS

特定の照会がアクセスするそれぞれの表ごとに、コンパイラーが最良のデータベース・パーティションを推奨するように指定します。それから、最良のデータベース・パーティションは ADVISE\_PARTITION 表に書き込まれます。照会は実行されません。

#### EVALUATE PARTITIONINGS

ADVISE\_PARTITION 表に指定された仮想データベース・パーティションを使って、コンパイラーが照会の推定パフォーマンスを取得するように指定します。

#### *host-variable*

*host-variable* (ホスト変数) のデータ・タイプは CHAR または VARCHAR でなければならず、その内容の長さは 254 を超えてはなりません。それより長いフィールドを指定すると、エラーになります (SQLSTATE 42815)。指定する値は、NO、YES、EXPLAIN、RECOMMEND INDEXES、または EVALUATE INDEXES でなければなりません。実際に指定する値が、指定した置換値より大きい場合は、入力の右側にブランクを入れる必要があります。先行ブランクは使用できません (SQLSTATE 42815)。すべての入力値は、大文字小文字を区別しないものとして処理されます。*host-variable* が標識変数を伴う場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

#### 注

- Explain 機能は、データの取り込み先である Explain 表を修飾するときに、スキーマとして以下の ID を使用します。

## SET CURRENT EXPLAIN MODE

- 動的 SQL のセッション許可 ID
- 静的 SQL のステートメント許可 ID

そのスキーマは、一連の Explain 表に関連付けられている場合もあれば、別のスキーマの下で一連の Explain 表を参照する別名に関連付けられている場合もあります。そのスキーマの下で Explain 表が検出されなかった場合、Explain 機能は、SYSTOOLS スキーマの下に Explain 表があるかどうかをチェックし、その表を使用しようとしています。

- 静的 SQL ステートメントの Explain 情報は、PREP または BIND コマンドの EXPLAIN オプションの使用によってキャプチャーすることができます。EXPLAIN オプションの ALL の値が指定され、CURRENT EXPLAIN MODE のレジスター値が NO の場合には、実行時に動的 SQL ステートメントの Explain 情報がキャプチャーされます。CURRENT EXPLAIN MODE レジスターの値が NO 以外の場合、EXPLAIN BIND オプションの値は無視されます。
- RECOMMEND INDEXES と EVALUATE INDEXES は特殊モードで、それらを設定するために使えるのは SET CURRENT EXPLAIN MODE ステートメントだけです。これらのモードは PREP または BIND オプションを使って設定することはできません。また、SET CURRENT EXPLAIN SNAPSHOT ステートメントを使用しても動作しません。
- Explain 機能が活動化される場合、現行の許可 ID に Explain 表に対する INSERT 特権が必要です。この特権がない場合には、エラー (SQLSTATE 42501) が発生します。
- ルーチンから SQL ステートメントの Explain 情報を取り出す場合は、MODIFIES SQL DATA の SQL データ・アクセス標識を指定して、ルーチンを定義しなければなりません (SQLSTATE 42985)。
- 特殊レジスターが REOPT に設定され、実行時の再最適化のために SQL ステートメントが修飾されない場合 (すなわち、ステートメントが入力変数を持っていないか、REOPT BIND オプションが NONE に設定されている場合) は、Explain 情報はキャプチャーされません。REOPT BIND オプションが ONCE に設定されている場合、Explain 情報は、ステートメントが最初に再最適化されるときの 1 回だけキャプチャーされます。ステートメントがキャッシュに入れられた後は、後続の実行では、それ以上の Explain 情報はこのステートメントに関して獲得されません。
- Explain 機能が使用可能で、REOPT BIND オプションが ONCE に設定されていて、すでにキャッシュに入れられている SQL ステートメントを実行しようとした場合は、入力変数の現行値を使ってステートメントがコンパイルおよび再最適化され、それにしたがって Explain 表にデータが取り込まれます。このステートメントのために新たに生成されるアクセス・プランは、キャッシュに入れられず、実行されません。このキャッシュ・ステートメントを並行して実行する他のアプリケーションは引き続き稼働し、このステートメントを実行するための新しい要求は、すでにキャッシュに入れられたアクセス・プランを採用します。
- 静的または動的 SQL ステートメントが入力変数を持っていて、REOPT BIND オプションが ONCE または ALWAYS に設定されている場合は、CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターの REOPT という値は、EXPLAIN および EXPLSNAP BIND オプションの値をバインド時にオーバーライドします。

## 例

次のステートメントでは、以降の適格な動的 SQL ステートメントの Explain 情報をキャプチャーし、そのステートメントが実行されないように、CURRENT EXPLAIN MODE 特殊レジスターを設定しています。

```
SET CURRENT EXPLAIN MODE = EXPLAIN
```

## SET CURRENT EXPLAIN SNAPSHOT

SET CURRENT EXPLAIN SNAPSHOT ステートメントは、CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値を変更します。このステートメントは、トランザクションの制御下にはありません。

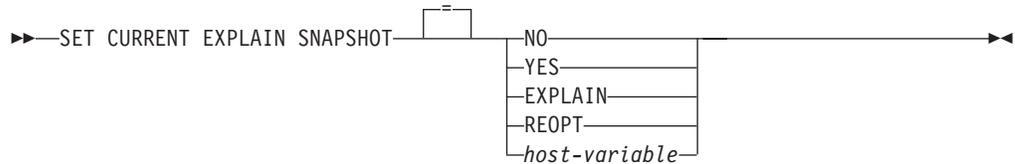
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文



### 説明

#### NO

Explain スナップショット機能を使用不可にします。スナップショットは取られません。NO は、特殊レジスターの初期値です。

#### YES

Explain スナップショット機能を使用可能にし、適格な動的 SQL ステートメントに対して内部表記のスナップショットを作成します。この情報は、EXPLAIN\_STATEMENT 表の SNAPSHOT 列に挿入されます。

EXPLAIN SNAPSHOT 機能は、Visual Explain での使用を意図しています。

#### EXPLAIN

Explain スナップショット機能を使用可能にし、準備済みの適格な動的 SQL ステートメントごとに内部表記のスナップショットを作成します。ただし、動的ステートメントは実行されません。

#### REOPT

Explain 機能を使用可能にし、実行時のステートメント再最適化の際 (すなわち、ホスト変数、特殊レジスター、グローバル変数、またはパラメーター・マー

## SET CURRENT EXPLAIN SNAPSHOT

カーの実際の値が使用可能になるときに、静的または動的 SQL ステートメントについての Explain 情報がキャプチャーされるようにします。

### *host-variable*

*host-variable* (ホスト変数) のデータ・タイプは CHAR または VARCHAR でなければならず、その内容の長さは 8 を超えてはなりません。それより長いフィールドを指定すると、エラーになります (SQLSTATE 42815)。このレジスターの値は、NO、YES、または EXPLAIN でなければなりません。実際に指定する値が、指定した置換値より大きい場合は、入力の右側にブランクを入れる必要があります。先行ブランクは使用できません (SQLSTATE 42815)。すべての入力値は、大文字小文字を区別しないものとして処理されます。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

## 注

- Explain 機能は、データの取り込み先である Explain 表を修飾するときに、スキーマとして以下の ID を使用します。
  - 動的 SQL のセッション許可 ID
  - 静的 SQL のステートメント許可 ID

そのスキーマは、一連の Explain 表に関連付けられている場合もあれば、別のスキーマの下で一連の Explain 表を参照する別名に関連付けられている場合もあります。そのスキーマの下で Explain 表が検出されなかった場合、Explain 機能は、SYSTOOLS スキーマの下に Explain 表があるかどうかをチェックし、その表を使用しようとしています。

- 静的 SQL ステートメントの Explain スナップショットは、PREP または BIND コマンドの EXPLSNAP オプションの使用によって取ることができます。EXPLSNAP オプションの ALL の値を指定し、CURRENT EXPLAIN SNAPSHOT のレジスター値が NO の場合には、実行時に動的 SQL ステートメントの Explain スナップショットが取られます。CURRENT EXPLAIN SNAPSHOT レジスターの値が NO 以外の場合、EXPLSNAP オプションは無視されます。
- Explain スナップショット機能が活動化される場合、現行の許可 ID には、Explain 表に対する INSERT 特権が必要です。この特権がないと、エラー (SQLSTATE 42501) になります。
- ルーチンから SQL ステートメントの Explain 情報を取り出す場合は、MODIFIES SQL DATA の SQL データ・アクセス標識を指定して、ルーチンを定義しなければなりません (SQLSTATE 42985)。
- 特殊レジスターが REOPT に設定され、実行時の再最適化のために SQL ステートメントが修飾されない場合 (すなわち、ステートメントが入力変数を持っていないか、REOPT BIND オプションが NONE に設定されている場合) は、Explain 情報はキャプチャーされません。REOPT BIND オプションが ONCE に設定されている場合、Explain スナップショット情報は、ステートメントが最初に再最適化される時の 1 回だけキャプチャーされます。ステートメントがキャッシュに入れられた後は、後続の実行では、それ以上の Explain 情報はこのステートメントに関して獲得されません。
- Explain 機能が使用可能で、REOPT BIND オプションが ONCE に設定されていて、すでにキャッシュに入れられている再最適化可能な SQL ステートメントを

実行しようとした場合は、入力変数の現行値を使ってステートメントがコンパイルおよび再最適化され、それにしたがって Explain スナップショットがキャプチャーされます。このステートメントのために新たに生成されるアクセス・プランは、キャッシュに入れられず、実行されません。このキャッシュ・ステートメントを並行して実行する他のアプリケーションは引き続き稼働し、このステートメントを実行するための新しい要求は、すでにキャッシュに入れられたアクセス・プランを採用します。

- 静的または動的 SQL ステートメントが入力変数を持っていて、REOPT BIND オプションが ONCE または ALWAYS に設定されている場合は、CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスタの REOPT という値は、EXPLAIN および EXPLSNAP BIND オプションの値をバインド時にオーバーライドします。

## 例

例 1: 以下のステートメントは、CURRENT EXPLAIN SNAPSHOT 特殊レジスタを設定して、以降の適格な動的 SQL ステートメントの Explain スナップショットを取り、そのステートメントを実行します。

```
SET CURRENT EXPLAIN SNAPSHOT = YES
```

例 2: 以下の例では、CURRENT EXPLAIN SNAPSHOT 特殊レジスタの現行値を検索して SNAP という名前のホスト変数に入れます。

```
EXEC SQL VALUES (CURRENT EXPLAIN SNAPSHOT) INTO :SNAP;
```

---

## SET CURRENT FEDERATED ASYNCHRONY

SET CURRENT FEDERATED ASYNCHRONY ステートメントは、CURRENT FEDERATED ASYNCHRONY 特殊レジスタに値を割り当てます。このステートメントは、トランザクションの制御下にはありません。

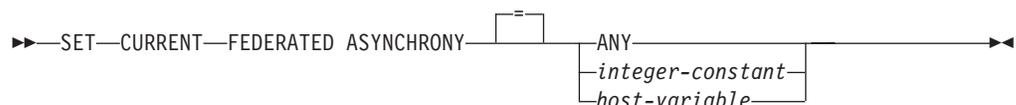
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文



## SET CURRENT FEDERATED ASYNCHRONY

### 説明

#### ANY

CURRENT FEDERATED ASYNCHRONY の値を -1 に指定します。この値は、ステートメントの実行に、データベース・マネージャーによって決定される度合いに基づいて非同期を使用できることを意味します。

#### *integer-constant*

0 から 32 767 まで (それぞれの値も含む) の範囲の整数値を指定します。ステートメントの実行に、指定した度合いに基づいて非同期を使用できます。SQL ステートメントが動的に準備される時点で、この値が 0 である場合は、そのステートメントの実行に非同期は使用されません。

#### *host-variable*

タイプが INTEGER の変数です。値は、0 から 32 767 まで (それぞれの値も含む) の範囲内か、または -1 (ANY を表す) でなければなりません。

*host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

### 注

- 静的 SQL ステートメントの非同期の度合いは、PREP または BIND コマンドの FEDERATED\_ASYNC オプションを使用して制御できます。
- 動的ステートメントがコマンド行プロセッサ (CLP) を介して発行される場合、CURRENT FEDERATED ASYNCHRONY 特殊レジスタの初期値は、**federated\_async** データベース・マネージャー構成パラメーターで決まります。動的ステートメントが、バインドされるアプリケーションの一部である場合、初期値は FEDERATED\_ASYNC バインド・オプションで決まります。

### 例

例 1: 次のステートメントは、CURRENT FEDERATED ASYNCHRONY 特殊レジスタの値を 0 に設定することで、非同期を使用不可にします。

```
SET CURRENT FEDERATED ASYNCHRONY = 0
```

例 2: 次のステートメントは、非同期の度合いを 5 に設定します。

```
SET CURRENT FEDERATED ASYNCHRONY 5
```

例 3: 次のステートメントは、CURRENT FEDERATED ASYNCHRONY 特殊レジスタの値を -1 に設定します。この値は、非同期の度合いはデータベース・マネージャーが決定することを指定します。

```
SET CURRENT FEDERATED ASYNCHRONY ANY
```

---

## SET CURRENT IMPLICIT XMLPARSE OPTION

SET CURRENT IMPLICIT XMLPARSE OPTION ステートメントは、CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスタの値を変更します。このステートメントは、トランザクションの制御下にありません。

## 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

必要ありません。

## 構文

```

▶▶ SET CURRENT IMPLICIT XMLPARSE OPTION [=]
 { string-constant | host-variable }

```

## 説明

### *string-constant*

文字ストリング定数です。値は、'PRESERVE WHITESPACE' または 'STRIP WHITESPACE' (大/小文字を区別しない) のいずれかの、キーワード間にこれ以上の追加のブランク文字を挿入しない、左揃えされたストリングでなければなりません。

### *host-variable*

タイプ CHAR または VARCHAR の変数です。ホスト変数の値は、'PRESERVE WHITESPACE' または 'STRIP WHITESPACE' (大/小文字を区別しない) のいずれかの、キーワード間にこれ以上の追加のブランク文字を挿入しない、左揃えされたストリングでなければなりません。固定長文字 *host-variable* を使用する場合は、値の右側をブランクで埋め込まなければなりません。ホスト変数は NULL に設定することはできません。

## 注

- CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスタの初期値は、'STRIP WHITESPACE' です。
- 動的または静的 SQL ステートメントのどちらも、この特殊レジスタにより影響を受けます。

## 例

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスタの値を 'PRESERVE WHITESPACE' に設定します。

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

---

## SET CURRENT ISOLATION

SET CURRENT ISOLATION ステートメントは、CURRENT ISOLATION 特殊レジスタに値を割り当てます。このステートメントは、トランザクションの制御下にありません。

## SET CURRENT ISOLATION

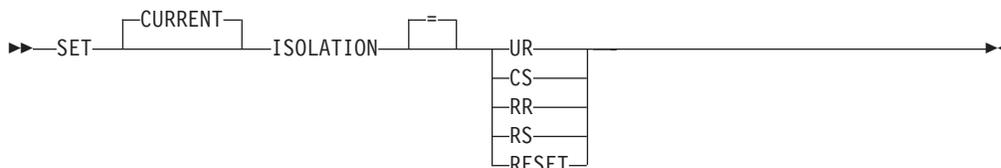
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文



### 説明

CURRENT ISOLATION 特殊レジスタの値は、RESET が指定されている場合、指定した値で置き換えられるかまたはブランクに設定されます。

### 注

#### • 互換性

- 以下の構文もサポートされています。
  - 等号 (=) の代わりに TO を指定できます。
  - UR の代わりに DIRTY READ を指定できます。
  - UR の代わりに READ UNCOMMITTED を指定できます。
  - READ COMMITTED が認識され、CS に更新されます。
  - CS の代わりに CURSOR STABILITY を指定できます。
  - RR の代わりに REPEATABLE READ を指定できます。
  - RR の代わりに SERIALIZABLE を指定できます。

---

## SET CURRENT LOCK TIMEOUT

SET CURRENT LOCK TIMEOUT ステートメントは、CURRENT LOCK TIMEOUT 特殊レジスタの値を変更します。このステートメントは、トランザクションの制御下にはありません。

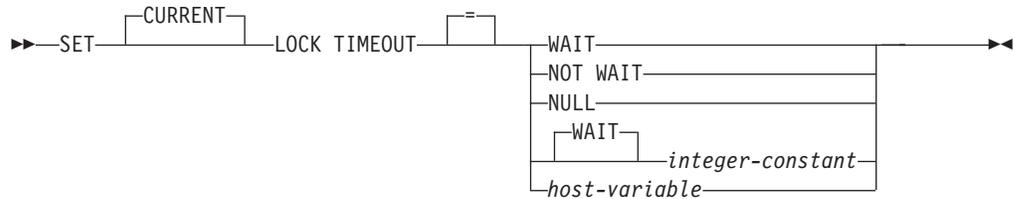
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また対話式に出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

必要ありません。

## 構文



## 説明

指定値は -1 から 32767 までの整数 (両端を含む) (SQLSTATE 428B7)、または NULL 値でなければなりません。

### WAIT

CURRENT LOCK TIMEOUT の値を -1 に設定します。この値は、ロックが解除されるか、デッドロックが検出されるまで、データベース・マネージャーが待機することを意味します (SQLSTATE 40001 または SQLSTATE 57033)。

### NOT WAIT

CURRENT LOCK TIMEOUT の値を 0 に設定します。この値は、獲得できないロックをデータベース・マネージャーが待機せず、エラーが戻されることを意味します (SQLSTATE 40001 または SQLSTATE 57033)。

### NULL

CURRENT LOCK TIMEOUT の値を設定解除するように指定します。ロックの待機の際には、**locktimeout** データベース構成パラメーターの値が使用されます。特殊レジスターに戻される値は、**locktimeout** の値が変更されると変化します。

### WAIT *integer-constant*

-1 から 32767 までの整数を指定します。-1 の値は、整数値なしで WAIT キーワードを指定することと等価です。0 の値は、NOT WAIT 節を指定するのと等価です。値が 1 から 32767 までの場合は、(ロックを獲得できない場合に) エラーが戻される前に、データベース・マネージャーはその秒数だけ待機します (SQLSTATE 40001 または SQLSTATE 57033)。

### *host-variable*

タイプが INTEGER の変数です。値は -1 から 32767 までの範囲内である必要があります。 *host-variable* が関連した標識変数を伴っていて、その標識変数の値が NULL 値を指定している場合、CURRENT LOCK TIMEOUT の値は設定解除されます。これは NULL キーワードを指定するのと等価です。

## 注

- 特殊レジスターの更新された値は、このステートメントが正常実行されると即時に有効になります。ステートメントの実行中に使用される特殊レジスター値はステートメント実行の初めに固定されるため、実行を開始したステートメントによ

## SET CURRENT LOCK TIMEOUT

って CURRENT LOCK TIMEOUT 特殊レジスターの更新された値が戻されるのは、SET LOCK TIMEOUT ステートメントが正常に完了した後になります。

### • 互換性

- Informix との互換性:

- TIMEOUT の代わりに MODE を指定できます。
- 等号 (=) 演算子の代わりに TO を指定できます。
- SET CURRENT LOCK TIMEOUT WAIT の代わりに SET LOCK WAIT を指定できます。
- SET CURRENT LOCK TIMEOUT NOT WAIT の代わりに SET LOCK NO WAIT を指定できます。

### 例

例 1: エラーを戻す前に 30 秒間待つよう、ロック・タイムアウト値を設定します。

```
SET CURRENT LOCK TIMEOUT 30
```

例 2: **locktimeout** データベース構成パラメーター値が代わりに使用されるように、ロック・タイムアウト値を設定解除します。

```
SET CURRENT LOCK TIMEOUT NULL
```

---

## SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ステートメントは、CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターの値を変更します。このステートメントは、トランザクションの制御下にはありません。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

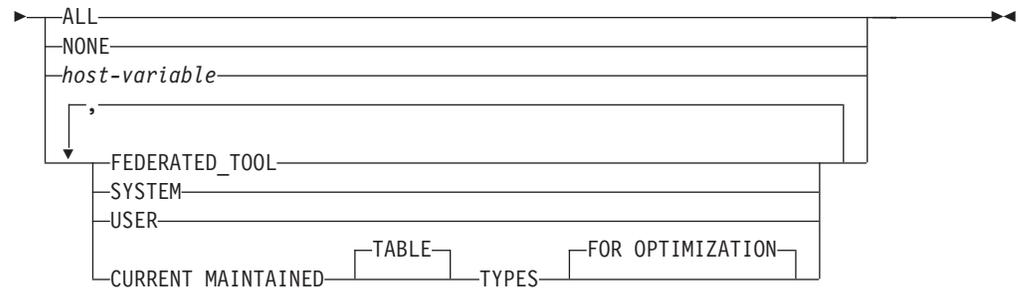
### 許可

必要ありません。

### 構文

```
▶▶ SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION [=]
```

## SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION



### 説明

#### ALL

動的 SQL 照会の処理を最適化する際に、この特殊レジスターによって制御されるすべての有効なタイプの保守されている表が、現在および将来に考慮されるよう指定します。

#### NONE

動的 SQL 照会の処理を最適化する際に、この特殊レジスターによって制御されるオブジェクト・タイプが考慮されないよう指定します。

#### FEDERATED\_TOOL

CURRENT QUERY OPTIMIZATION 特殊レジスターの値が 2 であるかまたは 5 より大きいときに、フェデレーテッド・ツールによって保守されているリフレッシュ据え置きマテリアライズ照会表が動的 SQL 照会の処理を最適化すると見なすことを指定します。

#### SYSTEM

動的 SQL 照会の処理を最適化する際に、システムによって保守されているリフレッシュ据え置きマテリアライズ照会表が考慮されるよう指定します。(即時マテリアライズ照会表は常に使用できます。)

#### USER

動的 SQL 照会の処理を最適化する際に、ユーザーが保守しているリフレッシュ据え置きマテリアライズ照会表が考慮されるよう指定します。

#### CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

このステートメントを実行する前の CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターの値。

#### host-variable

タイプ CHAR または VARCHAR の変数です。ホスト変数の内容の長さは、254 バイトを超えてはなりません (SQLSTATE 42815)。NULL に設定することはできません。host-variable が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

host-variable の文字は左寄せされていなければなりません。host-variable の内容は、特殊レジスターのキーワードとして指定できるキーワードをコンマで区切ってリストにした文字列です。大文字変換は行われなため、これらのキーワードはすべて大文字小文字を区別して指定しなければなりません。値の長さがホスト変数の長さ未満の場合は、値の右側を空白で埋め込まなければなりません。

## SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

### 注

- CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスタの初期値は SYSTEM です。
- 動的 SQL 照会の処理を最適化する際に、指定した表タイプが考慮されるようにするには、CURRENT REFRESH AGE 特殊レジスタをゼロ以外の値に設定しなければなりません。

### 例

例 1: CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスタを設定します。

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION SYSTEM = USER
```

例 2: CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスタの現行値を検索して CURMAINTYPES という名前のホスト変数に入れます。

```
EXEC SQL VALUES (CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION)
INTO :CURMAINTYPES
```

例 3: CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスタを値なしに設定します。

```
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION = NONE
```

---

## SET CURRENT MDC ROLLOUT MODE

SET CURRENT MDC ROLLOUT MODE ステートメントは、CURRENT MDC ROLLOUT MODE 特殊レジスタに値を割り当てます。この値は、マルチディメンション・クラスタリング (MDC) 表に適格である DELETE ステートメントに対して実行されるロールアウト・クリーンアップのタイプを指定します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文

```
→ SET CURRENT MDC ROLLOUT MODE { NONE | IMMEDIATE | DEFERRED | host-variable }
```

**説明****NONE**

削除操作中に MDC ロールアウト最適化は使用されないことを指定します。  
DELETE ステートメントは、ロールアウトできない DELETE ステートメントと同じようにして処理されます。

**IMMEDIATE**

DELETE ステートメントが適格である場合に MDC ロールアウト最適化が使用されることを指定します。表に RID 索引がある場合、削除処理中にそれらの索引は即時に更新されます。削除されたブロックは、トランザクションがコミットされた後に再利用できるようになります。

**DEFERRED**

DELETE ステートメントが適格である場合に MDC ロールアウト最適化が使用されることを指定します。表に RID 索引がある場合、索引の更新はトランザクションがコミットされるまで据え置かれます。このオプションを使用すると、削除処理は速くなり使用されるログ・スペースも少なくなります。削除されたブロックは索引の更新が完了するまで再利用できません。

*host-variable*

タイプが VARCHAR の変数です。 *host-variable* の長さは 17 バイト以下でなければなりません (SQLSTATE 42815)。ホスト変数の値は、「NONE」、「IMMEDIATE」、または「DEFERRED」(大/小文字を区別しない)のいずれかの、左揃えされたストリングでなければなりません。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

**注**

- ロールアウト処理に適格である後続の DELETE ステートメントは、CURRENT MDC ROLLOUT MODE 特殊レジスターの設定に従います。現在実行中のセッションは、この特殊レジスターを変更しても影響を受けません。
- SET CURRENT MDC ROLLOUT MODE ステートメントが実行される作業単位がロールバックされても、このステートメントの実行結果はロールバックされません。

**例**

ロールアウト処理に適格である次の DELETE ステートメントに対して据え置きクリーンアップ動作を指定します。

```
SET CURRENT MDC ROLLOUT MODE IMMEDIATE
```

---

**SET CURRENT OPTIMIZATION PROFILE**

SET CURRENT OPTIMIZATION PROFILE ステートメントは、CURRENT OPTIMIZATION PROFILE 特殊レジスターに値を割り当てます。この値は、動的 DML ステートメントを準備するときにオプティマイザーが使用しなければならない最適化プロファイルを指定します。このステートメントは、トランザクションの制御下にはありません。

## SET CURRENT OPTIMIZATION PROFILE

ステートメントが評価されると、最適化プロファイルの名前が妥当であるかどうかを検査されますが、プロファイル自体は、オプティマイザーが動的 DML ステートメントを検出するまで処理されません。

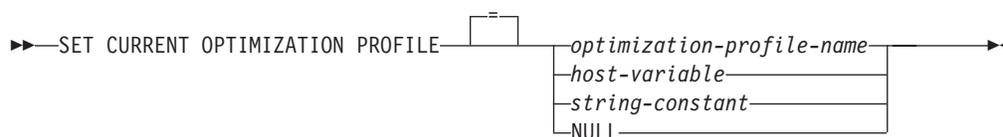
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文



### 説明

#### *optimization-profile-name*

最適化プロファイルの 2 部構成の名前。この名前は、リテラル、ホスト変数、または特殊レジスターを使用して指定できます。指定された名前は、CURRENT OPTIMIZATION PROFILE 特殊レジスターに入力される名前です。

指定した *optimization-profile-name* が修飾されていない場合、CURRENT DEFAULT SCHEMA レジスターの値が暗黙的な修飾子として使用されます。特殊レジスターのデフォルト値は NULL です。

#### *host-variable*

最適化プロファイルの名前を含む、タイプ CHAR または VARCHAR の変数です。NULL 標識を含んだホスト変数は、現行のパッケージに対して OPTPROFILE BIND オプションの値が指定されている場合にはその値を使用するということを示します。長さがゼロあるいは空白文字のみのホスト変数は、最適化プロファイルを使用しないということを示します。

ホスト変数は次の特性を満たしていなければなりません。

- スtringの内容が、単一 ID であるかまたは 2 部構成の ID (ピリオドで区切る) であり、先行ブランクがない。
- ID を区切りありまたは区切りなしにすることができる。
- Stringの内容が大文字に変換されていない。
- 区切りなしStringでは小文字および特殊文字を使用できない。
- 先頭文字が二重引用符である場合、終了二重引用符はピリオドの前に置かれるか、あるいはString内の最後の非ブランク文字でなければならない。
- ピリオドの後に続く先頭文字が二重引用符である場合、String内の最後の非ブランク文字は二重引用符でなければならない。

## SET CURRENT OPTIMIZATION PROFILE

- ID が区切られている場合、二重引用符を ID に含めるには、その文字を 2 回指定する。
- 区切り ID の内側にはないピリオドは区切り記号として扱われ、ストリング内に存在可能なピリオドの区切り文字は、1 つだけである。

### *string-constant*

最適化プロファイルの名前である文字ストリングとして定数を指定します。ストリング定数の内容は、ホスト変数と同じ特性を満たしている必要があります。

### NULL

CURRENT OPTIMIZATION PROFILE レジスターを NULL に設定します。

表 34 は、最適化プロファイルの命名規則に従って、レジスターを割り当てるのに使用可能なストリング・リテラルおよび ID の例を示しています。「SCHEMA」列および「NAME」列内の値は、OPT\_PROFILE 表に現れ得る形の最適化プロファイル名を表します。「有効なストリング・リテラル」列は、対応する「SCHEMA」列および「NAME」列の値で指定された最適化プロファイルと一致するストリング・リテラルを示しています。「有効な ID」列は、同じ最適化プロファイルを識別する ID を示しています。

表 34. ストリング・リテラルおよび ID の例

| SCHEMA | 名前          | 有効なストリング・リテラル            | 有効な ID                 |
|--------|-------------|--------------------------|------------------------|
| SIMMEN | BIG_PROF    | 'BIG_PROF'               | BIG_PROF               |
|        |             | 'SIMMEN.BIG_PROF'        | SIMMEN.BIG_PROF        |
|        |             | '"BIG_PROF"'             | "BIG_PROF"             |
|        |             | '"SIMMEN"."BIG_PROF"'    | "SIMMEN"."BIG_PROF"    |
| SIMMEN | low_profile | '"low_profile"'          | "low_profile"          |
|        |             | 'SIMMEN."low_profile"'   | SIMMEN."low_profile"   |
|        |             | '"SIMMEN"."low_profile"' | "SIMMEN"."low_profile" |
| eliaz  | DBA3        | 'DBA3'                   | DBA3                   |
|        |             | '"DBA3"'                 | "eliaz".DBA3           |
|        |             | '"eliaz".DBA3'           | "eliaz"."DBA3"         |
|        |             | '"eliaz"."DBA3"'         |                        |
| SNOW   | PROFILE1.0  | '"PROFILE1.0"'           | "PROFILE1.0"           |
|        |             | 'SNOW."PROFILE1.0"'      | SNOW."PROFILE1.0"      |
|        |             | '"SNOW"."PROFILE1.0"'    | "SNOW"."PROFILE1.0"    |

### 注

- レジスターの値が既存の最適化プロファイルの名前を指定している場合、後続の動的 DML ステートメントを準備するときには、指定された最適化プロファイルが使用されます。

## SET CURRENT OPTIMIZATION PROFILE

- レジスターの値が NULL である場合、後続の動的 DML ステートメントを準備するときには、OPTPROFILE BIND オプションで指定された最適化プロファイルがあれば、それが使用されます。
- レジスターの値が NULL であり、OPTPROFILE BIND オプションが設定されていない場合、後続の動的 DML ステートメントを準備するときに最適化プロファイルは使用されません。
- レジスターの値が空ストリング ("" ) である場合、OPTPROFILE BIND オプションが設定されているかどうかにかかわらず、後続の動的 DML ステートメントを準備するときには最適化プロファイルは使用されません。
- その後に CURRENT DEFAULT SCHEMA に変更を加えても、最適化プロファイルには影響しません。CURRENT OPTIMIZATION PROFILE レジスターの値は、SET CURRENT OPTIMIZATION PROFILE ステートメントが評価された時点で有効になる、2 部構成の名前で設定されます。使用される最適化プロファイルを変更する唯一の方法は、別の SET CURRENT OPTIMIZATION PROFILE ステートメントを実行することです。

### 例

例 1: ステートメント 1、2、および 3 には最適化プロファイル RICK.FOO が使用されます。ステートメント 4 には TOM.FOO が使用されます。

```
SET CURRENT SCHEMA = 'RICK'
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
statement 1
statement 2
SET CURRENT SCHEMA = 'TOM'
statement 3
SET CURRENT OPTIMIZATION PROFILE = 'FOO'
statement 4
```

例 2: 以下のステートメントを持つアプリケーションが、オプション OPTPROFILE("Foo") および QUALIFIER("John") を指定してバインドされました。ステートメント 1 には最適化プロファイル KAAREL.BAR が使用され、ステートメント 2 には最適化プロファイル "John"."Foo" が使用されます。

```
SET CURRENT SCHEMA = 'KAAREL'
SET CURRENT OPTIMIZATION PROFILE = 'BAR'
statement 1
SET CURRENT SCHEMA = "Tom"
SET CURRENT OPTIMIZATION PROFILE NULL
statement 2
```

例 3: 空ストリングは、最適化プロファイルを何も使用しないことを示す特殊値です。ステートメント 1 には最適化プロファイル "Hamid"."Foo" が使用され、ステートメント 2 には最適化プロファイルが何も使用されません。

```
SET CURRENT OPTIMIZATION PROFILE = '"Hamid"."Foo"
statement 1
SET CURRENT OPTIMIZATION PROFILE = ''
statement 2
```

## SET CURRENT PACKAGE PATH

SET CURRENT PACKAGE PATH ステートメントは、CURRENT PACKAGE PATH 特殊レジスターに値を割り当てます。このステートメントは、トランザクションの制御下にはありません。

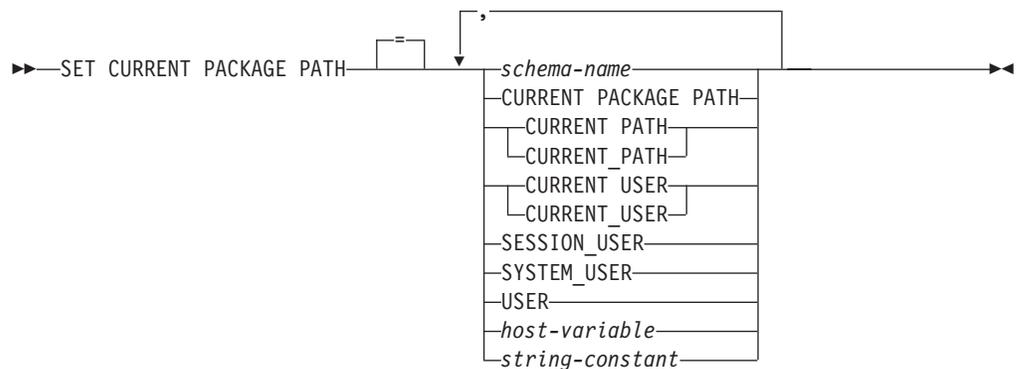
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

### 許可

必要ありません。

### 構文



### 説明

#### *schema-name*

スキーマを識別します。その名前は、空または空白だけの区切り ID であってはなりません (SQLSTATE 42815)。

#### **CURRENT PACKAGE PATH**

このステートメントが実行される前の CURRENT PACKAGE PATH 特殊レジスターの値。

#### **CURRENT PATH**

CURRENT PATH 特殊レジスターの値。

#### **CURRENT\_USER**

CURRENT\_USER 特殊レジスターの値。

#### **SESSION\_USER**

SESSION\_USER 特殊レジスターの値。

#### **SYSTEM\_USER**

SYSTEM\_USER 特殊レジスターの値。

#### **USER**

USER 特殊レジスターの値。

## SET CURRENT PACKAGE PATH

### *host-variable*

1 つ以上のスキーマ名をコンマで区切って指定します。ホスト変数は次の条件を満たしていなければなりません。

- 文字ストリング変数であること (CHAR または VARCHAR)。ホスト変数の内容の実際の長さは、CURRENT PACKAGE PATH 特殊レジスタの長さを超えてはなりません。
- NULL 値でないこと。標識変数が提供される場合、その値が NULL 値を示してはなりません。
- 空またはブランクのストリング、または 1 つ以上のスキーマ名をコンマで区切って指定すること。
- ホスト変数の実際の長さが内容よりも大きい場合は、右側をブランクで埋めること。
- CURRENT PACKAGE PATH、CURRENT PATH、CURRENT\_PATH、CURRENT USER、CURRENT\_USER、SESSION\_USER、SYSTEM\_USER、PATH、または USER を含まないこと。
- 空またはブランクだけを含む区切り ID を指定しないこと。

### *string-constant*

コンマで区切ったゼロ個以上のスキーマ名を含む文字ストリング定数を指定します。ストリング定数は次の条件を満たしていなければなりません。

- CURRENT PACKAGE PATH 特殊レジスタの最大長を超えない長さであること。
- CURRENT PACKAGE PATH、CURRENT PATH、CURRENT\_PATH、CURRENT USER、CURRENT\_USER、SESSION\_USER、SYSTEM\_USER、PATH、または USER を含まないこと。
- 空またはブランクだけを含む区切り ID を指定しないこと。

## 規則

- 複数の同じスキーマがリストに現れた場合は、最初に現れるスキーマが使用されます (SQLSTATE 01625)。
- 指定できるスキーマの数は、CURRENT PACKAGE PATH 特殊レジスタの合計長によって限定されます。特殊レジスタ・ストリングは、指定されたそれぞれのスキーマ名を採って末尾ブランクを除去し、名前を二重引用符で囲み、そしてスキーマ名をコンマで区切ることによって構築されます。結果リストの長さが、特殊レジスタの最大長を超えることはできません (SQLSTATE 0E000)。
- スキーマ名が通常 ID の規則に準拠していない場合 (たとえば、小文字を含むスキーマ名や、通常 ID に指定できない文字を含むスキーマ名など) には、区切り文字で区切られているスキーマ名として指定する必要があり、ホスト変数内またはストリング定数内に指定することはできません。
- 特殊レジスタ (単一キーワードとして指定したもの) の現行値がパッケージ・パス内で使用されることを指示するには、特殊レジスタの名前をキーワードとして指定します。その代わりに、特殊レジスタの名前が区切り ID として指定される場合 (たとえば "USER") は、その値のスキーマ名 ('USER') として解釈されます。
- SET CURRENT PACKAGE PATH ステートメントに指定された値が変数であるか、スキーマ名であるかを判別するために、次の規則が使用されます。

- *name* が SQL プロシージャ内のパラメーターまたは SQL 変数と同じ場合は、*name* はパラメーターまたは SQL 変数として解釈され、*name* の値がパッケージ・パスに割り当てられます。
- *name* が SQL プロシージャ内のパラメーターまたは SQL 変数と同じでない場合は、*name* はスキーマ名として解釈され、*name* の値がパッケージ・パスに割り当てられます。

## 注

- **トランザクションの考慮事項:** SET CURRENT PACKAGE PATH ステートメントはコミット可能な操作ではありません。ROLLBACK は CURRENT PACKAGE PATH 特殊レジスターに影響を及ぼしません。
- **スキーマの存在検査:** CURRENT PACKAGE PATH 特殊レジスターがセットされる時点では、指定されたスキーマが存在することの検証は行われません。たとえば、つづりを誤ったスキーマが検出されない場合、それは後続の SQL の作動の仕方に影響する可能性があります。パッケージの実行時には、合致するパッケージへの許可が検査され、この許可検査に失敗した場合はエラーが戻されます (SQLSTATE 42501)。
- **ホスト変数またはストリング定数の内容:** ホスト変数またはストリング定数の内容は、スキーマ名のリストとして解釈されます。複数のスキーマ名を指定する場合は、それぞれの名前をコンマで区切る必要があります。リスト内の各スキーマ名は、通常 ID を形成するための規則に準拠するか、区切り ID として指定する必要があります。ホスト変数またはストリング定数の内容は大文字変換されません。
- **COBOL アプリケーション用の組み込み SQL に特有の制限:** SET CURRENT PACKAGE PATH ステートメントの右辺には、最大で 10 個のリテラル (非ホスト変数) を指定できます。そのような値の最大長は 130 (区切りなし) または 128 (区切りあり) です。

## 例

例 1: CURRENT PACKAGE PATH 特殊レジスターを、以下のスキーマのリストに設定します: MYPKGS, 'ABC E', SYSIBM

```
SET CURRENT PACKAGE PATH = MYPKGS, 'ABC E', SYSIBM
```

次のステートメントは、ホスト変数を結果リストの値に設定します。

```
SET :hvpklist = CURRENT PACKAGE PATH
```

ホスト変数の値は: "MYPKGS", "ABC E", "SYSIBM"

例 2: CURRENT PACKAGE PATH 特殊レジスターを、以下のスキーマのリストに設定します: "SCH4","SCH5" (ただし、:hvar1 は 'SCH4,SCH5' を含みます)

```
SET CURRENT PACKAGE PATH :hvar1
```

このステートメントの実行後の CURRENT PACKAGE PATH 特殊レジスターの値は: "SCH4","SCH5"

例 3: CURRENT PACKAGE PATH 特殊レジスターを、以下のスキーマのリストに設定します: "SCH1","SCH#2","SCH3","SCH4","SCH5" (ただし、:hvar1 は 'SCH4,SCH5' を含みます)

## SET CURRENT PACKAGE PATH

```
SET CURRENT PACKAGE PATH = SCH1,'SCH#2',"SCH3",:hvar1
```

このステートメントの実行後の CURRENT PACKAGE PATH 特殊レジスターの値は: "SCH1","SCH#2","SCH3","SCH4","SCH5"

例 4: CURRENT PACKAGE PATH 特殊レジスターをクリアします。

```
SET CURRENT PACKAGE PATH = ''
```

例 5: SUMMARIZE プロシージャの実行のために、"SCH\_PROD" スキーマ (:prodschema ホスト変数に含まれる) および "SCH\_PROD2" スキーマ (:prod2schema ホスト変数に含まれる) を、CURRENT PACKAGE PATH 特殊レジスターの末尾に一時的に付加します。それから、CURRENT PACKAGE PATH 特殊レジスターを以前の値に戻します。

```
SET :oldCPP = CURRENT PACKAGE PATH
```

```
SET CURRENT PACKAGE PATH = CURRENT PACKAGE PATH,:prodschema,:prod2schema
```

```
CALL SUMMARIZE(:V1,:V2)
```

```
SET CURRENT PACKAGE PATH = :oldCPP
```

例 6: CURRENT PACKAGE PATH 特殊レジスターを、区切り文字で区切られているスキーマ名のリストに設定します: "MY.SCHEMA" (組み込みピリオド)、"OLD SCHEMA" (組み込みブランク)。両方の区切り ID を含む単一のホスト変数を使用します。

```
hv = 'MY.SCHEMA', 'OLD SCHEMA'
```

```
SET CURRENT PACKAGE PATH = :hv
```

あるいは、両方の区切り ID を含む単一のストリング定数を使用します。

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

あるいは、区切り文字で区切られているスキーマのリストを使用します。

```
SET CURRENT PACKAGE PATH = 'MY.SCHEMA', 'OLD SCHEMA'
```

---

## SET CURRENT PACKAGESET

SET CURRENT PACKAGESET ステートメントは、それ以降の SQL ステートメントで使用されるパッケージの選択に使用されるスキーマ名 (コレクション ID) を設定します。このステートメントは、トランザクションの制御下にありません。

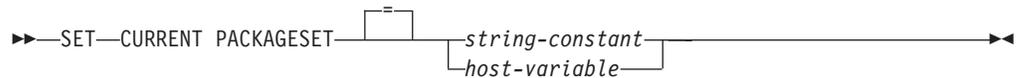
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。このステートメントは REXX ではサポートされません。

### 許可

必要ありません。

## 構文



## 説明

### *string-constant*

文字ストリング定数です。値が 128 バイトを超える場合は、先頭の 128 バイトだけが使用されます。

### *host-variable*

タイプ CHAR または VARCHAR の変数です。NULL に設定することはできません。値が 128 バイトを超える場合は、先頭の 128 バイトだけが使用されます。

## 注

- このステートメントを使用して、アプリケーションは、実行可能な SQL ステートメントのパッケージを選択する際に使用するスキーマ名を指定することができます。このステートメントは、クライアントで処理され、アプリケーション・サーバーへの流れはありません。
- COLLECTION BIND オプションを使用して、指定したスキーマ名を伴うパッケージを作成できます。
- DB2 for z/OS とは異なり、SET CURRENT PACKAGESET ステートメントは、CURRENT PACKAGESET 特殊レジスタのサポートなしでインプリメントされています。

## 例

TRYIT というアプリケーションがユーザー ID PRODUSA によってプリコンパイルされ、バインド・ファイルのデフォルトのスキーマ名は 'PRODUSA' になっていると想定します。その後、このアプリケーションは、異なる BIND オプションを使用して 2 回バインドされます。以下のコマンド行プロセッサのコマンドが使用されました。

```
DB2 CONNECT TO SAMPLE USER PRODUSA
DB2 BIND TRYIT.BND DATETIME USA
DB2 CONNECT TO SAMPLE USER PRODEUR
DB2 BIND TRYIT.BND DATETIME EUR COLLECTION 'PRODEUR'
```

これにより、TRYIT というパッケージが 2 つ作成されます。最初の BIND コマンドでは、'PRODUSA' というスキーマにパッケージが作成されます。2 番目 BIND コマンドでは、COLLECTION オプションに基づいて 'PRODEUR' というスキーマにパッケージが作成されます。

ここで、アプリケーション TRYIT に、次のステートメントが含まれていると想定します。

```
EXEC SQL CONNECT TO SAMPLE;
.
.
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 1
.
```

## SET CURRENT PACKAGESET

```
EXEC SQL SET CURRENT PACKAGESET 'PRODEUR'; 2
.
EXEC SQL SELECT HIREDATE INTO :HD FROM EMPLOYEE WHERE EMPNO='000010'; 3
```

- 1 このステートメントは、アプリケーションのデフォルトのパッケージである PRODUSA.TRYIT パッケージを使って実行されます。日付は、USA 形式で戻されます。
- 2 このステートメントは、パッケージ選択のスキーマ名を 'PRODEUR' に設定します。
- 3 SET CURRENT PACKAGESET ステートメントの結果として、このステートメントは PRODEUR.TRYIT パッケージを使用して実行されます。日付は、EUR 形式で戻されます。

---

## SET CURRENT QUERY OPTIMIZATION

SET CURRENT QUERY OPTIMIZATION ステートメントは、CURRENT QUERY OPTIMIZATION 特殊レジスターに値を割り当てます。この値は、動的 SQL ステートメントの準備の時点で使用される最適化手法の現行クラスを指定します。このステートメントは、トランザクションの制御下にはありません。

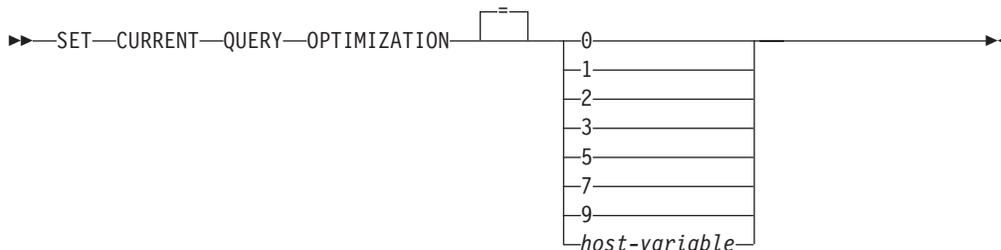
### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

### 構文



### 説明

#### *optimization-class*

*optimization-class* (最適化クラス) は、整数定数、または実行時に適切な値が入られるホスト変数の名前として指定することができます。クラスの概要を以下に示します。

- 0 アクセス・プランの生成に、最低限の最適化が行われることを指定しま

す。このクラスは、適切な索引が付けられた表にアクセスする単純な動的 SQL の場合に最も適しています。

- 1 アクセス・プランの生成に、DB2 バージョン 1 に匹敵する最適化を行うことを指定します。
- 2 DB2 バージョン 1 よりも高度な最適化を指定します。ただし、特にきわめて複雑な照会の場合、レベル 3 やそれ以降よりも最適化コストは大幅に低くなります。
- 3 アクセス・プランの生成に、中程度の最適化を行うことを指定します。
- 5 アクセス・プランの生成に、かなり高度な最適化を行うことを指定します。動的 SQL 照会が複雑な場合には、アクセス・プランの選択にかかる時間を制限するのに発見的手法の規則が使用されます。可能な場合、照会では基礎となる基本表ではなくマテリアライズ照会表が使用されます。
- 7 アクセス・プランの生成に、かなり高度な最適化を行うことを指定します。5 とほとんど同じですが、発見的手法の規則が使用されない点が異なります。
- 9 アクセス・プランの生成に、最大限の最適化を行うことを指定します。これにより、評価対象のアクセス・プランの数は大幅に増大します。このクラスは、大規模な表を使用するきわめて複雑で、実行に長時間を要する照会に対して、より良いアクセス・プランを生成することを判断する場合に使うようにしてください。EXPLAIN とパフォーマンス測定値を使用することにより、効率的なプランが生成されたかどうかを検証することができます。

#### *host-variable*

データ・タイプは INTEGER です。値は、0 から 9 の範囲内である必要があります (SQLSTATE 42815)。ただし、値は、0、1、2、3、5、7、または 9 のいずれかでなければなりません。*host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

## 注

- CURRENT QUERY OPTIMIZATION レジスターを特定の値に設定すると、一連の照会書き直し規則が有効になり、特定の最適化変数が特定の値になります。該当のクラスの最適化手法が、動的 SQL ステートメントの準備の過程で使用されます。
- 一般に、最適化クラスの変更は、アプリケーションの実行時間、コンパイル時間、および必要なリソースに影響を与えます。多くのステートメントは、デフォルトの照会最適化クラスを用いて適切な最適化が行われます。動的 SQL ステートメントに対して、動的 PREPARE が消費するリソースが、照会の実行に必要なリソースのかなりの部分を占める場合には、低い照会最適化クラス (特にクラス 1 と 2) が動的 SQL ステートメントに適している場合があります。より高いレベルの最適化クラスは、消費するリソースがどれだけ増えるかを検討し、より良いアクセス・プランが生成されたことを確認して初めて、選択するようにしてください。

## SET CURRENT QUERY OPTIMIZATION

- 照会最適化クラスは、0 から 9 の範囲でなければなりません。この範囲外のクラスは、エラーになります (SQLSTATE 42815)。この範囲内でサポートされていないクラスを指定すると、警告 (SQLSTATE 01608) が戻され、より低い次の照会最適化クラスで置き換えられます。たとえば、照会最適化クラス 6 は 5 に置き換えられます。
- 動的に準備されるステートメントは、最近、実行された SET CURRENT QUERY OPTIMIZATION ステートメントによって設定された最適化クラスを使用します。SET CURRENT QUERY OPTIMIZATION ステートメントがまだ実行されていない場合、照会最適化クラスは **dft\_queryopt** データベース構成パラメーターによって決まります。
- 静的にバインドされたステートメントでは、CURRENT QUERY OPTIMIZATION 特殊レジスターを使用しません。したがって、このレジスターはそれらのステートメントに影響を与えません。静的にバインドされたステートメントに対する必要な最適化クラスの指定には、プリプロセスまたはバインドの過程で QUERYOPT オプションが使用されます。QUERYOPT の指定がない場合は、**dft\_queryopt** データベース構成パラメーターによって指定されたデフォルト値が使用されます。
- SET CURRENT QUERY OPTIMIZATION ステートメントが実行される作業単位がロールバックされても、このステートメントの実行結果はロールバックされません。

### 例

例 1: この例は、最も程度の高い最適化を選択する方法を示しています。

```
SET CURRENT QUERY OPTIMIZATION 9
```

例 2: 以下の例は、照会の中で CURRENT QUERY OPTIMIZATION 特殊レジスターを使用する方法を示しています。

以下の例は、SYSCAT.PACKAGES カタログ・ビューを使用して、CURRENT QUERY OPTIMIZATION 特殊レジスターの現行値と同じ設定でバインドされたすべてのプランを検索しています。

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT PKGNAME, PKGSHEMA FROM SYSCAT.PACKAGES
WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

---

## SET CURRENT REFRESH AGE

SET CURRENT REFRESH AGE ステートメントは、CURRENT REFRESH AGE 特殊レジスターの値を変更します。このステートメントは、トランザクションの制御下にはありません。

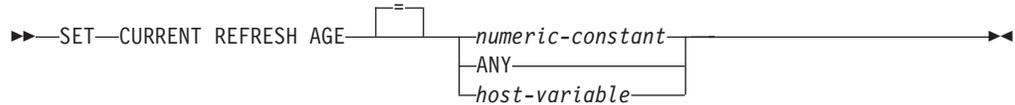
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

必要ありません。

## 構文



## 説明

*numeric-constant*

タイム・スタンプ期間を表す DECIMAL(20,6) 値。この値には、0 または 99 999 999 999 999 を指定しなければなりません (値のマイクロ秒部分は無視されるので、任意の値を指定できます)。

## ANY

これは 99 999 999 999 999 の簡潔な表記です。

*host-variable*

タイプ DECIMAL(20,6) の変数、または DECIMAL(20,6) に割り当て可能な他のタイプ。NULL に設定することはできません。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。 *host-variable* の値には、0 または 99 999 999 999 999 を指定しなければなりません。

## 注

- CURRENT REFRESH AGE 特殊レジスタの初期値はゼロです。
- CURRENT REFRESH AGE は、指定した値によって置き換えられます。値は、0 または 99 999 999 999 999 を指定しなければなりません。99 999 999 999 999 という値は、9999 年、99 カ月、99 日、99 時間、99 分、99 秒を表します。

CURRENT REFRESH AGE の値が 0 の場合は、この特殊レジスターから影響を受けるマテリアライズ照会表は、照会の処理を最適化するために使用されません。CURRENT REFRESH AGE の値が 99 999 999 999 999 の場合は、この特殊レジスターから影響を受けるマテリアライズ照会表は、照会の処理を最適化するために使用できますが、それが可能なのは、CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターの値がそれらを含んでおり、CURRENT QUERY OPTIMIZATION 特殊レジスターが 2 か、5 以上の値に設定されている場合だけです。この特殊レジスターによって影響を受けるマテリアライズ照会表は、REFRESH DEFERRED MAINTAINED BY USER と REFRESH DEFERRED MAINTAINED BY SYSTEM です。

CURRENT QUERY OPTIMIZATION 特殊レジスターが 2 か、5 以上の値に設定されている場合、REFRESH IMMEDIATE MAINTAINED BY SYSTEM マテリアライズ照会表は常に、照会の処理を最適化するために使用できます。

CURRENT QUERY OPTIMIZATION 特殊レジスターが 2 か、5 以上の値に設定されていて、CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターの値が ALL に設定されているか FEDERATED\_TOOL を含む場

## SET CURRENT REFRESH AGE

合に、REFRESH DEFERRED MAINTAINED BY FEDERATED\_TOOL マテリアライズ照会表は照会の処理を最適化するために使用できます。

- **CURRENT REFRESH AGE** 特殊レジスターをゼロ以外の値に設定する場合は、注意が必要です。CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターによって指定される表タイプは、基礎となる基本表の値を表していない可能性があります。そのような表を使用して照会の処理を最適化する場合、照会結果は基礎表内のデータを正確に表していない可能性があります。とはいえ、基礎データが変化していないことが分かっている場合、あるいはキャッシュに入れられたデータに関する知識に基づいて照会結果のエラーの度合いを受け入れるつもりである場合、これはさほど気になる問題にならないことがあります。
- タイム・スタンプの算術演算では、CURRENT REFRESH AGE の値として 99 999 999 999 999 を使用することはできません。その結果が、日付の有効範囲外になるからです (SQLSTATE 22008)。

### 例

例 1: 以下のステートメントは、CURRENT REFRESH AGE 特殊レジスターを設定します。

```
SET CURRENT REFRESH AGE ANY
```

例 2: 以下の例では、CURRENT REFRESH AGE 特殊レジスターの現行値を検索して CURMAXAGE という名前のホスト変数に入れます。前の例で設定された値は 9999999999999999.000000 です。

```
EXEC SQL VALUES (CURRENT REFRESH AGE) INTO :CURMAXAGE;
```

---

## SET ENCRYPTION PASSWORD

SET ENCRYPTION PASSWORD ステートメントは、ENCRYPT、DECRYPT\_BIN、および DECRYPT\_CHAR 関数によって使用されるパスワードを設定します。このパスワードは DB2 認証には関連付けられず、データの暗号化および暗号化解除にのみ使用されます。

このステートメントは、トランザクションの制御下にありません。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また対話式に出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

## 構文

```

▶▶ SET ENCRYPTION PASSWORD host-variable | string-constant

```

## 説明

ENCRYPTION PASSWORD は、パスワード・ベースの暗号化のための ENCRYPT、DECRYPT\_BIN、および DECRYPT\_CHAR 組み込み関数によって使用できます。長さは 6 から 127 バイトの間でなければなりません。自動大文字変換は行われないため、文字はすべて大文字小文字を区別して指定しなければなりません。

### *host-variable*

タイプ CHAR または VARCHAR の変数です。 *host-variable* の長さは 6 から 127 バイトの間でなければなりません (SQLSTATE 428FC)。 NULL に設定することはできません。大文字変換は行われないため、文字はすべて大文字小文字を区別して指定します。

### *string-constant*

文字ストリング定数です。長さは 6 から 127 バイトの間でなければなりません (SQLSTATE 428FC)。

## 注

- ENCRYPTION PASSWORD の初期値は空ストリング ('') です。
- *host-variable* または *string-constant* は、標準 DB2 メカニズムでデータベース・サーバーに送信されます。

## 例

例 1: 以下のステートメントは、ENCRYPTION PASSWORD を設定します。

```
SET ENCRYPTION PASSWORD = 'Gre89Ea'
```

---

## SET EVENT MONITOR STATE

SET EVENT MONITOR STATE ステートメントは、イベント・モニターの活動化、または非活動化を行います。 イベント・モニターの現在の状態 (アクティブまたは非アクティブ) は、EVENT\_MON\_STATE 組み込み関数によって判別することができます。 SET EVENT MONITOR STATE ステートメントは、トランザクションの制御下にありません。

## 呼び出し

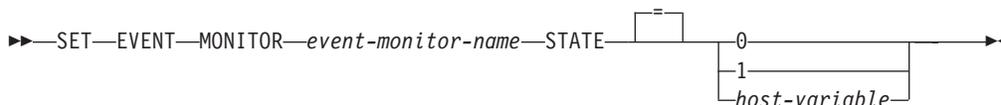
このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

## SET EVENT MONITOR STATE

### 許可

このステートメントの許可 ID が持つ特権には、DBADM または SYSADM 権限が含まれている必要があります。

### 構文



### 説明

#### *event-monitor-name*

活動化または非活動化するイベント・モニターを指定します。この名前は、カタログに存在しているイベント・モニターを指定していなければなりません (SQLSTATE 42704)。

#### *new-state*

*new-state* (新しい状態) は、整数定数として、または実行時に適切な値が入られるホスト変数の名前として指定することができます。指定可能な値は、次のとおりです。

- 0** 指定したイベント・モニターを非活動化することを指定します。
- 1** 指定したイベント・モニターを活動化することを指定します。そのイベント・モニターはすでにアクティブであってはなりません。そうでない場合、警告 (SQLSTATE 01598) が出されます。

#### *host-variable*

データ・タイプは INTEGER です。指定する値は、0 または 1 でなければなりません (SQLSTATE 42815)。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

### 規則

- 定義できるイベント・モニターの数には制限はありませんが、同時にアクティブ化できるイベント・モニターの数 は 32 までです (SQLSTATE 54030)。
- イベント・モニターを活動化するには、そのイベント・モニターが作成されたトランザクションはコミットされていなければなりません (SQLSTATE 55033)。この規則は、(1 つの作業単位内で) イベント・モニターを作成し、そのモニターを活動化し、その後で、トランザクションをロールバックするのを防止します。
- イベント・モニター・ファイルの数またはサイズが、 CREATE EVENT MONITOR ステートメントの MAXFILES または MAXFILESIZE に指定された値を超える場合には、エラー (SQLSTATE 54031) になります。
- イベント・モニターのターゲット・パス (CREATE EVENT MONITOR ステートメントにより指定) が、他のイベント・モニターですでに使用されている場合、エラー (SQLSTATE 51026) になります。

**注**

- イベント・モニターを活動化すると、それに対応するカウンターはいずれもリセットされます。
- **WRITE TO TABLE** イベント・モニターは、**SET EVENT MONITOR STATE** を使用して開始されると、**SYSCAT.EVENTMONITORS** カタログ・ビューの **EVMON\_ACTIVATES** 列を更新します。セット演算が実行された作業単位が何らかの理由でロールバックされると、そのカタログ更新は失われます。イベント・モニターが再開したときに、ロールバックされた **EVMON\_ACTIVATES** 値が再使用されます。
- イベント・モニターを実行するデータベース・パーティションがアクティブでない場合は、次回そのデータベース・パーティションを活動化した時点で、イベント・モニターも活動化されます。
- イベント・モニターは、活動化の後に、明示的に非活動化されるか、インスタンスがリサイクルされるまで、自動始動のイベント・モニターのように動作します。つまり、データベース・パーティションが非活動化された時点でイベント・モニターがアクティブであれば、そのデータベース・パーティションがそれ以降に再び活動化された時点で、イベント・モニターも明示的に再活動化されます。
- データベースが非活動化するときに、アクティビティ・イベント・モニターがアクティブである場合、キューにあるバックログされたアクティビティ・レコードはすべて廃棄されます。確実にすべてのアクティビティ・イベント・モニター・レコードを入手し、何も廃棄されないようにするには、データベースを非活動化する前に、まずアクティビティ・イベント・モニターを明示的に非活動化します。アクティビティ・イベント・モニターを明示的に非活動化した場合、キューにあるバックログされたアクティビティ・レコードはすべて、イベント・モニターが非活動化される前に処理されます。

**例**

例 1: SMITHPAY という名前のイベント・モニターを活動化します。

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

例 2: MYSAMPLE という複数パーティション・データベースがあり、それぞれのデータベース・パーティションを 0 および 2 と想定します。パーティション 2 はまだアクティブではありません。

データベース・パーティション 0 上で次のようにします。

```
CONNECT TO MYSAMPLE;
CREATE EVENT MONITOR MYEVMON ON DBPARTITIONNUM 2;
SET EVENT MONITOR MYEVMON STATE 1;
```

MYSAMPLE がデータベース・パーティション 2 で活動化すると、MYEVMON も自動的に活動化します。この動作は、**SET EVENT MONITOR MYEVMON STATE 0** を実行するか、パーティション 2 が停止するまで続きます。

---

**SET INTEGRITY**

**SET INTEGRITY** ステートメントは、以下の目的で使用されます。

## SET INTEGRITY

- 1 つ以上の表に対して必要な保全性処理を実行することによって、それらの表の SET INTEGRITY ペンディング状態 (以前の「チェック・ペンディング状態」) を解除する。
- 1 つ以上の表に対して必要な保全性処理を実行しないで、それらの表の SET INTEGRITY ペンディング状態を解除する。
- 1 つ以上の表を SET INTEGRITY ペンディング状態にする。
- 1 つ以上の表をフル・アクセス状態にする。
- 1 つ以上のステージング表の内容を整理する。

表のロード後またはアタッチ後に表の保全性処理を実行するためにステートメントを使用する場合、システムは、制約に違反する追加部分だけを検査するという、増分的な表の処理を実行できます。サブジェクト表がマテリアライズ照会表かステージング表であり、その基礎表でロード、アタッチ、デタッチの各操作が実行される場合、システムは、増分的にマテリアライズ照会表をリフレッシュしたり、基礎表のデルタ部分だけを使用して、増分的にステージング表に伝搬したりすることができます。ただし、システムでは、そのような最適化を実行できないので、代わりに、データ保全性を確保するための完全保全性処理を実行する場合があります。完全保全性処理は、制約違反がないか表全体を検査すること、マテリアライズ照会表の定義を再計算すること、またはステージング表を不整合としてマーク付けすることで行われます。後者の方法の場合、関連するマテリアライズ照会表のフル・リフレッシュが必要であることを意味します。また、INCREMENTAL オプションを指定して、増分処理を明示的に要求できる場合があります。

SET INTEGRITY ステートメントは、トランザクションの制御下にあります。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

SET INTEGRITY ステートメントの実行に必要な特権は、目的によって以下のように異なります。

- 必要な保全性処理を実行して、表の SET INTEGRITY ペンディング状態を解除する場合。

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 以下に対する CONTROL 特権
  - 保全性処理が実行される表および、例外表がこのような表の 1 つ以上に提供されている場合、例外表の INSERT 特権
  - ステートメントによって暗黙的に SET INTEGRITY ペンディング状態にされる、下層外部キー表、下層即時マテリアライズ照会表、および下層即時ステージング表のすべて。

- LOAD 権限 (条件付き)。適切な特権の提供として LOAD 権限を考慮する前に、以下の条件のすべてが満たされる必要があります。
  - 必要な保全性処理に以下のアクションは伴いません。
    - マテリアライズ照会表のリフレッシュ
    - ステージング表への伝搬
    - 生成済み列または ID 列の更新
  - 例外表が 1 つ以上の表に対して提供されているならば、保全性処理が実行されている表および関連する例外表への保全性処理期間中に、必要なアクセスが与えられます。すなわち、
    - 保全性処理が実行されるそれぞれの表に対する SELECT および DELETE 特権、および
    - 例外表に対する INSERT 特権
- SYSADM または DBADM 権限
- 必要な保全性処理を実行しないで、表の SET INTEGRITY ペンディング状態を解除する場合。

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 処理対象表に対する CONTROL 特権。ステートメントによって暗黙的に SET INTEGRITY ペンディング状態にされるそれぞれの下層外部キー表、下層即時マテリアライズ照会表、下層即時ステージング表に対する CONTROL 特権。
- LOAD 権限
- SYSADM または DBADM 権限
- 表を SET INTEGRITY ペンディング状態にする場合。

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 以下に対する CONTROL 特権
  - 指定された表
  - ステートメントによって SET INTEGRITY ペンディング状態にされる下層外部キー表
  - ステートメントによって SET INTEGRITY ペンディング状態にされる下層即時マテリアライズ照会表
  - ステートメントによって SET INTEGRITY ペンディング状態にされる下層即時ステージング表
- LOAD 権限
- SYSADM または DBADM 権限
- 表をフル・アクセス状態にする場合。

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- フル・アクセス状態にする表に対する CONTROL 特権
- LOAD 権限
- SYSADM または DBADM 権限

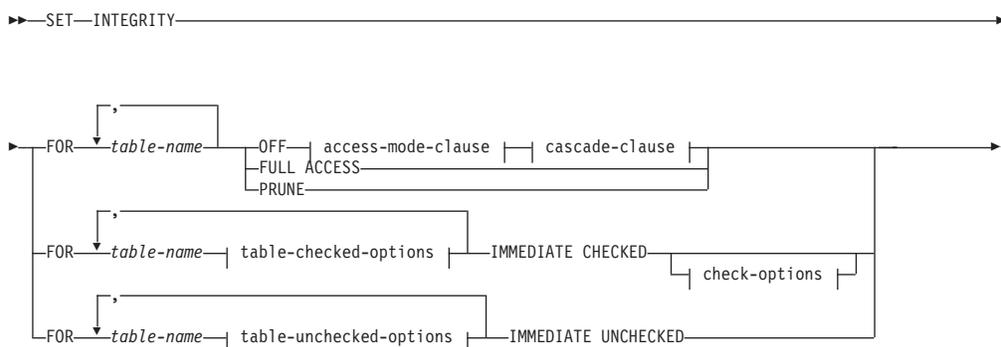
## SET INTEGRITY

- ステージング表を整理する場合。

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 整理する表に対する CONTROL 特権
- SYSADM または DBADM 権限

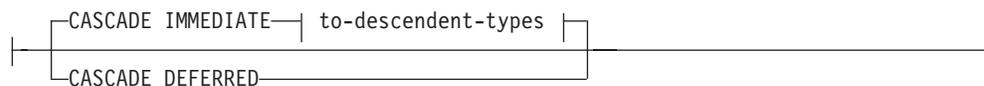
### 構文



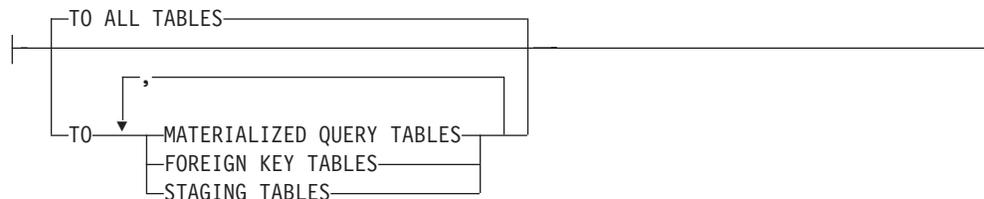
#### access-mode-clause:



#### cascade-clause:



#### to-descendent-types:



#### table-checked-options:



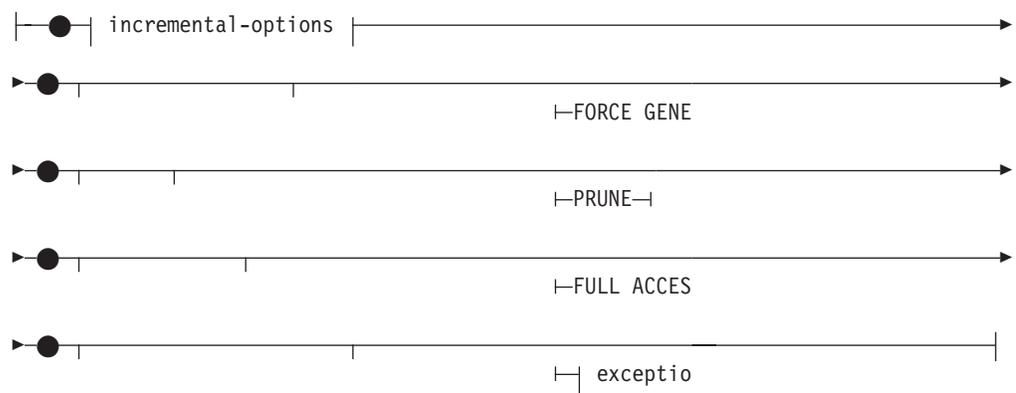
**online-options:**



**query-optimization-options:**



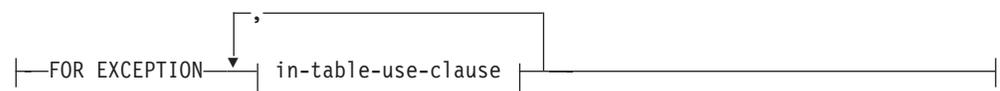
**check-options:**



**incremental-options:**



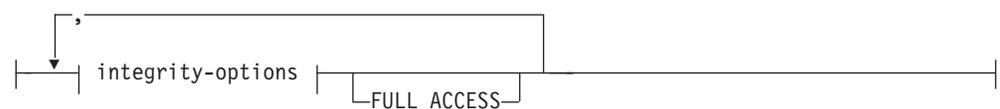
**exception-clause:**



**in-table-use-clause:**

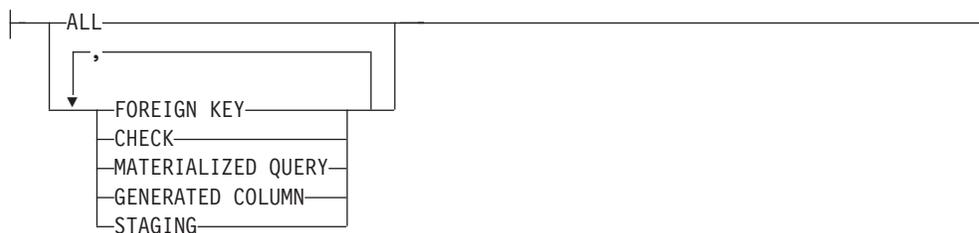


**table-unchecked-options:**



## SET INTEGRITY

### integrity-options:



### 説明

#### FOR *table-name*

保全性処理を行う表 (複数可) を指定します。これは、カタログに記述されている表でなければならず、ビュー、カタログ表、または型付き表を対象にすることはできません。

#### OFF

表を SET INTEGRITY ペンディング状態にすることを指定します。SET INTEGRITY ペンディング状態にある表に対しては、極めて限定されたアクティビティーのみが許されます。

#### *access-mode-clause*

SET INTEGRITY ペンディング状態のときの表の読み取り可否を指定します。

#### NO ACCESS

表を SET INTEGRITY ペンディング・アクセスなし状態にすることを指定します。この状態では、表への読み取りまたは書き込みアクセスは許可されません。

#### READ ACCESS

表を SET INTEGRITY ペンディング読み取りアクセス状態にすることを指定します。この状態では、表の追加部分以外への読み取りアクセスが許可されています。このオプションは、SET INTEGRITY ペンディング・アクセスなし状態の表に対しては許可されていません (SQLSTATE 428FH)。

#### *cascade-clause*

SET INTEGRITY ステートメントで参照される表の SET INTEGRITY ペンディング状態を下層表にすぐにカスケードするかどうかを指定します。

#### CASCADE IMMEDIATE

SET INTEGRITY ペンディング状態を下層表にすぐに拡張することを指定します。

#### *to-descendent-types*

SET INTEGRITY ペンディング状態をすぐにカスケードする下層表のタイプを指定します。

#### TO ALL TABLES

SET INTEGRITY ペンディング状態を呼び出しリストにある表のすべての下層表に対してすぐにカスケードすることを指定します。下層表には、呼び出しリストの表の下層である、あるいは下層外部キー表の下層である、すべての下層外部キー表、即時ステージング表、および即時マテリアライズ照会表が含まれます。

TO ALL TABLES を指定することは、TO FOREIGN KEY TABLES、TO MATERIALIZED QUERY TABLES、TO STAGING TABLES をすべて同じステートメントに指定することと等価です。

#### TO MATERIALIZED QUERY TABLES

TO MATERIALIZED QUERY TABLES だけを指定する場合、SET INTEGRITY ペンディング状態は、すぐに、下層即時マテリアライズ照会表に対してだけカスケードされます。他の下層表は、表の SET INTEGRITY ペンディング状態が解除されたときに、必要に応じて SET INTEGRITY ペンディング状態になるように設定できます。TO FOREIGN KEY TABLES と TO MATERIALIZED QUERY TABLES の両方を指定する場合、SET INTEGRITY ペンディング状態はすぐに、すべての下層外部キー表、呼び出しリストにある表のすべての下層即時マテリアライズ照会表、下層外部キー表の下層であるすべての即時マテリアライズ照会表にカスケードされます。

#### TO FOREIGN KEY TABLES

SET INTEGRITY ペンディング状態を下層外部キー表にすぐにカスケードすることを指定します。他の下層表は、表の SET INTEGRITY ペンディング状態が解除されたときに、必要に応じて SET INTEGRITY ペンディング状態になるように設定できます。

#### TO STAGING TABLES

SET INTEGRITY ペンディング状態を下層ステージング表にすぐにカスケードすることを指定します。他の下層表は、表の SET INTEGRITY ペンディング状態が解除されたときに、必要に応じて SET INTEGRITY ペンディング状態になるように設定できます。TO FOREIGN KEY TABLES と TO STAGING TABLES の両方を指定する場合、SET INTEGRITY ペンディング状態はすぐに、すべての下層外部キー表、呼び出しリストにある表のすべての下層即時ステージング表、下層外部キー表の下層であるすべての即時ステージング表にカスケードされます。

#### CASCADE DEFERRED

呼び出しリストに含まれる表だけを SET INTEGRITY ペンディング状態にすることを指定します。下層表の状態は未変更のままになります。下層外部キー表は、その親表の制約違反を検査するときに、暗黙的に SET INTEGRITY ペンディング状態になるように設定できます。下層即時マテリアライズ照会表と下層即時ステージング表は、基礎表のいずれかで健全性違反を検査するときに、暗黙的に SET INTEGRITY ペンディング状態になるように設定できます。

*cascade-clause* を指定しない場合は、SET INTEGRITY ペンディング状態がすぐにすべての下層表にカスケードされます。

#### IMMEDIATE CHECKED

必要な健全性処理を表に対して実行することによって、表の SET INTEGRITY ペンディング状態を解除することを指定します。これは、SYSCAT.TABLES カタログ・ビューの STATUS 列と CONST\_CHECKED 列に設定されている情報に基づいて行われます。すなわち、

- 表が、リストで指定され、SET INTEGRITY ペンディング状態にあり、さらに中間上層もリストに含まれる表の下層外部キー表、下層マテリアライズ照

会表、または下層ステージング表でない限り、STATUS 列の値は「C」(表は SET INTEGRITY ペンディング状態にあるという意味) でなければなりません。そうでない場合は、エラーが戻されます (SQLSTATE 51027)。

- 検査する表が SET INTEGRITY ペンディング状態にある場合、CONST\_CHECKED の値は、どの保全性オプションを検査するかを示します。

表の SET INTEGRITY ペンディング状態が解除されたときに、その下層表は、必要に応じて SET INTEGRITY ペンディング状態になります。下層表が SET INTEGRITY ペンディング状態になったことを示す警告が戻されます (SQLSTATE 01586)。

表がシステムによって保守されるマテリアライズ照会表であれば、照会に基づいてデータが検査され、必要に応じてリフレッシュされます。(IMMEDIATE CHECKED は、ユーザーが保守するマテリアライズ照会表には使用できません。) 表がステージング表であれば、その照会定義に基づいてデータが検査され、必要に応じて伝搬されます。

子表の保全性を検査する場合、以下のようになります。

- 親を SET INTEGRITY ペンディング状態にすることはできません。
- それぞれの親は、同じ SET INTEGRITY ステートメントで制約違反を検査される必要があります。

即時マテリアライズ照会表がリフレッシュされる場合、あるいは差分がステージング表に伝搬される場合、以下のようになります。

- 基礎表を SET INTEGRITY ペンディング状態にすることはできません。
- それぞれの基礎表は、同じ SET INTEGRITY ステートメントで検査される必要があります。

これ以外の場合には、エラーになります (SQLSTATE 428A8)。

### *table-checked-options*

#### *online-options*

処理中の表のアクセス可能性を指定します。

#### **ALLOW NO ACCESS**

他のユーザーは、非コミット読み取り分離レベルを使用している場合を除き、処理中の表にアクセスできないことを指定します。

#### **ALLOW READ ACCESS**

他のユーザーは処理中の表に対して読み取り専用アクセスを持つことを指定します。

#### **ALLOW WRITE ACCESS**

他のユーザーは処理中の表に対して読み取り/書き込みアクセスを持つことを指定します。

#### **GENERATE IDENTITY**

表に ID 列が含まれている場合に、SET INTEGRITY ステートメントによって値を生成することを指定します。GENERATE IDENTITY オプションを指定した場合、SET INTEGRITY ステートメントによって ID 列の値が生成されるのは、デフォルトでは、追加された行についてのみになります。表のすべての行 (追加された行、ロードされた行、既存の

行) の ID 列の値を SET INTEGRITY ステートメントによって生成するには、GENERATE IDENTITY オプションと一緒に NOT INCREMENTAL オプションを指定する必要があります。GENERATE IDENTITY オプションを指定しない場合は、表のすべての行の現在の ID 列の値が未変更のままになります。

#### *query-optimization-options*

**REFRESH DEFERRED** マテリアライズ照会表の保守に関する照会最適化オプションを指定します。

#### **ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY**

**CURRENT REFRESH AGE** 特殊レジスターが「ANY」に設定されている場合に、*table-name* の保守で REFRESH DEFERRED マテリアライズ照会表を使用することによって、*table-name* の保守に使用する照会を最適化できるようにすることを指定します。*table-name* が REFRESH DEFERRED マテリアライズ照会表でない場合は、エラーが戻されます (SQLSTATE 428FH)。REFRESH IMMEDIATE マテリアライズ照会表は、常に照会の最適化のときに考慮されません。

#### *check-options*

##### *incremental-options*

#### **INCREMENTAL**

表の追加部分 (もしあれば) に対して保全性処理を適用することを指定します。この要求が満たされない場合 (つまり、システムが表全体でデータ保全性検査を実行する必要があると判断する場合は、エラーが戻されます (SQLSTATE 55019)。

#### **NOT INCREMENTAL**

表全体に対して保全性処理を適用することを指定します。表がマテリアライズ照会表である場合、マテリアライズ照会表定義が再計算されます。表に少なくとも 1 つの制約が定義されている場合、このオプションを指定すると、下層外部キー表と下層即時マテリアライズ照会表が完全処理されます。表がステージング表の場合は、不整合状態に設定されます。

*incremental-options* 節を指定しない場合、システムは増分処理が可能かどうかを判断します。それが可能でなければ、表全体が検査されます。

#### **FORCE GENERATED**

表に式生成列が含まれている場合は、式に基づいてその値が計算され、列に保管されます。このオプションを指定しない場合は、等価チェック制約が有効であるかのように、現行値が式の算出値と比較されます。表の保全性が増分的に処理される場合、生成された列は追加部分についてのみ計算されます。

#### **PRUNE**

このオプションは、ステージング表の場合にのみ指定できます。ステージング表の内容を整理すること、ステージング表を矛盾状態にすることを指定します。*table-name* リストに含まれている表がステージング表

でなければ、エラーが戻されます (SQLSTATE 428FH)。  
 INCREMENTAL 検査オプションも指定されている場合、エラーが戻されます (SQLSTATE 428FH)。

### FULL ACCESS

SET INTEGRITY ステートメントの実行後に表を完全にアクセス可能にすることを指定します。

呼び出しリストにある基礎表 (従属即時マテリアライズ照会表または従属即時ステージング表を持つ基礎表) が増分的に処理される場合、その基礎表は、SET INTEGRITY ステートメントの実行後に、必要に応じてデータ移動なし状態に設定されます。増分的にリフレッシュ可能なすべての従属の即時マテリアライズ照会表とステージング表の SET INTEGRITY ペンディング状態が解除されると、基礎表は、自動的にデータ移動なし状態からフル・アクセス状態になります。IMMEDIATE CHECKED オプションと一緒に FULL ACCESS オプションを指定すると、基礎表は、データ移動なし状態をバイパスして、直接にフル・アクセス状態になります。リフレッシュされたことのない従属即時マテリアライズ照会表は、後続の REFRESH TABLE ステートメントですべてが再計算される可能性があり、伝搬元の表の追加部分を持たない従属即時ステージング表は、不整合状態のフラグが設定される可能性があります。

呼び出しリストにある基礎表が、完全処理を必要とするか、従属即時マテリアライズ照会表または従属即時ステージング表を持たない場合、その基礎表は、FULL ACCESS オプションが指定されているかどうかに関係なく、SET INTEGRITY ステートメントの実行後に直接にフル・アクセス状態に設定されます。

### *exception-clause*

#### FOR EXCEPTION

チェック対象の制約に違反している行を例外表に移動することを指定します。エラーが検出されても、表の SET INTEGRITY ペンディング状態は解除されます。1 つ以上の行が例外表に移されたことを示す警告が戻されます (SQLSTATE 01603)。

FOR EXCEPTION 節の指定がない場合に、制約違反が生じると、最初に検出された違反だけが戻されます (SQLSTATE 23514)。表のいずれかに違反がある場合は、すべての表が SET INTEGRITY ペンディング状態のままになります。

制約違反をチェックする場合は、違反が検出された場合に SET INTEGRITY ステートメントがロールバックされる事態を回避するために、常に FOR EXCEPTION オプションを使用することをお勧めします。

#### IN *table-name*

制約違反行の移動元の表を指定します。検査される各表ごとに、1 つの例外表を指定する必要があります。この節は、マテリアライズ照会表またはステージング表には指定できません (SQLSTATE 428A7)。

**USE *table-name***

エラー行の移動先にする例外表を指定します。

**FULL ACCESS**

ステートメントの唯一の操作として FULL ACCESS オプションを指定すると、表は保全性違反の再チェックなしでフル・アクセス状態になります。ただし、リフレッシュされたことのない従属即時マテリアライズ照会表は、後続の REFRESH TABLE ステートメントですべての再計算が必要になる可能性があります。伝搬元の表のデルタ部分を持たない従属即時ステージング表は、不整合状態に変更される可能性があります。このオプションは、データ移動なし状態またはアクセスなし状態でありながら、SET INTEGRITY ペンディング状態ではない表にのみ指定できます (SQLSTATE 428FH)。

**PRUNE**

このオプションは、ステージング表の場合にのみ指定できます。ステージング表の内容を整理すること、ステージング表を矛盾状態にすることを指定します。*table-name* リストに含まれている表がステージング表でなければ、エラーが戻されます (SQLSTATE 428FH)。

*table-unchecked-options**integrity-options*

表の SET INTEGRITY ペンディング状態を解除するときバイパスする、必要な保全性処理のタイプを定義するために使用します。

**ALL**

すべての必要な保全性処理を実行しないで、表の SET INTEGRITY ペンディング状態をすぐに解除します。

**FOREIGN KEY**

必要な外部キー制約検査を実行しないで、表の SET INTEGRITY ペンディング状態を解除します。

**CHECK**

必要なチェック制約検査を実行しないで、表の SET INTEGRITY ペンディング状態を解除します。

**MATERIALIZED QUERY**

必要なマテリアライズ照会表のリフレッシュを実行しないで、表の SET INTEGRITY ペンディング状態を解除します。

**GENERATED COLUMN**

必要な生成列制約検査を実行しないで、表の SET INTEGRITY ペンディング状態を解除します。

**STAGING**

必要なステージング表へのデータ伝搬を実行しないで、表の SET INTEGRITY ペンディング状態を解除します。

特定タイプの保全性処理がバイパス対象として設定された後に、表に対する他の保全性処理が必要なければ、表の SET INTEGRITY ペンディング状態はすぐに解除されます。

**FULL ACCESS**

SET INTEGRITY ステートメントの実行後に表が完全にアクセス可能になることを指定します。

呼び出しリストにある基礎表が増分的に処理され、従属即時マテリアライズ照会表または従属即時ステージング表を持つ場合、その基礎表は、SET INTEGRITY ステートメントの実行後に、必要に応じてデータ移動なし状態に設定されます。増分的にリフレッシュ可能なすべての従属の即時マテリアライズ照会表とステージング表の SET INTEGRITY ペンディング状態が解除されると、基礎表は、自動的にデータ移動なし状態からフル・アクセス状態になります。IMMEDIATE UNCHECKED オプションと一緒に FULL ACCESS オプションを指定すると、基礎表は、データ移動なし状態をバイパスして、直接にフル・アクセス状態になります。リフレッシュされたことのない従属即時マテリアライズ照会表は、後続の REFRESH TABLE ステートメントですべてが再計算される可能性があり、伝搬元の表の追加部分を持たない従属即時ステージング表は、不整合状態のフラグが設定される可能性があります。

呼び出しリストにある基礎表が、完全処理を必要とするか、従属即時マテリアライズ照会表または従属即時ステージング表を持たない場合、その基礎表は、FULL ACCESS オプションが指定されているかどうかに関係なく、SET INTEGRITY ステートメントの実行後に直接にフル・アクセス状態に設定されます。

IMMEDIATE UNCHECKED オプションと一緒に FULL ACCESS オプションを指定した場合に、ステートメントが表の SET INTEGRITY ペンディング状態を解除しなければ、エラーが戻されます (SQLSTATE 428FH)。

**IMMEDIATE UNCHECKED**

以下のいずれかを指定します。

- 必要な保全性処理をいずれも実行しないで、表の SET INTEGRITY ペンディング状態をすぐに解除すること。
- IMMEDIATE CHECKED オプションを使用した後続の SET INTEGRITY ステートメントで表の SET INTEGRITY ペンディング状態を解除するときに、表に必要な保全性処理のうち、1 つ以上のタイプの処理をバイパスすること。

このオプションを使用する前に、このオプションがデータ保全性に対して持つ意味合いをよく検討してください。下の『注』のセクションを参照してください。

**注**

- SET INTEGRITY に関連した制限状態のいずれかが表に及ぼす影響:
  - 読み取りアクセス状態またはアクセスなし状態の表については、INSERT、UPDATE、DELETE を実行できません。さらに、そのような状態の表にその種の変更を加える必要のあるステートメントはリジェクトされます。たとえば、アクセスなし状態にある従属表にカスケードする親表の行の削除は実行できません。
  - アクセスなし状態の表については、SELECT を実行できません。さらに、アクセスなし状態の表への読み取りアクセスが必要なステートメントはリジェクトされます。

- 表に新しく追加される制約は、通常、ただちに適用されます。ただし、表が SET INTEGRITY ペンディング状態の場合は、表の SET INTEGRITY ペンディング状態が解除されるまで、新しい制約の検査は据え置かれます。表が SET INTEGRITY ペンディング状態にある場合に、新しい制約を追加すると、データの妥当性がリスクにさらされるので、表は SET INTEGRITY ペンディング・アクセスなし状態になります。
- CREATE INDEX ステートメントでは、読み取りアクセス状態またはアクセスなし状態にある表を参照できません。同様に、主キー制約またはユニーク制約を追加する ALTER TABLE ステートメントでは、読み取りアクセス状態またはアクセスなし状態にある表を参照できません。
- 読み取りアクセス状態またはアクセスなし状態の表については、IMPORT ユーティリティを実行できません。
- EXPORT ユーティリティは、アクセスなし状態の表については実行できませんが、読み取りアクセス状態の表については実行できます。表が読み取りアクセス状態の場合、EXPORT ユーティリティは、追加部分以外のデータだけをエクスポートします。
- 読み取りアクセス状態、アクセスなし状態、データ移動なし状態の表については、表の中でのデータ移動を伴う可能性がある操作 (REORG、REDISTRIBUTE、分散キーの更新、マルチディメンション・クラスタリング・キーの更新、レンジ・クラスタリング・キーの更新、表パーティション・キーの更新 など) を実行できません。
- LOAD、BACKUP、RESTORE、UPDATE STATISTICS、RUNSTATS、REORGCHK、LIST HISTORY、ROLLFORWARD の各ユーティリティは、フル・アクセス状態、読み取りアクセス状態、アクセスなし状態、データ移動なし状態の表に対して実行できます。
- ALTER TABLE、COMMENT、DROP TABLE、CREATE ALIAS、CREATE TRIGGER、CREATE VIEW、GRANT、REVOKE、SET INTEGRITY の各ステートメントでは、フル・アクセス状態、読み取りアクセス状態、アクセスなし状態、データ移動なし状態の表を参照できます。ただし、結果的に表がアクセスなし状態にされる場合もあります。
- アクセスなし状態の表に従属しているパッケージ、ビュー、およびその他のオブジェクトは、実行時にその表がアクセスされると、エラーを戻します。読み取りアクセス状態の表に従属しているパッケージは、実行時にその表に対して挿入、更新、削除の操作が試行されると、エラーを戻します。

SET INTEGRITY ステートメントによる違反行の除去は、削除イベントではありません。したがって、SET INTEGRITY ステートメントではトリガーは活動化されません。同様に、FORCE GENERATED オプションを使用して生成された列を更新しても、トリガーは活動化されません。

- 状態が許すときには、増分処理が使用されます。増分処理はより効率的です。INCREMENTAL オプションは多くの場合必要ありません。しかし、保全性検査が確実に増分的に行われることを保証するため、このオプションが必要になります。システムが、データ保全性を確保するために完全処理が必要だと判断すると、エラーが戻されます (SQLSTATE 55019)。
- IMMEDIATE UNCHECKED 節の使用に関する警告 :

- この節は、ユーティリティー・プログラムで使用することを意図しているの  
で、アプリケーション・プログラムによる使用はお勧めしません。定義されて  
いる保全性指定を満たさない表にデータが存在する場合に、IMMEDIATE  
UNCHECKED オプションを使用すると、不正確な照会結果が戻されることが  
あります。

必要な保全性処理を実行しないで表の SET INTEGRITY ペンディング状態を  
解除したという事実は、カタログに記録されます (SYSCAT.TABLES ビューの  
CONST\_CHECKED 列の関連バイトが 'U' に設定されます)。これは、特定の  
制約に関するデータ保全の責任はユーザーにあることを示しています。この値  
は、以下のいずれかの条件が満たされるまで変更されません。

- OFF オプションを指定した SET INTEGRITY ステートメントで表を参照す  
ることによって、表を SET INTEGRITY ペンディング状態に戻した場合。  
その時点で、CONST\_CHECKED 列にある 'U' 値が 'W' 値に変更されま  
す。これは、データ保全性の責任が以前はユーザーにあったと見なされてい  
たのに対し、現在はシステムがデータを検査する必要があることを示してい  
ます。
- 検査されていないすべての表の制約をドロップした場合。

'W' 状態は 'N' 状態と違って、保全性が以前はシステムではなくユーザーに  
よって検査されていたことを記録しています。ユーザーが NOT  
INCREMENTAL オプションを指定した SET INTEGRITY ... IMMEDIATE  
CHECKED ステートメントを発行すると、システムは、表全体のデータ保全性  
を再検査 (または、マテリアライズ照会表で完全リフレッシュを実行) してか  
ら、'W' 状態を 'Y' 状態に変更します。IMMEDIATE UNCHECKED が指定  
されるか、NOT INCREMENTAL が指定されない場合、'W' 状態は変更され  
て 'U' 状態に戻され、一部のデータがまだシステムで検査されていないことを  
記録されます。後者の場合 (NOT INCREMENTAL が指定されない場合) は、  
警告が戻されます (SQLSTATE 01636)。

基礎表の保全性が IMMEDIATE UNCHECKED 節を使用して検査された場合、  
基礎表の CONST\_CHECKED 列にある 'U' の値は、以下の表の対応する  
CONST\_CHECKED 列に伝搬されます。

- 従属即時マテリアライズ照会表
- 従属据え置きマテリアライズ照会表
- 従属ステージング表

従属即時マテリアライズ照会表の場合、この伝搬は、基礎表の SET  
INTEGRITY ペンディング状態が解除されるとき、およびマテリアライズ照会  
表がリフレッシュされるときに必ず行われます。従属据え置きマテリアライズ  
照会表の場合、この伝搬は、マテリアライズ照会表がリフレッシュされるとき  
に必ず行われます。従属ステージング表の場合、この伝搬は、基礎表の SET  
INTEGRITY ペンディング状態が解除されるときに必ず行われます。従属のマ  
テリアライズ照会表とステージング表の CONST\_CHECKED 列に示される、  
これらの伝搬された 'U' の値は、これらのマテリアライズ照会表とステー  
ジング表が、IMMEDIATE UNCHECKED オプションによって必要な保全性処理が  
バイパスされた基礎表に従属することを記録しています。

マテリアライズ照会表の場合、基礎表によって伝搬された `CONST_CHECKED` 列の 'U' の値は、マテリアライズ照会表が完全にリフレッシュされ、すべての基礎表の対応する `CONST_CHECKED` 列に 'U' の値がなくなるまで、そのまま変わりません。リフレッシュが行われたら、マテリアライズ照会表の `CONST_CHECKED` 列にある 'U' の値は、'Y' に変更されます。

ステージング表の場合、基礎表によって伝搬された `CONST_CHECKED` 列の 'U' の値は、ステージング表の対応する据え置きマテリアライズ照会表がリフレッシュされるまで、そのまま変わりません。リフレッシュが行われたら、ステージング表の `CONST_CHECKED` 列にある 'U' の値は、'Y' に変更されます。

- 子表とその親表が `IMMEDIATE CHECKED` オプションを指定した同じ `SET INTEGRITY` ステートメントで検査される場合に、親表で制約を完全に検査する必要がある場合は、子表の外部キー制約の `CONST_CHECKED` 列に 'U' の値があるかどうかに関係なく、子表では外部キー制約が検査されます。
- `LOAD INSERT` または `ALTER TABLE ATTACH` を使用してデータを追加した後、`IMMEDIATE CHECKED` オプションを指定した `SET INTEGRITY` ステートメントによって表の制約違反を検査します。表に対する増分処理が可能かどうかは、システムが判断します。可能な場合には、追加部分だけが保全性違反を検査されます。不可能な場合には、システムは、表全体の保全性違反を検査します。
- 次のステートメントについて考慮します。

#### **SET INTEGRITY FOR T IMMEDIATE CHECKED**

システムが完全なリフレッシュを必要とする状況、または表全体の保全性 (`INCREMENTAL` オプションは指定できない) を検査する状況は、以下のとおりです。

- T が `SET INTEGRITY` ペンディング状態になっている間に、T そのものに新しい制約が追加された場合。
- T、その親、またはその基礎表に対する `LOAD REPLACE` 操作が生じた場合。
- T、その親、またはその基礎表に対する最後の保全性検査の後に、`NOT LOGGED INITIALLY WITH EMPTY TABLE` オプションが活動化された場合。
- 完全処理のカスケード効果により、T の親 (T がマテリアライズ照会表かステージング表である場合には、基礎表) について、増分的ではない方法で保全性が検査された場合。
- 表またはその親 (またはマテリアライズ照会表またはステージング表の基礎表) を含む表スペースが、ある時点までロールフォワードされ、表およびその親 (表がマテリアライズ照会表またはステージング表の場合は基礎表) が別の表スペースに存在する場合。
- T がマテリアライズ照会表で、最後のリフレッシュ後に、T に対する直接の `LOAD REPLACE` または `LOAD INSERT` 操作が行われる場合。
- 上記の完全処理の条件が満たされない場合、システムは、追加部分の保全性だけを検査しようとするか、ユーザーがステートメント `SET INTEGRITY FOR T IMMEDIATE CHECKED` に `NOT INCREMENTAL` オプションを指定していなければ、増分リフレッシュを実行します (マテリアライズ照会表の場合)。

## SET INTEGRITY

- 保全性処理の過程でエラーが発生すると、(元の表からの削除や例外表への挿入を含め) すべての処理結果がロールバックされます。
- **FORCE GENERATED** オプションを指定して発行された **SET INTEGRITY** ステートメントが、ログ・スペースの不足のために失敗する場合、使用できるアクティブなログ・スペースを増やし、**SET INTEGRITY** ステートメントを再発行します。別の方法としては、**GENERATED COLUMN** オプションと **IMMEDIATE UNCHECKED** オプションを指定した **SET INTEGRITY** ステートメントによって、表の生成列の検査を回避します。その後、**IMMEDIATE CHECKED** オプションを指定し **FORCE GENERATED** オプションを指定しない **SET INTEGRITY** ステートメントを実行して、表に他の保全性違反 (該当する場合) があるかどうかを検査し、表の **SET INTEGRITY** ペンディング状態を解除します。表の **SET INTEGRITY** ペンディング状態が解除されたら、**UPDATE** ステートメントのキーワード **DEFAULT** に生成列を割り当てることによって、生成列をそのデフォルト値 (生成値) に更新できます。このことは、範囲に基づいて複数の検索済み更新ステートメントを使用する方法 (それぞれの後にコミットされる) と、断続的なコミットを使用したカーソル・ベースによる方法のいずれかを使用することで、実現されます。カーソル・ベースによる方法を使用した断続的なコミットの後で、ロックを保存する場合には、『**WITH HOLD**』カーソルを使用する必要があります。
- **SET INTEGRITY** ステートメントまたは **LOAD** コマンドの **CASCADE DEFERRED** オプション、または **ATTACH** 節を指定した **ALTER TABLE** ステートメントによって **SET INTEGRITY** ペンディング状態にされ、**SET INTEGRITY** ステートメントの **IMMEDIATE CHECKED** オプションによって保全性違反を検査される表については、その下層外部キー表、下層即時マテリアライズ照会表、下層即時ステージング表が必要に応じて **SET INTEGRITY** ペンディング状態にされます。
  - 表全体の保全性違反が検査される場合は、その下層外部キー表、下層即時マテリアライズ照会表、下層即時ステージング表が **SET INTEGRITY** ペンディング状態にされます。
  - 表の保全性違反が増分的に検査される場合は、その下層即時マテリアライズ照会表とステージング表が **SET INTEGRITY** ペンディング状態にされ、その下層外部キー表は元の状態のままになります。
  - 表を検査する必要がまったくない場合、その下層即時マテリアライズ照会表、下層ステージング表、および下層外部キー表は、元の状態のままにされます。
- **SET INTEGRITY** ステートメントまたは **LOAD** コマンドの **CASCADE DEFERRED** オプションによって **SET INTEGRITY** ペンディング状態にされ、**SET INTEGRITY** ステートメントの **IMMEDIATE UNCHECKED** オプションによって **SET INTEGRITY** ペンディング状態を解除される表については、その下層外部キー表、下層即時マテリアライズ照会表、下層即時ステージング表が必要に応じて **SET INTEGRITY** ペンディング状態にされます。
  - 表が **REPLACE** モードでロードされた場合は、その下層外部キー表、下層即時マテリアライズ照会表、下層即時ステージング表が **SET INTEGRITY** ペンディング状態にされます。
  - 表が **INSERT** モードでロードされた場合は、その下層即時マテリアライズ照会表とステージング表が **SET INTEGRITY** ペンディング状態にされ、その下層外部キー表は元の状態のままになります。

- 表がロードされていない場合、その下層即時マテリアライズ照会表、下層ステージング表、および下層外部キー表は、元の状態のままにされます。
- 通常、SET INTEGRITY ステートメントの実行には長い時間がかかります。したがって、ロック・タイムアウトが原因でステートメント全体がロールバックされるリスクを軽減するために、まず WAIT オプションを指定した SET CURRENT LOCK TIMEOUT ステートメントを実行してから SET INTEGRITY ステートメントを実行し、トランザクションのコミット後にその特殊レジスターを元の値にリセットできます。ただし、CURRENT LOCK TIMEOUT 特殊レジスターは、特定セットのロック・タイプだけに影響を与えます。
- ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY オプションを使用する場合は、REFRESH DEFERRED マテリアライズ照会表の保守の順序が正しいことを確認してください。たとえば、2つのマテリアライズ照会表 MQT1 と MQT2 があり、それぞれのマテリアライズ照会が同じ基礎表を共有するとします。この場合、MQT2 のマテリアライズ照会は、基礎表ではなく MQT1 を使用して計算できます。この 2 つのマテリアライズ照会表を保守するために別々のステートメントを使用し、MQT2 を最初に保守する場合、システムは、MQT2 の保守のために、まだ保守されていない MQT1 の内容を使用することを選択する可能性があります。その場合、MQT1 には現在のデータが入りますが、両方の保守をほとんど同時に実行したとしても、MQT2 には失効したデータが入る可能性があります。1 つではなく 2 つの SET INTEGRITY ステートメントを使用する場合は、MQT1 を最初に保守するのが正しい順序になります。
- SET INTEGRITY ステートメントを使用して、ロードまたは追加が行われた基本表の保全性処理を実行する場合は、従属の REFRESH IMMEDIATE マテリアライズ照会表と PROPAGATE IMMEDIATE ステージング表も同じ SET INTEGRITY ステートメントで処理することによって、SET INTEGRITY の処理の最終段階で、それらの従属表を SET INTEGRITY ペンディング・アクセスなし状態にする動作を回避することをお勧めします。ただし、従属の REFRESH IMMEDIATE マテリアライズ照会表と PROPAGATE IMMEDIATE ステージング表を多数抱えている基本表の場合は、メモリー制約のために、基本表と同じステートメントですべての従属表を処理することが不可能な場合もあります。
- FORCE GENERATED オプションまたは GENERATE IDENTITY オプションを指定した場合に、生成される列がユニーク索引の一部になっていれば、SET INTEGRITY ステートメントはエラーを戻します (SQLSTATE 23505)。また、ユニーク索引の中で重複キーを検出すると、処理をロールバックします。このエラーは、処理対象表に例外表がある場合でも戻されます。

このシナリオは、以下の状況で発生する可能性があります。

- 表に対して LOAD コマンドを実行し、その後に SET INTEGRITY ステートメントを実行する場合に、ロード操作の実行時にファイル・タイプ修飾子として GENERATEDOVERRIDE または IDENTITYOVERRIDE を指定していた状況。このシナリオを回避するために、ファイル・タイプ修飾子として GENERATEDOVERRIDE の代わりに GENERATEDIGNORE または GENERATEDMISSING、IDENTITYOVERRIDE の代わりに IDENTITYIGNORE または IDENTITYMISSING を使用することをお勧めします。これらの推奨修飾子を使用すれば、SET INTEGRITY ステートメントの実行時に、式生成列または ID 列の処理が必要なくなります。

- 式生成列の式を変更する ALTER TABLE ステートメントの後に SET INTEGRITY ステートメントを実行する状況。

このようなシナリオが発生した後に、表の SET INTEGRITY ペンディング状態を解除するには、以下のようにします。

- 列値を再生成するために FORCE GENERATED オプションまたは GENERATE IDENTITY オプションを使用しないでください。その代わりに、IMMEDIATE CHECKED オプションと FOR EXCEPTION オプションを併用して、生成列の式に違反している行を例外表に移動します。その後、それらの行を例外表から対象表に挿入し直せば、正しい式が生成され、ユニーク・キーの検査が実行されます。この場合、再処理の必要があるのは、生成列の式に違反していた行だけなので、表全体を再処理する必要はありません。
- 処理対象表にパーティションがアタッチされている場合は、まずそれらのパーティションをデタッチしてから、以下の黒丸の箇条書きリストに挙げられている操作を実行します。その後、それらのパーティションを再アタッチしてから、SET INTEGRITY ステートメントによって、アタッチしたパーティションの健全性処理を別途実行します。
- 保護対象表に SET INTEGRITY ステートメントを例外表と共に指定する場合は、表に関する以下のすべての基準を満たす必要があります。そうでなければ、エラーが戻されます (SQLSTATE 428A5)。
  - 両方の表が同じセキュリティ・ポリシーによって保護されている必要があります。
  - 保護対象表の列のデータ・タイプが DB2SECURITYLABEL の場合は、例外表の対応する列のデータ・タイプも DB2SECURITYLABEL でなければなりません。
  - 保護対象表の列がセキュリティ・ラベルによって保護されている場合、例外表の対応する列も同じセキュリティ・ラベルで保護されている必要があります。
- 互換性
  - 以前のバージョンの DB2 との互換性:
    - SET INTEGRITY の代わりに SET CONSTRAINTS を指定できます。
    - MATERIALIZED QUERY の代わりに SUMMARY を指定できます。

### 例

例 1: 以下は、表の SET INTEGRITY ペンディング状態と、SET INTEGRITY に関連したアクセス制限状態についての情報を提供する照会の例です。SUBSTR を使用して、SYSCAT.TABLES の CONST\_CHECKED 列の個々のバイトを抽出しています。第 1 バイトは外部キー制約、第 2 バイトはチェック制約、第 5 バイトはマテリアライズ照会表の健全性、第 6 バイトは生成列制約、第 7 バイトはステージング表の健全性、第 8 バイトはデータ・パーティション制約をそれぞれ表します。STATUS は SET INTEGRITY ペンディング状態を示し、ACCESS\_MODE は SET INTEGRITY に関連したアクセス制限状態を示します。

```
SELECT TABNAME, STATUS, ACCESS_MODE,
 SUBSTR(CONST_CHECKED,1,1) AS FK_CHECKED,
 SUBSTR(CONST_CHECKED,2,1) AS CC_CHECKED,
 SUBSTR(CONST_CHECKED,5,1) AS MQT_CHECKED,
```

```

SUBSTR(CONST_CHECKED,6,1) AS GC_CHECKED,
SUBSTR(CONST_CHECKED,7,1) AS STG_CHECKED,
SUBSTR(CONST_CHECKED,8,1) AS DP_CHECKED
FROM SYSCAT.TABLES

```

例 2: PARENT 表を SET INTEGRITY ペンディング・アクセスなし状態にして、すぐに SET INTEGRITY ペンディング状態を下層表にカスケードします。

```

SET INTEGRITY FOR PARENT OFF
NO ACCESS CASCADE IMMEDIATE

```

例 3: PARENT 表を SET INTEGRITY ペンディング読み取りアクセス状態にしますが、すぐには SET INTEGRITY ペンディング状態を下層表にカスケードしません。

```

SET INTEGRITY FOR PARENT OFF
READ ACCESS CASCADE DEFERRED

```

例 4: FACT\_TABLE という名前の表の保全性を検査します。保全性違反が検出されなければ、表の SET INTEGRITY ペンディング状態は解除されます。保全性違反が検出されれば、ステートメント全体がロールバックされ、表は SET INTEGRITY ペンディング状態のままになります。

```

SET INTEGRITY FOR FACT_TABLE IMMEDIATE CHECKED

```

例 5: SALES 表と PRODUCTS 表の保全性を検査し、保全性に違反している行を SALES\_EXCEPTIONS および PRODUCTS\_EXCEPTIONS という名前の例外表にそれぞれ移動します。保全性違反があってもなくても、SALES 表と PRODUCTS 表の両方の SET INTEGRITY ペンディング状態が解除されます。

```

SET INTEGRITY FOR SALES, PRODUCTS IMMEDIATE CHECKED
FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS,
IN PRODUCTS USE PRODUCTS_EXCEPTIONS

```

例 6: MANAGER 表の FOREIGN KEY 制約検査を使用可能にし、EMPLOYEE 表の CHECK 制約検査を IMMEDIATE UNCHECKED オプションによってバイパスします。

```

SET INTEGRITY FOR MANAGER FOREIGN KEY,
EMPLOYEE CHECK IMMEDIATE UNCHECKED

```

例 7: 2 つの ALTER TABLE ステートメントを使用して、チェック制約と外部キーを EMP\_ACT 表に追加します。OFF オプションを指定した SET INTEGRITY ステートメントによって表を SET INTEGRITY ペンディング状態にすると、2 つの ALTER TABLE ステートメントの実行時に制約検査がすぐに行われることはなくなります。IMMEDIATE CHECKED オプションを指定した 1 つの SET INTEGRITY ステートメントを使用して、追加した両方の制約を表の 1 回のパススルーによって検査します。

```

SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK
(EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY
(EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
FOR EXCEPTION IN EMP_ACT USE EMP_ACT_EXCEPTIONS

```

例 8: 生成済み列を正しい値で更新します。

## SET INTEGRITY

### SET INTEGRITY FOR SALES IMMEDIATE CHECKED FORCE GENERATED

例 9: REFRESH IMMEDIATE マテリアライズ照会表 (SALES\_SUMMARY) の基礎表 (SALES) に (LOAD INSERT を使用して) いくつかのソースからデータを追加します。SALES のデータ保全性を増分的に検査し、SALES\_SUMMARY を増分的にリフレッシュします。このシナリオで SALES の保全性検査と SALES\_SUMMARY のリフレッシュが増分的に行われるのは、システムが増分的な処理を選択するからです。SALES 表については、ALLOW READ ACCESS オプションを使用して、表のロード部分の保全性検査中にも既存データの並行読み取りを可能にします。

```
LOAD FROM 2000_DATA.DEL OF DEL
INSERT INTO SALES ALLOW READ ACCESS;
LOAD FROM 2001_DATA.DEL OF DEL
INSERT INTO SALES ALLOW READ ACCESS;
SET INTEGRITY FOR SALES ALLOW READ ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS;
REFRESH TABLE SALES_SUMMARY;
```

例 10: SALES という名前のデータ・パーティション表に新しいパーティションをアタッチします。SALES 表の追加データの制約違反を増分的に検査し、従属の SALES\_SUMMARY 表を増分的にリフレッシュします。両方の表で ALLOW WRITE ACCESS オプションを使用して、保全性検査中にも並行更新を可能にします。

```
ALTER TABLE SALES
ATTACH PARTITION STARTING (100) ENDING (200)
FROM SOURCE;
SET INTEGRITY FOR SALES ALLOW WRITE ACCESS, SALES_SUMMARY ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN SALES
USE SALES_EXCEPTIONS;
```

例 11: SALES という名前のデータ・パーティション表からパーティションをデータタッチします。従属の SALES\_SUMMARY 表を増分的にリフレッシュします。

```
ALTER TABLE SALES
DETACH PARTITION 2000_PART INTO ARCHIVE_TABLE;
SET INTEGRITY FOR SALES_SUMMARY
IMMEDIATE CHECKED;
```

例 12: 新しいユーザー管理マテリアライズ照会表の SET INTEGRITY ペンディング状態を解除します。

```
CREATE TABLE YEARLY_SALES
AS (SELECT YEAR, SUM(SALES) AS SALES
FROM FACT_TABLE GROUP BY YEAR)
DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED BY USER

SET INTEGRITY FOR YEARLY_SALES
ALL IMMEDIATE UNCHECKED
```

---

## SET PASSTHRU

SET PASSTHRU ステートメントは、データ・ソースのネイティブ SQL を、直接そのデータ・ソースに送信するセッションをオープンおよびクローズします。このステートメントは、トランザクションの制御下にはありません。

## 呼び出し

このステートメントは、対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

このステートメントの許可 ID が持つ特権には、以下の事柄を行う許可がなければなりません。

- データ・ソースにパススルーする
- データ・ソースでのセキュリティーの制限を満たす

## 構文

```

▶▶ SET PASSTHRU { server-name | RESET }

```

## 説明

*server-name*

パススルー・セッションをオープンするデータ・ソースを指定します。

*server-name* (サーバー名) は、カタログに記述されているデータ・ソースを指定していなければなりません。

## RESET

パススルー・セッションをクローズします。

## 注

- 以下の制約事項が Microsoft SQL Server、Sybase、および Oracle のデータ・ソースに適用されます。
  - パススルー・モードでは、ユーザー定義のトランザクションを Microsoft SQL Server や Sybase のデータ・ソースに対して使用できません。なぜなら、Microsoft SQL Server および Sybase では、ユーザー定義のトランザクションで指定できる SQL ステートメントが制約されるからです。パススルー・モードで処理される SQL ステートメントは DB2 によって構文解析されないのので、ユーザーが指定した SQL ステートメントがユーザー定義のトランザクションで使用できるかどうかを検出できません。
  - Microsoft SQL Server および Sybase データ・ソース上では、COMPUTE 節はサポートされていません。
  - Microsoft SQL Server、Oracle、および Sybase データ・ソース上では、DDL ステートメントはトランザクション・セマンティクスの対象外です。操作が完了した時点で Microsoft SQL Server、Oracle または Sybase によって自動的にコミットされます。ロールバックが行われても、DDL はロールバックされません。

## 例

例 1: データ・ソース BACKEND に対するパススルー・セッションを開始します。

```

strcpy (PASS_THRU, "SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;

```

## SET PASSTHRU

例 2: PREPARE ステートメントを使ってパススルー・セッションを開始します。

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL PREPARE STMT FROM :PASS_THRU;
EXEC SQL EXECUTE STMT;
```

例 3: パススルー・セッション終了します。

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

例 4: PREPARE および EXECUTE ステートメントを使って、パススルー・セッションを終了します。

```
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL PREPARE STMT FROM :PASS_THRU_RESET;
EXEC SQL EXECUTE STMT;
```

例 5: データ・ソースに移動するセッションをオープンし、このデータ・ソースにある表のクラスター索引を作成し、それからパススルー・セッションを終了します。

```
strcpy (PASS_THRU,"SET PASSTHRU BACKEND");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU;
EXEC SQL PREPARE STMT pass-through mode
FROM "CREATE UNIQUE
 CLUSTERED INDEX TABLE_INDEX
ON USER2.TABLE table is not an
WITH IGNORE DUP KEY"; alias
EXEC SQL EXECUTE STMT;
strcpy (PASS_THRU_RESET,"SET PASSTHRU RESET");
EXEC SQL EXECUTE IMMEDIATE :PASS_THRU_RESET;
```

---

## SET PATH

SET PATH ステートメントは、CURRENT PATH 特殊レジスターの値を変更します。このステートメントは、トランザクションの制御下にはありません。

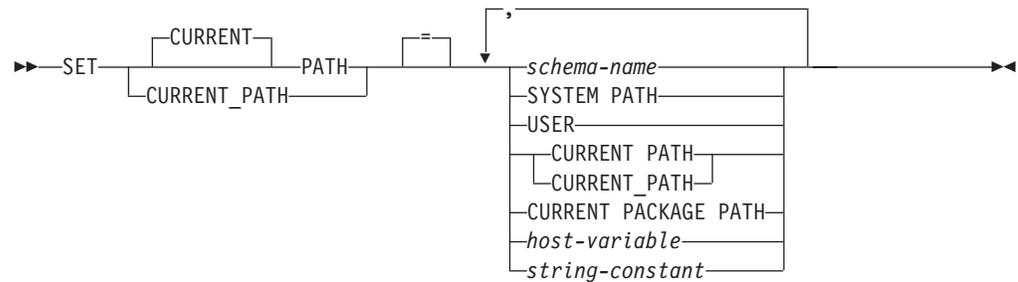
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

### 許可

必要ありません。

## 構文



## 説明

### *schema-name*

これは、1つの部分だけからなる名前で、アプリケーション・サーバーに存在するスキーマを指定します。そのスキーマの存否の検査は、パス設定時には行われません。たとえば *schema-name* (スキーマ名) のつづりが誤っていると、エラーを捉えることができず、以降の SQL 操作に影響を及ぼします。

### SYSTEM PATH

この値を指定すると、スキーマ名として "SYSIBM"、"SYSFUN"、"SYSPROC"、"SYSIBMADM" を指定したのと同じことになります。

### USER

USER 特殊レジスタの値。

### CURRENT PATH

このステートメントを実行する前の CURRENT PATH 特殊レジスタの値。

### CURRENT PACKAGE PATH

CURRENT PACKAGE PATH 特殊レジスタの値。

### *host-variable*

タイプ CHAR または VARCHAR の変数です。 *host-variable* の内容の長さは、128 バイトを超えてはなりません (SQLSTATE 42815)。 NULL に設定することはできません。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

*host-variable* の文字は左寄せされていなければなりません。 *host-variable* にスキーマ名を指定する場合は、英大文字への変換はなされないため、すべての文字を大/小文字の区別も含めて正確に指定する必要があります。

### *string-constant*

128 バイトの最大長を超えない文字ストリング定数。

## 規則

- SQL パスの中に 1 つのスキーマ名を複数回指定することはできません (SQLSTATE 42732)。
- 指定できるスキーマの数は、CURRENT PATH 特殊レジスタの合計長によって限定されます。特殊レジスタのストリングは、指定した各スキーマの名前から後続ブランクを除き、二重引用符で区切り、必要に応じてスキーマ名の中で使わ

## SET PATH

れている引用符を反復させ、スキーマ名をコンマで区切ったものになります。結果のストリングの長さが 2048 バイトを超えてはなりません (SQLSTATE 42907)。

### 注

- CURRENT PATH 特殊レジスターの初期値は、"SYSIBM"、"SYSFUN"、"SYSPROC"、"SYSIBMADM"、"X" です (X は USER 特殊レジスターの値)。
- SYSIBM スキーマを指定する必要はありません。それが SQL パスに含まれていない場合、暗黙のうちに最初のスキーマであると見なされます (この場合 CURRENT PATH 特殊レジスターには入れられません)。
- CURRENT PATH 特殊レジスターは、動的 SQL ステートメント内のユーザー定義データ・タイプ、プロシージャ、および関数を解決するために使用する SQL パスを指定します。静的 SQL ステートメント内のユーザー定義データ・タイプおよび関数の解決に使用する SQL パスは、FUNCPATH BIND オプションによって指定されます。
- 互換性
  - 以前のバージョンの DB2 との互換性:
    - CURRENT PATH の代わりに CURRENT FUNCTION PATH を指定できます。

### 例

例 1: 以下のステートメントは、CURRENT PATH 特殊レジスターを設定します。

```
SET PATH = FERMAT, "McDrw #8", SYSIBM
```

例 2: 以下の例では、CURRENT PATH 特殊レジスターの現行値を検索して CURPATH という名前のホスト変数に入れます。

```
EXEC SQL VALUES (CURRENT PATH) INTO :CURPATH;
```

例 1 での設定を使った場合、値は "FERMAT"、"McDrw #8"、"SYSIBM" になります。

---

## SET ROLE

SET ROLE ステートメントは、セッションの許可 ID が特定のロールのメンバーであることを確認します。許可 ID は、ロールが許可 ID に付与されるかまたはメンバーとして許可 ID を持つグループまたはロールに付与されると、そのロールのメンバーシップを取得します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

必要ありません。

## 構文

```
→ SET ROLE role-name →
```

## 説明

*role-name*

セッションの許可 ID がメンバーとして所属する、検査対象のロールを指定します。 *role-name* は、現行サーバーの既存のロールを識別するものでなければなりません (SQLSTATE 42704)。セッションの許可 ID が *role-name* のメンバーではない場合、エラーが戻されます (SQLSTATE 42501)。

## 注

- 許可 ID にすでに付与されているロールはすべて、許可検査で使用されます。SET ROLE ステートメントは、この許可検査でどのロールを使用するかということには影響を与えません。許可 ID がメンバーとして所属するロールの変更は、GRANT ROLE および REVOKE ROLE ステートメントを使用して行います。

## 例

例 1: ユーザー WALID にロール EDITOR が付与されており、ロール AUTHOR は付与されていません。WALID が EDITOR ロールのメンバーであることを確認します。

```
SET ROLE EDITOR
```

例 2: WALID が AUTHOR ロールのメンバーではないことを確認します。以下のステートメントはエラーを戻します (SQLSTATE 42501)。

```
SET ROLE AUTHOR
```

---

## SET SCHEMA

SET SCHEMA ステートメントは、CURRENT SCHEMA 特殊レジスターの値を変更します。このステートメントは、トランザクションの制御下にはありません。

DYNAMICRULES BIND オプションを使ってパッケージがバインドされている場合、このステートメントは、修飾されていないデータベース・オブジェクト参照に使用される修飾子に影響を与えません。

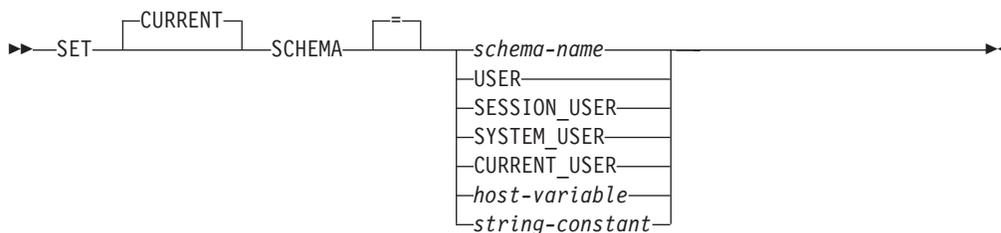
## 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また対話式に出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

必要ありません。

## 構文



## 説明

*schema-name*

これは、1 つの部分だけからなる名前で、アプリケーション・サーバーに存在するスキーマを指定します。名前の長さは、128 バイトを超えてはなりません (SQLSTATE 42815)。そのスキーマの存否の検査は、スキーマ設定時には行われません。 *schema-name* (スキーマ名) のつづりが誤っていると、エラーをキャッチすることができず、以降の SQL ステートメントの実行の仕方に影響を及ぼす可能性があります。

**USER**

USER 特殊レジスタの値。

**SESSION\_USER**

SESSION\_USER 特殊レジスタの値。

**SYSTEM\_USER**

SYSTEM\_USER 特殊レジスタの値。

**CURRENT\_USER**

CURRENT\_USER 特殊レジスタの値。

*host-variable*

タイプ CHAR または VARCHAR の変数です。 *host-variable* の内容の長さは、128 バイトを超えてはなりません (SQLSTATE 42815)。 NULL に設定することはできません。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 42815)。

*host-variable* の文字は左寄せされていなければなりません。 *host-variable* にスキーマ名を指定する場合は、英大文字への変換はなされないため、すべての文字を大/小文字の区別も含めて正確に指定する必要があります。

*string-constant*

128 バイトの最大長を超えない文字ストリング定数。

## 規則

- 指定した値が *schema-name* の規則に適合しない場合、エラー (SQLSTATE 3F000) が発生します。
- CURRENT SCHEMA 特殊レジスタの値は、すべての動的 SQL ステートメント (データベース・オブジェクトへの非修飾参照がある CREATE SCHEMA ステートメントを除く) でスキーマ名として使用されます。
- QUALIFIER BIND オプションは、静的 SQL ステートメントで非修飾データベース・オブジェクト名の修飾子として使用するスキーマ名を指定します。

**注**

- CURRENT SCHEMA 特殊レジスターの初期値は USER と同じです。
- CURRENT SCHEMA 特殊レジスターを設定しても、CURRENT PATH 特殊レジスターには影響しません。したがって、CURRENT SCHEMA は SQL パスおよび関数に含まれず、プロシージャおよびユーザー定義タイプの解決ではこれらのオブジェクトを見つけることができない場合があります。SQL パスに現行スキーマ値を含めるには、SET SCHEMA ステートメントを発行するときに、SET SCHEMA ステートメントからスキーマ名を含む SET PATH ステートメントも発行してください。
- CURRENT SQLID は CURRENT SCHEMA の同義語として受け入れられ、SET CURRENT SQLID ステートメントは SET CURRENT SCHEMA ステートメントと同じ影響を及ぼします。ステートメント許可変更など、他の影響はありません。

**例**

例 1: 以下のステートメントは、CURRENT SCHEMA 特殊レジスターを設定します。

```
SET SCHEMA RICK
```

例 2: 以下の例では、CURRENT SCHEMA 特殊レジスターの現行値を検索して CURSCHEMA という名前のホスト変数に入れます。

```
EXEC SQL VALUES (CURRENT SCHEMA) INTO :CURSCHEMA;
```

値は、前の例で設定された RICK になります。

---

**SET SERVER OPTION**

SET SERVER OPTION ステートメントは、ユーザーまたはアプリケーションがフェデレーテッド・データベースに接続中に有効となる、サーバー・オプションの設定値を指定します。接続が終了すると、このサーバー・オプションの以前の設定値が復元されます。このステートメントは、トランザクションの制御下ではありません。

**呼び出し**

このステートメントは、対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

**許可**

必要ありません。

**構文**

```
►►—SET SERVER OPTION—server-option-name—TO—string-constant—►►
►—FOR—SERVER—server-name—►►
```

## SET SERVER OPTION

### 説明

*server-option-name*

設定するサーバー・オプションを指名します。

**TO** *string-constant*

*server-option-name* の設定を、文字ストリング定数として指定します。

**SERVER** *server-name*

*server-option-name* が適用されるデータ・ソースを指定します。これは、カタログに記述されているサーバーでなければなりません。

### 注

- サーバー・オプション名は、大文字または小文字で入力することができます。
- ユーザーまたはアプリケーションがフェデレーテッド・データベースに接続する際、1 つまたは複数の SET SERVER OPTION ステートメントをサブミットすることができます。このステートメント (複数可) は、接続が確立した後、最初に処理される作業単位の初めに指定する必要があります。
- SYSCAT.SERVEROPTIONS は SET SERVER OPTION ステートメントに基づいては更新されません。この変更内容は現行接続だけに影響を与えるからです。
- 静的 SQL では、SET SERVER OPTION ステートメントの使用によって影響を受けるのは静的 SQL ステートメントの実行だけです。SET SERVER OPTION ステートメントを使用しても、オプティマイザーによって生成されたプランは影響を受けません。

### 例

例 1: DJDB というフェデレーテッド・データベースに、ORASERV という Oracle データ・ソースを定義します。ORASERV は、プランのヒントを使用できないように構成されます。しかし、DBA は新しいアプリケーションを試験的に実行するため、プランのヒントを使用できるようにすることを希望しています。実行を終了すると、プランのヒントは再度使用不可になります。

```
CONNECT TO DJDB;
strcpy(stmt,"set server option plan_hints to 'Y' for server oraserv");
EXEC SQL EXECUTE IMMEDIATE :stmt;
strcpy(stmt,"select c1 from ora_t1 where c1 > 100"); /*Generate plan hints*/
EXEC SQL PREPARE s1 FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :hv;
```

例 2: すべての Oracle 8 データ・ソースで、サーバー・オプション PASSWORD を 'Y' (データ・ソースでパスワードを妥当性検査する) に設定しました。しかし、特定の Oracle 8 データ・ソース (フェデレーテッド・データベース DJDB に ORA8A—passwords と定義されているデータ・ソース) にアクセスするために、アプリケーションがフェデレーテッド・データベースに接続するような特定のセッションの場合、パスワードを妥当性検査する必要はありません。

```
CONNECT TO DJDB;
strcpy(stmt,"set server option password to 'N' for server ora8a");
EXEC SQL PREPARE STMT_NAME FROM :stmt;
EXEC SQL EXECUTE STMT_NAME FROM :stmt;
strcpy(stmt,"select max(c1) from ora8a_t1");
```



## SET SESSION AUTHORIZATION

128 バイトを超えてはなりません (SQLSTATE 28000)。 NULL に設定することはできません。 *host-variable* が標識変数を伴っている場合、その標識変数の値は NULL 値以外でなければなりません (SQLSTATE 28000)。

*host-variable* の文字は左寄せされていなければなりません。ホスト変数で *authorization-name* を指定する場合は、大文字への変換が行われなため、すべての文字を大文字で指定する必要があります。

### *string-constant*

128 バイトの最大長を超えない文字ストリング定数。

## ALLOW ADMINISTRATION

同じ作業単位内でこのステートメントよりも前に SQL スキーマ・ステートメントを指定できることを示します。

## 規則

- SESSION\_USER 特殊レジスターで指定される値は、タイプ USER の許可 ID の規則に従わなければなりません (SQLSTATE 42602)。
- OWNER BIND オプションは、静的 SQL ステートメントに使用される許可 ID を指定します。
- このステートメントは、開かれている WITH HOLD カーソルがない、新しい作業単位の最初のステートメント (SET 特殊レジスター・ステートメントを除く) として発行します (SQLSTATE 25001)。この制限には、SET 特殊レジスター・ステートメント以外のステートメントに対するすべての PREPARE 要求も含まれます。
- SESSION\_USER 特殊レジスターの値は、 DYNAMICRULES(RUN) BIND オプションでバインドされたパッケージのすべての動的 SQL ステートメントに対する許可 ID として使用されます (これには、パッケージがルーチンで使用されていない場合の INVOKERUN および DEFINERUN も含まれます)。パッケージが DYNAMICRULES オプションに基づいて所有者、起動側、または定義元の許可に使用される場合、このステートメントは、パッケージ内から発行される動的 SQL ステートメントに対して効果がありません。

## 注

- SET SESSION AUTHORIZATION ステートメントを使用して、セッション許可 ID を変更することができます。セッション許可 ID とは、接続の現在のユーザーを表すもので、 DYNAMICRULES 実行パッケージ内の動的 SQL に比較してすべての許可検査を行うときに DB2 が考慮に入れる許可 ID です。 SESSION\_USER 特殊レジスターは、このセッション許可 ID の現行値を表示するのに使用できます。
- 新しい接続における SESSION\_USER 特殊レジスターの初期値は、 SYSTEM\_USER 特殊レジスターの値と同じです。
- このステートメントで指定されているセッション許可 ID のグループ情報は、ステートメントの実行時に獲得されます。
- SESSION\_USER 特殊レジスターの設定は、 CURRENT SCHEMA または CURRENT PATH 特殊レジスターには影響を与えません。
- SESSION\_USER 特殊レジスターの設定中にエラーが発生した場合、レジスターは直前の値に復帰します。

- このステートメントを使用して、複数の異なるユーザーが同一の接続を再利用できるようにしないでください。元の接続所有者がもっていた `SESSION_USER` 特殊レジスター値を変更する能力が、各ユーザーに継承されてしまうからです。このステートメントは、特権のチェックを `SYSTEM_USER` の値に依存しており、最初の接続許可 ID は `SET SESSION AUTHORIZATION` ステートメントによって変更されません。さらに、このステートメントでは、接続の再利用に影響を与える以下の振る舞いに対処できません。
  - `CONNECT` 特権には、新しい許可 ID のチェックが行われません。
  - 更新可能特殊レジスターの内容がリセットされません。特に、`ENCRYPTION PASSWORD` 特殊レジスターの内容が変更されず、新しい許可 ID で暗号化または暗号化解除できてしまいます。
  - 宣言済みグローバル一時表の内容が影響を受けず、新しい許可 ID からアクセスできてしまいます。
  - リモート・サーバーへの既存のリンクがリセットされません。
- `ALLOW ADMINISTRATION` 節が指定される場合、以下のタイプのステートメントまたは操作を `SET SESSION AUTHORIZATION` ステートメントよりも先行させることができます。
  - データ定義言語 (DDL)。これにはセーブポイントの定義やグローバル一時表の宣言が含まれますが、`SET INTEGRITY` は含まれません。
  - `GRANT` および `REVOKE` ステートメント
  - `LOCK TABLE` ステートメント
  - `COMMIT` および `ROLLBACK` ステートメント
  - 特殊レジスターの `SET`
  - グローバル変数の `SET`

## 例

例 1: 以下のステートメントは、`SESSION_USER` 特殊レジスターを設定します。

```
SET SESSION_USER = RAJIV
```

例 2: セッション許可 ID (`SESSION_USER` 特殊レジスター) を、ステートメント発行元の接続を確立する際に使用されたシステム許可 ID の値にします。

```
SET SESSION AUTHORIZATION SYSTEM_USER
```

---

## SET 変数

`SET` 変数ステートメントは、変数に値を割り当てます。このステートメントは、トランザクションの制御下にありません。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

遷移変数を参照するには、トリガー作成者の許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- 割り当ての左辺で参照されているすべての列に対する UPDATE 特権、および右辺で参照されているすべての列に対する SELECT 特権。
- 表 (トリガーのサブジェクト表) での CONTROL 特権
- SYSADM または DBADM 権限

割り当てステートメントの右辺でグローバル変数が参照される場合、ステートメントの許可 ID は以下の特権を保持する必要があります。

- グローバル変数に対する READ 特権

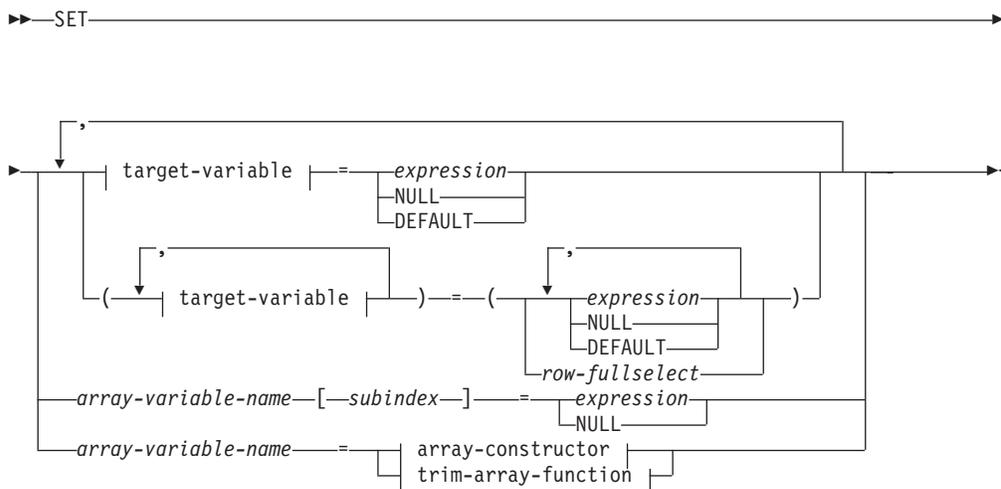
割り当てステートメントの左辺でグローバル変数に値が割り当てられる場合、ステートメントの許可 ID は以下の特権を保持する必要があります。

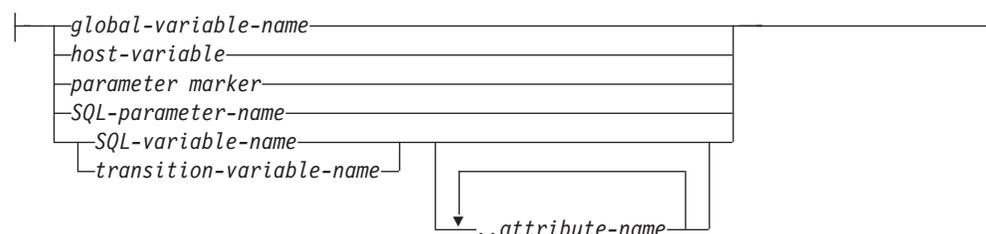
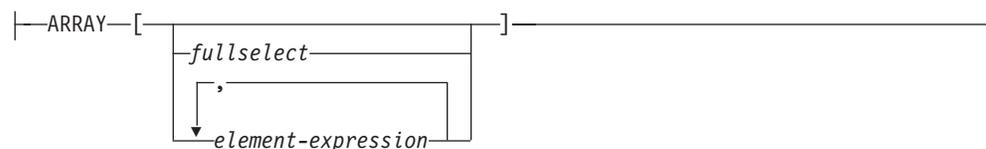
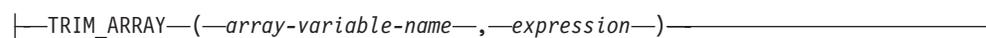
- グローバル変数に対する WRITE 特権

加えて、割り当ての右辺で *row-fullselect* を使ってこのステートメントを実行するには、参照されるそれぞれの表またはビューに対して、ステートメントの許可 ID が以下の少なくとも 1 つの特権を持つ必要があります。

- SELECT 特権
- CONTROL 特権
- SYSADM または DBADM 権限

## 構文



**target-variable:****array-constructor:****trim-array-function:****説明****target-variable**

割り当てのターゲット変数を識別します。同じ変数を表す *target-variable* を複数指定することはできません (SQLSTATE 42701)。

*global-variable-name*

割り当てのターゲットとなるグローバル変数を指定します。

*global-variable-name* は、現在のサーバーに存在するグローバル変数を識別するものでなければなりません (SQLSTATE 42704)。

*host-variable*

割り当てのターゲットとなるホスト変数を指定します。

*parameter-marker*

割り当てのターゲットとなるパラメーター・マーカを指定します。

*SQL-parameter-name*

割り当てのターゲットとなるパラメーターを識別します。そのパラメーターは、CREATE PROCEDURE ステートメントの *parameter-declaration* で指定しなければならず、さらに OUT または INOUT パラメーターとして定義しなければなりません。

*SQL-variable-name*

割り当てターゲットである SQL 変数を識別します。SQL 変数は、使用する前に宣言しておかなければなりません。

*transition-variable-name*

移行行で更新する列を識別します。 *transition-variable-name* は、新しい値を識別する相関名によってオプションで修飾されている、トリガーのサブジェクト表にある列を識別していなければなりません (SQLSTATE 42703)。

*..attribute name*

設定されている構造化タイプの属性 (属性割り当て という) を指定します。指定する *SQL-variable-name* または *transition-variable-name* は、ユーザー定義の構造化タイプで定義されていなければなりません (SQLSTATE 428DP)。 *..attribute-name* は、構造化タイプの属性でなければなりません (SQLSTATE 42703)。 *..attribute name* 節が含まれない割り当てを、従来型の割り当てといいます。

*array-variable-name*

SQL 変数またはパラメーターを指定します。変数またはパラメーターは配列型でなければなりません (SQLSTATE 428H0)。

*[subindex]*

配列のどのエレメントが割り当てのターゲットとなるかを指定する数式。副索引は、位取りがゼロの厳密な数値タイプでなければならず (SQLSTATE 428H1)、NULL にすることはできません。その値は、1 と、配列に定義された最大カーディナリティーとの間でなければなりません (SQLSTATE 2202E)。

*expression*

割り当てのターゲットの新しい値を指定します。この *expression* (式) として、『式』で説明されているタイプの式はいずれも使用することができます。スカラー *fullselect* で使用される場合を除き、列関数を組み込むことはできません (SQLSTATE 42903)。 *CREATE TRIGGER* ステートメントのコンテキストにおいて、*expression* は OLD および NEW 遷移変数への参照を含むことができます。遷移変数は、*correlation-name* で修飾されていなければなりません (SQLSTATE 42702)。

*array-constructor*

配列コンストラクターは、配列データ・タイプの値を作成するために使用されます。

*fullselect*

単一の列を戻す全選択です。全選択によって戻される値は、配列のエレメントになります。配列のカーディナリティーは、全選択によって戻される行数と同じです。配列内のエレメントの順序を指定するために *ORDER BY* 節を全選択で使用できます。使用しない場合、順序は定義されません。配列の基本タイプは「結果データ・タイプの規則」に基づいて決定されます。基本タイプは、*CREATE TYPE* (配列) ステートメントで許容されるいずれかのタイプでなければなりません (SQLSTATE 429C2)。

*element-expression*

配列内のエレメントの値を定義する式。配列のカーディナリティーは、エレメント式の数と同じです。最初の *element-expression* は副索引 1 を持つ配列エレメントに割り当てられます。2 番目の *element-expression* は副索引 2 を持つ配列エレメントに割り当てられ、それ以降も同様になります。すべてのエレメント式のデータ・タイプは互換でなければなりません (SQLSTATE 429C2)。配列の基本タイプは「結果データ・タイプの規則」に基づいて決定されます。基本タイプは、*CREATE TYPE* (配列) ステートメントで許容されるいずれかのタイプでなければなりません (SQLSTATE 429C2)。

*trim-array-function*

配列の引数のコピーを戻します。その際、指定した数のエレメントが配列の末尾から除去されます。

*array-variable-name*

SQL 変数またはパラメーターを指定します。変数またはパラメーターは配列型でなければなりません (SQLSTATE 428H0)。

*expression*

*array-variable-name* のコピーから切り取られるエレメント数を指定する数式。 *expression* は、0 と *array-variable-name* のカーディナリティーとの間でなければなりません (SQLSTATE 2202E)。結果のカーディナリティーは、切り取られるエレメントの数だけ少なくなります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

**NULL**

NULL 値を指定します。属性のデータ・タイプに特別にキャストされた場合を除いて、NULL を属性割り当ての値にすることはできません (SQLSTATE 429B9)。

**DEFAULT**

デフォルト値が使用されることを指定します。

SQL プロシージャでは、静的 SQL ステートメントに対してのみ DEFAULT 節を指定できます。ただし、例外として、動的 SQL ステートメント内で *target-variable* がグローバル変数である場合には、DEFAULT 節を指定できます。

*target-variable* が列であれば、挿入される値は、どのように列が表に定義されているかによって異なります。

- 列が WITH DEFAULT 節で定義されている場合、値は、その列に定義されたデフォルトに設定されます (『ALTER TABLE』の *default-clause* を参照してください)。
- 列が IDENTITY 節で定義されている場合、値はデータベース・マネージャーによって生成されます。
- 列が WITH DEFAULT 節、IDENTITY 節、または NOT NULL 節のいずれも指定せずに定義されている場合、値は NULL になります。
- 列が NOT NULL 節で定義されている場合で、次のいずれかに該当する場合には、列に DEFAULT キーワードを指定できません (SQLSTATE 23502)。
  - IDENTITY 節が使用されていない
  - WITH DEFAULT 節が使用されていない
  - DEFAULT NULL が使用されている

その列に対して DEFAULT キーワードを指定できません (SQLSTATE 23502)。

*target-variable* が SQL 変数であれば、挿入される値は、変数宣言に指定または暗黙指定されているデフォルトになります。

*target-variable* がグローバル変数である場合、挿入される値は、変数の作成時に指定されたデフォルト値です。

*target-variable* が SQL プロシージャ内の SQL 変数または SQL パラメータ、ホスト変数、またはパラメーター・マーカーであれば、DEFAULT キーワードは指定できません (SQLSTATE 42608)。

### *row-fullselect*

割り当てに指定されているターゲット変数の数に対応する列数とともに、単一行を返す全選択です。値は、対応するターゲット変数それぞれに割り当てられます。row-fullselect の結果が行なしであれば、NULL 値が割り当てられます。CREATE TRIGGER ステートメントのコンテキストにおいて、row-fullselect には OLD および NEW 遷移変数への参照を含めることができます。その際、どの遷移変数が使用されるかを指定するために correlation-name で修飾する必要があります (SQLSTATE 42702)。結果に行が複数ある場合、エラーが返されます (SQLSTATE 21000)。

## 規則

- 式から割り当てる値、NULL、DEFAULT、または row-fullselect の数は、割り当てに指定されている target-variables の数に一致していなければなりません (SQLSTATE 42802)。
- SET 変数ステートメントでは、1 つのステートメントで SQL 変数と遷移変数を割り当てることができません (SQLSTATE 42997)。
- トリガー、関数、メソッド、動的コンパウンド・ステートメントの内部で、あるいはこれらのいずれかのオブジェクトによって直接的または間接的に呼び出されるプロシージャの内部で、グローバル変数を割り当てることはできません (SQLSTATE 428GX)。
- 配列コンストラクターまたは ARRAY\_AGG の結果である配列が値として割り当てられる場合には、配列の基本タイプとターゲット変数の基本タイプは同じでなければなりません (SQLSTATE 42821)。
- 割り当てが SET A[idx] = rhs (A は配列変数名、idx は副索引として使用される式、rhs は配列エレメントと同じタイプの式) という形式である場合、
  1. A が NULL 値であれば、A を空の配列に設定します。
  2. 配列 A のカーディナリティーを C とします。
  3. idx が C 以下であれば、idx によって識別される位置の値は rhs の値で置き換えられます。
  4. idx が C より大きい場合には、
    - a. 位置 i (i は C より大きく idx より小さい) の値は NULL 値に設定されます。
    - b. 位置 idx の値は rhs の値に設定されます。
    - c. A のカーディナリティーは idx に設定されます。

## 注

- 特定の割り当て規則に従って、値がターゲット変数に割り当てられます。
- SQL プロシージャの割り当てステートメントは、SQL 割り当て規則に準拠していなければなりません。string 割り当てでは、検索割り当て規則が使用されます。

- 特殊レジスターの名前 (PATH など) と一致した ID で変数が宣言されている場合には、意図せずに特殊レジスターに割り当てられてしまわないように、その変数を引用符で区切ってください (例えば、PATH という変数が整数として宣言されている場合は SET "PATH" = 1;)。
- 複数の割り当てが組み込まれている場合、それぞれの *expression* および *row-fullselect* は、割り当てが実行される前に評価されます。そのため、式または行の全選択でのターゲット変数への参照は常に、単一 SET ステートメントでの割り当ての前のターゲット変数の値となります。
- 特殊タイプとして定義された ID 列が更新された場合は、まずすべての計算がソース・タイプで行われます。その結果は、値が列に実際に割り当てられる前に、ソース・タイプから定義された特殊タイプにキャストされます。(計算に先立って、元の値がソース・タイプにキャストされることはありません。)
- ID 列に対する SET ステートメントで DB2 によって値が生成されるようにするには、DEFAULT キーワードを使用します。

```
SET NEW.EMPNO = DEFAULT
```

この例では、NEW.EMPNO が ID 列として定義されており、この列の更新に使用される値は DB2によって生成されます。

- ID 列に生成されるシーケンス値の使用に関する詳細、および ID 列で値が最大値を超えた場合の詳細は、『INSERT』を参照してください。

## 例

例 1: 現在トリガー・アクションが実行されている行の給与の列を 50000 に設定します。

```
SET NEW_VAR.SALARY = 50000;
```

または

```
SET (NEW_VAR.SALARY) = (50000);
```

例 2: 現在トリガー・アクションが実行されている行の給与と歩合の列を、それぞれ 50000 および 8000 に設定します。

```
SET NEW_VAR.SALARY = 50000, NEW_VAR.COMM = 8000;
```

または

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (50000, 8000);
```

例 3: 現在トリガー・アクションが実行されている行の給与と歩合の列を、更新される行に関連した部門の従業員の平均給与および平均歩合にそれぞれ設定します。

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM)
 = (SELECT AVG(SALARY), AVG(COMM)
 FROM EMPLOYEE E
 WHERE E.WORKDEPT = NEW_VAR.WORKDEPT);
```

例 4: 現在トリガー・アクションが実行されている行の給与と歩合の列を、それぞれ 10000、および元の (つまり SET ステートメントの実行前の) 給与値に設定します。

```
SET NEW_VAR.SALARY = 10000, NEW_VAR.COMM = NEW_VAR.SALARY;
```

または

## SET 変数

```
SET (NEW_VAR.SALARY, NEW_VAR.COMM) = (10000, NEW_VAR.SALARY);
```

例 5: SQL 変数 p\_salary を 10 % ずつ増加させます。

```
SET p_salary = p_salary + (p_salary * .10)
```

例 6: SQL 変数 p\_salary を NULL 値に設定します。

```
SET p_salary = NULL
```

例 7: SQL 配列変数 p\_phonenumbers を、固定値の配列に設定します。

```
SET p_phonenumbers = ARRAY[9055553907, 4165554213, 4085553678]
```

例 8: SQL 配列変数 p\_phonenumbers を、PHONENUMBER 表から取得される数値の配列に設定します。

```
SET p_phonenumbers = ARRAY[SELECT NUMBER
FROM PHONENUMBERS
WHERE EMPID = 624]
```

例 9: SQL 配列変数 p\_phonenumbers の最後のエレメントを除去します。

```
SET p_phonenumbers = TRIM_ARRAY(p_phonenumbers, 1)
```

例 10: p\_mynumber を、配列変数 p\_phonenumbers の最初のエレメントと 10 番目のエレメントに割り当てます。最初の割り当ての後、p\_phonenumbers のカーディナリティーは 1 です。2 度目の割り当ての後、カーディナリティーは 10 になり、エレメント 1 から 9 には暗黙的に NULL 値が割り当てられます。

```
SET p_phonenumbers[1] = p_mynumber
```

```
SET p_phonenumbers[10] = p_mynumber
```

例 11: SECURITY.USERS という表には、データベースに接続可能な各ユーザーごとに 1 行が含まれています。現在時刻と許可レベルをグローバル変数 userinfo.gv\_connect\_time および userinfo.gv\_auth\_level にそれぞれ割り当てます。

```
SET userinfo.gv_connect_time = CURRENT_TIMESTAMP,
userinfo.gv_auth_level = (
SELECT AUTHLEVEL FROM SECURITY.USERS
WHERE USERID = CURRENT USER)
```

---

## SIGNAL

SIGNAL ステートメントは、エラーまたは警告条件を通知するために使用されます。これを使用すると、指定した SQLSTATE とオプションのメッセージ・テキストが、エラーまたは警告とともに戻されます。

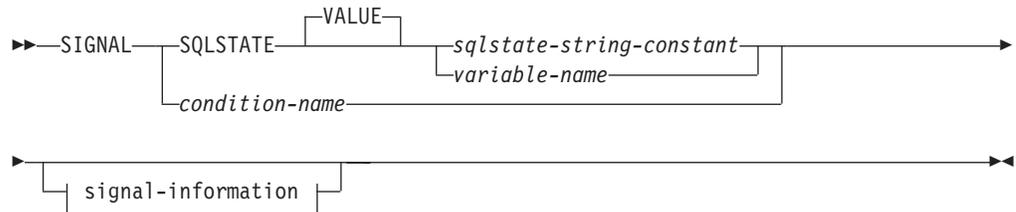
### 呼び出し

このステートメントは、SQL プロシージャまたは動的コンパウンド・ステートメントに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

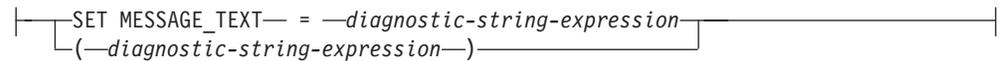
### 許可

必要ありません。

## 構文



### signal-information:



## 説明

### SQLSTATE VALUE *sqlstate-string-constant*

指定されたストリング定数が SQLSTATE を表します。この定数は、正確に 5 文字の文字ストリング定数でなければならず、SQLSTATE の規則に従っていません。

- 各文字は、数字 ('0' から '9')、またはアクセントのない大文字の英字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は '00' にはできません。これは正常な完了を示します。

動的コンパウンド・ステートメント、トリガー、SQL 関数、または SQL メソッドのコンテキストでは、以下の規則も適用されます。

- SQLSTATE クラス (最初の 2 文字) はエラー・クラスではないため、'01' または '02' にはできません。
- SQLSTATE クラスが数字 '0' から '6' または文字 'A' から 'H' で始まっている場合、サブクラス (最後の 3 文字) は 'I' から 'Z' の範囲の文字で始まっていないとできません。
- SQLSTATE クラスが数字 '7'、'8'、'9'、または文字 'I' から 'Z' で始まっている場合、サブクラスとして '0' から '9' または 'A' から 'Z' のいずれでも使用することができます。

SQLSTATE がこれらの規則に従っていない場合、エラーが戻されます (SQLSTATE 428B3)。

### SQLSTATE VALUE *variable-name*

指定する変数名はタイプ CHAR(5) でなければなりません。ステートメント実行時のこの変数の値は、NULL であってはならず、また *sqlstate-string-constant* に記述されているものと同じ規則に準拠している必要があります。SQLSTATE がこれらの規則に従っていない場合、エラーが戻されます (SQLSTATE 428B3)。

### *condition-name*

条件の名前を指定します。条件名は、SIGNAL ステートメントを含むコンパウ

ンド・ステートメント内、またはそのようなコンパウンド・ステートメントがネストしているコンパウンド・ステートメント内で宣言しなければなりません (SQLSTATE 42737)。

#### SET MESSAGE\_TEXT =

エラーまたは警告を記述する文字列を指定します。文字列は SQLCA の SQLERRMC フィールドに返されます。実際の文字列が 70 バイトを超えている場合は、警告なしで切り捨てられます。

#### *diagnostic-string-expression*

エラー条件を記述する最高 70 バイトの文字列を戻すタイプ CHAR または VARCHAR の式。文字列は 70 バイトを超えると切り捨てられます。

#### *(diagnostic-string-expression)*

エラー条件を記述する最高 70 バイトの文字列を戻すタイプ CHAR または VARCHAR の式。文字列は 70 バイトを超えると切り捨てられます。このオプションは、以前のバージョンの DB2 との互換性のために、CREATE TRIGGER ステートメントの有効範囲内でのみ提供されます。通常は使用しないでください。

## 注

- SIGNAL ステートメントが発行される場合、戻される SQLCODE は以下の SQLSTATE に基づいています。
  - 指定した SQLSTATE クラスが '01' か '02' のいずれかである場合、警告か、見つからないことを示す条件が戻され、SQLCODE は +438 に設定されます。
  - それ以外の場合、例外条件が戻され、SQLCODE は -438 に設定されます。

SQLCA の他のフィールドは、以下のように設定されます。

  - sqlerrd フィールドはゼロに設定されます
  - sqlwarn フィールドはブランクに設定されます
  - sqlerrmc は MESSAGE\_TEXT の先頭の 70 バイトに設定されます
  - sqlerrml は sqlerrmc の長さか、SET MESSAGE\_TEXT 節が指定されていない場合にはゼロに設定されます
  - sqlerrp は ROUTINE に設定されます
- SQLSTATE 値は 2 文字のクラス・コード値からなり、その後 3 文字のサブクラス・コード値が続きます。クラス・コード値は実行の成功状態または不成功状態のクラスを表します。

有効な SQLSTATE 値はいずれも SIGNAL ステートメントで使用できます。ただし、プログラマーがアプリケーション用に予約された範囲に基づいて新しい SQLSTATE を定義することをお勧めします。これにより、将来のリリースでデータベース・マネージャーによって定義される可能性のある SQLSTATE 値を誤って使用してしまうのを避けることができます。

- 文字 '7' から '9'、または 'I' から 'Z' で始まる SQLSTATE クラスは定義可能です。これらのクラス内では、サブクラスを定義することができます。
- 文字 '0' から '6'、または 'A' から 'H' で始まる SQLSTATE クラスはデータベース・マネージャー用に予約されています。これらのクラス内では、文字

'0' から 'H' で始まるサブクラスはデータベース・マネージャ用に予約されています。文字 'I' から 'Z' で始まるサブクラスは定義可能です。

## 例

顧客番号がアプリケーションに認識されていないときにアプリケーション・エラーを通知する、オーダー・システムの SQL プロシージャです。ORDERS 表には CUSTOMER 表に対する外部キーが含まれており、オーダーを入れるためには CUSTNO が存在していることが必要になります。

```
CREATE PROCEDURE SUBMIT_ORDER
(IN ONUM INTEGER, IN CNUM INTEGER,
 IN PNUM INTEGER, IN QNUM INTEGER)
SPECIFIC SUBMIT_ORDER
MODIFIES SQL DATA
LANGUAGE SQL
BEGIN
 DECLARE EXIT HANDLER FOR SQLSTATE VALUE '23503'
 SIGNAL SQLSTATE '75002'
 SET MESSAGE_TEXT = 'Customer number is not known';
 INSERT INTO ORDERS (ORDERNO, CUSTNO, PARTNO, QUANTITY)
 VALUES (ONUM, CNUM, PNUM, QNUM);
END
```

---

## TRANSFER OWNERSHIP

TRANSFER OWNERSHIP ステートメントは、データベース・オブジェクトの所有権を転送します。

### 呼び出し

このステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

### 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- オブジェクトの所有権
- SECADM 権限

### 構文

```
►►—TRANSFER OWNERSHIP OF—| objects |—TO—| new-owner |—PRESERVE PRIVILEGES—►►
```

オブジェクト:

## TRANSFER OWNERSHIP

|                          |                                                                    |
|--------------------------|--------------------------------------------------------------------|
| ALIAS                    | <i>alias-name</i>                                                  |
| CONSTRAINT               | <i>table-name.constraint-name</i>                                  |
| DATABASE PARTITION GROUP | <i>db-partition-group-name</i>                                     |
| EVENT MONITOR            | <i>event-monitor-name</i>                                          |
| FUNCTION                 | <i>function-name</i> ( ( <i>data-type</i> ) )                      |
| SPECIFIC FUNCTION        | <i>specific-name</i>                                               |
| FUNCTION MAPPING         | <i>function-mapping-name</i>                                       |
| INDEX                    | <i>index-name</i>                                                  |
| INDEX EXTENSION          | <i>index-extension-name</i>                                        |
| METHOD                   | <i>method-name</i> ( ( <i>data-type</i> ) ) FOR <i>type-name</i>   |
| SPECIFIC METHOD          | <i>specific-name</i>                                               |
| NICKNAME                 | <i>nickname</i>                                                    |
| PACKAGE                  | <i>schema-name.</i> <i>package-id</i> [VERSION <i>version-id</i> ] |
| PROCEDURE                | <i>procedure-name</i> ( ( <i>data-type</i> ) )                     |
| SPECIFIC PROCEDURE       | <i>specific-name</i>                                               |
| SCHEMA                   | <i>schema-name</i>                                                 |
| SEQUENCE                 | <i>sequence-name</i>                                               |
| TABLE                    | <i>table-name</i>                                                  |
| TABLE HIERARCHY          | <i>root-table-name</i>                                             |
| TABLESPACE               | <i>tablespace-name</i>                                             |
| TRIGGER                  | <i>trigger-name</i>                                                |
| TYPE                     | <i>type-name</i> [DISTINCT]                                        |
| TYPE MAPPING             | <i>type-mapping-name</i>                                           |
| VARIABLE                 | <i>variable-name</i>                                               |
| VIEW                     | <i>view-name</i>                                                   |
| VIEW HIERARCHY           | <i>root-view-name</i>                                              |
| XSROBJECT                | <i>xsrobject-name</i>                                              |

### new-owner:

|              |                           |
|--------------|---------------------------|
| USER         | <i>authorization-name</i> |
| SESSION_USER |                           |
| SYSTEM_USER  |                           |

## 説明

### ALIAS *alias-name*

所有権を転送する別名を指定します。*alias-name* (別名) は、カタログに記述されている別名を指定する名前ではありません (SQLSTATE 42704)。

別名の所有権が転送されると、SYSCAT.TABLES カタログ・ビューにある別名の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

**CONSTRAINT** *table-name.constraint-name*

所有権を転送する制約を指定します。 *table-name.constraint-name* (表名.制約名) の組み合わせは、制約とそれが制約する表を指定していなければなりません。 *constraint-name* (制約名) は、カタログに記述されている制約を指定していなければなりません (SQLSTATE 42704)。

制約の所有権が転送されると、SYSCAT.TABCONST カタログ・ビューにある制約の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

- 制約が FOREIGN KEY 制約である場合、SYSCAT.REFERENCES カタログ・ビューにある OWNER 列が、新規所有者の許可 ID に置き換えられます。
- 制約が PRIMARY KEY または UNIQUE 制約である場合、この制約のために暗黙的に作成された索引の SYSCAT.INDEXES カタログ・ビューにある OWNER 列が、新規所有者の許可 ID に置き換えられます。索引が存在しており、この場合にそれが再利用されるのであれば、索引の所有者は変更されません。

**DATABASE PARTITION GROUP** *db-partition-group-name*

所有権を転送するデータベース・パーティション・グループを指定します。 *db-partition-group-name* は、カタログに記述されているデータベース・パーティション・グループを指定していなければなりません (SQLSTATE 42704)。

データベース・パーティション・グループの所有権が転送されると、SYSCAT.DBPARTITIONGROUPS カタログ・ビューにあるデータベース・パーティション・グループの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

**EVENT MONITOR** *event-monitor-name*

所有権を転送するイベント・モニターを指定します。 *event-monitor-name* (イベント・モニター名) は、すでにカタログに存在するイベント・モニターを指定していなければなりません (SQLSTATE 42704)。

イベント・モニターの所有権が転送されると、SYSCAT.EVENTMONITORS カタログ・ビューにあるイベント・モニターの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

指定されたイベント・モニターがアクティブになっている場合、エラーが戻されます (SQLSTATE 429BT)。

所有権を転送する WRITE TO FILE イベント・モニターのターゲット・パスにイベント・ファイルが存在する場合、そのイベント・ファイルは削除されません。

WRITE TO TABLE イベント・モニターの所有権が転送される場合、SYSCAT.EVENTTABLES カタログ・ビューの表情報は保持されます。

**FUNCTION**

所有権を転送する関数を指定します。指定する関数インスタンスは、カタログに記述されたユーザー定義関数、または関数テンプレートでなければなりません。CREATE TYPE (特殊) および CREATE TYPE (構造化) ステートメントで暗黙的に生成された関数の所有権は転送できません (SQLSTATE 429BT)。

関数のインスタンスを指定する方法としては、次のようにいくつかの方法があります。

**FUNCTION** *function-name*

所有権を転送する特定の関数を指定します。*function-name* (関数名) の関数インスタンスが 1 つだけ存在している場合にのみ有効です。このように指定する関数には、任意の数のパラメーターが定義されていても構いません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこの名前関数が存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定したスキーマまたは暗黙のスキーマに、この関数の特定インスタンスが複数存在する場合、エラーが戻されます (SQLSTATE 42725)。

**FUNCTION** *function-name (data-type,...)*

所有権を転送する関数を固有に指定する関数シグニチャーを指定します。関数選択のアルゴリズムは使用されません。

*function-name*

所有権を転送する関数の名前を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

*(data-type,...)*

指定するデータ・タイプは、CREATE FUNCTION ステートメント上で指定されたタイプおよび位置に一致していなければなりません。データ・タイプの数とデータ・タイプを論理的に連結した値を使用して、所有権を転送する特定の関数を識別します。

*data-type* が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE FUNCTION ステートメントにおける指定に完全に一致していなければなりません。

$0 < n < 25$  は REAL を意味し、 $24 < n < 54$  は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

FOR BIT DATA 属性は、一致検索のためのシグニチャーの一部であるとは見なされません。したがって、例えば、シグニチャーの中に CHAR

FOR BIT DATA が指定されている場合、それは CHAR とだけ定義されている関数と一致し、シグニチャーに CHAR とだけ指定されているものは、CHAR FOR BIT DATA と指定されている関数と一致することになります。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つ関数がない場合は、エラーが戻されます (SQLSTATE 42883)。

関数の所有権が転送されると、SYSCAT.ROUTINES カタログ・ビューにある関数の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **SPECIFIC FUNCTION** *specific-name*

関数の作成時に指定された特定の関数名、またはデフォルト値として使用された特定の関数名を使用して、所有権を転送する特定のユーザー定義関数を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。 *specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定関数のインスタンスを指定していなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42704)。

特定の関数の所有権が転送されると、SYSCAT.ROUTINES カタログ・ビューにある特定の関数の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **FUNCTION MAPPING** *function-mapping-name*

所有権を転送する関数マッピングを指定します。 *function-mapping-name* (関数マッピング名) は、カタログに記述されている関数マッピングを指定していなければなりません (SQLSTATE 42704)。

関数マッピングの所有権が転送されると、SYSCAT.FUNCMAPPINGS カタログ・ビューにある関数マッピングの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **INDEX** *index-name*

所有権を転送する索引または SPECIFICATION ONLY 指定の索引を指定します。 *index-name* (索引名) は、カタログに記述されている索引または SPECIFICATION ONLY 指定の索引を識別していなければなりません (SQLSTATE 42704)。

索引の所有権が転送されると、SYSCAT.INDEXES カタログ・ビューにある索引の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

索引が定義されている表がグローバル一時表である場合、索引の所有権は転送できません (SQLSTATE 429BT)。

#### **INDEX EXTENSION** *index-extension-name*

所有権を転送する索引拡張を指定します。 *index-extension-name* (索引拡張名) は、カタログに記述されている索引拡張を指定する名前ではなければなりません (SQLSTATE 42704)。

索引拡張の所有権が転送されると、SYSCAT.INDEXEXTENSIONS カタログ・ビューにある索引拡張の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

**METHOD**

所有権を転送するメソッドを指定します。指定するメソッドの本体は、カタログに記述されているメソッドでなければなりません (SQLSTATE 42704)。

**CREATE TYPE** ステートメントによって暗黙的に生成されたメソッドの所有権は転送できません (SQLSTATE 429BT)。

メソッド本体を指定する方法としては、次のようにいくつかの方法があります。

**METHOD** *method-name*

所有権を転送する特定のメソッドを指定します。この方法は、名前 *method-name* およびサブジェクト・タイプ *type-name* のメソッド・インスタンスが 1 つしかない場合にのみ有効です。この方法を用いる場合は、メソッドにいくつのパラメーターが定義されていても構いません。タイプ *type-name* に、指定された名前のメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定されたデータ・タイプに、そのメソッドの特定のインスタンスが複数存在する場合も、エラーが戻されます (SQLSTATE 42725)。

**METHOD** *method-name (data-type,...)*

所有権を転送するメソッドを固有に指定するメソッド・シングニチャーを指定します。メソッド選択のアルゴリズムは使用されません。

*method-name*

所有権を転送するメソッドの名前を指定します。指定する名前は、修飾なしの ID でなければなりません。

*(data-type, ...)*

指定するデータ・タイプは、**CREATE TYPE** または **ALTER TYPE** ステートメント上で指定されたタイプおよび位置に一致していなければなりません。データ・タイプの数とデータ・タイプを論理的に連結した値を使用して、所有権を転送する特定のメソッド・インスタンスを識別します。

*data-type* が修飾なしの場合は、SQL パス上でスキーマを検索してタイプ名が決定されます。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、**FLOAT()** を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、**CREATE TYPE** ステートメントで指定された値と完全に一致していなければなりません。

$0 < n < 25$  は REAL を意味し、 $24 < n < 54$  は DOUBLE を意味するので、**FLOAT(*n*)** のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定されたデータ・タイプに、指定されたシグニチャーを持つメソッドが存在しない場合は、エラーが戻されます (SQLSTATE 42883)。

#### **FOR** *type-name*

所有権を転送する、指定したメソッドのタイプを指定します。名前は、カタログに記述されているタイプを示すものでなければなりません (SQLSTATE 42704)。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないタイプ名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイル/バインド・オプションによって、修飾子のないタイプ名の修飾子が暗黙指定されます。

メソッドの所有権が転送されると、SYSCAT.ROUTINES カタログ・ビューにあるメソッドの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **SPECIFIC METHOD** *specific-name*

所有権を転送する特定のメソッドを指定します。特定名 (*specific-name*) に修飾子が付いていない場合、動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のない特定名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイル/バインド・オプションによって、修飾子のない固有名の修飾子が暗黙指定されます。*specific-name* に指定される名前は、メソッドの名前でなければなりません。そうでない場合は、エラーが戻されます (SQLSTATE 42704)。

特定のメソッドの所有権が転送されると、SYSCAT.ROUTINES カタログ・ビューにある特定のメソッドの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **NICKNAME** *nickname*

所有権を転送するニックネームを指定します。*nickname* (ニックネーム) は、カタログに記述されているニックネームでなければなりません (SQLSTATE 42704)。

ニックネームの所有権が転送されると、SYSCAT.TABLES カタログ・ビューにあるニックネームの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **PACKAGE** *schema-name.package-id*

所有権を転送するパッケージを指定します。スキーマ名が指定されていない場合、パッケージ ID は暗黙的にデフォルト・スキーマで修飾されます。スキーマ名およびパッケージ ID は、明示的または暗黙的に指定されたバージョン ID とともに、カタログに記述されているパッケージを指定していなければなりません (SQLSTATE 42704)。

#### **VERSION** *version-id*

所有権を転送するパッケージ・バージョンを指定します。値が指定されない場合には、空ストリングがバージョンのデフォルトになり、このパッケージの所有権が転送されます。パッケージ名が同じでバージョンが異なる複数のパッケージが存在する場合には、*version-id* (バージョン ID) が TRANSFER OWNERSHIP ステートメントに指定されているパッケージの所有権のみが転送されます。次のような場合は、バージョン ID を二重引用符で区切ってください。

## TRANSFER OWNERSHIP

- バージョン ID が VERSION(AUTO) プリコンパイラー・オプションによって生成された場合
- バージョン ID が数字で始まる場合
- バージョン ID が小文字であったり、大/小文字混合である場合

ステートメントをオペレーティング・システムのコマンド・プロンプトから呼び出す場合は、各二重引用符の区切り文字の前に円記号を置いて、オペレーティング・システムによって区切り文字が外されないようにします。

パッケージの所有権が転送されると、SYSCAT.PACKAGES カタログ・ビューにあるパッケージの BOUNDBY 列の値が、新規所有者の許可 ID に置き換えられます。

SQL プロシージャに関連したパッケージの所有権は転送できません (SQLSTATE 429BT)。

### PROCEDURE

所有権を転送するプロシージャを指定します。指定するプロシージャ・インスタンスは、カタログに記述されたプロシージャでなければなりません。

プロシージャ・インスタンスを指定する方法としては、次のようにいくつかの方法があります。

#### PROCEDURE *procedure-name*

所有権を転送する特定のプロシージャを指定します。この方法は、スキーマに *procedure-name* のプロシージャが 1 つだけ存在している場合にのみ有効です。この方法で指定するプロシージャには、パラメーターがいくつ定義されていても構いません。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。指定したスキーマまたは暗黙のスキーマにこの名前プロシージャが存在しない場合は、エラーが戻されます (SQLSTATE 42704)。指定したスキーマまたは暗黙のスキーマに、このプロシージャの特定インスタンスが複数存在する場合、エラーが戻されます (SQLSTATE 42725)。

#### PROCEDURE *procedure-name (data-type,...)*

所有権を転送するプロシージャを固有に指定するプロシージャ・シグニチャーを指定します。

##### *procedure-name*

所有権を転送するプロシージャのプロシージャ名を指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

##### *(data-type,...)*

指定するデータ・タイプは、CREATE PROCEDURE ステートメント上で指定されたタイプおよび位置に一致していなければなりません。デー

タ・タイプの数とデータ・タイプを論理的に連結した値を使用して、所有権を転送する特定のプロシージャを識別します。

*data-type* が修飾なしの場合は、SQL パス上でスキーマを検索することによってタイプ名が決定されます。REFERENCE タイプに指定するデータ・タイプ名にも同様の規則が当てはまります。

パラメーター化データ・タイプの長さ、精度、または位取りを指定する必要はありません。空の括弧をコーディングすることによって、一致データ・タイプの検索時にそれらの属性を無視するように指定することができます。

パラメーター値が異なるデータ・タイプ (REAL または DOUBLE) を示しているため、FLOAT() を使用することはできません (SQLSTATE 42601)。

ただし、長さ、精度、または位取りをコーディングする場合、その値は、CREATE PROCEDURE ステートメントにおける指定に完全に一致していなければなりません。

$0 < n < 25$  は REAL を意味し、 $24 < n < 54$  は DOUBLE を意味するので、FLOAT(*n*) のタイプは、*n* に定義された値と一致している必要はありません。マッチングは、タイプが REAL か DOUBLE かに基づいて行われます。

指定したスキーマまたは暗黙のスキーマに、指定したシグニチャーを持つプロシージャがない場合は、エラー (SQLSTATE 42883) が戻されます。

プロシージャの所有権が転送されると、SYSCAT.ROUTINES カタログ・ビューにあるプロシージャの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

関連パッケージを持つ SQL プロシージャの所有権を転送すると、パッケージの所有者も新規所有者に暗黙的に転送されます。

#### **SPECIFIC PROCEDURE** *specific-name*

プロシージャの作成時に指定された特定の名前、またはデフォルト値として使用された特定の名前を使用して、所有権を転送する特定のプロシージャを指定します。動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

*specific-name* (特定名) は、指定したスキーマまたは暗黙のスキーマの特定プロシージャのインスタンスを指定していなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 42704)。

特定のプロシージャの所有権が転送されると、SYSCAT.ROUTINES カタログ・ビューにある特定のプロシージャの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **SCHEMA** *schema-name*

所有権を転送するスキーマを指定します。*schema-name* に指定するスキーマ名は、カタログに記述されているスキーマを識別するものでなければなりません (SQLSTATE 42704)。

## TRANSFER OWNERSHIP

スキーマの所有権が転送されると、SYSCAT.SCHEMATA カタログ・ビューにあるスキーマの OWNER 列および DEFINER 列の値が、新規所有者の許可 ID に置き換えられます。

システム定義スキーマの所有権 (所有者は SYSIBM) は転送できません (SQLSTATE 42832)。

### SEQUENCE *sequence-name*

所有権を転送するシーケンスを指定します。 *sequence-name* (シーケンス名) は、カタログに記述されているシーケンスを指定していなければなりません (SQLSTATE 42704)。

シーケンスの所有権が転送されると、SYSCAT.SEQUENCES カタログ・ビューにあるスキーマの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

### TABLE *table-name*

所有権を転送する表を指定します。 *table-name* は、データベースに存在する表を指定していなければならず (SQLSTATE 42704)、宣言済み一時表を指定してはなりません (SQLSTATE 42995)。

表の所有権が転送されると、以下ようになります。

- SYSCAT.TABLES カタログ・ビューにある表の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。
- SYSCAT.TABDEP カタログ・ビューにある表のすべての従属オブジェクトの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

表階層内の副表の所有権は転送できません (SQLSTATE 429BT)。

フェデレーテッド・システムでは、透過性 DDL を使用して作成されたりリモート表の所有権は転送できます。リモート表の所有権を転送しても、表に関連したニックネームの所有権は転送されません。そうしたニックネームの所有権は、TRANSFER OWNERSHIP ステートメントを使用して、明示的に転送できます。

### TABLE HIERARCHY *root-table-name*

所有権を転送する、型付き表階層のルート表である型付き表を指定します。 *root-table-name* で指定する型付き表は、型付き表階層のルート表でなければなりません (SQLSTATE 428DR)。また、データベースに存在する型付き表を参照する必要があります (SQLSTATE 42704)。

表階層の所有権が転送されると、以下ようになります。

- SYSCAT.TABLES カタログ・ビューにあるルート表およびそのすべての副表の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。
- SYSCAT.TABDEP カタログ・ビューにある表のすべての従属オブジェクトおよびそのすべての副表の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

### TABLESPACE *tablespace-name*

所有権を転送する表スペースを指定します。 *tablespace-name* (表スペース名) は、カタログに記述されている表スペースを指定していなければなりません (SQLSTATE 42704)。

表スペースの所有権が転送されると、SYSCAT.TABLESPACES カタログ・ビューにある表スペースの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **TRIGGER** *trigger-name*

所有権を転送するトリガーを指定します。*trigger-name* (トリガー名) は、カタログに記述されているトリガーを指定していなければなりません (SQLSTATE 42704)。

トリガーの所有権が転送されると、SYSCAT.TRIGGERS カタログ・ビューにあるトリガーの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **TYPE** *type-name*

所有権を転送するユーザー定義タイプを指定します。*type-name* (タイプ名) は、カタログに記述されているタイプを指定していなければなりません (SQLSTATE 42704)。DISTINCT 節が指定されている場合、*type-name* (タイプ名) は、カタログに記述されている特殊タイプを指定していなければなりません (SQLSTATE 42704)。

動的 SQL ステートメントでは、CURRENT SCHEMA 特殊レジスターが、修飾子のないオブジェクト名の修飾子として使用されます。静的 SQL ステートメントでは、QUALIFIER のプリコンパイルまたはバインド・オプションによって、修飾子のないオブジェクト名の修飾子が暗黙指定されます。

タイプの所有権が転送されると、SYSCAT.DATATYPES カタログ・ビューにあるタイプの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **TYPE MAPPING** *type-mapping-name*

所有権を転送するユーザー定義データ・タイプ・マッピングを指定します。*type-mapping-name* (タイプ・マッピング名) は、カタログに記述されているデータ・タイプ・マッピングを指定していなければなりません (SQLSTATE 42704)。

タイプ・マッピングの所有権が転送されると、SYSCAT.TYPEMAPPINGS カタログ・ビューにあるタイプ・マッピングの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **VARIABLE** *variable-name*

所有権が転送されるオブジェクトは、作成されたグローバル変数であることを示します。*variable-name* は、現在のサーバーに存在するグローバル変数を識別するものでなければなりません (SQLSTATE 42704)。

グローバル変数が転送されると、SYSCAT.VARIABLES カタログ・ビューにあるグローバル変数の OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

#### **VIEW** *view-name*

所有権を転送するビューを指定します。*view-name* (ビュー名) は、データベースに存在しているビューを指定していなければなりません (SQLSTATE 42704)。

ビューの所有権が転送されると、以下のようになります。

- SYSCAT.VIEWS カタログ・ビューにあるビューの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。
- SYSCAT.TABDEP カタログ・ビューにあるビューのすべての従属オブジェクトの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

## TRANSFER OWNERSHIP

ビュー階層内のサブビューの所有権は転送できません (SQLSTATE 429BT)。

### **VIEW HIERARCHY** *root-view-name*

所有権を転送する、型付きビュー階層のルート・ビューである型付きビューを指定します。 *root-view-name* で指定する型付きビューは、型付きビュー階層のルート・ビューでなければなりません (SQLSTATE 428DR)。また、データベースに存在する型付きビューを参照する必要があります (SQLSTATE 42704)。

ビュー階層の所有権が転送されると、以下のようになります。

- SYSCAT.VIEWS カタログ・ビューにあるルート・ビューおよびそのすべてのサブビューの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。
- SYSCAT.TABDEP カタログ・ビューにあるビューのすべての従属オブジェクトおよびそのすべてのサブビューの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

### **XSROBJECT** *xsobject-name*

所有権を転送する XSR オブジェクトを指定します。 *xsobject-name* は、カタログに記述されている XSR オブジェクトを識別するものでなければなりません (SQLSTATE 42704)。

XSR オブジェクトの所有権が転送されると、SYSCAT.XSROBJECTS カタログ・ビューにある XSR オブジェクトの OWNER 列の値が、新規所有者の許可 ID に置き換えられます。

### **USER** *authorization-name*

オブジェクトの所有権の転送先となる許可 ID を指定します。

### **SESSION\_USER**

SESSION\_USER 特殊レジスターの値を、オブジェクトの所有権の転送先となる許可 ID として使用することを指定します。

### **SYSTEM\_USER**

SYSTEM\_USER 特殊レジスターの値を、オブジェクトの所有権の転送先となる許可 ID として使用することを指定します。

### **PRESERVE PRIVILEGES**

所有権を転送するオブジェクトの現行所有者が、転送後もオブジェクトに対する既存の特権を引き続き保持することを指定します。例えば、ビューの作成時にビューの作成者に付与された特権は、所有権が別のユーザーに転送された後でも、元の所有者によって引き続き保持されます。

## 規則

- 大半のシステム定義オブジェクトの所有権 (所有者は SYSIBM) は転送できません (SQLSTATE 42832)。ただし、フィックスパック 2 からは、SYSIBM が OWNER 列に含まれており、SYSIBM が DEFINER 列に含まれていない、暗黙的に作成されたスキーマ・オブジェクトの所有権を転送することができます。
- 名前が 'SYS' で始まるスキーマの所有権は転送できません (SQLSTATE 42832)。
- 以下のオブジェクトの所有権は明示的に転送できません (SQLSTATE 429BT)。
  - 表階層内の副表 (これはルート階層表とともに転送されます)
  - ビュー階層内のサブビュー (これはルート階層ビューとともに転送されます)
  - グローバル一時表で定義されている索引

- ユーザー定義タイプの作成時に暗黙的に生成されるメソッドまたは関数
- SQL プロシージャに依存するパッケージ (これは SQL プロシージャとともに転送されます)
- アクティブになっているイベント・モニター (これは、アクティブでないときに転送できます)
- SECADM 権限を持つ許可 ID は、まだオブジェクトの所有者となっていない場合、そのオブジェクトの所有権を自分自身に転送できません (SQLSTATE 42502)。

## 注

- オブジェクトの作成の一部として付与された、現行所有者が持っている特権はすべて、新規所有者に転送されます。現行所有者がオブジェクトに対する特権を取り消されてから、その特権を再び付与された場合、その特権は転送されません。フィックスパック 2 からは、まだ転送されていない、暗黙的に作成されたスキーマ・オブジェクトの場合、新規所有者はそのスキーマに関して CREATEIN 特権、DROPIN 特権、および ALTERIN 特権を与えられ、これらの特権を他のユーザーに与えることもできます。
- データベース・オブジェクトの所有権を転送する場合、新規所有者は、オブジェクトの従属関係で示されているように、基本オブジェクトに対する特権のセットを持っていない限りなりません。これは、オブジェクトの存在を未変更のまま維持するために必要なものです。新規所有者は、オブジェクトの存在を維持するためにそれらの特権が必要でなければ、オブジェクトの作成に必要な特権を必要としません。

以下に例を示します。

- 基礎表に対して SELECT および INSERT の従属関係を持つビューについて考慮します。所有権の転送が成功するためには、ビューの新規所有者が保持する特権に、少なくとも SELECT (GRANT OPTION があってもなくても可) および INSERT (GRANT OPTION があってもなくても可) が含まれている必要があります。従属関係が SELECT WITH GRANT OPTION と INSERT WITH GRANT OPTION である場合、ビューの新規所有者が保持する特権には、少なくとも SELECT WITH GRANT OPTION と INSERT WITH GRANT OPTION が含まれていなければなりません。
- ルーチンに対して従属関係を持つビューについて考慮します。ビューの新規所有者が保持する特権には、少なくとも従属ルーチンに対する EXECUTE が含まれている必要があります。
- 表に対して従属関係を持つトリガーについて考慮します。トリガーの新規所有者が保持する特権には、トリガーの従属関係によって示される、表に対する同じ特権のセットが含まれている必要があります。トリガーが定義されている表に対する ALTER 特権は必須ではありません。

以下の表は、その他のデータベース・オブジェクトが依存するオブジェクトを記述した、システム・カタログ・ビューをリストしています。

表 35. その他のオブジェクトが依存するオブジェクトを記述したカタログ・ビュー

| データベース・オブジェクト | システム・カタログ・ビュー   |
|---------------|-----------------|
| CONSTRAINT    | SYSCAT.CONSTDEP |

表 35. その他のオブジェクトが依存するオブジェクトを記述したカタログ・ビュー (続き)

| データベース・オブジェクト   | システム・カタログ・ビュー                                      |
|-----------------|----------------------------------------------------|
| FUNCTION        | SYSCAT.ROUTINEDEP; SYSCAT.ROUTINES<br>(ソース派生関数の場合) |
| INDEX           | SYSCAT.INDEXDEP                                    |
| INDEX EXTENSION | SYSCAT.INDEXEXTENSIONDEP                           |
| METHOD          | SYSCAT.ROUTINEDEP                                  |
| PACKAGE         | SYSCAT.PACKAGEDEP                                  |
| PROCEDURE       | SYSCAT.ROUTINEDEP                                  |
| TABLE           | SYSCAT.TABDEP                                      |
| TRIGGER         | SYSCAT.TRIGDEP                                     |
| VIEW            | SYSCAT.TABDEP                                      |
| XSROBJECT       | SYSCAT.XSROBJECTDEP                                |

別のオブジェクトに依存するデータベース・オブジェクトの所有権を正常に転送するには、データベース・オブジェクトの新規所有者は、その従属関係の従属オブジェクトに対する一定の特権を保持する必要があります。

- 従属オブジェクトがシーケンスである場合、新規所有者はそのシーケンスに対する USAGE 特権を持っている必要があります。
- 従属オブジェクトが関数、メソッド、またはプロシージャである場合、新規所有者は、その関数、メソッド、またはプロシージャに対する EXECUTE 特権を持っている必要があります。
- 従属オブジェクトがパッケージである場合、新規所有者はそのパッケージに対する EXECUTE 特権を持っている必要があります。
- 従属オブジェクトが XSR オブジェクトである場合、新規所有者はその XSR オブジェクトに対する USAGE 特権を持っている必要があります。

それ以外の従属関係の従属オブジェクトの場合、該当するシステム・カタログ・ビューの TABAUTH 列を使用して、新規所有者が保持する必要がある特権を判別してください。

- オブジェクトの所有権をその所有者に転送しようとする場合、警告が戻されます (SQLSTATE 01676)。
- 次のデータベース・オブジェクトには所有者がないため、その所有権を転送できません。それは、監査ポリシー、バッファー・プール、ロール、セキュリティ・ラベル、セキュリティ・ラベル・コンポーネント、セキュリティ・ポリシー、サーバー、トランスフォーメーション関数、トラステッド・コンテキスト、ユーザー・マッピング、およびラッパーの各オブジェクトです。  
SYSCAT.AUDITPOLICIES、SYSCAT.BUFFERPOOLS、SYSCAT.CONTEXTS、  
SYSCAT.ROLES、SYSCAT.SECURITYLABELS、  
SYSCAT.SECURITYLABELCOMPONENTS、SYSCAT.SECURITYPOLICIES、  
SYSCAT.SERVERS、SYSCAT.TRANSFORMS、SYSCAT.USEROPTIONS、および  
SYSCAT.WRAPPEERS カタログ・ビューには OWNER 列がないことに注意してください。
- 所有権が転送されたオブジェクトのスキーマ名が自動的に変わることはありません。

- 互換性

- 他の SQL ステートメントとの整合性を保つには、以下のようにします。
  - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できません。

## 例

例 1: 表 T1 の所有権を PAUL に転送します。

```
TRANSFER OWNERSHIP OF TABLE WALID.T1
TO USER PAUL PRESERVE PRIVILEGES
```

SYSCAT.TABLES カタログ・ビューにある表 WALID.T1 の OWNER 列の値が、'PAUL' に置き換えられます。Paul には表 WALID.T1 に対する次のような特権が暗黙的に付与されます (表の以前の所有者がその表に対する特権を何も失っていないと想定)。それは、CONTROL および ALTER、DELETE、INDEX、INSERT、SELECT、UPDATE、REFERENCE (WITH GRANT OPTION) です。

例 2: JOHN が表 T1 および T2 を作成し、MIKE が表 JOHN.T1 および JOHN.T2 に対する SELECT 特権を保持していると想定します。MIKE は、表 JOHN.T1 および JOHN.T2 に依存するビュー V1 を作成します。ビュー V1 の所有権を、DBADM 権限を持つ HENRY に転送します。

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER HENRY PRESERVE PRIVILEGES
```

SYSCAT.VIEWS カタログ・ビューにあるビュー V1 の OWNER 列の値が、'HENRY' に置き換えられます。SYSCAT.TABAUTH に新規行が追加されます。その値は、GRANTOR = 'SYSIBM'、GRANTEE = 'HENRY'、および TABNAME = 'V1' です。

例 3: DBADM 権限を持つ HENRY が、表 T1 に依存するトリガー TR1 を作成すると想定します。トリガー TR1 の所有権を、DBADM 権限を保持しない WALID に転送します。

```
TRANSFER OWNERSHIP OF TRIGGER TR1
TO USER WALID PRESERVE PRIVILEGES
```

Walid が DBADM 権限を保持していなくても、トリガーの所有権は正常に転送されます。

例 4: JOHN が表 T1 および T2 を作成し、MIKE が表 JOHN.T1 に対する SELECT 特権と、表 JOHN.T2 に対する CONTROL 特権を保持していると想定します。PAUL は、表 JOHN.T1 および JOHN.T2 に対する SELECT 特権を保持しています。MIKE は、表 JOHN.T1 および JOHN.T2 に依存するビュー V1 を作成します。ビューには、SYSCAT.TABAUTH に SELECT 特権の 1 つの項目と、SYSCAT.TABDEP に表 JOHN.T1 および JOHN.T2 の 2 つの SELECT 従属関係があります。ビュー V1 の所有権を、通常ユーザーである PAUL に転送します。

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER PAUL PRESERVE PRIVILEGES
```

Paul が表 JOHN.T2 に対する CONTROL 特権を保持していなくても、ビューの所有権は正常に転送されます。Paul は、ビューの存在を保持するために、表 JOHN.T1 および JOHN.T2 に対する SELECT 特権だけを必要とします。(ビュー

## TRANSFER OWNERSHIP

には SELECT 特権しかありません。なぜなら、Paul はビューの作成時に両方の表に対する CONTROL 特権を保持しておらず、結果としてビューに対する CONTROL が付与されていないからです。) SYSCAT.VIEWS カタログ・ビューにあるビュー V1 の OWNER 列の値が、'PAUL' に置き換えられます。SYSCAT.TABDEP カタログ・ビューにあるビュー V1 の OWNER 列の値が、'PAUL' に置き換えられます。SYSCAT.TABAUTH に新規行が追加されます。その値は、GRANTOR = 'SYSIBM'、GRANTEE = 'PAUL'、および TABNAME = 'V1' です。

例 5: JOHN が表 T1 を作成し、PUBLIC が JOHN.T1 に対する SELECT 特権を保持していると想定します。PAUL は JOHN.T1 に対する SELECT 特権を明示的に保持し、表 JOHN.T1 に依存するビュー V1 を作成します。ビュー V1 の所有権を MIKE に転送します。MIKE は DBADM ではありませんが、特別なグループ PUBLIC を介して、ビューの所有権を取得するのに必要な特権を保持しています。

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER MIKE PRESERVE PRIVILEGES
```

Mike は PUBLIC を介して表 JOHN.T1 に対する SELECT 特権を保持しているため、ビューの所有権は正常に転送されます。SYSCAT.VIEWS カタログ・ビューにあるビュー V1 の OWNER 列の値が、'MIKE' に置き換えられます。SYSCAT.TABDEP カタログ・ビューにあるビュー V1 の OWNER 列の値が、'MIKE' に置き換えられます。SYSCAT.TABAUTH に新規行が追加されます。その値は、GRANTOR = 'SYSIBM'、GRANTEE = 'MIKE'、および TABNAME = 'V1' です。

例 6: 例 5 と同様に、JOHN が表 T1 を作成し、ロール R1 が JOHN.T1 に対する SELECT 特権を保持していると想定します。PAUL は JOHN.T1 に対する SELECT 特権を明示的に保持し、表 JOHN.T1 に依存するビュー V1 を作成します。ビュー V1 の所有権を MIKE に転送します。MIKE は DBADM ではありませんが、ロール R1 のメンバーシップを介して、ビューの所有権を取得するのに必要な特権を保持しています。

```
TRANSFER OWNERSHIP OF VIEW V1
TO USER MIKE PRESERVE PRIVILEGES
```

Mike はロール R1 のメンバーシップを介して表 JOHN.T1 に対する SELECT 特権を保持しているため、ビューの所有権は正常に転送されます。SYSCAT.VIEWS カタログ・ビューにあるビュー V1 の OWNER 列の値が、'MIKE' に置き換えられます。SYSCAT.TABDEP カタログ・ビューにあるビュー V1 の OWNER 列の値が、'MIKE' に置き換えられます。SYSCAT.TABAUTH に新規行が追加されます。その値は、GRANTOR = 'SYSIBM'、GRANTEE = 'MIKE'、および TABNAME = 'V1' です。

---

## UPDATE

UPDATE ステートメントは、表、ビュー、またはニックネームの行で、あるいは指定された全選択の基礎になる表、ニックネーム、またはビューの行で、指定された列の値を更新します。ビューに対する更新操作に INSTEAD OF トリガーが定義されていない場合、ビューの行を更新することは、そのビューの基本表の行を更新することでもあります。このようなトリガーが定義されている場合は、トリガーが

代わりに実行されます。ニックネームを使用して行を更新することは、そのニックネームが参照するデータ・ソース・オブジェクト中の行を更新することでもあります。

このステートメントの形式は以下のとおりです。

- 検索条件付き UPDATE 形式は、1 つまたは複数の行 (任意指定の検索条件によって決まる) を更新する場合に使用されます。
- 位置指定 UPDATE 形式は、1 行 (カーソルの現在位置によって決まる) だけを更新する場合に使用されます。

## 呼び出し

UPDATE ステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントを使用して出すことができます。このステートメントは、動的に作成できる実行可能ステートメントです。

## 許可

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。

- ターゲット表、ビュー、またはニックネームに対する UPDATE 特権
- 更新するそれぞれの列に対する UPDATE 特権
- ターゲット表、ビュー、またはニックネームに対する CONTROL 特権
- SYSADM または DBADM 権限

割り当て式に *row-fullselect* (行全選択) を含める場合には、ステートメントの許可 ID に、参照される表、ビュー、またはニックネームのそれぞれに対して、以下の特権が少なくとも 1 つ含まれている必要があります。

- SELECT 特権
- CONTROL 特権
- SYSADM または DBADM 権限

副照会によって参照される表、ビュー、またはニックネームのそれぞれに対して、このステートメントの許可 ID が持つ特権には以下の少なくとも 1 つが含まれている必要があります。

- SELECT 特権
- CONTROL 特権
- SYSADM または DBADM 権限

ステートメントの処理に使用されるパッケージが SQL92 規則を使用してプリコンパイルされており (オプション *LANGLEVEL* の値を *SQL92E* または *MIA* と指定)、UPDATE ステートメントの検索条件付き形式で *assignment-clause* の右側または *search-condition* のいずれかの個所に表、ビュー、またはニックネームの列への参照が含まれている場合、このステートメント許可 ID が持つ特権には、さらに以下の少なくとも 1 つが含まれている必要があります。

- SELECT 特権
- CONTROL 特権

# UPDATE

- SYSADM または DBADM 権限

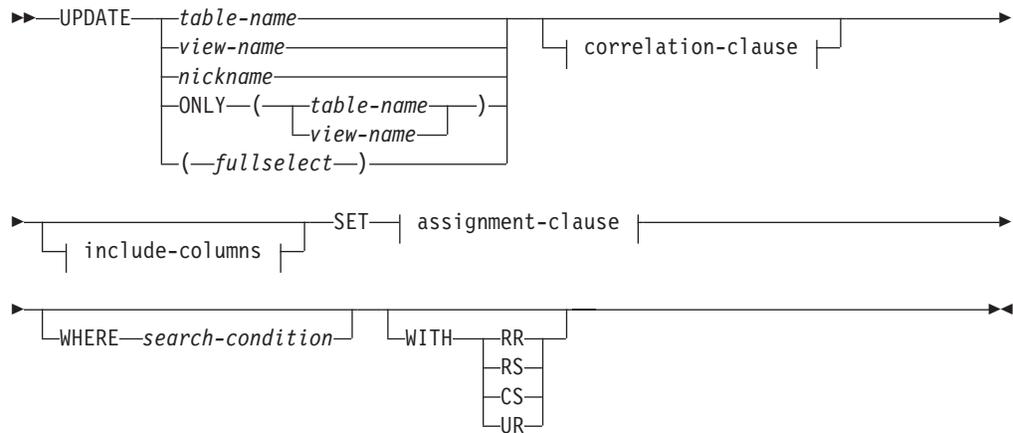
指定した表またはビューが ONLY キーワードの後にくる場合、ステートメントの許可 ID が持つ特権にも、指定した表またはビューの副表またはサブビューごとに SELECT 特権が含まれている必要があります。

静的 UPDATE ステートメントの場合、GROUP 特権はチェックされません。

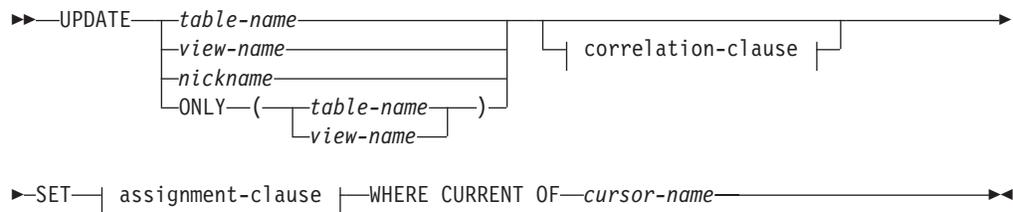
更新操作の対象がニックネームの場合は、データ・ソースでステートメントが実行されないうちは、そのデータ・ソース上のオブジェクトに対する特権は考慮されません。この時点で、データ・ソースに接続するために使用される許可 ID は、データ・ソースのオブジェクトに対して操作を行うのに必要な特権を持っている必要があります。ステートメントの許可 ID は、データ・ソースの別の許可 ID へマップできます。

## 構文

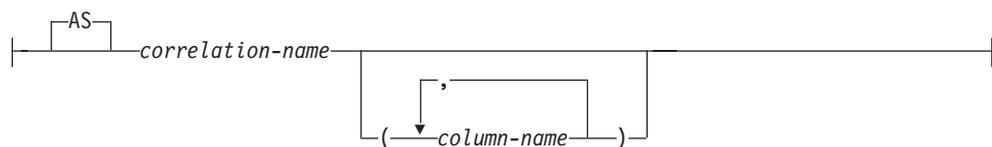
### 検索更新:

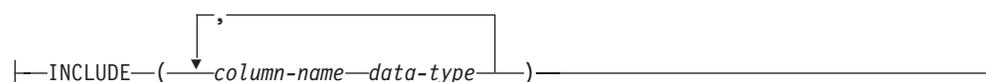
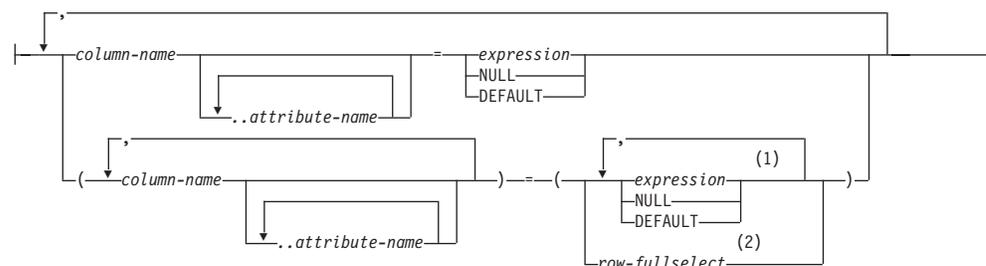


### 位置指定更新:



### correlation-clause:



**include-columns:****assignment-clause:****注:**

- 1 式、NULL、および DEFAULT の数は、列名の数と一致している必要があります。
- 2 選択リストの列の数は、列名の数と一致している必要があります。

**説明****table-name、view-name、nickname、または (fullselect)**

更新操作の対象のオブジェクトを指定します。この名前は、カタログに記述されている表、ビュー、またはニックネームを指定する名前だけでなく、カタログ表、カタログ表のビュー (更新可能な SYSSTAT ビューを除く)、システム保守のマテリアライズ照会表、または更新操作用に INSTEAD OF トリガーが定義されていない読み取り専用のビューを指定することはできません。

*table-name* が型付き表である場合は、このステートメントを使用して、その表またはそれに関係する副表の行を更新できます。WHERE 節で設定または参照できるのは、指定した表の列だけです。位置指定 UPDATE の場合は、FROM 節に指定されているのと同じ表、ビュー、またはニックネームを、関連するカーソルにも ONLY を使用せずに指定しなければなりません。

更新操作のオブジェクトが全選択である場合、全選択は、CREATE VIEW ステートメントの説明の『注』にある、『更新可能ビュー』の項目で定義されているように、更新可能になっている必要があります。

**ONLY (table-name)**

型付き表の場合に適用できます。ONLY キーワードは、指定した表のデータだけにステートメントを適用し、その表に関係する副表の行は更新できないことを指定します。位置指定 UPDATE の場合は、FROM 節に指定されているのと同じ表を、関連するカーソルにも ONLY キーワードを使用して指定しなければなりません。*table-name* が型付き表でない場合は、このステートメントに ONLY キーワードを使用しても効果はありません。

**ONLY (view-name)**

型付きビューの場合に適用できます。ONLY キーワードは、指定されたビュー

のデータだけにステートメントを適用し、その表に関係するサブビューの行は更新できないことを指定します。位置指定 UPDATE の場合は、FROM 節に指定されているのと同じビューを、関連するカーソルにも ONLY を指定して指定しなければなりません。view-name が型付きビューでない場合は、このステートメントに ONLY キーワードを使用しても効果はありません。

#### correlation-clause

search-condition や assignment-clause で、表、ビュー、ニックネーム、または全選択の指定に使用できます。correlation-clause についての説明は、『副選択』の説明にある『table-reference』を参照してください。

#### include-columns

全選択の FROM 節にネストされているとき、table-name や view-name などの列と一緒に UPDATE ステートメントの中間結果表に組み込まれている列セットを指定します。include-columns は、table-name や view-name で指定されている列のリストの最後に付加されます。

#### INCLUDE

UPDATE ステートメントの中間結果表に組み込まれる列のリストを指定します。

#### column-name

UPDATE ステートメントの中間結果表の列を指定します。名前は、他の組み込み列や、table-name または view-name の列と同じ名前であってはなりません (SQLSTATE 42711)。

#### data-type

組み込み列のデータ・タイプを指定します。データ・タイプは、CREATE TABLE ステートメントでサポートされているものでなければなりません。

#### SET

この跡に、列名への値の割り当てを指定します。

#### assignment-clause

##### column-name

更新する列を指定します。column-name は、指定された表、ビュー、またはニックネームの更新可能列か、INCLUDE 列を識別しなければなりません。型付き表のオブジェクト ID 列は更新できません (SQLSTATE 428DZ)。..attribute-name を付けて指定しない限り、1 つの列を複数回指定することはできません (SQLSTATE 42701)。

INCLUDE 列を指定した場合、列名は修飾できません。

位置指定 UPDATE の場合：

- カーソルの select-statement に update-clause を指定した場合、この assignment-clause の各列名は、その update-clause にも指定されていなければなりません。
- カーソルの select-statement に update-clause を指定せず、アプリケーションのプリコンパイル時に LANGLEVEL MIA または SQL92E が指定されていた場合には、更新可能な列の名前はいずれも指定することができません。

- カーソルの *select-statement* に *update-clause* 節を指定せず、アプリケーションのプリコンパイル時に `LANGLEVEL SAA1` を明示的にまたはデフォルト値として指定していた場合には、列は更新できません。

#### *..attribute-name*

設定されている構造化タイプの属性 (属性割り当て という) を指定します。指定される *column-name* は、ユーザー定義構造化タイプで定義されているものでなければなりません (SQLSTATE 428DP)。 *attribute-name* は、*column-name* の構造化タイプの属性でなければなりません (SQLSTATE 42703)。 *..attribute-name* 節と関係のない割り当ては、通常の割り当て と見なされます。

#### *expression*

列の新しい値を指定します。この *expression* (式) として、『式』で説明されているタイプの式はいずれも使用することができます。スカラー `fullselect` で使用される場合を除き、列関数を組み込むことはできません (SQLSTATE 42903)。

*expression* には、UPDATE ステートメントのターゲット表の列への参照を含めることができます。更新対象の行ごとに、式の中のそのような列の値は、行の更新前のその行の列の値になります。

式に、`INCLUDE` 列への参照を含めることはできません。

#### NULL

NULL 値を指定します。NULL 可能列にのみ指定することができます (SQLSTATE 23502)。NULL が特に属性のデータ・タイプにキャストされたのでない限り、属性割り当ての値として NULL を使用することはできません (SQLSTATE 429B9)。

#### DEFAULT

対応する列の表における定義方法に基づくデフォルト値を使用することを指定します。挿入される値は、その列の定義方法によって異なります。

- 式に基づいて生成された列として列が定義されている場合は、その式に基づいた列の値がシステムによって生成されます。
- 列が `IDENTITY` 節で定義されている場合、値はデータベース・マネージャーによって生成されます。
- 列が `WITH DEFAULT` 節で定義されている場合、値は、その列に定義されたデフォルトに設定されます (『ALTER TABLE』の *default-clause* を参照してください)。
- 列の定義に `NOT NULL` 節が使用されたが `GENERATED` 節が使用されなかった場合、また `WITH DEFAULT` 節が使用されていない場合や `DEFAULT NULL` が使用されている場合は、その列に `DEFAULT` キーワードを指定することはできません (SQLSTATE 23502)。
- `ROW CHANGE TIMESTAMP` 節を使って列が定義された場合、値はデータベース・マネージャーによって生成されます。

生成された列が `GENERATED ALWAYS` 節で定義されている場合は、`DEFAULT` 以外の値を挿入することはできません (SQLSTATE 428C9)。

属性割り当てでは、`DEFAULT` キーワードを値として使用することはできません (SQLSTATE 429B9)。

データ・ソースが DEFAULT 構文をサポートしていない場合に、割り当てで DEFAULT キーワードを値として使用してニックネームに対する更新を行うことはできません。

#### *row-fullselect*

割り当て式に指定した列名 の数と同じ数の列を含む 1 つの行を戻す全選択です。値は、それぞれ対応する列名 に割り当てられます。この *row-fullselect* の結果の行がない場合は、NULL 値が割り当てられます。

*row-fullselect* (行全選択) には、UPDATE ステートメントのターゲット表の列に対する参照を含めることができます。更新対象の行ごとに、式の中のそのような列の値は、行の更新前のその行の列の値になります。結果に行が複数ある場合、エラーが返されます (SQLSTATE 21000)。

### WHERE

この後に、更新する行を識別する条件を指定します。この節は、省略することも、検索条件を指定することも、またはカーソル名を指定することもできます。この節を省略すると、表、ビュー、またはニックネームのすべての行が更新されます。

#### *search-condition*

副照会以外の検索条件の各列名 は、表、ビュー、またはニックネームの列を指定していなければなりません。検索条件に、同じ表が UPDATE と副照会の両方の基本オブジェクトである副照会が含まれている場合、行が更新される前に、その副照会が完全に評価されます。

検索条件は、表、ビュー、またはニックネームの各行に適用され、検索条件の結果が「真」の行が更新されます。

検索条件に副照会が含まれる場合、その副照会は、検索条件が 1 つの行に適用されるたびに実行され、その結果は検索条件の適用に使用されるものと見なされます。実際には、相関参照が含まれていない副照会は一度実行されるのに対し、相関参照を含む副照会は各行ごとに一度ずつ実行しなければならない場合があります。

### CURRENT OF *cursor-name*

更新操作で使用するカーソルを指定します。『DECLARE CURSOR』で説明されているように、*cursor-name* は、宣言済みカーソルを指定しなければなりません。プログラムで、UPDATE ステートメントよりも前に、該当の DECLARE CURSOR ステートメントがなければなりません。

指定する表、ビュー、またはニックネームは、そのカーソルの SELECT ステートメントの FROM 節でも指定されていなければならず、またそのカーソルの結果表が読み取り専用であってはなりません。(読み取り専用の結果表については、『DECLARE CURSOR』を参照してください。)

UPDATE ステートメントが実行される時点で、そのカーソルは行に位置づけられていなければなりません。その行が更新されます。

この書式の UPDATE は、カーソルが次のものを参照している場合は使用できません (SQLSTATE 42828)。

- INSTEAD OF UPDATE トリガーが定義されているビュー
- ビューを定義する全選択の選択リストに OLAP 関数が含まれているビュー

- WITH ROW MOVEMENT 節を使用して直接または間接的に定義されたビュー

**WITH**

UPDATE ステートメントが実行される分離レベルを指定します。

**RR**

反復可能読み取り

**RS**

読み取り固定

**CS**

カーソル固定

**UR**

非コミット読み取り

ステートメントのデフォルト分離レベルは、ステートメントがバインドされているパッケージの分離レベルです。WITH 節はニックネームには影響を与えません。ニックネームは常にステートメントのデフォルトの分離レベルを使用します。

**規則**

- **トリガー:** UPDATE ステートメントによってトリガーの実行が引き起こされる場合があります。トリガーが他のステートメントの実行を引き起こす場合や、更新値に起因するエラーが発生する場合があります。ビューに対する更新操作を行うと INSTEAD OF トリガーが起動する場合は、そのトリガーによって実行される更新に対して妥当性、参照保全、および制約が検査されます。トリガーを起動させたビューやその基礎表に対する検査は行われません。
- **割り当て:** 更新値は、特定の割り当て規則に従って列に割り当てられます。
- **妥当性:** 更新される列のユニーク索引がある場合には、その表 (またはビューの基本表) に適用される制約に更新された行は、適合していなければなりません。

WITH CHECK OPTION を使用して定義されていないビューが使用される場合、行が変更され、その結果、それらの行がそのビューの定義に適合しないこととなる場合があります。そのような行は、ビューの基本表で更新され、そのビューには現れなくなります。

WITH CHECK OPTION を用いて定義されたビューを使用する場合、更新された行は、そのビューの定義に従っていなければなりません。この状況に関連する規則については、『CREATE VIEW』を参照してください。

- **チェック制約:** 更新値は、表に定義されているチェック制約の検査条件を満たしていなければなりません。

チェック制約が定義されている表に対する UPDATE では、更新される各行ごとに一度、更新される各列に対して制約条件が評価されます。UPDATE ステートメントが処理される時点で、更新される列を参照しているチェック制約だけが検査されます。

- **参照保全:** 更新規則が RESTRICT で、1 つまたは複数の従属行が存在する場合には、親のユニーク・キーの値は変更できません。ただし、NO ACTION の更新規則では、更新ステートメントの完了時にすべての子が親キーを持つ場合、親の

ユニーク・キーを更新することができます。 NULL 以外の外部キーの更新値は、関連する親表の主キーの値に等しくなければなりません。

- **XML 値:** XML 列の値を更新する場合、新しい値は整形 XML 文書でなければなりません (SQLSTATE 2200M)。
- **セキュリティー・ポリシー:** 指定された表または指定されたビューの基本表がセキュリティー・ポリシーで保護されている場合、セッション許可 ID は、以下を許可するラベル・ベースのアクセス制御 (LBAC) クレデンシャルを持っている必要があります。
  - 更新対象となる保護されたすべての列に対する書き込みアクセス (SQLSTATE 42512)
  - RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL オプションを使って生成されたセキュリティー・ポリシーに関して DB2SECURITYLABEL 列に明示的に与えられる値に対する書き込みアクセス (SQLSTATE 23523)
  - 更新対象となるすべての行に対する読み取りおよび書き込みアクセス (SQLSTATE 42519)

さらに、DB2SECURITYLABEL 列に暗黙的な値が使用される場合には、セキュリティー・ポリシーの書き込みアクセスに関するセキュリティー・ラベルもまた、セッション許可 ID に付与されている必要があります (SQLSTATE 23523)。このような暗黙的な値は、以下の場合に使用される可能性があります。

- DB2SECURITYLABEL 列が更新される列のリストに含まれていない (そのため、SESSION 許可 ID の書き込みアクセスのセキュリティー・ラベルに暗黙的に更新される)
- DB2SECURITYLABEL 列の値が明示的に提供されているが、セッション許可 ID がその値に対する書き込みアクセスを持たず、OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL オプションを使ってセキュリティー・ポリシーが生成されている

## 注

- 更新値が制約のいずれかに違反している場合、または UPDATE ステートメントの実行時に他のエラーが発生した場合、行は更新されません。複数の行が更新される順序は、決められていません。
- WITH ROW MOVEMENT 節を使用して定義されたビューへの更新は、ビューの基礎表に対する削除操作および挿入操作を引き起こす可能性があります。詳細は、CREATE VIEW ステートメントの説明を参照してください。
- UPDATE ステートメントの実行が完了すると、SQLCA の SQLERRD(3) の値は更新操作に修飾された行の数を示します。SQL プロシージャ・ステートメントでは、値は GET DIAGNOSTICS ステートメントの ROW\_COUNT 変数を使用して検索できます。SQLERRD(5) フィールドには、活動化されたすべてのトリガーによって挿入、削除、または更新された行の数が入れます。
- 適切なロックがすでに存在している場合を除き、正常な UPDATE ステートメントの実行によって、1 つまたは複数の排他ロックが獲得されます。そのようなロックが解放されるまで、更新された行には、その更新を行ったアプリケーション・プロセス以外はアクセスできません (非コミット読み取り分離レベルを使用

するアプリケーションを除く)。 ロッキングについては、COMMIT、ROLLBACK、および LOCK TABLE の各ステートメントの説明を参照してください。

- 型付き表の列分布統計を更新する場合は、列を最初に生成した副表を指定しなければなりません。
- 同じ構造化タイプの列で複数の属性割り当てが行われる場合は、SET 節で (括弧付きで挿入された SET 節では左から右の順番で) 指定された順に属性が割り当てられます。
- 属性割り当てでは、ユーザー定義構造化タイプの属性に対して mutator メソッドが呼び出されます。たとえば、割り当て `st..a1=x` は、割り当て `st = st..a1(x)` で mutator メソッドを使用した場合と同じ働きをします。
- 通常の割り当ての場合、指定された列に対しては 1 つの割り当てしか行われませんが、属性割り当てでは、1 つの列が複数の割り当てのターゲット列になることができます (ただし、通常の割り当てでターゲット列として指定されていない場合)。
- 特殊タイプとして定義された ID 列が更新された場合は、まずすべての計算がソース・タイプで行われます。その結果は、値が列に実際に割り当てられる前に、ソース・タイプから定義された特殊タイプにキャストされます。(計算に先立って、元の値がソース・タイプにキャストされることはありません。)
- ID 列に対する SET ステートメントで DB2 によって値が生成されるようにするには、DEFAULT キーワードを使用します。

```
SET NEW.EMPNO = DEFAULT
```

この例では、NEW.EMPNO が ID 列として定義されており、この列の更新に使用される値は DB2によって生成されます。

- ID 列に生成されるシーケンス値の使用に関する詳細、または ID 列で値が最大値を超えた場合の詳細は、『INSERT』を参照してください。
- パーティション表では、UPDATE WHERE CURRENT OF *cursor-name* 操作によって、あるデータ・パーティションから別のデータ・パーティションへ行を移動することができます。その後、カーソルはその行の位置を示さなくなるため、その行に対して UPDATE WHERE CURRENT OF *cursor-name* 変更を行うことはできなくなります。ただし、カーソルの次の行は取り出すことが可能です。
- ROW CHANGE TIMESTAMP 節を使って列が定義された場合、その値は行の更新時に常に変更されます。列が SET リスト内に明示的に指定されていない場合でも、データベース・マネージャーはその行に関する値を生成します。値はデータベース・パーティション内の各表パーティションごとに固有で、行の更新時を近似的に示すタイム・スタンプに設定されます。

## 例

- 例 1: EMPLOYEE 表において、従業員番号 (EMPNO) '000290' のジョブ (JOB) を 'LABORER' に変更します。

```
UPDATE EMPLOYEE
SET JOB = 'LABORER'
WHERE EMPNO = '000290'
```

- 例 2: PROJECT 表において、部門 (DEPTNO) 'D21' が担当しているすべてのプロジェクトについて、プロジェクトのスタッフ・レベル (PRSTAFF) を 1.5 増やします。

## UPDATE

```
UPDATE PROJECT
 SET PRSTAFF = PRSTAFF + 1.5
 WHERE DEPTNO = 'D21'
```

- 例 3: 部門 (WORKDEPT) 'E21' の管理者以外の全従業員が一時的に配置替えになったとします。このことは、EMPLOYEE 表において、そのジョブ (JOB) を NULL 値に、給与額 (SALARY、BONUS、COMM) をゼロに変更することにより示されます。

```
UPDATE EMPLOYEE
 SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
 WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

このステートメントは、次のように書き換えることもできます。

```
UPDATE EMPLOYEE
 SET (JOB, SALARY, BONUS, COMM) = (NULL, 0, 0, 0)
 WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

- 例 4: 従業員番号 000120 の従業員の給与と歩合の列を、それぞれ更新後の行の部門の従業員の平均給与と平均歩合に更新します。

```
UPDATE (SELECT EMPNO, SALARY, COMM,
 AVG(SALARY) OVER (PARTITION BY WORKDEPT),
 AVG(COMM) OVER (PARTITION BY WORKDEPT)
 FROM EMPLOYEE E) AS E(EMPNO, SALARY, COMM, AVGSAL, AVGCOMM)
 SET (SALARY, COMM) = (AVGSAL, AVGCOMM)
 WHERE EMPNO = '000120'
```

上のステートメントは、意味的には次のステートメントと同等ですが、EMPLOYEE 表へのアクセスを一度しか必要としません。それに対し、次のステートメントでは、EMPLOYEE 表を二度指定します。

```
UPDATE EMPLOYEE EU
 SET (EU.SALARY, EU.COMM)
 =
 (SELECT AVG(ES.SALARY), AVG(ES.COMM)
 FROM EMPLOYEE ES
 WHERE ES.WORKDEPT = EU.WORKDEPT)
 WHERE EU.EMPNO = '000120'
```

- 例 5: C プログラムにおいて、EMPLOYEE 表の行を表示し、必要に応じて、特定の従業員のジョブ (JOB) を、キーボードから入力した新しいジョブに変更します。

```
EXEC SQL DECLARE C1 CURSOR FOR
 SELECT *
 FROM EMPLOYEE
 FOR UPDATE OF JOB;
```

```
EXEC SQL OPEN C1;
```

```
EXEC SQL FETCH C1 INTO ... ;
if (strcmp (change, "YES") == 0)
 EXEC SQL UPDATE EMPLOYEE
 SET JOB = :newjob
 WHERE CURRENT OF C1;
```

```
EXEC SQL CLOSE C1;
```

- 例 6: これらの例では、列オブジェクトの属性を変化させます。

以下のタイプと表が存在すると想定します。

```

CREATE TYPE POINT AS (X INTEGER, Y INTEGER)
 NOT FINAL WITHOUT COMPARISONS
 MODE DB2SQL

CREATE TYPE CIRCLE AS (RADIUS INTEGER, CENTER POINT)
 NOT FINAL WITHOUT COMPARISONS
 MODE DB2SQL

CREATE TABLE CIRCLES (ID INTEGER, OWNER VARCHAR(50), C CIRCLE)

```

以下の例では、CIRCLES 表を更新して、OWNER 列と、ID が 999 の CIRCLE 列の RADIUS 属性を変更します。

```

UPDATE CIRCLES
 SET OWNER = 'Bruce'
 C..RADIUS = 5
 WHERE ID = 999

```

以下の例では、999 で識別される円の中心の X 座標と Y 座標を転置します。

```

UPDATE CIRCLES
 SET C..CENTER..X = C..CENTER..Y,
 C..CENTER..Y = C..CENTER..X
 WHERE ID = 999

```

以下は、上の 2 つのステートメントを別の方法で書いた例です。この例では、上の例に示した 2 つのステートメントの働きを結合させています。

```

UPDATE CIRCLES
 SET (OWNER,C..RADIUS,C..CENTER..X,C..CENTER..Y) =
 ('Bruce',5,C..CENTER..Y,C..CENTER..X)
 WHERE ID = 999

```

- 例 7: DOCID が '001' の場合に、DOCUMENTS 表の XMLDOC 列を、XMLTEXT 表から選択した解析済みの文字ストリングに更新します。

```

UPDATE DOCUMENTS SET XMLDOC =
 (SELECT XMLPARSE(DOCUMENT C1 STRIP WHITESPACE)
 FROM XMLTEXT WHERE TEXTID = '001')
 WHERE DOCID = '001'

```

---

## VALUES INTO

VALUES INTO ステートメントは、0 行か 1 行から成る結果表を作成して、その行の値をホスト変数に割り当てます。

### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、動的に作成できない実行可能ステートメントです。

### 許可

必要ありません。

### 構文



```
EXEC SQL VALUES (SESSION_USER)
INTO GV_SESS_USER;
```

## VALUES

VALUES ステートメントは、照会の 1 つの形式です。これは、アプリケーション・プログラムに組み込むことも、または対話式に発行することも可能です。

## WHENEVER

WHENEVER ステートメントは、指定した例外条件が発生した時点で実行するアクションを指定します。

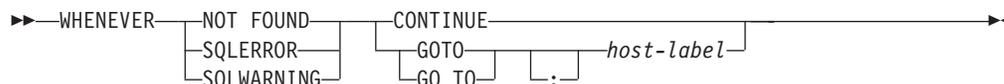
### 呼び出し

このステートメントは、アプリケーション・プログラムに組み込む方法のみ可能です。これは、実行可能ステートメントではありません。このステートメントは REXX ではサポートされません。

### 許可

必要ありません。

### 構文



### 説明

NOT FOUND、SQLERROR、または SQLWARNING の各節は、例外条件のタイプの指定に使用されます。

#### NOT FOUND

SQLCODE が +100、または SQLSTATE が '02000' になる条件を指定します。

#### SQLERROR

SQLCODE が負になる条件を指定します。

#### SQLWARNING

警告状態 (SQLWARN0 が 'W') または SQL 戻りコードが +100 以外の正の値になる条件を指定します。

CONTINUE または GO TO の各節は、指定したタイプの例外条件が生じた場合に行うアクションを指定します。

#### CONTINUE

ソース・プログラムの次に続く命令を実行します。

#### GOTO または GO TO *host-label*

*host-label* で識別されるステートメントに制御を渡します。 *host-label* には、単

## WHENEVER

一のトークンを指定します。オプションとして、その先頭にコロンを付けることができます。トークンの形式は、ホスト言語によって異なります。

### 注

WHENEVER ステートメントには、以下の 3 つのタイプがあります。

- **WHENEVER NOT FOUND**
- **WHENEVER SQLERROR**
- **WHENEVER SQLWARNING**

プログラムの実行可能な SQL ステートメントはいずれも、各タイプの暗黙のまたは明示的な WHENEVER ステートメントの有効範囲内にあります。WHENEVER ステートメントの有効範囲は、プログラムのステートメントの実行順序ではなく、ステートメントのリスト順序に関連しています。

SQL ステートメントは、ソース・プログラムでその SQL ステートメントよりも前に指定されている各タイプの最後の WHENEVER ステートメントの有効範囲内にあります。いずれかのタイプの WHENEVER ステートメントが SQL ステートメントよりも前に指定されていない場合、その SQL ステートメントは、CONTINUE が指定されたそのタイプの暗黙の WHENEVER ステートメントの有効範囲内にあります。

### 例

次の C の例では、エラーが発生した場合に HANDLERR へ進みます。警告コードを生成された場合は、プログラムの通常フローを続行します。データが戻されない場合には、ENDDATA に進みます。

```
EXEC SQL WHENEVER SQLERROR GOTO HANDLERR;
EXEC SQL WHENEVER SQLWARNING CONTINUE;
EXEC SQL WHENEVER NOT FOUND GO TO ENDDATA;
```

---

## WHILE

WHILE ステートメントは、指定した条件が真である間、ステートメント、またはステートメントのグループの実行を繰り返します。

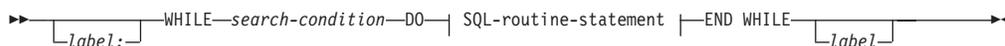
### 呼び出し

このステートメントは、SQL プロシージャまたは動的コンパウンド・ステートメントに組み込むことができます。このステートメントは実行可能ステートメントではなく、動的に準備することはできません。

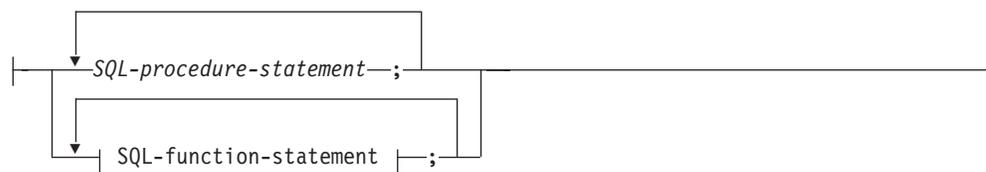
### 許可

WHILE ステートメントを呼び出すために、特権は必要ありません。ただし、ステートメントの許可 ID には、WHILE ステートメントに組み込まれている SQL ステートメントおよび検索条件を呼び出すために必要な特権がなければなりません。

## 構文



### SQL-routine-statement:



## 説明

### label

WHILE ステートメントのラベルを指定します。開始ラベルを指定した場合、それを LEAVE および ITERATE ステートメントで指定することができます。終了ラベルを指定する場合、そのラベルは開始ラベルと同じでなければなりません。

### search-condition

ループが実行される前に評価される条件を指定します。条件が真であれば、ループ内の SQL-procedure-statement が処理されます。

### SQL-procedure-statement

ループ内で実行する SQL ステートメントを指定します。SQL-procedure-statement は、SQL プロシージャのコンテキスト内でのみ使用できます。『コンパウンド SQL (プロシージャ)』で、SQL-procedure-statement を参照してください。

### SQL-function-statement

ループ内で実行する SQL ステートメントを指定します。SQL-function-statement は、SQL 関数または SQL メソッドのコンテキスト内でのみ使用できます。『FOR』で、SQL-function-statement を参照してください。

## 例

以下の例では、WHILE ステートメントを使用して、FETCH から SET ステートメントまでを繰り返します。SQL 変数 *v\_counter* の値が、IN パラメータ *deptNumber* で識別される部門内の従業員数の半分より少ない間は、WHILE ステートメントは FETCH および SET ステートメントを引き続き実行します。条件が真でなくなれば、WHILE ステートメントは制御のフローを渡し、カーソルがクローズされます。

```
CREATE PROCEDURE DEPT_MEDIAN
 (IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
LANGUAGE SQL
BEGIN
 DECLARE v_numRecords INTEGER DEFAULT 1;
 DECLARE v_counter INTEGER DEFAULT 0;
 DECLARE c1 CURSOR FOR
 SELECT CAST(salary AS DOUBLE)
 FROM staff
 WHERE DEPT = deptNumber
```

## WHILE

```
ORDER BY salary;
DECLARE EXIT HANDLER FOR NOT FOUND
SET medianSalary = 6666;
SET medianSalary = 0;
SELECT COUNT(*) INTO v_numRecords
FROM staff
WHERE DEPT = deptNumber;
OPEN c1;
WHILE v_counter < (v_numRecords / 2 + 1) DO
 FETCH c1 INTO medianSalary;
 SET v_counter = v_counter + 1;
END WHILE;
CLOSE c1;
END
```

---

## 付録 A. DB2 技術情報の概説

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

**注:** DB2 インフォメーション・センター のトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、[ibm.com](http://www.ibm.com)<sup>®</sup> にある DB2 インフォメーション・センター を参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks<sup>®</sup> 資料などのその他の DB2 技術情報には、オンライン ([ibm.com](http://www.ibm.com)) でアクセスできます。DB2 Information Management ソフトウェア・ライブラリー・サイト (<http://www.ibm.com/software/data/sw-library/>) にアクセスしてください。

### 資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、[db2docs@ca.ibm.com](mailto:db2docs@ca.ibm.com) まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

IBM Information Management 製品の使用をより容易にするために IBM にご協力いただける場合は、コンシューマビリティに関する次のアンケートにご回答ください。<http://www.ibm.com/software/data/info/consumability-survey/>

## DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、DB2 ライブラリーについて説明しています。DB2 ライブラリーに関する詳細な説明については、[www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order) にある IBM Publications Center にアクセスしてください。英語の DB2 バージョン 9.5 のマニュアル (PDF 形式) とその翻訳版は、[www.ibm.com/support/docview.wss?rs=71&uid=swg2700947](http://www.ibm.com/support/docview.wss?rs=71&uid=swg2700947) からダウンロードできます。

この表には印刷資料が入手可能かどうかを示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: DB2 インフォメーション・センター は、PDF やハードコピー資料よりも頻繁に更新されます。

表 36. DB2 の技術情報

| 資料名                                              | 資料番号         | 印刷資料が入手可能かどうか | 最終更新       |
|--------------------------------------------------|--------------|---------------|------------|
| 管理 API リファレンス                                    | SC88-4431-02 | 入手可能          | 2009 年 4 月 |
| 管理ルーチンおよびビュー                                     | SC88-4435-02 | 入手不可          | 2009 年 4 月 |
| コール・レベル・インターフェース ガイド<br>およびリファレンス 第 1 巻          | SC88-4433-02 | 入手可能          | 2009 年 4 月 |
| コール・レベル・インターフェース ガイド<br>およびリファレンス 第 2 巻          | SC88-4434-02 | 入手可能          | 2009 年 4 月 |
| コマンド・リファレンス                                      | SC88-4432-02 | 入手可能          | 2009 年 4 月 |
| データ移動ユーティリティ<br>ガイドおよびリファレンス                     | SC88-4421-02 | 入手可能          | 2009 年 4 月 |
| データ・リカバリーと<br>高可用性 ガイドおよび<br>リファレンス              | SC88-4423-02 | 入手可能          | 2009 年 4 月 |
| データ・サーバー、<br>データベース、および<br>データベース・オブジェクト<br>のガイド | SC88-4259-02 | 入手可能          | 2009 年 4 月 |
| データベース・セキュリティ<br>ガイド                             | SC88-4418-02 | 入手可能          | 2009 年 4 月 |
| ADO.NET および OLE<br>DB アプリケーション<br>の開発            | SC88-4425-02 | 入手可能          | 2009 年 4 月 |

表 36. DB2 の技術情報 (続き)

| 資料名                                                                     | 資料番号         | 印刷資料が入手可能<br>かどうか | 最終更新       |
|-------------------------------------------------------------------------|--------------|-------------------|------------|
| 組み込み SQL アプリケーションの開発                                                    | SC88-4426-02 | 入手可能              | 2009 年 4 月 |
| Java アプリケーションの開発                                                        | SC88-4427-02 | 入手可能              | 2009 年 4 月 |
| Perl および PHP アプリケーションの開発                                                | SC88-4428-02 | 入手不可              | 2009 年 4 月 |
| SQL および外部ルーチンの開発                                                        | SC88-4429-02 | 入手可能              | 2009 年 4 月 |
| データベース・アプリケーション開発の基礎                                                    | GC88-4430-02 | 入手可能              | 2009 年 4 月 |
| DB2 インストールおよび管理 概説 (Linux および Windows 版)                                | GC88-4439-02 | 入手可能              | 2009 年 4 月 |
| 国際化対応ガイド                                                                | SC88-4420-02 | 入手可能              | 2009 年 4 月 |
| メッセージ・リファレンス 第 1 巻                                                      | GI88-4109-01 | 入手不可              | 2009 年 4 月 |
| メッセージ・リファレンス 第 2 巻                                                      | GI88-4110-01 | 入手不可              | 2009 年 4 月 |
| マイグレーション・ガイド                                                            | GC88-4438-02 | 入手可能              | 2009 年 4 月 |
| Net Search Extender 管理およびユーザズ・ガイド                                       | SC88-4630-02 | 入手可能              | 2009 年 4 月 |
| パーティションおよびクラスタリングのガイド                                                   | SC88-4419-02 | 入手可能              | 2009 年 4 月 |
| Query Patroller 管理およびユーザズ・ガイド                                           | SC88-4611-01 | 入手可能              | 2009 年 4 月 |
| IBM データ・サーバー・クライアント機能概説およびインストール                                        | GC88-4441-02 | 入手不可              | 2009 年 4 月 |
| DB2 サーバー機能概説およびインストール                                                   | GC88-4440-02 | 入手可能              | 2009 年 4 月 |
| Spatial Extender および Geodetic Data Management Feature ユーザズ・ガイドおよびリファレンス | SC88-4629-02 | 入手可能              | 2009 年 4 月 |
| SQL リファレンス 第 1 巻                                                        | SC88-4436-02 | 入手可能              | 2009 年 4 月 |
| SQL リファレンス 第 2 巻                                                        | SC88-4437-02 | 入手可能              | 2009 年 4 月 |

## DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

表 36. DB2 の技術情報 (続き)

| 資料名                                | 資料番号         | 印刷資料が入手可能<br>かどうか | 最終更新       |
|------------------------------------|--------------|-------------------|------------|
| システム・モニター ガ<br>イドおよびリファレン<br>ス     | SC88-4422-02 | 入手可能              | 2009 年 4 月 |
| <i>Text Search</i> ガイド             | SC88-4424-01 | 入手可能              | 2009 年 4 月 |
| 問題判別ガイド                            | GI88-4108-02 | 入手不可              | 2009 年 4 月 |
| データベース・パフォ<br>ーマンスのチューニン<br>グ      | SC88-4417-02 | 入手可能              | 2009 年 4 月 |
| <i>Visual Explain</i> チュー<br>トリアル  | SC88-4449-00 | 入手不可              |            |
| 新機能                                | SC88-4445-02 | 入手可能              | 2009 年 4 月 |
| ワークロード・マネー<br>ジャー ガイドおよびリ<br>ファレンス | SC88-4446-02 | 入手可能              | 2009 年 4 月 |
| <i>pureXML</i> ガイド                 | SC88-4447-02 | 入手可能              | 2009 年 4 月 |
| <i>XQuery</i> リファレンス               | SC88-4448-02 | 入手不可              | 2009 年 4 月 |

表 37. DB2 Connect 固有の技術情報

| 資料名                                                           | 資料番号         | 印刷資料が入手可能<br>かどうか | 最終更新       |
|---------------------------------------------------------------|--------------|-------------------|------------|
| DB2 Connect <i>Personal</i><br><i>Edition</i> 概説およびイン<br>ストール | GC88-4443-02 | 入手可能              | 2009 年 4 月 |
| DB2 Connect サーバー<br>機能 概説およびインス<br>トール                        | GC88-4444-02 | 入手可能              | 2009 年 4 月 |
| DB2 Connect ユーザー<br>ズ・ガイド                                     | SC88-4442-02 | 入手可能              | 2009 年 4 月 |

表 38. Information Integration の技術情報

| 資料名                                                                                                       | 資料番号         | 印刷資料が入手可能<br>かどうか | 最終更新       |
|-----------------------------------------------------------------------------------------------------------|--------------|-------------------|------------|
| <i>Information Integration</i> :<br>フェデレーテッド・シ<br>ステム 管理ガイド                                               | SC88-4166-01 | 入手可能              | 2008 年 3 月 |
| <i>Information Integration</i> :<br>レプリケーションおよ<br>びイベント・パブリッ<br>シングのための<br><i>ASNCLP</i> プログラム・<br>リファレンス | SC88-4167-02 | 入手可能              | 2008 年 3 月 |

表 38. Information Integration の技術情報 (続き)

| 資料名                                               | 資料番号         | 印刷資料が入手可能かどうか | 最終更新       |
|---------------------------------------------------|--------------|---------------|------------|
| Information Integration: フェデレーテッド・データ・ソース 構成ガイド   | SC88-4185-01 | 入手不可          |            |
| Information Integration: SQL レプリケーションガイドおよびリファレンス | SC88-4168-01 | 入手可能          | 2008 年 3 月 |
| Information Integration: レプリケーションとイベント・パブリッシング 概説 | GC88-4187-01 | 入手可能          | 2008 年 3 月 |

## DB2 の印刷資料の注文方法

DB2 の印刷資料が必要な場合、オンラインで購入することができますが、すべての国および地域で購入できるわけではありません。DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。DB2 PDF ドキュメンテーション DVD の一部のソフトコピー・ブックは、印刷資料では入手できないことに留意してください。例えば、「DB2 メッセージ・リファレンス」はどちらの巻も印刷資料としては入手できません。

DB2 PDF ドキュメンテーション DVD で利用できる DB2 の印刷資料の大半は、IBM に有償で注文することができます。国または地域によっては、資料を IBM Publications Center からオンラインで注文することもできます。お客様の国または地域でオンライン注文が利用できない場合、DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。DB2 PDF ドキュメンテーション DVD に収録されている資料の中には、印刷資料として提供されていないものもあります。

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5>) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは <http://www.ibm.com/shop/publications/order> の「ご注文について」をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。
- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようになります。
  1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
    - IBM Directory of world wide contacts ([www.ibm.com/planetwide](http://www.ibm.com/planetwide))
    - IBM Publications Web サイト (<http://www.ibm.com/shop/publications/order>)  
国、地域、または言語を選択し、お客様の所在地に該当する Publications ホ

ーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。

2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、1022 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

---

## コマンド行プロセッサから SQL 状態ヘルプを表示する

DB2 は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサを開いて以下のように入力します。

```
? sqlstate or ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

例えば、? 08003 を指定すると SQL 状態 08003 のヘルプが表示され、? 08 を指定するとクラス・コード 08 のヘルプが表示されます。

---

## 異なるバージョンの DB2 インフォメーション・センターへのアクセス

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>です。

DB2 バージョン 9 のトピックを扱っている DB2 インフォメーション・センターの URL は <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>です。

DB2 バージョン 8 のトピックについては、バージョン 8 のインフォメーション・センターの URL <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>にアクセスしてください。

---

## DB2 インフォメーション・センターでの希望する言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、DB2 インフォメーション・センターでは英語でトピックが表示されます。

- Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。
  1. Internet Explorer の「ツール」->「インターネット オプション」->「言語...」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
    - リストに新しい言語を追加するには、「追加...」ボタンをクリックします。

## DB2 インフォメーション・センターでの希望する言語でのトピックの表示

注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。

- リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「上に移動」ボタンをクリックします。
- 3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。
- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようになります。
  1. 「ツール」->「オプション」->「詳細」ダイアログの「言語」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
    - リストに新しい言語を追加するには、「追加...」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
    - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「上に移動」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

ブラウザとオペレーティング・システムの組み合わせによっては、オペレーティング・システムの地域の設定も希望のロケールと言語に変更しなければならない場合があります。

---

## コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

ローカルにインストールされた DB2 インフォメーション・センター を更新するには、以下のことを行う必要があります。

1. コンピューター上の DB2 インフォメーション・センター を停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。非管理者および非 root の DB2 インフォメーション・センター は常にスタンドアロン・モードで実行されます。を参照してください。
2. 更新機能を使用することにより、どんな更新が利用できるかを確認します。インストールする更新がある場合は、更新機能を使用してそれを入手およびインストールできます。

注: ご使用の環境において、インターネットに接続されていないマシンに DB2 インフォメーション・センター の更新をインストールする必要がある場合は、

## コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

インターネットに接続されていて DB2 インフォメーション・センター がインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングする必要があります。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。

更新パッケージが入手可能な場合、更新機能を使用してパッケージを入手します。ただし、更新機能は、スタンドアロン・モードでのみ使用できます。

3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の DB2 インフォメーション・センター を再始動します。

**注:** Windows Vista の場合、下記のコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを起動するには、ショートカットを右クリックしてから、「**管理者として実行**」を選択します。

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センター を更新するには、以下のようになります。

1. DB2 インフォメーション・センター を停止します。
  - Windows では、「スタート」 → 「コントロール パネル」 → 「管理ツール」 → 「サービス」をクリックします。次に、「DB2 インフォメーション・センター」サービスを右クリックして「**停止**」を選択します。
  - Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv95 stop
```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
  - Windows の場合:
    - a. コマンド・ウィンドウを開きます。
    - b. インフォメーション・センターがインストールされているパスにナビゲートします。DB2 インフォメーション・センター は、デフォルトで `Program_files¥IBM¥DB2 Information Center¥Version 9.5` ディレクトリーにインストールされます。ここで、`Program_files` は Program Files ディレクトリーのロケーションを表します。
    - c. インストール・ディレクトリーから `doc¥bin` ディレクトリーにナビゲートします。
    - d. 次のように `help_start.bat` ファイルを実行します。

```
help_start.bat
```
  - Linux の場合:
    - a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センター は `/opt/ibm/db2ic/V9.5` ディレクトリーにインストールされています。
    - b. インストール・ディレクトリーから `doc/bin` ディレクトリーにナビゲートします。
    - c. 次のように `help_start` スクリプトを実行します。

```
help_start
```

システムのデフォルト Web ブラウザーが起動し、スタンドアロンのインフォメーション・センターが表示されます。

3. 「更新」ボタン (🔄) をクリックします。インフォメーション・センターの右側のパネルで、「更新の検索 (Find Updates)」をクリックします。既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「更新のインストール」をクリックします。
5. インストール・プロセスが完了したら、「完了」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。

- Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help\_end.bat ファイルを実行します。

```
help_end.bat
```

注: help\_end バッチ・ファイルには、help\_start バッチ・ファイルを使用して開始したプロセスを安全に終了するのに必要なコマンドが含まれています。

help\_start.bat は、Ctrl-C や他の方法を使用して終了しないでください。

- Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help\_end スクリプトを実行します。

```
help_end
```

注: help\_end スクリプトには、help\_start スクリプトを使用して開始したプロセスを安全に終了するのに必要なコマンドが含まれています。他の方法を使用して、help\_start スクリプトを終了しないでください。

7. DB2 インフォメーション・センター を再始動します。

- Windows では、「スタート」 → 「コントロール パネル」 → 「管理ツール」 → 「サービス」をクリックします。次に、「DB2 インフォメーション・センター」サービスを右クリックして「開始」を選択します。

- Linux では、以下のコマンドを入力します。

```
/etc/init.d/db2icdv95 start
```

更新された DB2 インフォメーション・センター に、更新された新しいトピックが表示されます。

---

## DB2 チュートリアル

DB2 チュートリアルは、DB2 製品のさまざまな機能について学習するのを支援します。この演習をとおして段階的に学習することができます。

### はじめに

インフォメーション・センター (<http://publib.boulder.ibm.com/infocenter/db2help/>) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

### DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML* ガイド」の『**pureXML™**』

XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

「*Visual Explain* チュートリアル」の『**Visual Explain**』

Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

---

## DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

### DB2 ドキュメンテーション

トラブルシューティング情報は、「DB2 問題判別ガイド」、またはDB2 インフォメーション・センターの『データベースの基本』セクションにあります。ここでは、DB2 診断ツールおよびユーティリティを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および DB2 データベース製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

### DB2 Technical Support の Web サイト

現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。

Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト ([http://www.ibm.com/software/data/db2/support/db2\\_9/](http://www.ibm.com/software/data/db2/support/db2_9/)) にアクセスしてください。

---

## ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

**個人使用:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

**商業的使用:** これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態を提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。



---

## 付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711  
東京都港区六本木 3-2-12  
日本アイ・ビー・エム株式会社  
法務・知的財産  
知的財産権ライセンス渉外

**以下の保証は、国または地域の法律に沿わない場合は、適用されません。** IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書は、IBM 以外の Web サイトおよびリソースへのリンクまたは参照を含む場合があります。IBM は、本書より参照もしくはアクセスできる、または本書からリンクされた IBM 以外の Web サイトもしくは第三者のリソースに対して一切の責任を負いません。IBM 以外の Web サイトにリンクが張られていることにより IBM が当該 Web サイトを推奨するものではなく、またその内容、使用もしくはサイトの所有者について IBM が責任を負うことを意味するものではありません。また、IBM は、お客様が IBM Web サイトから第三者の存在を知ることになった場合にも (もしくは、IBM Web サイトから第三者へのリンクを使用した場合にも)、お客様と第三者との間のいかなる取引に対しても一切責任を負いません。従って、お客様は、IBM が上記の外部サイトまたはリソースの利用について責任を負うものではなく、また、外部サイトまたはリソースからアクセス可能なコンテンツ、サービス、

製品、またはその他の資料一切に対して IBM が責任を負うものではないことを承諾し、同意するものとします。第三者により提供されるソフトウェアには、そのソフトウェアと共に提供される固有の使用条件が適用されます。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited  
Office of the Lab Director  
8200 Warden Avenue  
Markham, Ontario  
L6G 1C7  
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性があります。その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確認できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があります、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. \_年を入れる\_. All rights reserved.

## 商標

IBM、IBM ロゴ、ibm.com は、International Business Machines Corporation の米国およびその他の国における商標または登録商標です。現時点での IBM の商標リストについては、[www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml) の「Copyright and trademark information」をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- Intel、Intel (ロゴ)、Intel Inside、Intel Inside (ロゴ)、Intel Centrino、Intel Centrino (ロゴ)、Celeron、Intel Xeon、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。



# 索引

日本語、数字、英字、特殊文字の順に配列されています。なお、濁音と半濁音は清音と同等に扱われています。

## [ア行]

アセンブラー・アプリケーション・ホスト変数 741  
暗黙スキーマ  
    GRANT (データベース権限) ステートメント 768  
    REVOKE (データベース権限) ステートメント 867  
暗黙接続  
    CONNECT ステートメント 238  
一時表  
    OPEN ステートメント 839  
イベント・モニター  
    CREATE EVENT MONITOR ステートメント 265  
    DROP ステートメント 705  
    FLUSH EVENT MONITOR ステートメント 758  
    SET EVENT MONITOR STATE ステートメント 947  
エラー  
    カーソル 839  
エラー・メッセージ  
    トリガー実行 570  
    戻りコード 9  
    FETCH ステートメント 755  
    UPDATE ステートメント 1004  
オブジェクト ID (OID)  
    列  
        概要 469  
    CREATE TABLE ステートメント 469  
    CREATE VIEW ステートメント 635

## [カ行]

カーソル  
    アクティブ・セットとの関連付け 839  
    アプリケーションでの使用の準備 839  
    位置の移動、FETCH の使用 755  
    オープン 839  
    オープンの位置 755  
    クローズ状態、事前条件 839  
    結果表の関係 673  
    現在行 755  
    更新可能性の判別 673  
    作業単位  
        条件の状態 673  
    作業単位の終了 905  
    削除、検索条件の詳細 688  
    宣言  
        SQL ステートメントの構文 673

カーソル (続き)  
    定義 673  
    表内の位置、FETCH の結果 755  
    プログラムの使用法 673  
    未確定 673  
    読み取り専用  
        条件 673  
    WITH HOLD  
        COMMIT ステートメントのロック節、影響 218  
カーソル名  
    クローズ、CLOSE ステートメント 203  
    ALLOCATE 21  
開始キー値 390  
外部キー  
    制約名規則 469  
    ALTER TABLE を使用した追加 74  
    ALTER TABLE を使用したドロップ 74  
型付きビュー  
    サブビューの定義 635  
カタログ  
    COMMENT ステートメント 205  
関数  
    カタログへのコメントの追加 205  
    トランスフォーム 566  
関数指定子の構文エレメント 16  
関数テンプレート  
    説明 367  
キー値  
    開始 390  
    停止 390  
疑問符  
    EXECUTE パラメーター・マーカー 743  
疑問符 (?) EXECUTE パラメーター・マーカー 743  
キャッシュ  
    EXECUTE ステートメント 743  
行  
    値の割り当て、ホスト変数へ、SELECT INTO 911  
    値の割り当て、ホスト変数へ、VALUES INTO 1015  
    エラーにつながる制限事項 815  
    カーソル、結果表内の位置 673  
    カーソル、FETCH でのクローズの影響 203  
    行データのロック、INSERT ステートメント 815  
    索引 373  
    索引キーと UNIQUE 節 373  
    削除 688  
    挿入 815  
    特権の付与 802  
    列値の更新、UPDATE ステートメント 1004  
    ロック、WITH HOLD のカーソルに対する影響 673  
    FETCH ステートメントでのカーソル 839  
    FETCH 要求、カーソル行選択 673

## 行全選択

UPDATE ステートメント 1004

行を基準にした、列の定位置更新 1004

## 許可

索引に関する制御権の付与 777

索引に対する PUBLIC 制御権 777

データベース操作に関する制御権の付与 768

取り消し 867

付与、スキーマに対する作成の 789

PUBLIC、スキーマに対する作成 789

組み込み SQL アプリケーション

概要 9

文字ストリング形式のステートメント 741

EXECUTE IMMEDIATE ステートメント 741

## クローズ状態

カーソル 839

## 結果セット

戻す

SQL プロシージャ 229

結果セットを戻す

SQL プロシージャから 229

## 結合

CREATE TABLE ステートメント 469

## 検索条件

DELETE

行の選択 688

UPDATE における

引数と規則 1004

## 更新

更新可能なビュー 635

DB2 インフォメーション・センター 1027

構造化照会言語 (SQL)

制御ステートメント 14

構造化タイプ

CREATE TRANSFORM ステートメント 566

DROP ステートメント 705

## 構文

エレメント 16

関数指定子 16

共通エレメント 16

説明 vii

プロシージャ指定子 16

メソッド指定子 16

## コメント

カタログ表 205

SQL 静的ステートメント 9

## ご利用条件

資料の使用 1030

## コンテナ

CREATE TABLESPACE ステートメント 542

コンテナ節

CREATE TABLESPACE ステートメント 542

コンパウンド SQL

動的、変数 221

コンパウンド SQL (組み込み) ステートメント

ブロックへのステートメントの結合 225

# [サ行]

## サーバー

特権の付与 796

## 作業単位 (UOW)

開始時にカーソルはクローズ状態 839

終了 218

終了時に準備済みステートメントが破棄される 844

準備済みステートメントの参照 844

準備済みステートメントの破棄 844

取り消し 905

変更を保管しないで終了 905

COMMIT ステートメント 218

ROLLBACK ステートメントの効果 905

## 索引

カタログ指定コメント、追加 205

## 削除

DROP ステートメントの使用 705

主キー、一致における使用 74

制御権の付与 777, 802

挿入された行の値に対応する 815

## 特権

取り消し 875

## 名前

主キー制約 469

ユニーク制約 469

名前変更 860

ユニーク・キー、一致における使用 74

## 削除可能なビュー

説明 635

## 作成

データベース、権限の付与 768

## 作動不能

トリガー 570

ビュー 635

サポートされる SQL ステートメント 1

## サマリー表

定義 469

## 算術

パラメーター・マーカー 844

## 参照制約

カタログへのコメントの追加 205

## シーケンス

DROP ステートメント 705

実行可能 SQL ステートメント 9

実行不能 SQL ステートメント

プリコンパイラ要件 9

呼び出し 9

## シノニム

CREATE ALIAS ステートメント 254

DROP ALIAS ステートメント 705

## 従属オブジェクト

DROP ステートメント 705

## 終了

作業単位 218, 905

主キー  
 追加  
   ALTER TABLE ステートメント 74  
   CREATE TABLE ステートメント 469  
 ドロップ 74  
 必要な特権 802  
 準備済み SQL ステートメント  
 実行 743  
 情報の取得  
   DESCRIBE INPUT ステートメント 695  
   DESCRIBE OUTPUT ステートメント 698  
 ホスト変数の置換 743  
 条件処理ルーチン  
 宣言 229  
 資料  
 印刷 1022  
   注文 1025  
 概要 1021  
 使用に関するご利用条件 1030  
 PDF 1022  
 スキーマ  
 暗黙的  
   権限の取り消し 867  
   権限の付与 768  
   カタログへのコメントの追加 205  
   CREATE SCHEMA ステートメント 443  
 ステートメント  
 ストリング  
   作成 741  
   PREPARE ステートメント 844  
 ALTER WRAPPER 185  
 LEAVE 826  
 SET CURRENT LOCK TIMEOUT 928  
 SET CURRENT PACKAGE PATH 937  
 ストアド・プロシージャ  
   CREATE PROCEDURE ステートメント 416  
 ストレージ  
   作業単位のバックアウト、ROLLBACK 905  
 ストレージ構造  
   ALTER BUFFERPOOL ステートメント 25  
   ALTER TABLESPACE ステートメント 123  
   CREATE BUFFERPOOL ステートメント 259  
   CREATE TABLESPACE ステートメント 542  
 セーブポイント  
   解放 858  
   ROLLBACK TO SAVEPOINT 905  
 生成列  
   CREATE TABLE ステートメント 469  
 制約  
   カタログへのコメントの追加 205  
   ドロップ  
     ALTER TABLE を使用 74  
     ALTER TABLE を使用した追加 74  
 セキュリティー  
   CONNECT ステートメント 238

セキュリティ管理者権限 (SECADM)  
 取り消し 867  
 GRANT (データベース権限) ステートメント 768  
 セキュリティー・ラベル (LBAC)  
 ポリシー  
   ALTER SECURITY POLICY ステートメント 56  
   CREATE SECURITY POLICY ステートメント 450  
 ALTER SECURITY LABEL COMPONENT ステートメント  
 53  
 CREATE SECURITY LABEL COMPONENT ステートメン  
 ト 446  
 CREATE SECURITY LABEL ステートメント 449  
 GRANT (セキュリティ・ラベル) ステートメント 791  
 REVOKE (セキュリティ・ラベル) ステートメント 888  
 宣言  
   挿入、プログラムへの 814  
 全選択  
   CREATE VIEW ステートメント 635  
 挿入可能なビュー 635

## [タ行]

タイプ 2 索引 373  
 単一行の選択 911  
 単精度浮動小数点データ・タイプ 469  
 チェック制約  
   ALTER TABLE ステートメント 74  
   CREATE TABLE ステートメント 469  
   INSERT ステートメント 815  
 チュートリアル  
   トラブルシューティング 1030  
   問題判別 1030  
   Visual Explain 1029  
 データ  
   保全性  
     ロック 827  
 データベース  
   アクセス  
     権限の付与 768  
     CREATE TABLESPACE ステートメント 542  
 データベース管理スペース (DMS)  
   表スペース  
     CREATE TABLESPACE ステートメント 542  
 データベース権限  
   付与  
     GRANT (データベース権限) ステートメント 768  
 データベース・パーティション・グループ  
   カタログへのコメントの追加 205  
   作成 263  
   パーティションの追加 28  
   パーティションのドロップ 28  
   分散マップの作成 263  
 データ・タイプ  
 構造化  
   ALTER TYPE (構造化) ステートメント 149  
   CREATE TYPE (構造化) ステートメント 598

## データ・タイプ (続き)

抽象 149, 598

### 特殊

CREATE TYPE (特殊) ステートメント 592

### ユーザー定義

CREATE TYPE (特殊) ステートメント 592

ALTER TYPE ステートメント 149

CREATE TYPE (構造化) ステートメント 598

停止キー値 390

## 動的 SQL

### カーソル

DECLARE CURSOR ステートメント 9

コンパウンド・ステートメント 221

呼び出し、ステートメント 9

DESCRIBE INPUT ステートメント

説明 695

DESCRIBE OUTPUT ステートメント

説明 698

EXECUTE IMMEDIATE ステートメント

説明 741

EXECUTE ステートメント

説明 743

呼び出し SQL ステートメント 9

FETCH ステートメント

説明 755

呼び出し SQL ステートメント 9

OPEN ステートメント 9

PREPARE ステートメント

説明 844

呼び出し SQL ステートメント 9

DESCRIBE を使用した 695, 698

## 特殊タイプ

CREATE TYPE (特殊) ステートメント 592

DROP ステートメント 705

特記事項 1033

## 特権

### 取り消し

REVOKE ステートメント 897

### パッケージ

取り消し 877, 897

### database

取り消し 886

### INDEX

取り消し 875

## トラブルシューティング

オンライン情報 1030

チュートリアル 1030

## トランスフォーメーション

### 関数

CREATE TRANSFORM ステートメント 566

DROP ステートメント 705

## トリガー

エラー・メッセージ 570

型付き表 570

カタログへのコメントの追加 205

## トリガー (続き)

### 更新

UPDATE ステートメント 1004

作動不能 570

ドロップ 705

CREATE TRIGGER ステートメント 570

INSERT ステートメント 815

## トリガー SQL ステートメント

SET 変数 979

## [ナ行]

### 名前

行の削除 688

### ニックネーム

説明 403

### 特権

制御権の付与 802

取り消し 897

付与 802

## [ハ行]

### パーティション・キー

追加 74

ドロップ 74

表作成時の定義 469

パーティション・キーでのハッシュ 469

### パーティション・マップ

データベース・パーティション・グループのための作成  
263

バイナリー・ラージ・オブジェクト (BLOB) 469

### バインディング

すべての特権の取り消し 877

GRANT ステートメント 779

### パッケージ

カタログ・コメント 205

削除 705

作成する権限

付与 768

### 特権

付与 779

REVOKE (パッケージ特権) ステートメントを使用した  
取り消し 877

REVOKE (表、ビュー、またはニックネーム特権) ステ  
ートメントを使用した取り消し 897

ALTER TABLE STATEMENT 74

COMMIT ステートメントのカーソルへの影響 218

バッファー挿入 815

バッファー・プール 259

サイズの設定 25, 259

ページ・サイズ 259

DROP ステートメントを使用した削除 705

### パフォーマンス

パーティション・キーに関する推奨事項 469

パラメーター・マーカー  
   型付き 844  
   パスワード規則 844  
   非型付き 844  
   EXECUTE ステートメント 743  
   OPEN ステートメント 839  
   PREPARE ステートメント 844  
 ハンドラー  
   宣言 229  
 ビュー  
   カタログへのコメントの追加 205  
   更新可能 635  
   コントロール特権  
     制限事項 802  
     付与 802  
   削除可能 635  
   作成 635  
   作動不能 635  
   スキーマ 443  
   挿入可能 635  
   特権の取り消し 897  
   特権の付与 802  
   ビュー定義の欠落の回避、WITH CHECK OPTION 1004  
   ビュー表への行の挿入 815  
   別名 254, 705  
   読み取り専用 635  
   列名 635  
   列を基準にして行を更新 1004  
   DROP ステートメントを使用した削除 705  
   WITH CHECK OPTION 1004  
 ビュー名  
   ALTER VIEW ステートメント内 158  
 表  
   一時  
     OPEN ステートメントでの 839  
   型付き、およびトリガー 570  
   行と列を基準にした更新、UPDATE ステートメント 1004  
   行の挿入 815  
   共用アクセスの制限、LOCK TABLE ステートメント 827  
   結合  
     パーティション・キーに関する考慮事項 469  
   索引 373  
   削除  
     DROP ステートメントの使用 705  
   作成  
     権限の付与 768  
     SQL ステートメントの説明 469  
   作成の許可 469  
   スキーマ 443  
   生成列 74  
   追加  
     カタログへのコメント 205  
     列、ALTER TABLE 74  
   特権の取り消し 897  
   特権の付与 802

表 (続き)  
   名前  
     ALTER TABLE ステートメント 74  
     LOCK TABLE ステートメント内 827  
   名前変更 860  
   別名 254, 705  
   変更 74  
   例外 949  
   標識変数  
     説明 741  
   標準  
     動的 SQL の設定規則 973  
   表スペース  
   索引  
     CREATE TABLE ステートメント 469  
   作成  
     CREATE TABLESPACE ステートメント 542  
   追加  
     カタログへのコメント 205  
   特権の取り消し 895  
   特権の付与 800  
   ドロップ  
     DROP ステートメント 705  
   名前変更 858  
   バッファ・プール 259  
   ページ・サイズ 542  
   DROP ステートメントを使用した削除 705  
   ID  
     CREATE TABLE ステートメント 469  
   表名  
     CREATE TABLE ステートメント 469  
   プリコンパイル  
     外部テキスト・ファイル 814  
     実行不能 SQL ステートメント 9  
     INCLUDE ステートメント 814  
     SQLCA 814  
     SQLDA 814  
   プロシージャ  
     作成 416, 436  
     作成の許可  
       CREATE PROCEDURE (SQL) ステートメント 436  
       CREATE PROCEDURE (外部) ステートメント 416  
     CALL ステートメント 195  
   プロシージャ指定子の構文エレメント 16  
   プロシージャ・コンパウンド・ステートメント 229  
   分離レベル  
     DELETE ステートメントにおける 688, 815, 911, 1004  
   並行性制御  
     LOCK TABLE ステートメント 827  
   別名  
     カタログへのコメントの追加 205  
     CREATE ALIAS ステートメント 254  
     DROP ステートメントを使用した削除 705  
   ヘルプ  
     言語の構成 1026  
     SQL ステートメント 1026

## 変換

文字ストリングから実行可能 SQL 741

## ホスト変数

アクティブ・セットとカーソルとのリンク 839

行の値の割り当て

SELECT INTO ステートメント 911

VALUES INTO ステートメント 1015

行への挿入 815

組み込み SQL ステートメント 9

ステートメント・ストリング 844

## 宣言

カーソル 673

BEGIN DECLARE SECTION ステートメント 194

END DECLARE SECTION ステートメント 740

パラメーター・マーカー置換 743

BEGIN DECLARE SECTION ステートメント 194

END DECLARE SECTION ステートメント 740

EXECUTE IMMEDIATE ステートメント 741

FETCH ステートメント 755

REXX アプリケーション 194

## 保全制約

カタログへのコメントの追加 205

## [マ行]

### マテリアライズ照会表 (MQT)

定義 469

REFRESH TABLE ステートメント 853

### 未確定カーソル 673

メソッド指定子の構文エレメント 16

### 文字ストリング

SQL ステートメントとしての実行 741

SQL ステートメント・ストリング、作成規則 741

### 戻りコード

組み込みステートメント 9

実行可能 SQL ステートメント 9

### 問題判別

チュートリアル 1030

利用できる情報 1030

## [ヤ行]

### ユーザー定義関数

CREATE FUNCTION ステートメント

外部スカラー 288

外部表 315

説明 287

ソースまたはテンプレート 344

OLE DB 外部表 335

SQL スカラー、表、または行 356

DROP ステートメント 705

REVOKE (データベース権限) ステートメント 867

### ユーザー定義タイプ

カタログへのコメントの追加 205

構造化タイプ 469

### ユーザー定義タイプ (続き)

特殊データ・タイプ

CREATE TABLE ステートメント 469

CREATE TRANSFORM ステートメント 566

CREATE TYPE (特殊) ステートメント 592

### 有効範囲

追加された列の定義 74

ALTER TABLE ステートメントを使用した追加 74

ALTER VIEW ステートメントを使用した追加 158

CREATE TABLE ステートメントによる定義 469

CREATE VIEW ステートメント 635

### ユニーク制約

ALTER TABLE ステートメント 74

ALTER TABLE を使用した追加 74

ALTER TABLE を使用したドロップ 74

CREATE TABLE ステートメント 469

### ユニーク・キー

ALTER TABLE ステートメント 74

CREATE TABLE ステートメント 469

### 読み取り専用カーソル

未確定 673

読み取り専用ビュー 635

## [ラ行]

### ラベル

GOTO 767

### ラベル・ベースのアクセス制御 (LBAC)

ALTER SECURITY LABEL COMPONENT ステートメント 53

ALTER SECURITY POLICY ステートメント 56

CREATE SECURITY LABEL COMPONENT ステートメント 446

CREATE SECURITY LABEL ステートメント 449

CREATE SECURITY POLICY ステートメント 450

GRANT (セキュリティー・ラベル) ステートメント 791

GRANT (免除) ステートメント 773

REVOKE (セキュリティー・ラベル) ステートメント 888

REVOKE (免除) ステートメント 871

### リモート・アクセス

正常に接続された場合 238

正常に接続されなかった場合 238

CONNECT ステートメント

サーバー情報のみ、オペランドがない 238

EXCLUSIVE MODE、専用接続 238

ON SINGLE DBPARTITIONNUM、専用接続 238

SHARE MODE、コネクタ以外の場合は読み取り専用 238

### 例外表

SET INTEGRITY ステートメント 949

### レコード

行データのロック 815

### 列

値の挿入 815

カタログへのコメントの追加 205

更新 1004

列 (続き)  
 索引キーの作成 373  
 制約名  
 FOREIGN KEY 規則 469  
 追加特権の付与 802  
 名前  
 INSERT ステートメント 815  
 ALTER TABLE ステートメントを使用した追加 74  
 NULL 値  
 ALTER TABLE ステートメント、防止 74  
 列オプション  
 CREATE TABLE ステートメント 469  
 ロード  
 付与、データベース権限 768  
 ログ  
 初期ロギングを行わない表の作成 469  
 ロケータ  
 ASSOCIATE LOCATORS ステートメント 188  
 FREE LOCATOR ステートメント 764  
 ロック  
 作業単位の終了 905  
 制限、アクセス 827  
 COMMIT ステートメント 218  
 INSERT ステートメントに関するデフォルトの規則 815  
 LOCK TABLE ステートメント 827  
 UPDATE の間 1004

## A

ALIAS 節  
 COMMENT ステートメント 205  
 DROP ステートメント 705  
 ALL PRIVILEGES 節  
 GRANT ステートメント (表、ビュー、またはニックネーム) 802  
 REVOKE 表、ビュー、またはニックネーム特権 897  
 ALLOCATE CURSOR ステートメント  
 説明 21  
 ALTER AUDIT POLICY ステートメント 22  
 ALTER BUFFERPOOL ステートメント 25  
 ALTER DATABASE PARTITION GROUP ステートメント 28  
 ALTER DATABASE ステートメント 32  
 ALTER FUNCTION ステートメント 35  
 ALTER HISTOGRAM TEMPLATE ステートメント 38  
 ALTER METHOD ステートメント 39  
 ALTER NICKNAME ステートメント  
 説明 40  
 ALTER NODEGROUP ステートメント  
 「ALTER DATABASE PARTITION GROUP」を参照 28  
 ALTER PROCEDURE (SQL) ステートメント 52  
 ALTER PROCEDURE (外部) ステートメント 48  
 ALTER PROCEDURE (ソース派生) ステートメント 51  
 ALTER SECURITY LABEL COMPONENT ステートメント  
 53  
 ALTER SECURITY POLICY ステートメント 56  
 ALTER SEQUENCE ステートメント 60

ALTER SERVER ステートメント 64  
 ALTER SERVICE CLASS ステートメント 67  
 ALTER TABLE ステートメント  
 構文図 74  
 必要な許可 74  
 例 74  
 ALTER TABLESPACE ステートメント  
 説明 123  
 ALTER THRESHOLD ステートメント 133  
 ALTER TRUSTED CONTEXT ステートメント 140  
 ALTER TYPE (構造化) ステートメント 149  
 ALTER USER MAPPING ステートメント 156  
 ALTER VIEW ステートメント  
 許可 158  
 構文図 158  
 説明 158  
 ALTER WORK ACTION SET ステートメント 160  
 ALTER WORK CLASS SET ステートメント 173  
 ALTER WORKLOAD ステートメント 179  
 ALTER WRAPPER ステートメント 185  
 ALTER XSROBJECT ステートメント 187  
 ALTER 節  
 GRANT ステートメント (表、ビュー、またはニックネーム) 802  
 REVOKE ステートメント、特権の取り消し 897  
 AS 節  
 CREATE VIEW ステートメント 635  
 ASC 節  
 CREATE INDEX ステートメント 373  
 ASSOCIATE LOCATORS ステートメント 188  
 ASUTIME  
 CREATE FUNCTION (外部スカラ) ステートメントにおける 288  
 CREATE FUNCTION (外部表) ステートメントにおける 315  
 CREATE PROCEDURE ステートメントにおける 416, 436  
 AUDIT ステートメント 190

## B

BEGIN DECLARE SECTION ステートメント  
 説明 194  
 必要な許可 194  
 呼び出し規則 194  
 BIGINT データ・タイプ  
 CREATE TABLE ステートメント 469  
 BINDADD パラメーター  
 特権の付与 768  
 BLOB (バイナリー・ラージ・オブジェクト)  
 CREATE TABLE ステートメント 469  
 BUFFERPOOL 節  
 ALTER TABLESPACE ステートメント 123  
 CREATE TABLESPACE ステートメント 542  
 DROP ステートメント 705

## C

- CALL ステートメント  
説明 195
- CASCADE 削除規則 469
- CASE ステートメント 201
- CCSID (コード化文字セット ID)
  - CREATE TABLE ステートメント 469
  - DECLARE GLOBAL TEMPORARY TABLE ステートメントにおける 680
- CHAR VARYING データ・タイプ 469
- CHARACTER VARYING データ・タイプ 469
- CHARACTER データ・タイプ 469
- CHECK 節
  - CREATE VIEW ステートメント 635
- CLOB (文字ラージ・オブジェクト)  
データ・タイプ  
列の作成 469
- CLOSE ステートメント 203
- CLUSTER 節
  - CREATE INDEX ステートメント 373
- COLLID
  - CREATE FUNCTION (外部スカラー) ステートメントにおける 288
  - CREATE FUNCTION (外部表) ステートメントにおける 315
  - CREATE PROCEDURE ステートメントにおける 416, 436
- COLUMN 節
  - COMMENT ステートメント 205
- COMMENT ON ステートメントの FUNCTION 節 205
- COMMENT ステートメント 205
- COMMIT ON RETURN
  - CREATE PROCEDURE ステートメントにおける 416, 436
- COMMIT ステートメント  
説明 218
- CONNECT TO ステートメント
  - 正常に接続された場合 238, 246
  - 正常に接続されなかった場合 238, 246
- CONNECT ステートメント
  - 新しいパスワード情報 238
  - アプリケーション・サーバー情報 238
  - 暗黙接続 238
  - オペランドがない、戻り情報 238
  - 現行のサーバーから切断する 238
  - タイプ 2 246
- CONNECT パラメーター
  - GRANT...ON DATABASE ステートメント 768
- CONSTRAINT 節 205
- CONTINUE 節
  - WHENEVER ステートメント 1017
- CONTROL 節
  - 取り消し 897
  - GRANT ステートメント (表、ビュー、またはニックネーム) 802
- CONTROL パラメーター
  - パッケージに関する特権の取り消し 877
- COPY
  - CREATE INDEX ステートメント 373
- CREATE ALIAS ステートメント  
説明 254
- CREATE AUDIT POLICY ステートメント 256
- CREATE BUFFERPOOL ステートメント  
説明 259
  - except-on-db-partitions-clause 259
- CREATE DATABASE PARTITION GROUP ステートメント 263
- CREATE DISTINCT TYPE ステートメント
  - CREATE TYPE (特殊) ステートメントを参照 592
- CREATE EVENT MONITOR ステートメント 265
- CREATE FUNCTION MAPPING ステートメント  
説明 367
- CREATE FUNCTION (OLE DB 外部表) ステートメント 335
- CREATE FUNCTION (SQL スカラー、表、または行) ステートメント 356
- CREATE FUNCTION (外部スカラー) ステートメント 288
- CREATE FUNCTION (外部表) ステートメント 315
- CREATE FUNCTION ステートメント
  - 外部表 315
  - 説明 287
  - OLE 外部表 335
  - SQL スカラー、表、または行 356
- CREATE FUNCTION (ソース派生またはテンプレート) ステートメント 344
- CREATE FUNCTION (ソース派生) ステートメント 344
- CREATE HISTOGRAM TEMPLATE ステートメント 371
- CREATE INDEX EXTENSION ステートメント 390
- CREATE INDEX ステートメント
  - 索引キー内の column-name 373
  - 説明 373
  - XML 列 373
- CREATE INDEX ステートメント内の CLOSE 373
- CREATE INDEX ステートメント内の FREEPAGE 373
- CREATE METHOD ステートメント  
説明 397
- CREATE NICKNAME ステートメント  
説明 403
- CREATE NODEGROUP ステートメント
  - CREATE DATABASE PARTITION GROUP ステートメント 263
- CREATE PROCEDURE (SQL) ステートメント 436
- CREATE PROCEDURE (外部) ステートメント 416
- CREATE PROCEDURE ステートメント
  - 条件処理ルーチン 229
  - 説明 416
  - 動的コンパウンド・ステートメント 221
  - ハンドラー・ステートメント 229
  - プロシージャ・コンパウンド・ステートメント 229
  - 変数 229
- CASE ステートメント 201
- DECLARE ステートメント 229
- FOR ステートメント 761
- GET DIAGNOSTICS ステートメント 765

CREATE PROCEDURE ステートメント (続き)  
 GOTO ステートメント 767  
 IF ステートメント 812  
 ITERATE ステートメント 824  
 LEAVE ステートメント 826  
 LOOP ステートメント 828  
 REPEAT ステートメント 862  
 RESIGNAL ステートメント 863  
 RETURN ステートメント 865  
 SIGNAL ステートメント 986  
 WHILE ステートメント 1018

CREATE PROCEDURE (ソース派生) ステートメント 432

CREATE ROLE ステートメント 443

CREATE SCHEMA ステートメント 443

CREATE SECURITY LABEL COMPONENT ステートメント 446

CREATE SECURITY LABEL ステートメント 449

CREATE SECURITY POLICY ステートメント 450

CREATE SEQUENCE ステートメント  
 説明 452

CREATE SERVER ステートメント  
 説明 465

CREATE SERVICE CLASS ステートメント 457

CREATE TABLE ステートメント  
 構文図 469

CREATE TABLESPACE ステートメント  
 説明 542

CREATE THRESHOLD ステートメント 557

CREATE TRANSFORM ステートメント 566

CREATE TRIGGER ステートメント  
 説明 570

CREATE TRUSTED CONTEXT ステートメント 583

CREATE TYPE MAPPING ステートメント  
 説明 624

CREATE TYPE (構造化) ステートメント 598

CREATE TYPE (特殊) ステートメント 592

CREATE TYPE (配列) ステートメント 590

CREATE USER MAPPING ステートメント  
 説明 630

CREATE VARIABLE ステートメント 632

CREATE VIEW ステートメント  
 説明 635

CREATE WORK ACTION SET ステートメント 650

CREATE WORK CLASS SET ステートメント 658

CREATE WORKLOAD ステートメント 664

CREATE WRAPPER ステートメント  
 説明 672

CREATETAB パラメーター  
 GRANT...ON DATABASE ステートメント 768

CURRENT DECFLOAT ROUNDING MODE 特殊レジスター  
 SET CURRENT DECFLOAT ROUNDING MODE ステートメント 916

CURRENT DEGREE 特殊レジスター  
 SET CURRENT DEGREE ステートメント 918

CURRENT EXPLAIN MODE 特殊レジスター  
 SET CURRENT EXPLAIN MODE ステートメント 920

CURRENT EXPLAIN SNAPSHOT 特殊レジスター  
 SET CURRENT EXPLAIN SNAPSHOT ステートメント 923

CURRENT FUNCTION PATH 特殊レジスター  
 SET CURRENT FUNCTION PATH ステートメント 970  
 SET CURRENT PATH ステートメント 970  
 SET PATH ステートメント 970

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスター  
 SET CURRENT IMPLICIT XMLPARSE OPTION ステートメント 926

CURRENT ISOLATION 特殊レジスター  
 SET CURRENT ISOLATION ステートメント 927

CURRENT OPTIMIZATION PROFILE 特殊レジスター  
 SET CURRENT OPTIMIZATION PROFILE ステートメント 933

CURRENT PATH 特殊レジスター  
 SET CURRENT FUNCTION PATH ステートメント 970  
 SET CURRENT PATH ステートメント 970  
 SET PATH ステートメント 970

CURRENT QUERY OPTIMIZATION 特殊レジスター  
 SET CURRENT QUERY OPTIMIZATION ステートメント 942

CURRENT REFRESH AGE 特殊レジスター  
 SET CURRENT REFRESH AGE ステートメント 944

CURSOR FOR RESULT SET 変数 21

## D

DB2 インフォメーション・センター  
 言語 1026  
 更新 1027  
 バージョン 1026  
 別の言語で表示する 1026

DB2 資料の印刷方法 1025

db2nodes.cfg ファイル  
 ALTER DATABASE PARTITION GROUP ステートメント 28  
 CONNECT (タイプ 1) ステートメント 238  
 CREATE DATABASE PARTITION GROUP ステートメント 263

DB2SECURITYLABEL データ・タイプ  
 CREATE TABLE ステートメント 469

DBADM (データベース管理) 権限  
 付与 768

DBCLOB データ・タイプ  
 CREATE TABLE ステートメント 469

DECLARE CURSOR ステートメント  
 説明 673

DECLARE GLOBAL TEMPORARY TABLE ステートメント  
 説明 680

DECLARE ステートメント  
 コンパウンド SQL 229  
 BEGIN DECLARE SECTION ステートメント 194  
 END DECLARE SECTION ステートメント 740

DELETE ステートメント  
 説明 688

DESCRIBE INPUT ステートメント 695  
DESCRIBE OUTPUT ステートメント 698  
DESCRIBE ステートメント  
概要 695  
準備済みステートメント  
DESCRIBE INPUT ステートメント 695  
DESCRIBE OUTPUT ステートメント 698  
説明 695  
DISCONNECT ステートメント 703  
DROP ステートメント  
説明 705  
トランスフォーム 705  
DROP ステートメントの NICKNAME 節 705

## E

END DECLARE SECTION ステートメント 740  
EXCLUSIVE MODE 接続 238  
EXECUTE IMMEDIATE ステートメント  
組み込み 9  
説明 741  
EXECUTE ステートメント  
組み込み 9  
説明 743  
EXPLAIN 可能ステートメント  
説明 750  
EXPLAIN ステートメント  
説明 750

## F

FETCH ステートメント  
実行のためのカーソル前提条件 755  
説明 755  
FIELDPROC 節  
ALTER TABLE ステートメント 74  
FLOAT データ・タイプ 469  
FLUSH EVENT MONITOR ステートメント 758  
FLUSH OPTIMIZATION PROFILE CACHE ステートメント  
759  
FLUSH PACKAGE CACHE ステートメント 761  
FOR BIT DATA 節  
CREATE TABLE ステートメント 469  
FOR ステートメント 761  
FOR 節  
CREATE TABLE ステートメント 469  
FOREIGN KEY 節 469  
FREE LOCATOR ステートメント 764  
FROM 節  
DELETE ステートメント 688

## G

GBPCACHE  
CREATE INDEX ステートメント内 373

GET DIAGNOSTICS ステートメント 765  
GO TO 節  
WHENEVER ステートメント 1017  
GOTO ステートメント 767  
GRANT (SETSESSIONUSER 特権) ステートメント 798  
GRANT (XSR オブジェクト特権) ステートメント 811  
GRANT (グローバル変数特権) ステートメント 775  
GRANT (サーバー特権) ステートメント  
説明 796  
GRANT (シーケンス特権) ステートメント  
説明 794  
GRANT (スキーマ特権) ステートメント  
説明 789  
GRANT ステートメント  
データベース権限  
説明 768  
ニックネーム特権 802  
パッケージ特権  
説明 779  
ビュー特権 802  
表、ビュー、またはニックネーム特権  
説明 802  
表特権 802  
CONTROL ON INDEX  
説明 777  
CREATE ON SCHEMA 789  
GRANT (セキュリティ・ラベル) ステートメント 791  
GRANT (パッケージ特権) ステートメント 779  
GRANT (表スペース特権) ステートメント  
説明 800  
GRANT (免除) ステートメント 773  
GRANT (ルーチン特権) ステートメント  
説明 785  
GRANT (ロール) ステートメント 782  
GRANT (ワークロード特権) ステートメント 809  
GRAPHIC データ・タイプ  
CREATE TABLE ステートメント 469

## I

ID 列  
CREATE TABLE ステートメント 469  
IF ステートメント 812  
IN  
CREATE TABLE ステートメント 469  
IN EXCLUSIVE MODE 節  
LOCK TABLE ステートメント 827  
IN SHARE MODE 節  
LOCK TABLE ステートメント 827  
INCLUDE ステートメント 814  
INCLUDE 節  
CREATE INDEX ステートメント 373  
INDEX キーワード  
DROP ステートメント 705  
INDEX 節  
COMMENT ステートメント 205

## INDEX 節 (続き)

- CREATE INDEX ステートメント 373
- GRANT ステートメント (表、ビュー、またはニッケネーム) 802
- REVOKE ステートメント、特権の取り消し 897

## INSERT

- 値の挿入 815
- エラーにつながる制限事項 815

## INSERT ステートメント

- 説明 815

## INSERT 節

- GRANT ステートメント (表、ビュー、またはニッケネーム) 802
- REVOKE ステートメント、特権の取り消し 897

## INTEGER データ・タイプ 469

## INTO 節

- 使用上の制限事項 815
- DESCRIBE ステートメント、SQLDA エリア名 695, 698
- FETCH ステートメント、ホスト変数の置換 755
- INSERT ステートメント、表またはビューの指定 815
- SELECT INTO ステートメント 911
- VALUES INTO ステートメント 1015

## IS 節

- COMMENT ステートメント 205

## ITERATE ステートメント 824

## L

## LBAC (ラベル・ベースのアクセス制御)

### 規則の免除

- GRANT (免除) ステートメント 773
- REVOKE (免除) ステートメント 871

### セキュリティ・ポリシー

- ALTER SECURITY POLICY ステートメント 56
- CREATE SECURITY POLICY ステートメント 450

### セキュリティ・ラベル

- ALTER SECURITY LABEL COMPONENT ステートメント 53
- CREATE SECURITY LABEL COMPONENT ステートメント 446
- CREATE SECURITY LABEL ステートメント 449
- GRANT (セキュリティ・ラベル) ステートメント 791
- REVOKE (セキュリティ・ラベル) ステートメント 888

### セキュリティ・ラベルのコンポーネント

- ALTER SECURITY LABEL COMPONENT ステートメント 53
- CREATE SECURITY LABEL COMPONENT ステートメント 446

### ALTER SECURITY LABEL COMPONENT ステートメント 53

- ALTER SECURITY POLICY ステートメント 56
- CREATE SECURITY LABEL COMPONENT ステートメント 446
- CREATE SECURITY LABEL ステートメント 449

## LBAC (ラベル・ベースのアクセス制御) (続き)

- CREATE SECURITY POLICY ステートメント 450
- GRANT (セキュリティ・ラベル) ステートメント 791
- GRANT (免除) ステートメント 773
- REVOKE (セキュリティ・ラベル) ステートメント 888
- REVOKE (免除) ステートメント 871

## LEAVE ステートメント 826

## LOAD パラメーター

- GRANT...ON DATABASE ステートメント 768

## LOCK TABLE ステートメント

- 説明 827

## LOOP ステートメント

- データベースの基本 828

## M

## MANAGED BY 節

- CREATE TABLESPACE ステートメント 542

## MERGE ステートメント

- 説明 830

## METHOD 節

- DROP ステートメント 705

## MODE キーワード

- LOCK TABLE ステートメント 827

## MQT (マテリアライズ照会表)

- 定義 469
- REFRESH TABLE ステートメント 853

## N

## NO ACTION 削除規則 469

## NOT FOUND 節

- WHENEVER ステートメント 1017

## NOT NULL 節

- CREATE TABLE ステートメント内 469

## NULL CALL

- CREATE TYPE (構造化) ステートメントでの 598

## O

## OF 節

- CREATE VIEW ステートメント 635

## OID

- オブジェクト ID (OID) を参照 469

## ON TABLE 節

- GRANT ステートメント 802
- REVOKE ステートメント 897

## ON UPDATE 節 469

## ON 節

- CREATE INDEX ステートメント 373

## ONLY 節

- DELETE ステートメント 688
- UPDATE ステートメント 1004

## on-db-partitions-clause

- CREATE TABLESPACE ステートメント 542

OPEN ステートメント 839  
OPTION 節  
    CREATE VIEW ステートメント 635

## P

PIECESIZE  
    CREATE INDEX ステートメント 373  
PREPARE ステートメント  
    組み込み 9  
    説明 844  
    動的宣言 844  
    OPEN ステートメントでの変数置換 839  
PROGRAM  
    DROP ステートメント 705  
PROGRAM TYPE  
    CREATE FUNCTION (外部スカラー) ステートメント 288  
    CREATE FUNCTION (外部表) ステートメント 315  
PUBLIC AT ALL LOCATIONS 802

## R

REAL SQL データ・タイプ  
    CREATE TABLE ステートメント 469  
REFERENCES 節  
    GRANT ステートメント (表、ビュー、またはニックネーム) 802  
    REVOKE ステートメント、特権の取り消し 897  
REFRESH TABLE ステートメント  
    説明 853  
    REFRESH DEFERRED 853  
    REFRESH IMMEDIATE 853  
RELEASE SAVEPOINT ステートメント 858  
RELEASE (接続) ステートメント 856  
RENAME TABLESPACE ステートメント 858  
RENAME ステートメント 860  
REPEAT ステートメント 862  
RESIGNAL ステートメント 863  
RESTRICT 削除規則 469  
RESULTSTATUS パラメーター 765  
RETURN ステートメント 865  
REVOKE (SETSESSIONUSER 特権) ステートメント 894  
REVOKE (XSR オブジェクト特権) ステートメント 904  
REVOKE (グローバル変数特権) ステートメント 873  
REVOKE (シーケンス特権) ステートメント  
    説明 890  
REVOKE ステートメント  
    サーバー特権 892  
    索引特権 875  
    スキーマ特権 886  
    データベース権限 867  
    ニックネーム特権 897  
    パッケージ特権 877  
    ビュー特権 897  
    表スペース特権 895

REVOKE ステートメント (続き)  
    表特権 897  
    ルーチン特権 882  
REVOKE (セキュリティ・ラベル) ステートメント 888  
REVOKE (パッケージ特権) ステートメント 877  
REVOKE (免除) ステートメント 871  
REVOKE (ロール) ステートメント 880  
REVOKE (ワークロード特権) ステートメント 903  
REXX 言語  
    END DECLARE SECTION、禁止 740  
ROLLBACK TO SAVEPOINT ステートメント  
    説明 905  
ROLLBACK ステートメント  
    構文 905  
    説明 905  
ROWCOUNT  
    GET DIAGNOSTICS ステートメント 765

## S

SAVEPOINT ステートメント  
    説明 908  
SCHEMA 節  
    COMMENT ステートメント 205  
    DROP ステートメント 705  
SCOPE 節  
    ALTER TABLE ステートメント 74  
    ALTER VIEW ステートメント 158  
    CREATE TABLE ステートメント 469  
    CREATE VIEW ステートメント 635  
SECADM  
    データベース権限  
        取り消し 867  
        GRANT (データベース権限) ステートメント 768  
    パラメーター  
        GRANT (データベース権限) ステートメント 768  
        REVOKE (データベース権限) ステートメント 867  
SECURED WITH 節  
    ALTER TABLE ステートメント 74  
    CREATE TABLE ステートメント 469  
SECURITY LABEL COMPONENT 節  
    COMMENT ステートメント 205  
    DROP ステートメント 705  
SECURITY LABEL 節  
    COMMENT ステートメント 205  
    DROP ステートメント 705  
SECURITY POLICY 節  
    COMMENT ステートメント 205  
    CREATE TABLE ステートメント 469  
    DROP ステートメント 705  
SELECT INTO ステートメント  
    説明 911  
SELECT ステートメント  
    カーソル  
        パラメーター・マーカーに関する規則 673  
    説明 913

SELECT ステートメント (続き)  
 評価  
 OPEN ステートメント・カーソルの結果表の 839  
 select ステートメント (SQL ステートメント構成体)  
 静的起動 9  
 定義 9  
 動的起動 9  
 SELECT 節  
 GRANT ステートメント (表、ビュー、またはニックネーム) 802  
 REVOKE ステートメント、特権の取り消し 897  
 SEQUENCE 節  
 COMMENT ステートメント 205  
 SET COMPILATION ENVIRONMENT ステートメント 913  
 SET CONNECTION ステートメント 914  
 SET CONSTRAINTS ステートメント 949  
 SET CURRENT DECFLOAT ROUNDING MODE ステートメント 916  
 SET CURRENT DEFAULT TRANSFORM GROUP ステートメント 917  
 SET CURRENT DEGREE ステートメント 918  
 SET CURRENT EXPLAIN MODE ステートメント 920  
 SET CURRENT EXPLAIN SNAPSHOT ステートメント 923  
 SET CURRENT FEDERATED ASYNCHRONY ステートメント 925  
 SET CURRENT FUNCTION PATH ステートメント 970  
 SET CURRENT IMPLICIT XMLPARSE OPTION ステートメント 926  
 SET CURRENT ISOLATION ステートメント 927  
 SET CURRENT LOCK TIMEOUT ステートメント 928  
 SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ステートメント 930  
 SET CURRENT MDC ROLLOUT MODE ステートメント 932  
 SET CURRENT OPTIMIZATION PROFILE ステートメント 933  
 SET CURRENT PACKAGE PATH ステートメント 937  
 SET CURRENT PACKAGESET ステートメント 940  
 SET CURRENT PATH ステートメント 970  
 SET CURRENT QUERY OPTIMIZATION ステートメント 942  
 SET CURRENT REFRESH AGE ステートメント 944  
 SET CURRENT SQLID ステートメント 973  
 SET ENCRYPTION PASSWORD ステートメント 946  
 SET EVENT MONITOR STATE ステートメント 947  
 SET INTEGRITY ステートメント 949  
 SET INTEGRITY ペンディング状態 949  
 SET NULL 削除規則 469  
 SET PASSTHRU ステートメント  
 説明 968  
 COMMIT ステートメントから独立 218  
 ROLLBACK ステートメントからの独立性 905  
 SET PATH ステートメント 970  
 SET ROLE ステートメント 972  
 SET SCHEMA ステートメント 973  
 SET SERVER OPTION ステートメント  
 説明 975  
 SET SERVER OPTION ステートメント (続き)  
 COMMIT ステートメントから独立 218  
 ROLLBACK ステートメントからの独立性 905  
 SET SESSION AUTHORIZATION ステートメント 977  
 SET 節  
 UPDATE ステートメント 1004  
 SET 変数ステートメント 979  
 SETSESSIONUSER 特権  
 GRANT (SETSESSIONUSER 特権) ステートメント 798  
 REVOKE (SETSESSIONUSER 特権) ステートメント 894  
 SET SESSION AUTHORIZATION ステートメントに必要な 977  
 SHARE MODE 接続 238  
 SHARE オプション  
 LOCK TABLE ステートメント 827  
 SIGNAL ステートメント 986  
 SMALLINT データ・タイプ  
 static SQL 469  
 SMS (システム管理スペース)  
 表スペース  
 CREATE TABLESPACE ステートメント 542  
 SPECIFIC FUNCTION 節  
 COMMENT ステートメント 205  
 SPECIFIC PROCEDURE 節  
 COMMENT ステートメント 205  
 SQL オブジェクト  
 削除 705  
 SQL ステートメント  
 組み込み 9  
 コンパウンド SQL (組み込み) 225  
 サポートされる 1  
 制御 14  
 対話式の入力 9  
 ヘルプを表示する 1026  
 呼び出し 9  
 ALLOCATE CURSOR 21  
 ALTER AUDIT POLICY 22  
 ALTER BUFFERPOOL 25  
 ALTER DATABASE 32  
 ALTER DATABASE PARTITION GROUP 28  
 ALTER FUNCTION 35  
 ALTER HISTOGRAM TEMPLATE 38  
 ALTER METHOD 39  
 ALTER NICKNAME 40  
 ALTER NODEGROUP (「ALTER DATABASE PARTITION GROUP」を参照) 28  
 ALTER PROCEDURE (SQL) 52  
 ALTER PROCEDURE (外部) 48  
 ALTER PROCEDURE (ソース派生) 51  
 ALTER SECURITY LABEL COMPONENT 53  
 ALTER SECURITY POLICY 56  
 ALTER SEQUENCE 60  
 ALTER SERVER 64  
 ALTER SERVICE CLASS 67  
 ALTER TABLE 74  
 ALTER TABLESPACE 123

## SQL ステートメント (続き)

ALTER THRESHOLD 133  
 ALTER TRUSTED CONTEXT 140  
 ALTER TYPE (構造化) 149  
 ALTER USER MAPPING 156  
 ALTER VIEW 158  
 ALTER WORK ACTION SET 160  
 ALTER WORK CLASS SET 173  
 ALTER WORKLOAD 179  
 ALTER XSROBJECT 187  
 ASSOCIATE LOCATORS 188  
 AUDIT 190  
 BEGIN DECLARE SECTION 194  
 CALL 195  
 CLOSE 203  
 COMMENT 205  
 COMMIT 218  
 CONNECT (タイプ 1) 238  
 CONNECT (タイプ 2) 246  
 CONTINUE、例外への応答 1017  
 CREATE ALIAS 254  
 CREATE AUDIT POLICY 256  
 CREATE BUFFERPOOL 259  
 CREATE DATABASE PARTITION GROUP 263  
 CREATE EVENT MONITOR 265  
 CREATE FUNCTION MAPPING 367  
 CREATE FUNCTION (OLE DB 外部表) 335  
 CREATE FUNCTION (SQL スカラー、表、または行) 356  
 CREATE FUNCTION (外部スカラー) 288  
 CREATE FUNCTION (外部表) 315  
 CREATE FUNCTION (ソース派生またはテンプレート)  
 344  
 CREATE FUNCTION (ソース派生) 344  
 CREATE FUNCTION、概要 287  
 CREATE HISTOGRAM TEMPLATE 371  
 CREATE INDEX 373  
 CREATE INDEX EXTENSION 390  
 CREATE METHOD 397  
 CREATE NICKNAME 403  
 CREATE NODEGROUP (CREATE DATABASE PARTITION  
 GROUP を参照) 263  
 CREATE PROCEDURE 416  
 CREATE PROCEDURE (SQL) 436  
 CREATE PROCEDURE (外部) 416  
 CREATE PROCEDURE (ソース派生) 432  
 CREATE ROLE 443  
 CREATE SCHEMA 443  
 CREATE SECURITY LABEL 449  
 CREATE SECURITY LABEL COMPONENT 446  
 CREATE SECURITY POLICY 450  
 CREATE SEQUENCE 452  
 CREATE SERVER 465  
 CREATE SERVICE CLASS 457  
 CREATE TABLE 469  
 CREATE TABLESPACE 542  
 CREATE THRESHOLD 557

## SQL ステートメント (続き)

CREATE TRANSFORM 566  
 CREATE TRIGGER 570  
 CREATE TRUSTED CONTEXT 583  
 CREATE TYPE MAPPING 624  
 CREATE TYPE (構造化) 598  
 CREATE TYPE (特殊) 592  
 CREATE TYPE (配列) 590  
 CREATE USER MAPPING 630  
 CREATE VARIABLE 632  
 CREATE VIEW 635  
 CREATE WORK ACTION SET 650  
 CREATE WORK CLASS SET 658  
 CREATE WORKLOAD 664  
 CREATE WRAPPER 672  
 DECLARE CURSOR 673  
 DECLARE GLOBAL TEMPORARY TABLE 680  
 DELETE 688  
 DESCRIBE 695  
 DESCRIBE INPUT 695  
 DESCRIBE OUTPUT 698  
 DISCONNECT 703  
 DROP 705  
 DROP TRANSFORM 705  
 END DECLARE SECTION 740  
 EXECUTE 743  
 EXECUTE IMMEDIATE 741  
 EXPLAIN 750  
 FETCH 755  
 FLUSH EVENT MONITOR 758  
 FLUSH OPTIMIZATION PROFILE CACHE 759  
 FLUSH PACKAGE CACHE 761  
 FREE LOCATOR 764  
 GRANT (SETSESSIONUSER 特権) 798  
 GRANT (XSR オブジェクト特権) 811  
 GRANT (グローバル変数特権) 775  
 GRANT (サーバー特権) 796  
 GRANT (索引特権) 777  
 GRANT (シーケンス特権) 794  
 GRANT (スキーマ特権) 789  
 GRANT (セキュリティ・ラベル) 791  
 GRANT (データベース権限) 768  
 GRANT (ニックネーム特権) 802  
 GRANT (パッケージ特権) 779  
 GRANT (ビュー特権) 802  
 GRANT (表スペース特権) 800  
 GRANT (表特権) 802  
 GRANT (免除) 773  
 GRANT (ルーチン特権) 785  
 GRANT (ロール) 782  
 GRANT (ワークロード特権) 809  
 INCLUDE 814  
 INSERT 815  
 LOCK TABLE 827  
 MERGE 830  
 OPEN 839

## SQL ステートメント (続き)

PREPARE 844  
 REFRESH TABLE 853  
 RELEASE SAVEPOINT 858  
 RELEASE (接続) 856  
 RENAME 860  
 RENAME TABLESPACE 858  
 REVOKE (SETSESSIONUSER 特権) 894  
 REVOKE (XSR オブジェクト特権) 904  
 REVOKE (グローバル変数特権) 873  
 REVOKE (サーバー特権) 892  
 REVOKE (索引特権) 875  
 REVOKE (シーケンス特権) 890  
 REVOKE (スキーマ特権) 886  
 REVOKE (セキュリティ・ラベル) 888  
 REVOKE (データベース権限) 867  
 REVOKE (ニックネーム特権) 897  
 REVOKE (パッケージ特権) 877  
 REVOKE (ビュー特権) 897  
 REVOKE (表スペース特権) 895  
 REVOKE (表特権) 897  
 REVOKE (免除) 871  
 REVOKE (ルーチン特権) 882  
 REVOKE (ロール) 880  
 REVOKE (ワークロード特権) 903  
 ROLLBACK 905  
 ROLLBACK TO SAVEPOINT 905  
 SAVEPOINT 908  
 SELECT 913  
 SELECT INTO 911  
 SET COMPILATION ENVIRONMENT 913  
 SET CONNECTION 914  
 SET CONSTRAINTS 949  
 SET CURRENT DECFLOAT ROUNDING MODE 916  
 SET CURRENT DEFAULT TRANSFORM GROUP 917  
 SET CURRENT DEGREE 918  
 SET CURRENT EXPLAIN MODE 920  
 SET CURRENT EXPLAIN SNAPSHOT 923  
 SET CURRENT FEDERATED ASYNCHRONY 925  
 SET CURRENT FUNCTION PATH 970  
 SET CURRENT IMPLICIT XMLPARSE OPTION 926  
 SET CURRENT ISOLATION 927  
 SET CURRENT MAINTAINED TABLE TYPES FOR  
 OPTIMIZATION 930  
 SET CURRENT MDC ROLLOUT MODE 932  
 SET CURRENT OPTIMIZATION PROFILE 933  
 SET CURRENT PACKAGESET 940  
 SET CURRENT PATH 970  
 SET CURRENT QUERY OPTIMIZATION 942  
 SET CURRENT REFRESH AGE 944  
 SET ENCRYPTION PASSWORD 946  
 SET EVENT MONITOR STATE 947  
 SET INTEGRITY 949  
 SET PASSTHRU 968  
 SET PATH 970  
 SET ROLE 972

## SQL ステートメント (続き)

SET SCHEMA 973  
 SET SERVER OPTION 975  
 SET SESSION AUTHORIZATION 977  
 SET 変数 979  
 TRANSFER OWNERSHIP 989  
 UPDATE 1004  
 VALUES 1017  
 VALUES INTO 1015  
 WHENEVER 1017  
 WITH HOLD  
     カーソル属性 673  
 SQL プロシージャ  
     条件処理ルーチン  
         宣言 229  
     動的コンパウンド・ステートメント 221  
     プロシージャ・コンパウンド・ステートメント 229  
     変数 221, 229  
     CASE ステートメント 201  
     DECLARE ステートメント 221, 229  
     FOR ステートメント 761  
     GET DIAGNOSTICS ステートメント 765  
     GOTO ステートメント 767  
     IF ステートメント 812  
     ITERATE ステートメント 824  
     LEAVE ステートメント 826  
     LOOP ステートメント 828  
     REPEAT ステートメント 862  
     RESIGNAL ステートメント 863  
     RETURN ステートメント 865  
     SIGNAL ステートメント 986  
     WHILE ステートメント 1018  
 SQL 変数 221, 229  
 SQL 戻りコード 9  
 SQL92 標準  
     動的 SQL の規則 973  
 SQLCA (SQL 連絡域)  
     UPDATE によって変更される項目 1004  
 SQLCA 構造  
     概要 9  
 SQLCA 節  
     INCLUDE ステートメント 814  
 SQLCODE  
     説明 9  
 SQLDA (SQL 記述子域)  
     ホスト変数の記述、OPEN ステートメント 839  
     DESCRIBE の必須の変数 695, 698  
     FETCH ステートメント 755  
 SQLDA 節  
     INCLUDE ステートメント 814  
 SQLERROR 節  
     WHENEVER ステートメント 1017  
 SQLSTATE  
     説明 9  
 SQL/XML  
     CREATE INDEX ステートメント 373

## static SQL

- ステートメント 9
- 呼び出し 9
- DECLARE CURSOR ステートメント 9
- FETCH ステートメント 9
- OPEN ステートメント 9
- SELECT 9

## STAY RESIDENT

- CREATE FUNCTION (外部スカラー) ステートメント 288
- CREATE FUNCTION (外部表) ステートメント 315
- CREATE PROCEDURE ステートメント 416, 436

## SYNONYM 節

- ALIAS 節の代わり 705

## system-containers

- CREATE TABLESPACE ステートメント 542

## T

### TABLE HIERARCHY 節

- DROP ステートメント 705

### TABLE 節

- COMMENT ステートメント 205
- CREATE FUNCTION (外部表) ステートメント 315
- DROP ステートメント 705

### TABLESPACE 節

- COMMENT ステートメント 205

### TIME データ・タイプ

- CREATE TABLE ステートメント 469

### TIMESTAMP データ・タイプ

- CREATE TABLE ステートメント 469

### TO 節

- GRANT ステートメント

- 索引特権 777
- スキーマ特権 789
- データベース権限 768
- パッケージ特権 779
- 表特権 802

- TRANSFER OWNERSHIP ステートメント 989

### TRIGGER 節

- COMMENT ステートメント 205

### TYPE 節

- COMMENT ステートメント 205
- DROP ステートメント 705

## U

### UNDER 節

- CREATE VIEW ステートメント 635

### UNIQUE 節

- ALTER TABLE ステートメント 74
- CREATE INDEX ステートメント 373
- CREATE TABLE ステートメント 469

### UPDATE ステートメント

- 行全選択 1004
- 説明 1004

### UPDATE 節

- GRANT ステートメント (表、ビュー、またはニックネーム) 802

- REVOKE ステートメント、特権の取り消し 897

### USING DESCRIPTOR 節

- OPEN ステートメント 839

### USING 節

- CREATE INDEX ステートメント 373
- FETCH ステートメント 755
- OPEN ステートメント、ホスト変数のリスト 839

## V

- VALUES INTO ステートメント 1015

- VALUES ステートメント 1017

### VALUES 節

- 値の数の規則 815
- ロード、1 行の 815

### VARCHAR データ・タイプ

- CREATE TABLE ステートメント 469

### VARIANT

- CREATE TYPE (構造化) ステートメント 598

### VIEW HIERARCHY 節

- DROP ステートメント 705

### VIEW 節

- CREATE VIEW ステートメント 635
- DROP ステートメント 705

### Visual Explain

- チュートリアル 1029

## W

### WHENEVER ステートメント

- 制御の流れの変更 9
- 説明 1017

- WHENEVER ステートメントの SQLWARNING 節 1017

### WHERE 節

- DELETE ステートメント 688
- UPDATE ステートメント 1004

### WHILE ステートメント

- 説明 1018

## X

### XML

- CREATE INDEX ステートメント 373

### XML データ

- CREATE INDEX ステートメント 373

### XML データに対する索引

- 構文とオプションの説明 373

- CREATE INDEX ステートメント 373

### XML 列

- CREATE INDEX ステートメント 373





Printed in Japan

SC88-4437-02



日本アイ・ビー・エム株式会社  
〒106-8711 東京都港区六本木3-2-12

Spine information:

**DB2 Version 9.5 for Linux, UNIX, and Windows**

**SQL リファレンス 第 2 巻**

