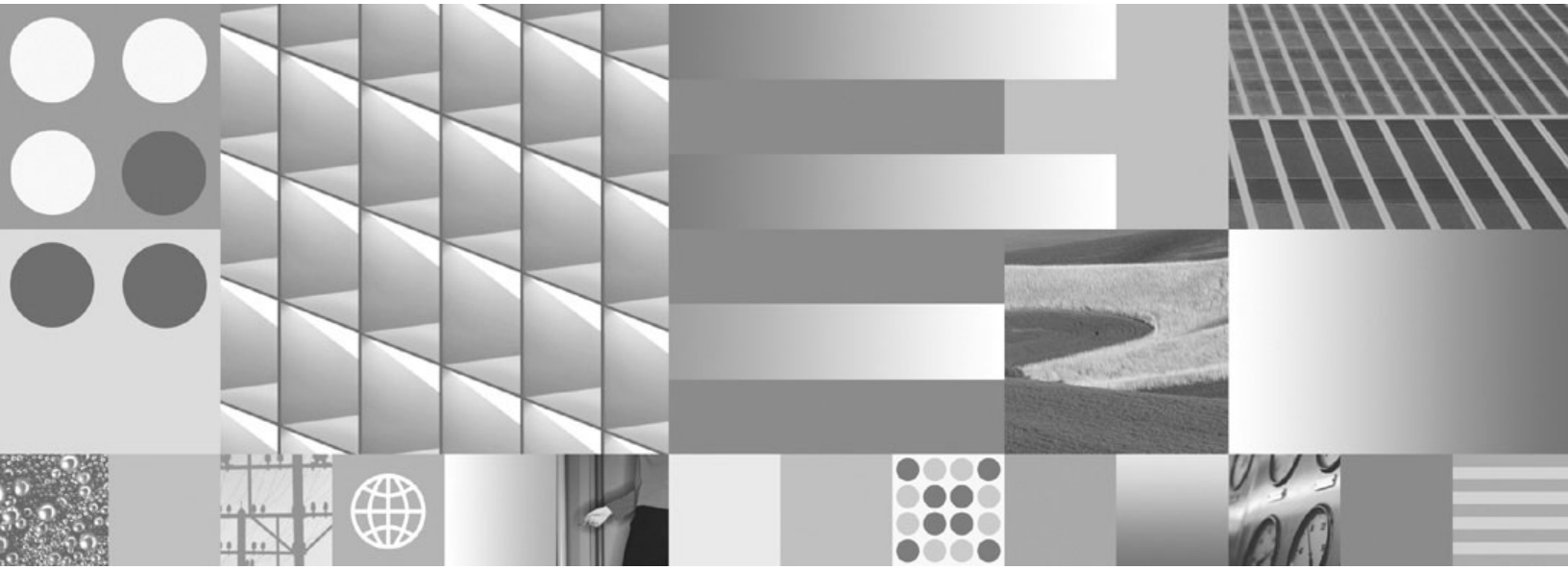
DB2 Version 9.5
for Linux, UNIX, and Windows

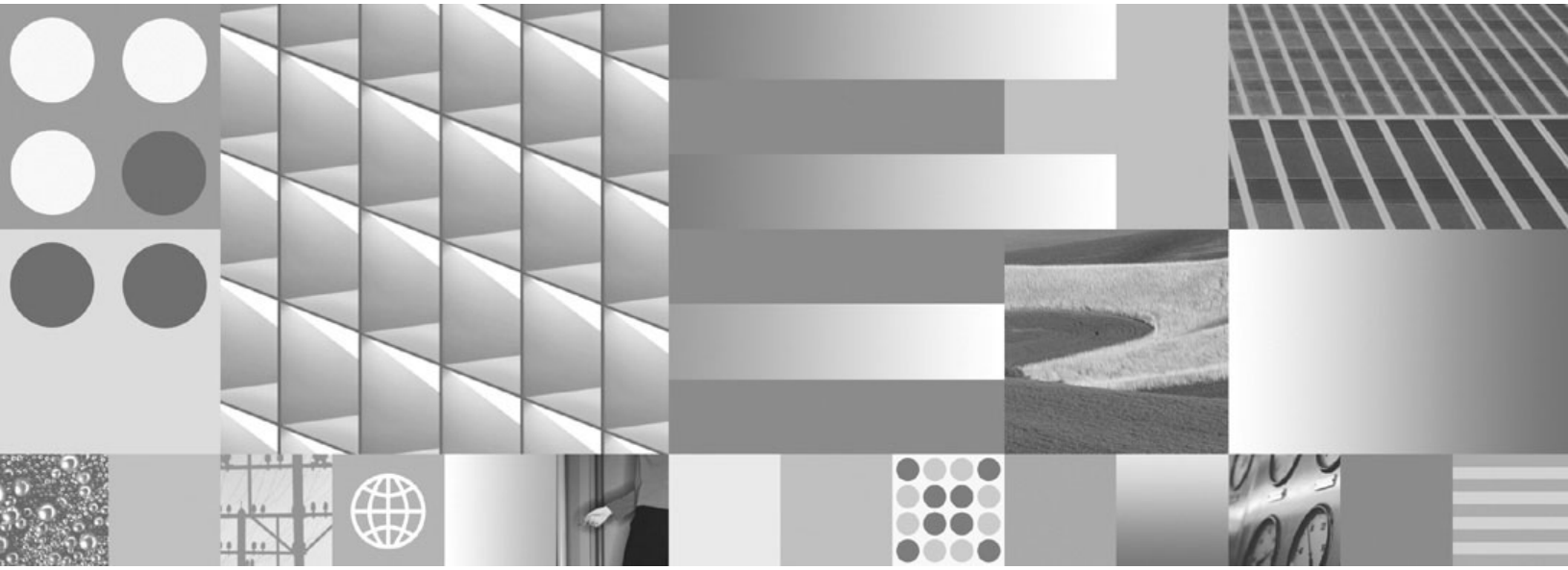


データ移動ユーティリティー ガイドおよびリファレンス

最終更新: 2009 年 4 月

SC88-4421-02
(英文原典：SC23-5847-02)

DB2 Version 9.5
for Linux, UNIX, and Windows



データ移動ユーティリティー ガイドおよびリファレンス

最終更新: 2009 年 4 月

SC88-4421-02
(英文原典：SC23-5847-02)

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、531 ページの『付録 F. 特記事項』に記載されている情報をお読みください。

原典： SC23-5847-02
DB2 Version 9.5
for Linux, UNIX, and Windows
Data Movement Utilities Guide and Reference
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

目次

本書について

本書では、以下の DB2® Database for Linux®, UNIX®, and Windows® データ移動ユーティリティーについて説明し、それらの使用法を示しています。

- エクスポートおよびインポート

  エクスポートおよびインポート・ユーティリティーは、表またはビューと、他のデータベースまたは表計算プログラムとの間、DB2 データベースどうしの間、および DB2 データベースと DB2® Connect™ を使用するホスト・データベースとの間で、データの移動を実行します。エクスポート・ユーティリティーは、データベースのデータをオペレーティング・システム・ファイルに移動します。それらのファイルを使用して、エクスポートされたデータを別のデータベースにインポートまたはロードすることができます。

- ロード

  ロード・ユーティリティーは、表の中へのデータの移動、既存の索引の拡張、および統計の生成を行います。大量のデータを扱う場合は、インポート・ユーティリティーよりロード・ユーティリティーのほうがデータを高速で移動できます。エクスポート・ユーティリティーを使用してエクスポートされたデータは、ロード・ユーティリティーを使用してロードすることができます。

  ロード・ユーティリティーをパーティション・データベース環境で使用する場合、大量のデータを分散して、複数の異なるデータベース・パーティションにロードすることができます。

データ移動オプションの完全なリストについては、DB2 V9.5 で使用可能なデータ移動オプションを参照してください。

第 1 部 データ移動のユーティリティーおよびリファレンス

第 1 章 DB2 V9.5 で使用可能なデータ移動オプション

DB2 V9.5 で使用可能なデータ移動オプションにはさまざまなものがあります。以下の表には、使用可能なデータ移動ツールおよびユーティリティーの概要が示されています。この表をガイドとして使用し、要件に合った最適なデータ移動オプションを判別する上で役立ててください。

表 1. DB2 V9.5 で使用可能なデータ移動オプション

ユーティリティー名	**ロード・ユーティリティー**
目的	新しく作成した表やすでにデータが入っている表に、大量のデータを効率よく移動します。
クロスプラットフォームの互換性	はい
最良事例の使用法	このユーティリティーは、パフォーマンスが最大の関心事である場合に最も適しています。このユーティリティーは、インポート・ユーティリティーの代わりに使用できます。これは、SQL の INSERTS を使用せず、フォーマット設定されたページを直接データベースに書き込むため、インポート・ユーティリティーよりも高速です。またロード・ユーティリティーには、データをログに記録しないオプションや、ロードされたデータのコピーを保管するために COPY オプションを使用するオプションがあります。ロード操作では、SMP および MPP 環境の CPU やメモリーなどのリソースを、十分に活用することができます。
参照	データのロード

ユーティリティー名	**db2move**
目的	db2move ユーティリティーを COPY オプションを設定して使用すると、スキーマ・テンプレート (データが入っていてもいなくても) をソース・データベースからターゲット・データベースにコピーしたり、スキーマ全体をソース・データベースからターゲット・データベースに移動することができます。 db2move ユーティリティーに IMPORT または EXPORT オプションを設定して使用すると、DB2 データベース間で大量の表を移動できるようになります。
クロスプラットフォームの互換性	はい

ユーティリティー名	db2move
最良事例の使用法	COPY オプションを設定して使用する場合、ソースとターゲットのデータベースは異なっていなければなりません。 COPY オプションは、スキーマ・テンプレートの作成時に便利です。クロスプラットフォームのバックアップおよびリストア操作のサポートがない場合には、データベースのクローン作成に IMPORT または EXPORT オプションを使用します。 IMPORT および EXPORT オプションは、db2look コマンドと一緒に使用します。
参照	• データ・サーバー、データベース、およびデータベース・オブジェクトのガイドの『スキーマのコピー』 • インポート表の再作成

ユーティリティー名	インポート・ユーティリティー
目的	外部ファイルから表、階層、ビュー、またはニックネームにデータを挿入します。
クロスプラットフォームの互換性	はい
最良事例の使用法	インポート・ユーティリティーは、以下の状態では、ロード・ユーティリティーの代わりに使用することができます。 • ターゲット表がビューである • ターゲット表に制約があり、ターゲット表を SET INTEGRITY ペンディング状態にしない • ターゲット表にトリガーがあり、それらを起動したい
参照	データのインポート

ユーティリティー名	エクスポート・ユーティリティー
目的	データベースから、いくつかある外部ファイル・フォーマットのいずれかにデータをエクスポートします。そうすると、後でデータをインポートまたはロードできます。
クロスプラットフォームの互換性	はい
最良事例の使用法	このユーティリティーは、データをさらに処理するかデータを別の表に移動するために外部ファイルにデータを保管する場合に、最も適しています。代わりに High Performance Unload (HPU) を使用できますが、別個に購入する必要があります。エクスポートは XML 列をサポートします。
参照	データのエクスポート

ユーティリティー名	ADMIN_COPY_SCHEMA プロシージャー
目的	1 つのスキーマ内のすべてのオブジェクトのコピーを作成し、それらのオブジェクトを新しいスキーマに再作成できるようにします。このコピー操作は、データベース内のデータの有無に関わらず実行できます。
クロスプラットフォームの互換性	はい
最良事例の使用法	このユーティリティーは、スキーマ・テンプレートの作成時に便利です。これは、ソース・スキーマの動作に影響を与えずにスキーマを試す場合 (例えば、新規索引をテストする場合) にも便利です。 ADMIN_COPY_SCHEMA プロシージャーと db2move ユーティリティーの主な違いには次のものがあります。 • ADMIN_COPY_SCHEMA プロシージャーは単一のデータベース上で使用されますが、db2move ユーティリティーは複数のデータベース間で使用されます。 • db2move ユーティリティーは、表や索引などの物理オブジェクトを作成できない場合に呼び出されると、失敗します。ADMIN_COPY_SCHEMA プロシージャーは、エラーをログに記録して続行します。 • ADMIN_COPY_SCHEMA プロシージャーは、「カーソルからロード」を使用してデータを 1 つのスキーマから別のスキーマに移動します。 db2move ユーティリティーは、「カーソルからロード」に似たリモート・ロードを使用します。これはデータをソース・データベースからプルします。
参照	データ・サーバー、データベース、およびデータベース・オブジェクトのガイドの『スキーマのコピー』

ユーティリティー名	リストア・ユーティリティー (REDIRECT オプションと GENERATE SCRIPT オプションを設定)
目的	既存のバックアップ・イメージにあるスクリプトを使用して、データベース全体を 1 つのシステムから別のシステムにコピーします。
クロスプラットフォームの互換性	制限があります。『参照』をご覧ください。
最良事例の使用法	このユーティリティーは、バックアップ・イメージが存在する場合に最も適しています。

ユーティリティー名	リストア・ユーティリティー (REDIRECT オプションと GENERATE SCRIPT オプションを設定)
参照	• *データ・リカバリーと高可用性 ガイド*およびリファレンスの『自動生成スクリプトを使用したリダイレクト・リストアの実行』 • *データ・リカバリーと高可用性 ガイド*およびリファレンスの『異なるオペレーティング・システムおよびハードウェア・プラットフォーム間のバックアップおよびリストア操作』

ユーティリティー名	db2relocatedb - データベースの再配置コマンド
目的	データベースを名前変更したり、1 つのデータベースまたはデータベースの一部を同じシステムまたは別のシステムに再配置します。
クロスプラットフォームの互換性	いいえ
最良事例の使用法	• このユーティリティーは、バックアップとリストアに時間がかかる場合に使用できます。 • このユーティリティーは、バックアップとリストアを使用してデータベースのコピーを移動または作成する代わりに使用できます。 • これは、テストなどの代替環境のためにデータベースのクローン作成を行うための簡単な方法としても使用されます。 • これは、表スペース・コンテナーをストレージ・デバイスの新しいセットに移動するために使用できます。
参照	コマンド・リファレンスの『db2relocatedb - データベースの再配置コマンド』

ユーティリティー名	スプリット・ミラー
目的	クローン・データベース、スタンバイ・データベース、またはバックアップ・データベースを作成します。
クロスプラットフォームの互換性	いいえ

ユーティリティー名	スプリット・ミラー
最良事例の使用法	• 主データベースで障害が発生した場合のダウン時間を短縮するためにスタンバイ・システムを作成します。 • バックアップ操作を、有効な実動マシンから切り離して分割データベースに移動します。 • テストなどの代替環境のためにデータベースのクローン作成を行うための簡単な方法として使用されます。
考慮事項	• データベースの分割バージョンでは、DMS 表スペースのみをバックアップできます。 • 通常、ストレージ・システムに搭載されている何らかのフラッシュ・コピー・テクノロジーと一緒に使用されます。 • 代わりに、データベースが中断した時にファイルのコピーを実行するという方法がありますが、この場合、データベースのストレージの量が倍になります。
参照	*データ・リカバリーと高可用性 ガイド*およびリファレンスの『オンライン・スプリット・ミラーおよびサスペンド入出力サポートによる高可用性』

第 2 章 エクスポート・ユーティリティー

エクスポート・ユーティリティーの概要

エクスポート・ユーティリティーは、SQL の SELECT ステートメントまたは XQuery ステートメントを使用してデータを抽出し、その情報をファイルに格納するためのユーティリティーです。出力ファイルを使用すれば、後にインポートやロードを利用してデータの移動をすることも、分析のためにデータをアクセス可能な状態にしておくこともできます。

エクスポート・ユーティリティーは、シンプルで柔軟性の高いデータ移動ユーティリティーです。このユーティリティーを実行するには、コントロール・センターを使用するか、CLP で EXPORT コマンドを実行するか、ADMIN_CMD ストアード・プロシージャーを呼び出すか、ユーザー・アプリケーションで db2Export API を呼び出します。

基本的なエクスポート操作で必須の項目は、以下のとおりです。

- エクスポートしたデータを格納するためのオペレーティング・システム・ファイルのパスと名前
- 入力ファイル内のデータのフォーマット
  エクスポート操作の出力ファイルのデータ・フォーマットとしては、IXF、WSF、DEL がサポートされています。
- エクスポートするデータの指定
  ほとんどのエクスポート操作では、エクスポートのために取得するデータを指定した SELECT ステートメントを指定する必要があります。型付き表をエクスポートする場合は、SELECT ステートメントを明示的に実行する必要はありません。指定しなければならないのは、階層内の副表のトラバース順序だけです。

IXF 形式のデータを移動する必要がある場合は、DB2 Connect でエクスポート・ユーティリティーを使用できます。

追加のオプション
エクスポート操作をカスタマイズするために使用できるパラメーターがいくつかあります。ファイル・タイプ修飾子には、データのフォーマット、日付と時刻のスタンプ、コード・ページを変更するためのオプションや、特定のデータ・タイプを別個のファイルに書き出すためのオプションなどが用意されています。**METHOD** パラメーターを使用すれば、エクスポート・データで使用する列名を変更して指定できます。

XML データ・タイプの列が 1 つ以上含まれている表からデータをエクスポートすることも可能です。エクスポート文書を格納する方法の詳細を指定するには、**XMLFILE**、**XML TO**、**XMLSAVESCHEMA** の各パラメーターを使用します。

エクスポート・ユーティリティーのパフォーマンスを改善するための方法もいくつかあります。エクスポート・ユーティリティーは、組み込み SQL アプリケーションであり、内部で SQL フェッチを実行するので、SQL 操作に当てはまる最適化の手法は、エクスポート・ユーティリティーにも当てはまります。例えば、大きなバ

ッファー・プール、索引作成、ソート・ヒープなどを活用できます。さらに、出力ファイルに関するデバイスの競合を最小化するために、コンテナーやロギング用デバイスとは異なるデバイスに出力ファイルを配置する、という方法もあります。

メッセージ・ファイル
エクスポート・ユーティリティーは、エラー・メッセージ、警告メッセージ、通知メッセージを標準の ASCII テキスト・メッセージ・ファイルに書き込みます。CLP 以外のすべてのインターフェースでは、そのファイルの名前を事前に **MESSAGES** パラメーターで指定しておく必要があります。 CLP を使用する場合にメッセージ・ファイルを指定しなければ、エクスポート・ユーティリティーは、メッセージを標準出力に書き込みます。

エクスポート・ユーティリティーを使用するために必要な特権と権限

特権は、データベース・リソースの作成、更新、削除、アクセスを可能にするためのものです。権限レベルは、データベース・マネージャーの保守とユーティリティーの高水準操作に各種の特権を対応付けるための手段になります。

特権と権限を組み合わせて、データベース・マネージャーとデータベース・オブジェクトに対するアクセスを制御します。アクセスできるのは、適切な許可 (必要な特権または権限) が与えられているオブジェクトだけになります。

SYSADM 権限または DBADM 権限か、エクスポート操作にかかわっている各表またはビューに対する CONTROL 特権または SELECT 特権が必要です。

LBAC で保護されたデータをエクスポートする場合は、セッション許可 ID に、エクスポートする行または列に対する読み取り許可が必要です。保護された行の読み取り許可を持たないセッション許可 ID では、その行をエクスポートできません。保護された列が SELECT ステートメントに含まれていて、セッション許可 ID にその列の読み取り許可がなければ、エクスポート・ユーティリティーは失敗し、エラー (SQLSTATE 42512) が返されます。

データのエクスポート

エクスポート・ユーティリティーを使用して、データベースからファイルにデータをエクスポートできます。そのファイルは、いずれかの外部ファイル形式になります。SQL SELECT ステートメントを提供するか、または型付き表の階層情報を提供することにより、エクスポートするデータを指定できます。

データベースからデータをエクスポートするには、SYSADM 権限、DBADM 権限、操作にかかわっているそれぞれの表またはビューに関する CONTROL 特権、SELECT 特権のいずれかが必要です。

エクスポート・ユーティリティーを実行するには、まずデータのエクスポート元になるデータベースに接続している (または、暗黙接続が可能な状態になっている) 必要があります。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。 Linux、UNIX、または Windows クライアントから Linux、UNIX、または Windows データベース・サーバーへのユーティリティーのアクセスは、DB2 Connect ゲートウェイまたはループバック環境を介してではなく、エンジンからの直接接続でなければなりません。

このユーティリティーは COMMIT ステートメントを実行するので、エクスポート・ユーティリティーの呼び出し前に COMMIT ステートメントまたは ROLLBACK ステートメントを実行して、すべてのトランザクションを完了し、すべてのロックを解放しておかなければなりません。アプリケーションが表にアクセスし、かつ切断するのに別々の接続を使用することに関する要件はありません。

エクスポート・ユーティリティーには、以下の制約事項が適用されます。

- このユーティリティーでは、構造化タイプ列を持つ表はサポートされません。

エクスポート・ユーティリティーを実行するには、コントロール・センターの「表のエクスポート」ノートブックを使用するか、アプリケーション・プログラミング・インターフェース (API) の db2Export を使用するか、コマンド行プロセッサー (CLP) で EXPORT コマンドを指定します。

「表のエクスポート」ノートブックの使用

「表のエクスポート」ノートブックを使用してデータをエクスポートするには、以下のようにします。

1. コントロール・センターから、「表」フォルダーまたは「ビュー」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
2. 対象となるフォルダーをクリックします。ウィンドウの右側部分 (コンテンツ・ペイン) に、既存の表またはビューがすべて表示されます。
3. コンテンツ・ペインで対象の表またはビューを右クリックし、ポップアップ・メニューから「エクスポート」を選択します。「表のエクスポート」がオープンします。

「表のエクスポート」についての詳細情報は、コントロール・センターのオンライン・ヘルプ機能を通して提供されます。

CLP による EXPORT コマンドの実行

シンプルなエクスポート操作の場合に指定する必要があるのは、SELECT ステートメントのターゲット・ファイル、ファイル形式、ソース・ファイルだけです。

CLP からデータをエクスポートするには、以下の EXPORT コマンドを入力します。

```
db2 export to filename of ixf select * from table
```

`filename` は、作成/エクスポートする出力ファイルの名前、`ixf` はファイル形式、`table` は、コピーするデータが含まれている表の名前です。

ただし、警告メッセージやエラー・メッセージを書き込むメッセージ・ファイルを指定することも可能です。そのためには、**MESSAGES** パラメーターとメッセージ・ファイル名 (この場合は `msg.txt`) を追加します。そのコマンドは、次のようになります。

```
db2 export to filename of ixf messages msgs.txt select * from table
```

構文と使用法の詳細については、「EXPORT コマンド」を参照してください。

XML データのエクスポート

XML データをエクスポートする際、結果として作成される QDM (XQuery データ・モデル) インスタンスは、エクスポートされるリレーショナル・データを含むメイン・データ・ファイルとは別のファイル (1 つまたは複数) に書き込まれます。XMLFILE オプションおよび XML TO オプションのどちらも指定されていない場合でも、そのようになります。デフォルトで、エクスポートされた QDM インスタンスはすべて同じ XML ファイルに連結されます。 XMLINSEPFILES ファイル・タイプ修飾子を使用して、各 QDM インスタンスが別のファイルに書き込まれるように指定できます。

ただし XML データは、メイン・データ・ファイル内で XML データ指定子 (XDS) によって表されます。 XDS は XDS という名前の XML タグによって表されるストリングであり、列内の実際の XML データについての情報を記述する属性が付随しています。そのような情報には、実際の XML データが含まれているファイルの名前、およびそのファイル内の XML データのオフセットおよび長さが含まれます。

エクスポートされた XML ファイルの宛先パスおよびベース名は、XML TO および XMLFILE オプションを使用して指定できます。 XML TO または XMLFILE オプションが指定されている場合、エクスポートされた XML ファイルの名前として XDS の FIL 属性に格納される名前の形式は xmlfilespec.xxx.xml です (xmlfilespec は XMLFILE オプションに指定された値、xxx はエクスポート・ユーティリティーによって生成された XML ファイルの順序番号)。そうでない場合、エクスポートされた XML ファイル名の形式は exportfilename.xxx.xml となります (exportfilename は EXPORT コマンドのために指定されるエクスポートされた出力ファイルの名前、xxx はエクスポート・ユーティリティーによって生成された XML ファイルの順序番号)。

デフォルトで、エクスポートされた XML ファイルはエクスポートされたデータ・ファイルのパスに書き込まれます。エクスポートされた XML ファイルのデフォルトのベース名は、エクスポートされたデータ・ファイル名に 3 桁の順序番号、および .xml 拡張子を付加したものです。

例

以下の例では、4 つの列および 2 つの行を含む表 USER.T1 を想定します。

```
C1 INTEGER
C2 XML
C3 VARCHAR(10)
C4 XML
```

表 2. USER.T1

C1	C2	C3	C4
2	<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</ to><from> Me</ from><heading>note1</heading> <body>Hello World!</body></note>	'char1'	<?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him</ to><from> Her</ from><heading>note2</heading>< body>Hello World!</body></note>

表 2. USER.T1 (続き)

C1	C2	C3	C4
4	NULL	'char2'	?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">to>Us</ to><from> Them</ from><heading>note3</heading> <body>Hello World!</body></note>

例 1

以下のコマンドは、USER.T1 の内容を Delimited ASCII (DEL) 形式でファイル "/mypath/t1export.del" にエクスポートします。 XML TO および XMLFILE オプションが指定されていないので、列 C2 および C4 に含まれる XML 文書はメインのエクスポート・ファイル "/mypath" と同じパスに書き込まれます。これらのファイルのベース名は、"t1export.del.xml" です。 XMLSAVESCHEMA オプションは、XML スキーマ情報がエクスポート手順の実行中に保存されることを示します。

```
EXPORT TO /mypath/t1export.del OF DEL XMLSAVESCHEMA SELECT * FROM USER.T1
```

エクスポート・ファイル "/mypath/t1export.del" には、以下が含まれます。

```
2,"<XDS FIL='t1export.del.001.xml' OFF='0' LEN='144' />","char1",
"<XDS FIL='t1export.del.001.xml' OFF='144' LEN='145' />"
4,,"char2","<XDS FIL='t1export.del.001.xml' OFF='289'
LEN='145' SCH='S1.SCHEMA_A' />"
```

エクスポートされた XML ファイル "/mypath/t1export.del.001.xml" には、以下が含まれます。

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
   <from>Me</from><heading>note1</heading><body>Hello World!</body>
   </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him
   </to><from>Her</from><heading>note2</heading><body>Hello World!
   </body></note><?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">
   <to>Us</to><from>Them</from>heading>note3</heading><body>
   Hello World!</body></note>
```

例 2

以下のコマンドは、USER.T1 の内容を DEL 形式でファイル "t1export.del" にエクスポートします。列 C2 および C4 に含まれる XML 文書は、パス "/home/user/xmlpath" に書き込まれます。 XML ファイルはベース名 "xmldocs" を使用して名前が付けられ、複数のエクスポートされた XML 文書が同じ XML ファイルに書き込まれます。 XMLSAVESCHEMA オプションは、XML スキーマ情報がエクスポート手順の実行中に保存されることを示します。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs XMLSAVESCHEMA SELECT * FROM USER.T1
```

エクスポートされた DEL ファイル "/home/user/t1export.del" には、以下が含まれます。

```
2,"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='144' />","char1",
"<XDS FIL='xmldocs.001.xml' OFF='144' LEN='145' />"
4,,"char2","<XDS FIL='xmldocs.001.xml' OFF='289'
LEN='145' SCH='S1.SCHEMA_A' />"
```

エクスポートされた XML ファイル "/home/user/xmlpath/xmldocs.001.xml" には、以下のものが含まれます。

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
   <from>Me</from><heading>note1</heading><body>Hello World!</body>
   </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00">
   <to>Him</to><from>Her</from><heading>note2</heading><body>
   Hello World!</body></note><?xml version="1.0" encoding="UTF-8" ?>
   <note time="14:00:00"><to>Us</to><from>Them</from><heading>
   note3</heading><body>Hello World!</body></note>
```

例 3

以下のコマンドは例 2 と似ていますが、それぞれのエクスポートされた XML 文書が別の XML ファイルに書き込まれることが異なります。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs MODIFIED BY XMLINSEPFILES XMLSAVESCHEMA
SELECT * FROM USER.T1
```

エクスポート・ファイル "/mypath/t1export.del" には、以下が含まれます。

```
2,"<XDS FIL='xmldocs.001.xml' />","char1","XDS FIL='xmldocs.002.xml' />"
4,,"char2","<XDS FIL='xmldocs.004.xml' SCH='S1.SCHEMA_A' />"
```

エクスポートされた XML ファイル "/home/user/xmlpath/xmldocs.001.xml" には、以下のものが含まれます。

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
   <from>Me</from><heading>note1</heading><body>Hello World!</body>
   </note>
```

エクスポートされた XML ファイル "/home/user/xmlpath/xmldocs.002.xml" には、以下のものが含まれます。

```
?xml version="1.0" encoding="UTF-8" ?>note time="13:00:00">to>Him/to>
   from>Her/from>heading>note2/heading>body>Hello World!/body>
   /note>
```

エクスポートされた XML ファイル "/home/user/xmlpath/xmldocs.004.xml" には、以下のものが含まれます。

```
<?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00"><to>Us</to>
   <from>Them</from><heading>note3</heading><body>Hello World!</body>
   </note>
```

例 4

次のコマンドは、 XQuery の結果を XML ファイルに書き込みます。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs MODIFIED BY XMLNODECLARATION select
xmlquery( '$m/note/from/text()' passing by ref c4 as "m" returning sequence)
   from USER.T1
```

エクスポートされた DEL ファイル "/mypath/t1export.del" には、以下が含まれます。

```
"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='3' />"
"<XDS FIL='xmldocs.001.xml' OFF='3' LEN='4' />"
```

エクスポートされた XML ファイル "/home/user/xmlpath/xmldocs.001.xml" には、以下のものが含まれます。

```
HerThem
```

注: 特定の XQuery の結果は、整形式 XML 文書を生成しません。そのため、上記でエクスポートされたファイルを XML 列に直接インポートすることはできません。

LBAC で保護されたデータのエクスポートに関する考慮事項

LBAC (ラベル・ベースのアクセス制御) で保護されたデータをエクスポートするときに、実際にエクスポートされるのは、LBAC 信用証明情報に読み取り権限があるデータに限られます。

LBAC 信用証明情報に行の読み取り権限がなければ、その行はエクスポートされませんが、エラーは返されません。LBAC 信用証明情報に列の読み取り権限がなければ、エクスポート・ユーティリティーは失敗し、エラー (SQLSTATE 42512) が返されます。

データ・タイプ DB2SECURITYLABEL の列の値は、区切り文字で囲まれた生データとしてエクスポートされます。元のデータに区切り文字が含まれていれば、区切り文字が二重になります。エクスポート値を構成するバイトにそれ以外の変更は加えられません。したがって、DB2SECURITYLABEL データが含まれているデータ・ファイルには、改行や用紙送りなどの出力不能の ASCII 文字が組み込まれる可能性があります。

データ・タイプ DB2SECURITYLABEL の列の値を理解できる形式でエクスポートするには、SELECT ステートメントで SECLABEL_TO_CHAR スカラー関数を使用して、その値をセキュリティー・ラベル・ストリング・フォーマットに変換できます。

例

以下の例では、出力を DEL 形式にして、ファイル myfile.del に書き込みます。データのエクスポート元は、以下のステートメントで作成された REPS という名前の表です。

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

次に示すのは、row_label 列の値をデフォルト形式でエクスポートする例です。

```
db2 export to myfile.del of del select * from reps
```

このデータ・ファイルの内容を読もうとしても、ほとんどのテキスト・エディターでは、あまり読みやすくない状態で表示されます。row_label 列の値には、いくつかの ASCII 制御文字が含まれている可能性が高いからです。

次に示すのは、row_label 列の値をセキュリティー・ラベル・ストリング・フォーマットでエクスポートする例です。

```
db2 export to myfile.del of del select SECLABEL_TO_CHAR
(row_label,'DATA_ACCESS_POLICY'), id, name from reps
```

この例で作成したデータ・ファイルの抜粋を以下に示します。セキュリティー・ラベルの形式が読みやすくなっています。

```
...
"Secret:():Epsilon 37", 2005, "Susan Liu"
"Secret:():(Epsilon 37,Megaphone,Cloverleaf)", 2006, "Johnny Cogent"
"Secret:():(Megaphone,Cloverleaf)", 2007, "Ron Imron"
...
```

表のエクスポートに関する考慮事項

標準的なエクスポート操作では、既存の表に挿入されているかロードされているデータを選択的に出力します。ただし、表全体をエクスポートして、後からインポート・ユーティリティーによって表を再作成することも可能です。

表をエクスポートするには、PC/IXF ファイル形式を指定する必要があります。そのようにして保管した表を索引と一緒に再作成するには、インポート・ユーティリティーを CREATE モードで使用します。ただし、以下のような条件が存在する場合は、エクスポート IXF ファイルに一部の情報が保管されません。

- 索引列名に 16 進値 0x2B または 0x2D が含まれる
- 表に XML 列が含まれる。
- 表がマルチディメンション・クラスター (MDC) 表である。
- 表に表パーティション・キーが含まれる。
- コード・ページ変換が原因で索引名が 128 バイトより大きくなっている。
- 表が保護されている。
- EXPORT コマンドに SELECT * FROM *tablename* 以外のアクション・ストリングが含まれている。
- エクスポート・ユーティリティーで **METHOD N** パラメーターが指定されている。

失われる表属性のリストについては、「表のインポートに関する考慮事項」を参照してください。保管されない情報がある場合は、表の再作成時に警告 SQL27984W が返されます。

注: インポートの CREATE モードは、非推奨になります。表のキャプチャーと再作成には、db2look ユーティリティーを使用してください。

索引情報

索引で指定した列名に「-」か「+」のいずれかの文字が含まれる場合は、索引情報は収集されず、警告 SQL27984W が戻されます。エクスポート・ユーティリティーは処理を完了し、エクスポート・データに影響はありません。ただし、索引情報は IXF ファイルに保管されません。したがって、db2look ユーティリティーを使用して、索引を別途作成する必要があります。

スペースに関する制限

エクスポートするデータが、エクスポート作業を行う OS のファイル・システムの使用可能領域を超えると、エクスポート操作は失敗します。その場合は、WHERE 節で条件を指定することにより、選択されるデータの量を制限して、エクスポート先のファイル・システムにうまく収まるようにしてください。すべてのデータをエクスポートするために、エクスポート・ユーティリティーを複数回実行することができます。

他のファイル形式の表
エクスポート時に IXF ファイル形式を使用しない場合は、出力ファイルにレコード・データは組み込まれますが、ターゲット表の記述は組み込まれません。表とデータを再作成するには、ターゲット表を作成してから、ロード・ユーティリティーまたはインポート・ユーティリティーを使用してその表にデータを取り込みます。元の表定義をキャプチャーし、対応するデータ定義言語 (DDL) を生成するには、db2look ユーティリティーを使用します。

型付き表のエクスポートに関する考慮事項

DB2 エクスポート・ユーティリティーを使用すれば、後からインポートできる状態で型付き表のデータを移動できます。エクスポートでは、特定の順序を指定し、中間のフラット・ファイルを作成することによって、型付き表の階層構造間でデータを移動します。

エクスポート・ユーティリティーでは、型付き表を操作するときに、出力ファイルに格納するデータを制御するために、ターゲット表の名前だけを指定しますが、オプションとして WHERE 節も指定できます。副選択ステートメントも記述できますが、そのためには、ターゲット表の名前と WHERE 節だけを指定します。階層をエクスポートするときに、全選択や SELECT ステートメントを指定することはできません。

トラバース順序を使用した階層の保存
型付き表は、階層になっている場合も可能です。階層間でデータを移動するには、いくつかの方法があります。

- 1 つの階層からそれと同一の階層への移動
- 1 つの階層から、より大きな階層のサブセクションへの移動
- 大きな階層のサブセクションから別の階層への移動

階層内の型の識別方法はデータベースによって違います。つまり、同じ型でもデータベースが異なると ID が違います。そのため、それらのデータベース間でデータを移動する場合、データを正しく移動するために同じ型のマッピングを実行する必要があります。

型付き表で使用するマッピングのことを*トラバース順序*といいます。つまり、階層内のすべてのスーパー表と副表を上から下へ、左から右へ進む順序です。エクスポート操作中にそれぞれの型の行が書き出される前に、ID が索引値に変換されます。この索引値は、1 から、階層内の関連する型の数までの範囲のいずれかの数になります。階層内を特定の順序 (トラバース順序) で移動する場合、それぞれの型に番号を付けることによって索引値が生成されます。図 1 は、以下のような 4 つの有効なトラバース順序のある階層を示しています。

- Person、Employee、Manager、Architect、Student
- Person、Student、Employee、Manager、Architect
- Person、Employee、Architect、Manager、Student
- Person、Student、Employee、Architect、Manager

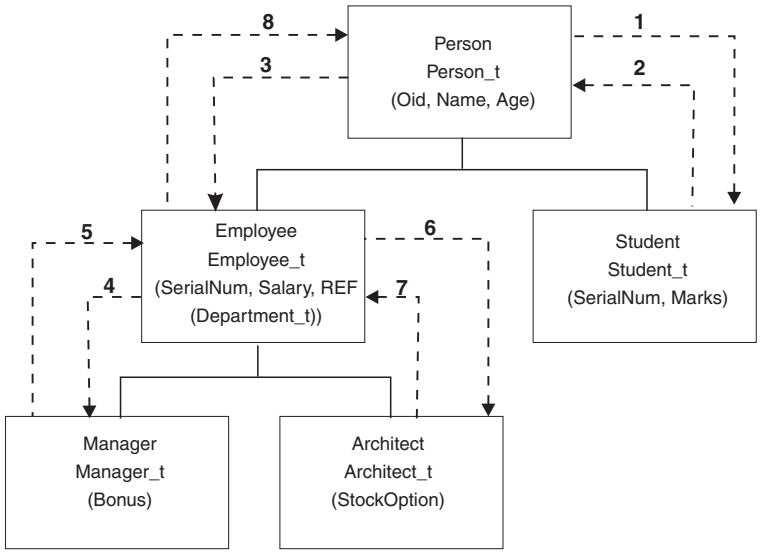


図 1. 階層の例

表階層間でデータを移動する場合、トラバース順序は他のデータとの相対関係でデータがどこに移動するかを決めるものとなるため重要です。トラバース順序には、*デフォルト*と*ユーザー指定*という 2 つのタイプがあります。

デフォルトのトラバース順序
デフォルトのトラバース順序では、関連するすべての型が、階層内の特定の開始点から到達可能な階層内の型をすべて参照します。デフォルト順序には階層内のすべての表が組み込まれており、それぞれの表の順序は OUTER 順序述部で使用されている方式に従ったものです。例えば、図 1 のデフォルトのトラバース順序は、点線で示されているとおり、Person、Student、Employee、Manager、Architect になります。

ファイル・フォーマットが違うと、デフォルトのトラバース順序の動作が違います。 PC/IXF ファイル・フォーマットにデータをエクスポートすると、関連のあるすべての型、それらの定義、および関連している表に関するレコードが作成されます。また、エクスポート・ユーティリティーによって索引値からそれぞれの表へのマッピングが完了します。 PC/IXF ファイル・フォーマットで作業をする場合は、デフォルトのトラバース順序を使用してください。

ASC、DEL、または WSF ファイル・フォーマットでは、ソースとターゲットの階層の構造が同じであっても、型付き行および型付き表の作成順が違う可能性があります。その場合、デフォルトのトラバース順序で階層を進むうちに時間の差が発生することになります。デフォルトのトラバース順序を使用する場合、ソースとターゲットのどちらにおいても、それぞれの型の作成日時が階層内を移動するためのトラバース順序を決定します。ソース階層およびターゲット階層の両方で、それぞれの型の作成順序が同じであること、およびソースとターゲットの構造が同じであることを確かめてください。この条件が満たされない場合は、ユーザー指定のトラバース順序を選択してください。

ユーザー指定のトラバース順序
ユーザー指定のトラバース順序では、使用する関連のある型をユーザーがトラバー

ス順序リストで定義します。デフォルトのトラバース順序では、階層内のすべての表がエクスポートされますが、この順序では、階層をトラバースする方法とエクスポートする副表の概略を指定します。

トラバース順序を定義するときには、開始点と階層を下るパスを指定しますが、副表は *pre-order* 方式でトラバースする必要があります。つまり、階層内のそれぞれの分岐の最下位にまで達しないと、新しい分岐のトラバースを開始できません。エクスポート・ユーティリティーは、指定されたトラバース順序がこの条件に違反していないかどうかを検査します。その条件を満たすための 1 つの方法は、階層の最上部 (ルート表) から下の階層 (副表) に進んで最も低い副表に達してから、そのスーパー表に戻り、次の「右端の」副表まで達したら、またそのスーパー表のさらに上に戻って、その副表を下っていき、このような過程を続けていく、ということです。

階層のトラバース順序を制御する場合は、エクスポートおよびインポート・ユーティリティーで必ず同じトラバース順序が使用されるようにしてください。

例 1

以下の例は、図 1 の階層構造に基づいています。階層全体をエクスポートするには、以下のコマンドを入力します。

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO entire_hierarchy.ixf OF IXF HIERARCHY STARTING Person
```

パラメーター **HIERARCHY STARTING** を Person に設定しているので、表 PERSON からデフォルトのトラバース順序が始まります。

例 2

階層全体をエクスポートするものの、20 歳以上の人のデータだけを抽出する場合は、以下のコマンドを入力します。

```
DB2 CONNECT TO Source_db
   DB2 EXPORT TO entire_hierarchy.del OF DEL HIERARCHY (Person,
      Employee, Manager, Architect, Student) WHERE Age>=20
```

パラメーター **HIERARCHY** を Person, Employee, Manager, Architect, Student に設定しているので、ユーザー指定のトラバース順序になっています。

ID 列のエクスポートに関する考慮事項

エクスポート・ユーティリティーでは、ID 列が含まれている表のデータをエクスポートできます。ただし、ID 列がある場合は、出力ファイルの形式に関する選択肢が限定されます。

エクスポート操作で指定する SELECT ステートメントが SELECT * FROM *tablename* という形式であり、そのステートメントで METHOD オプションを使用しない場合は、ID 列のプロパティーを IXF ファイルにエクスポートできます。IMPORT コマンドの REPLACE_CREATE オプションと CREATE オプションを使用すれば、ID 列のプロパティーを組み込んで表を再作成することが可能です。ただし、タイプ GENERATED ALWAYS の ID 列が含まれている表からエクスポート IXF ファイルを作成した場合は、インポート操作の実行時に `identityignore` ファ

イル・タイプ修飾子を指定しない限り、データ・ファイルを正常にインポートできません。そうでない場合は、すべての行がリジェクトされ、SQL3550W が生成されます。

注: IMPORT コマンドの CREATE オプションと REPLACE_CREATE オプションは非推奨で、将来のリリースでは廃止される可能性があります。

LOB のエクスポートに関する考慮事項

ラージ・オブジェクト (LOB) 列が含まれている表をエクスポートする場合は、1 つの LOB 値につき最大で 32 KB までのデータをエクスポートし、そのデータと残りの列データを同じファイルに格納する、というのがデフォルトのアクションになります。32 KB を超える LOB 値をエクスポートする場合は、切り捨てを避けるために、LOB データを別のファイルに書き込む必要があります。

LOB を独自のファイルに書き込むことを指定するには、lobsinfile ファイル・タイプ修飾子を使用します。この修飾子を指定すると、エクスポート・ユーティリティーは、LOBS TO 節で指定されているディレクトリーに LOB データを格納します。 LOBS TO または LOBFILE を使用すると、lobsinfile ファイル・タイプ修飾子が暗黙的にアクティブ化されます。デフォルトで、LOB 値は、エクスポート・リレーショナル・データが書き込まれるのと同じパスに書き込まれます。 LOBS TO オプションに 1 つ以上のパスが指定されていると、エクスポート・ユーティリティーは、各パスの間を循環し、正常な LOB 値をそれぞれ適切な LOB ファイルに書き込みます。さらに、LOBFILE オプションを使用して、出力 LOB ファイルの名前を指定することもできます。LOBFILE オプションが指定されている場合、lobfilename のフォーマットは lobfilespec.xxx.lob となります。ここで lobfilespec は LOBFILE オプションに指定された値であり、xxx はエクスポート・ユーティリティーによって生成された LOB ファイルのシーケンス番号です。それ以外の場合、lobfilename は exportfilename.xxx.lob のフォーマットになります。ここで、exportfilename は EXPORT コマンドに指定されたエクスポートされる出力ファイルの名前であり、xxx はエクスポート・ユーティリティーによって生成された LOB ファイルのシーケンス番号です。

デフォルトの動作では、1 つのファイルに LOB が書き込まれますが、個々の LOB を別々のファイルに格納するように指定することも可能です。エクスポート・ユーティリティーは、複数の LOB を 1 つのファイルに格納する動作を有効にするために、LOB ロケーション指定子 (LLS) を生成します。この LLS は、LOB データをファイル内のどこに格納するのかを指定したストリングであり、エクスポート出力ファイルに書き込まれます。LLS のフォーマットは、obfilename.ext.nnn.mmm/ です。ここで、lobfilename.ext は LOB の入ったファイルの名前、nnn はファイル内の LOB のオフセット (バイト単位)、mmm は LOB の長さ (バイト単位) です。例えば、LLS が db2exp.001.123.456/ である場合、これは、LOB がファイル db2exp.001 にあり、ファイル内の 123 バイトのオフセットで始まり、長さが 456 バイトであることを示します。 LLS で示されたサイズが 0 の場合、LOB の長さは 0 であると見なされます。長さが -1 の場合には、LOB は NULL と見なされ、オフセットおよびファイル名は無視されます。

個々の LOB データを同じファイルに連結しない場合は、lobsinsepfiles ファイル・タイプ修飾子を使用して、それぞれの LOB を別々のファイルに書き込むようにします。

注: IXF ファイル・フォーマットには、LOB 列がログ記録されるかどうかなどの、列の LOB オプションが保管されません。そのため、インポート・ユーティリティーでは、1 GB 以上の大きさに定義された LOB 列の入っている表を再作成できません。

例 1次の例では、LOB を 1 つの DEL ファイルにエクスポートする方法を示しています (エクスポート LOB ファイルのベース名として、lobs1 を指定します)。

```
db2 export to myfile.del of del lobs to mylobs/
   lobfile lobs1 modified by lobsinfile
   select * from emp_photo
```

例 2次の例では、LOB を 1 つの DEL ファイルにエクスポートし、それぞれの LOB 値を別々のファイルに書き込み、LOB ファイルを 2 つのディレクトリーに格納する方法を示しています。

```
db2 export to myfile.del of del
lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
select * from emp_photo
```

リファレンス - エクスポート

EXPORT

データベースから、いくつかある外部ファイル形式のいずれかにデータをエクスポートします。ユーザーは、SQL SELECT ステートメントによって、または型付き表の階層情報によってエクスポートするデータを指定します。

26 ページの『エクスポート・ユーティリティーのファイル・タイプ修飾子』へのクイック・リンク。

許可

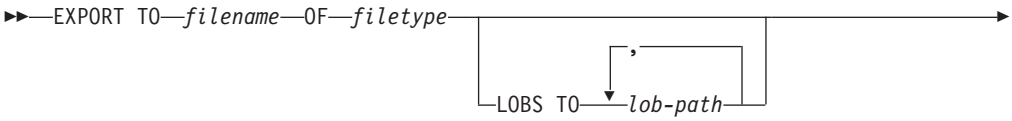
以下のいずれか。

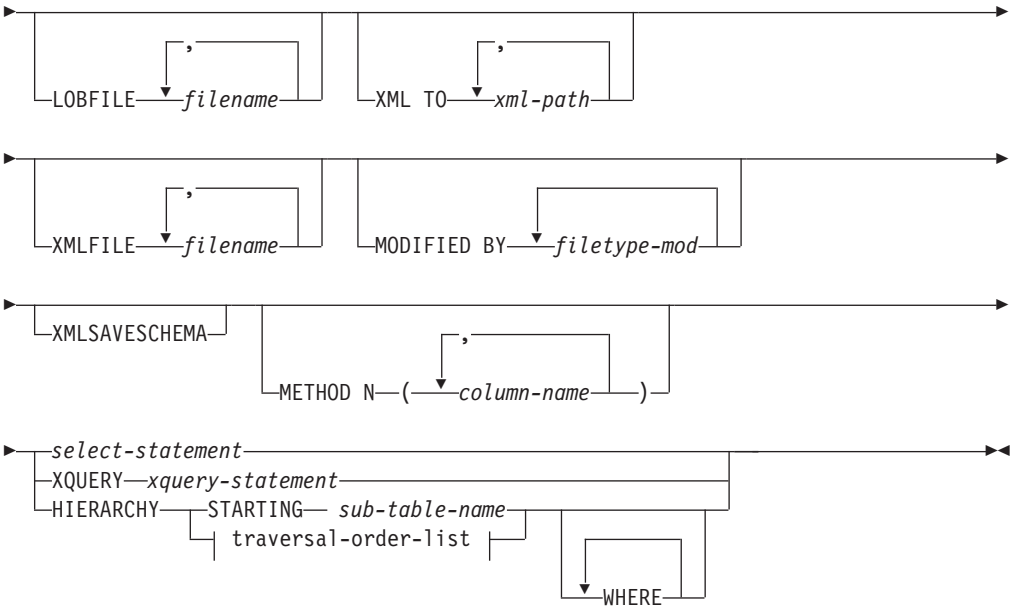
- *sysadm*
- *dbadm*

または、関係するそれぞれの表またはビューに対する CONTROL または SELECT 特権

必要な接続

コマンド構文





traversal-order-list:

コマンド・パラメーター

HIERARCHY *traversal-order-list*

指定した走査順序を使用して副階層をエクスポートします。すべての副表は、PRE-ORDER 方式でリストされていなければなりません。最初の副表名が、SELECT ステートメントのターゲット表名として使用されます。

HIERARCHY STARTING *sub-table-name*

デフォルトの走査順序 (ASC、DEL、または WSF ファイルの OUTER 順序、または PC/IXF データ・ファイルに保管されている順序) を使用して、*sub-table-name* から始まる副階層をエクスポートします。

LOBFILE *filename*

LOB ファイルに 1 つ以上の基本ファイル名を指定します。最初の名前の名前スペースがいっぱいになると、2 番目の名前が使用され、以下 3 番目、4 番目と続きます。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

エクスポート操作中に LOB ファイルを作成するときに、まずこのリストから現行パス (*lob-path* で指定されたパス) に現行のベース名を追加してから、最初に 3 桁のシーケンス番号、次に 3 文字の ID lob を追加して、ファイル名が構成されます。たとえば、現行 LOB パスがディレクトリー /u/foo/lob/path/ で、現行 LOB ファイル名が bar の場合、 LOB ファイルは、 /u/foo/lob/path/bar.001.lob、 /u/foo/lob/path/bar.002.lob (以下 003、004 と続く) などのように作成されます。 LOB ファイル名で使用する 3 桁のシーケンス番号は、999 の後は 4 桁になり、9999 の後は 5 桁になる、といった具合に大きくなっていきます。

LOBS TO *lob-path*

LOB ファイルが保管される、ディレクトリーへの 1 つ以上のパスを指定します。 LOB パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの LOB が入ります。指定できるパスの最大数は 999 です。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

METHOD N *column-name*

出力ファイルで使用される 1 つ以上の列名を指定します。このパラメーターが指定されない場合、表の列名が使用されます。このパラメーターは WSF および IXF ファイルでのみ有効ですが、階層データをエクスポートするときは無効です。

MODIFIED BY *filetype-mod*

ファイル・タイプ修飾子オプションを指定します。 26 ページの『エクスポート・ユーティリティーのファイル・タイプ修飾子』を参照してください。

OF *filetype*

次のような出力ファイルのデータ・フォーマットを指定します。

- DEL (区切り文字付き ASCII フォーマット)。さまざまなデータベース・マネージャーやファイル・マネージャー・プログラムで使用します。
- WSF (ワークシート・フォーマット)。以下のプログラムで使用します。
  - Lotus® 1-2-3®
  - Lotus Symphony

  BIGINT または DECIMAL データをエクスポートする場合、タイプ DOUBLE の範囲内の値のみが正確にエクスポートされます。この範囲内にない値もエクスポートされますが、オペレーティング・システムによっては、これらの値のインポートまたはエクスポートの結果、データに間違いが生じる場合があります。
- IXF (統合交換フォーマット、PC バージョン) は、プロプラエタリー・バイナリー・フォーマットです。

select-statement

エクスポートされるデータを戻す SELECT または XQUERY ステートメントを指定します。このステートメントによってエラーが発生する場合、メッセージ・ファイル (または標準出力) にメッセージが書き込まれます。エラー・コードが SQL0012W、SQL0347W、SQL0360W、SQL0437W、または SQL1824W である場合、エクスポート操作は続行します。これ以外のエラー・コードの場合、操作は停止します。

TO *filename*

すでに存在するファイルの名前を指定した場合、エクスポート・ユーティリティーはファイルの内容を上書きします。情報の追加は行いません。

XMLFILE *filename*

XML ファイルのための 1 つ以上の基本ファイル名を指定します。最初の名前の名前スペースがいっぱいになると、2 番目の名前が使用され、以下 3 番目、4 番目と続きます。

エクスポート操作中に XML ファイルを作成するときに、まずこのリストから現行パス (*xml-path* で指定されたパス) に現行のベース名を追加し、それに 3 桁のシーケンス番号を追加し、さらに 3 文字の ID xml を追加した

ファイル名が構成されます。たとえば、現行 XML パスがディレクトリー /u/foo/xml/path/ で、現行 XML ファイル名が bar の場合、XML ファイルは、/u/foo/xml/path/bar.001.xml、/u/foo/xml/path/bar.002.xml などのように作成されます。

XML TO *xml-path*
XML ファイルが保管されるディレクトリーを指す 1 つ以上のパスを指定します。 XML パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの XQuery データ・モデル (XDM) インスタンスが含まれることになります。 複数のパスが指定された場合、XDM インスタンスはそれらのパスに均等に分散されます。

XMLSAVESCHEMA
すべての XML 列について XML スキーマ情報を保管することを指定します。挿入時に XML スキーマに照らして妥当性検査されたエクスポート後の各 XML 文書に関しては、そのスキーマの完全修飾 SQL ID が、対応する XML Data Specifier (XDS) 内に (SCH) 属性として格納されます。エクスポート後の文書が XML スキーマに照らして妥当性検査されなかった場合や、スキーマ・オブジェクトがデータベース内にもう存在しない場合、対応する XDS に SCH 属性は組み入れられません。

SQL ID のスキーマと名前の各部分は、XML スキーマに対応する SYSCAT.XSROBJECTS カタログ表の行の ″OBJECTSCHEMA″ および ″OBJECTNAME″ の値として格納されます。

XMLSAVESCHEMA オプションには、整形式 XML 文書を生成しない XQuery シーケンスとの互換性がありません。

使用上の注意

- エクスポート操作を開始する前に、すべての表操作が完了し、すべてのロックがペンディング解除になっていることを確認してください。これは、WITH HOLD でオープンされた、すべてのカーソルをクローズした後で COMMIT または ROLLBACK を発行することによって行われます。
- SELECT ステートメントでは表の別名を使用できます。
- メッセージ・ファイルに置かれたメッセージには、メッセージ検索サービスから戻される情報が含まれています。各メッセージは新しい行から始まります。
- DEL フォーマット・ファイルへエクスポートするために 254 よりも長い文字データの列が選択されると、エクスポート・ユーティリティーは警告メッセージを生成します。
- PC/IXF インポートは、データベース間でデータを移動する場合に使用します。行区切り文字を含む文字データが区切り文字付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字を含むフィールドは長さが変わることがあります。
- ソースとターゲットのデータベースが両方とも同じクライアントからアクセス可能である場合、ファイルのコピーというステップは必要ありません。
- DB2 Connect を使用して、DB2 for OS/390®、DB2 for VM and VSE、および DB2 for OS/400® などの DRDA® から表をエクスポートすることができます。PC/IXF エクスポートだけがサポートされています。

- IXF 形式にエクスポートするときに、ID が IXF 形式でサポートされている最大サイズを超えていると、エクスポートは成功しますが、生成されるデータ・ファイルは、後から CREATE モードでインポート操作を実行するときに使用できません。SQL27984W が戻されます。
- Windows でディスケットにエクスポートするときに、1 枚のディスケットの容量を超えるデータが表に含まれている場合は、もう 1 枚のディスケットを用意することを求めるプロンプトがシステムから出され、マルチパート PC/IXF ファイル (マルチボリューム PC/IXF ファイル、論理分割 PC/IXF ファイルともいう) がそれぞれのディスケットに生成され、格納されます。最後のファイルを除く各ファイルには、DB2 CONTINUATION RECORD (または簡略形で "AC" レコード) が書き込まれます。つまり、各ファイルが論理的に分割されていることと、次のファイルの検出場所を示すレコードです。それらのファイルは、AIX® システムに転送して、インポート・ユーティリティーとロード・ユーティリティーで読み取ることができます。エクスポート・ユーティリティーは、 AIX システムから呼び出される場合、複数部分からなる PC/IXF ファイルを作成しません。詳しい使用法については、IMPORT コマンドまたは LOAD コマンドを参照してください。
- エクスポート・ユーティリティーは、提供される SELECT ステートメントが、`SELECT * FROM tablename` という形式である場合、 IXF ファイルの表の NOT NULL WITH DEFAULT 属性を保管します。
- 型付き表をエクスポートする場合、副選択ステートメントは、ターゲット表名と WHERE 節を指定することによってのみ表現することができます。階層をエクスポートするとき、全選択と*選択ステートメント* は指定できません。
- IXF 以外のファイル形式の場合は、階層の全探索の方法、およびエクスポートする副表とが DB2 に知らされるよう、全探索順序リストを指定することをお勧めします。このリストが指定されていないと、階層のすべての表がエクスポートされ、 OUTER 順序がデフォルトの順序になります。 OUTER 関数によって指定されるデフォルトの順序を使うこともできます。
- インポート操作時には、同じ全探索順序を使用してください。ロード・ユーティリティーでは、階層または副階層のロードはサポートされていません。
- 保護行のある表からデータをエクスポートする場合は、セッション許可 ID の保持する LBAC クリデンシャルのために、エクスポートされる行が制限されることがあります。セッション許可 ID に読み取りアクセスがない行はエクスポートされません。エラーも警告も出ません。
- セッション許可 ID の保持する LBAC クリデンシャルのために、エクスポートに含まれている 1 つ以上の保護列からの読み取りが許可されない場合、エクスポートは失敗し、エラー (SQLSTATE 42512) が戻されます。
- エクスポート・パッケージは `DATETIME ISO` フォーマットを使ってバインドされるので、ストリング表現へのキャスト時に、すべての日付/時刻/タイムスタンプが ISO フォーマットに変換されます。 CLP パッケージは、`DATETIME LOC` フォーマット (ロケール固有のフォーマット) を使ってバインドされるので、CLP DATETIME フォーマットが ISO と異なる場合は、CLP とエクスポートの間に動作上の不整合が見られることがあります。 たとえば、以下の SELECT ステートメントは、期待される結果を戻します。

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
   COL2
   ----------
```

```
05/10/2005
05/10/2005
05/10/2005
3 record(s) selected.
```

しかし、次のような、同じ select 節を使用した export コマンドは、そのような結果を戻しません。

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
   Number of rows exported: 0
```

次に、以下のように、LOCALE 日付フォーマットを ISO フォーマットに置き換えると、期待される結果が得られます。

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
   Number of rows exported: 3
```

エクスポート・ユーティリティーのファイル・タイプ修飾子

表 3. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式

修飾子	説明
lobsinfile	*lob-path* では、LOB データが含まれているファイルのパスを指定します。 各パスには、データ・ファイルの LOB ロケーション指定子 (LLS) によって参照されている LOB が少なくとも 1 つ入っているファイルが 1 つ以上含まれています。LLS は、LOB ファイル・パスに格納されているファイルの LOB の位置を示したストリング表記です。LLS の形式は、*filename.ext.nnn.mmm/* になります (*filename.ext* は、LOB が含まれているファイルの名前、*nnn* は、そのファイルに入っている LOB のオフセット (バイト単位)、*mmm* は、その LOB の長さ (バイト単位) です)。例えば、データ・ファイルにストリング db2exp.001.123.456/ が格納されている場合は、ファイル db2exp.001 のオフセット 123 に LOB が配置されていて、その長さは 456 バイトということになります。 EXPORT 使用時に lobsinfile 修飾子を指定すると、LOB データは、LOBS TO 節に指定されている位置に配置されます。そうでなければ、LOB データは、データ・ファイル・ディレクトリーに送信されます。LOBS TO 節では、LOB ファイルを格納するディレクトリーのパスを 1 つ以上指定します。LOB パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの LOB が入ります。 LOBS TO オプションまたは LOBFILE オプションを指定すると、LOBSINFILE の動作が暗黙的にアクティブになります。 NULL LOB を指定する場合は、サイズとして -1 を入力します。サイズとして 0 を指定すると、長さ 0 の LOB として処理されます。長さ -1 の NULL LOB の場合は、オフセットとファイル名が無視されます。例えば、NULL LOB の LLS は、db2exp.001.7.-1/ のようになります。
xmlinsepfiles	各 XQuery データ・モデル (XDM) インスタンスが別のファイルに書き込まれます。デフォルトでは、同じファイルの中で複数の値が連結されます。
lobsinsepfiles	各 LOB 値が別のファイルに書き込まれます。デフォルトでは、同じファイルの中で複数の値が連結されます。
xmlnodeclaration	XDM インスタンスが XML 宣言タグなしで書き込まれます。デフォルトでは、XDM インスタンスのエクスポート時に、エンコード属性を指定した XML 宣言タグが先頭に組み込まれます。

表 3. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
xmlchar	XDM インスタンスが文字コード・ページで書き込まれます。文字コード・ページは、codepage ファイル・タイプ修飾子で指定されている値か、その修飾子が指定されていない場合はアプリケーション・コード・ページになります。デフォルトでは、XDM インスタンスは、Unicode で書き込まれます。
xmlgraphic	EXPORT コマンドで xmlgraphic 修飾子を指定すると、アプリケーション・コード・ページまたは codepage ファイル・タイプ修飾子にかかわりなく、エクスポート XML 文書は、UTF-16 コード・ページでエンコードされます。

表 4. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式

修飾子	説明

chardel*x*	*x* は、単一文字ストリング区切りです。デフォルト値は、二重引用符 (″) です。文字ストリングを囲む二重引用符の代わりに指定の文字を使用します。[2] 文字ストリング区切りとして二重引用符を明示的に指定する場合は、以下のように指定します。
	`modified by chardel""`
	文字ストリング区切りとして単一引用符 (') を指定することもできます。その場合は、以下のようにします。
	`modified by chardel''`

codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、出力データ・セットに含まれているデータのコード・ページとして解釈されます。エクスポート操作の実行中に、文字データは、アプリケーション・コード・ページからこのコード・ページに変換されます。 DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。codepage 修飾子を lobsinfile 修飾子と一緒に使用することはできません。

coldel*x*	*x* は、単一文字列区切りです。デフォルト値は、コンマ (,) です。列の終わりを示すコンマの代わりに指定の文字を使用します。[2]
	以下の例では coldel; を指定しているので、エクスポート・ユーティリティーは、セミコロン文字 (;) をエクスポート・データの列区切りとして使用します。
	`db2 "export to temp of del modified by coldel;`
	`select * from staff where dept = 20"`

decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) の代わりにブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進値の接頭部として正符号を使用します。
decpt*x*	*x* は、小数点文字としてピリオドの代わりに使用する単一文字です。デフォルト値は、ピリオド (.) です。小数点文字として、ピリオドの代わりに指定の文字を使用します。[2]

表 4. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式 (続き)

修飾子	説明
nochardel	列データが文字区切りで囲まれなくなります。DB2 を使用してデータのインポートまたはロードを実行する場合は、このオプションを指定しないでください。このオプションは、文字区切りのないベンダー・データ・ファイルをサポートするために用意されています。正しくない使い方をすると、データが失われたり破損したりする可能性があります。 このオプションを chardelx または nodoubledel と一緒に指定することはできません。これらは、相互に排他的なオプションです。
nodoubledel	二重文字区切りの認識を抑止します。[2]

striplzeros	エクスポートするすべての 10 進数列から先行ゼロを除去します。
	
	次のような例を考察してみます。
	
	`db2 create table decimalTable ( c1 decimal( 31, 2 ) )`
	`db2 insert into decimalTable values ( 1.1 )`
	
	`db2 export to data of del select * from decimalTable`
	
	`db2 export to data of del modified by STRIPLZEROS`
	`select * from decimalTable`
	
	最初のエクスポート操作では、エクスポート・ファイル・データの内容が +0000000000000000000000000000001.10 になります。 2 番目の操作は、striplzeros 修飾子以外は最初の操作と同じですが、この場合は、エクスポート・ファイル・データの内容が +1.10 になります。

表 4. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式 (続き)

修飾子	説明
timestampformat="x"	x は、ソース・ファイルのタイム・スタンプの形式です。[4] 有効なタイム・スタンプ・エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (01 から 12 の 2 桁の数。 M および MMM とは相互に排他的) MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月のエレメントとは相互に排他的) H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 1 桁または 2 桁の数。) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数。 H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M (分) とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的) UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999990 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) タイム・スタンプ形式の例を以下に示します。 "YYYY/MM/DD HH:MM:SS.UUUUUU" MMM エレメントは、以下の値を生成します。 「Jan」、「Feb」、「Mar」、「Apr」、「May」、「Jun」、「Jul」、「Aug」、「Sep」、「Oct」、「Nov」、および「Dec」。「Jan」は 1 月と等しく、「Dec」は 12 月と等しいです。 ユーザー定義のタイム・スタンプ形式が含まれているデータを 'schedule' という表からエクスポートする例を以下に示します。 db2 export to delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" select * from schedule

表 5. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式

修飾子	説明
codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、出力データ・セットに含まれているデータのコード・ページとして解釈されます。エクスポート操作の実行中に、文字データは、このコード・ページからアプリケーション・コード・ページに変換されます。 DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。codepage 修飾子を lobsinfile 修飾子と一緒に使用することはできません。

表 6. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: WSF ファイル形式

修飾子	説明
1	Lotus 1-2-3 リリース 1 または Lotus 1-2-3 リリース 1a との互換性がある WSF ファイルを作成します。[5] これがデフォルトです。
2	Lotus Symphony リリース 1.0 との互換性がある WSF ファイルを作成します。[5]
3	Lotus 1-2-3 バージョン 2 または Lotus Symphony リリース 1.1 との互換性がある WSF ファイルを作成します。[5]
4	DBCS 文字が含まれている WSF ファイルを作成します。

注:

1. MODIFIED BY オプションでサポートされていないファイル・タイプを使用しようとしても、エクスポート・ユーティリティーからは警告が生成されません。その場合は、エクスポート操作が失敗し、エラー・コードが戻されます。
2. 区切り文字のオーバーライドとして使用できる文字に適用される制約事項については、『*データ移動のための区切り文字の制約事項*』を参照してください。
3. 通常、エクスポート・ユーティリティーは、以下のような形式で書き込みを行います。
   - 日付データを *YYYYMMDD* 形式で書き込みます。
   - char(date) データを *YYYY-MM-DD* 形式で書き込みます。
   - 時間データを *HH.MM.SS* 形式で書き込みます。
   - タイム・スタンプ・データを *YYYY-MM-DD-HH. MM.SS.uuuuuu* 形式で書き込みます。

   エクスポート操作の SELECT ステートメントで指定する日時列に含まれているデータも、このような形式になります。
4. タイム・スタンプ形式の場合は、月の記述子と分の記述子の間であいまいさが残らないように注意する必要があります。どちらも、M という文字を使用するからです。月のフィールドは、他の日付フィールドと隣接している必要があります。分のフィールドは、他の時刻フィールドと隣接している必要があります。あいまいなタイム・スタンプ形式の例を以下に示します。

   ```
   "M" (月または分のどちらにもとれる)
   "M:M" (月と分の区別がつかない)
   "M:YYYY:M" (両方とも月と解釈される)
   "S:M:YYYY" (時刻値と日付値の両方に隣接している)
   ```

あいまいな場合は、ユーティリティーによってエラー・メッセージが生成され、操作は失敗します。

あいまいでないタイム・スタンプ形式の例を以下に示します。

```
"M:YYYY" (M (月))
"S:M" (M (分))
"M:YYYY:S:M" (M (月)....M (分))
"M:H:YYYY:M:D" (M (分)....M (月))
```

5. これらのファイルの出力先として、特定の製品を指定することもできます。*filetype-mod* パラメーター・ストリングに、Lotus 1-2-3 の場合は L、Symphony の場合は S をそれぞれ指定します。指定できるのは、1 つの値または製品指定子だけです。
6. XML 列では、WSF ファイル形式はサポートされていません。
7. XMLFILE 節や XML TO 節を指定しなくても、すべての XDM インスタンスは、メイン・データ・ファイルとは別の XML ファイルに書き込まれます。デフォルトでは、エクスポート・データ・ファイルのパスに XML ファイルも書き込まれます。XML ファイルのデフォルトのベース名は、エクスポート・データ・ファイルの名前に拡張子 ″.xml″ を追加した形になります。
8. XMLNODECLARATION ファイル・タイプ修飾子を指定しない限り、すべての XDM インスタンスが書き込まれるときには、エンコード属性を指定した XML 宣言が先頭に組み込まれます。
9. ファイル・タイプ修飾子 XMLCHAR または XMLGRAPHIC を指定しない限り、デフォルトでは、すべての XDM インスタンスが Unicode で書き込まれます。
10. XML データと LOB データのデフォルトのパスは、メイン・データ・ファイルのパスです。XML ファイルのデフォルトのベース名は、メイン・データ・ファイルです。LOB ファイルのデフォルトのベース名は、メイン・データ・ファイルです。例えば、メイン・データ・ファイルが

    ```
    /mypath/myfile.del
    ```

    の場合、XML データおよび LOB データのデフォルト・パスは

    ```
    /mypath"
    ```

    となり、デフォルトの XML ファイルのベース名は

    ```
    myfile.del
    ```

    となり、デフォルトの LOB ファイルのベース名は

    ```
    myfile.del
    ```

    .

    LOB ファイルを生成するには、LOBSINFILE ファイル・タイプ修飾子を指定する必要があります。
11. エクスポート・ユーティリティーは、それぞれの LOB ファイルまたは XML ファイルに数値 ID を追加します。この ID は、0 を埋め込んだ 3 桁のシーケンス値 (つまり、

    ```
    .001
    ```

) から始まります。999 番目の LOB ファイルまたは XML ファイルの後は、ID にゼロが埋め込まれなくなります (例えば、1000 番目の LOG ファイルまたは XML ファイルの拡張子は、

```
.1000
```

になります)。数値 ID の後に、データ・タイプを示す 3 桁の文字タイプ ID (

```
.lob
```

または

```
.xml
```

) が追加されます。例えば、生成される LOB ファイルの名前は

```
myfile.del.001.lob
```

という形式、生成される XML ファイルの名前は

```
myfile.del.001.xml
```

という形式になります。

12. エクスポート・ユーティリティーでは、整形式でない文書の XDM インスタンスでも、XQuery を指定することによってエクスポートすることが可能です。ただし、そのエクスポート文書を XML 列に直接インポートしたりロードしたりすることはできません。XML 列には完全な文書しか組み込めないからです。

EXPORT コマンド (ADMIN_CMD プロシージャーを使用)

データベースから、いくつかある外部ファイル形式のいずれかにデータをエクスポートします。ユーザーは、SQL SELECT ステートメントによって、または型付き表の階層情報によってエクスポートするデータを指定します。

37 ページの『エクスポート・ユーティリティーのファイル・タイプ修飾子』へのクイック・リンク。

許可

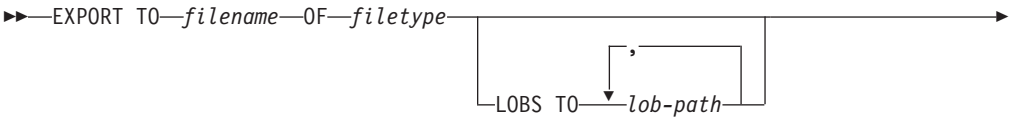
以下のいずれか。

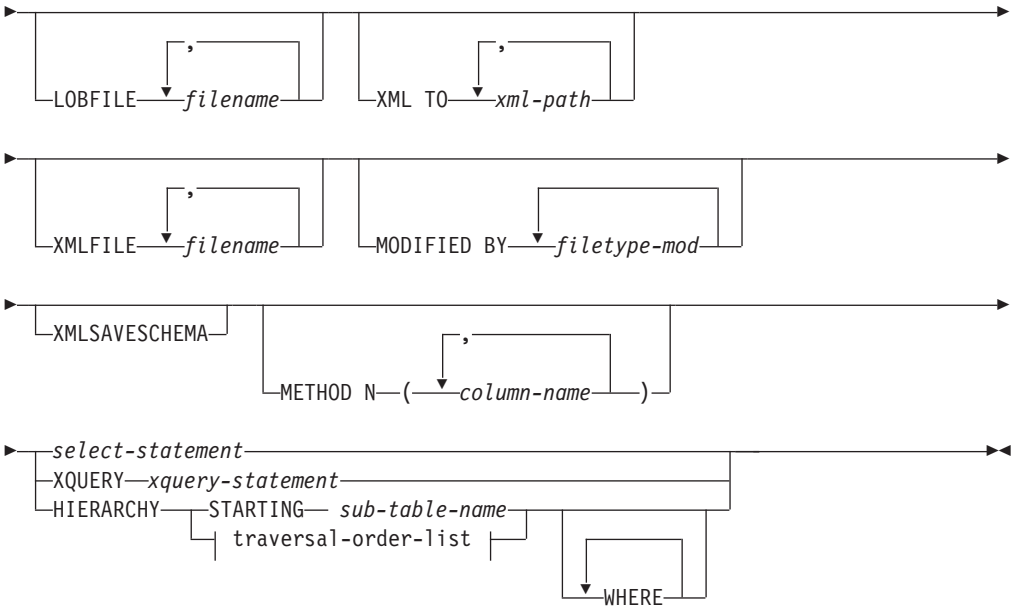
- *sysadm*
- *dbadm*

または、関係するそれぞれの表またはビューに対する CONTROL または SELECT 特権

必要な接続

コマンド構文





traversal-order-list:

コマンド・パラメーター

HIERARCHY *traversal-order-list*

指定した走査順序を使用して副階層をエクスポートします。すべての副表は、PRE-ORDER 方式でリストされていなければなりません。最初の副表名が、SELECT ステートメントのターゲット表名として使用されます。

HIERARCHY STARTING *sub-table-name*

デフォルトの走査順序 (ASC、DEL、または WSF ファイルの OUTER 順序、または PC/IXF データ・ファイルに保管されている順序) を使用して、*sub-table-name* から始まる副階層をエクスポートします。

LOBFILE *filename*

LOB ファイルに 1 つ以上の基本ファイル名を指定します。最初の名前の名前スペースがいっぱいになると、2 番目の名前が使用され、以下 3 番目、4 番目と続きます。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

エクスポート操作中に LOB ファイルを作成するときに、まずこのリストから現行パス (*lob-path* で指定されたパス) に現行のベース名を追加してから、最初に 3 桁のシーケンス番号、次に 3 文字の ID lob を追加して、ファイル名が構成されます。たとえば、現行 LOB パスがディレクトリー /u/foo/lob/path/ で、現行 LOB ファイル名が bar の場合、 LOB ファイルは、 /u/foo/lob/path/bar.001.lob、 /u/foo/lob/path/bar.002.lob (以下 003、004 と続く) などのように作成されます。 LOB ファイル名で使用する 3 桁のシーケンス番号は、999 の後は 4 桁になり、9999 の後は 5 桁になる、といった具合に大きくなっていきます。

LOBS TO *lob-path*

LOB ファイルが保管される、ディレクトリーへの 1 つ以上のパスを指定します。 LOB パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの LOB が入ります。指定できるパスの最大数は 999 です。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

METHOD N *column-name*

出力ファイルで使用される 1 つ以上の列名を指定します。このパラメーターが指定されない場合、表の列名が使用されます。このパラメーターは WSF および IXF ファイルでのみ有効ですが、階層データをエクスポートするときは無効です。

MODIFIED BY *filetype-mod*

ファイル・タイプ修飾子オプションを指定します。 37 ページの『エクスポート・ユーティリティーのファイル・タイプ修飾子』を参照してください。

OF *filetype*

次のような出力ファイルのデータ・フォーマットを指定します。

- DEL (区切り文字付き ASCII フォーマット)。さまざまなデータベース・マネージャーやファイル・マネージャー・プログラムで使用します。
- WSF (ワークシート・フォーマット)。以下のプログラムで使用します。
  - Lotus 1-2-3
  - Lotus Symphony

  BIGINT または DECIMAL データをエクスポートする場合、タイプ DOUBLE の範囲内の値のみが正確にエクスポートされます。この範囲内にない値もエクスポートされますが、オペレーティング・システムによっては、これらの値のインポートまたはエクスポートの結果、データに間違いが生じる場合があります。
- IXF (統合交換フォーマット、PC バージョン) は、プロプラエタリー・バイナリー・フォーマットです。

select-statement

エクスポートされるデータを戻す SELECT または XQUERY ステートメントを指定します。このステートメントによってエラーが発生する場合、メッセージ・ファイル (または標準出力) にメッセージが書き込まれます。エラー・コードが SQL0012W、SQL0347W、SQL0360W、SQL0437W、または SQL1824W である場合、エクスポート操作は続行します。これ以外のエラー・コードの場合、操作は停止します。

TO *filename*

すでに存在するファイルの名前を指定した場合、エクスポート・ユーティリティーはファイルの内容を上書きします。情報の追加は行いません。

XMLFILE *filename*

XML ファイルのための 1 つ以上の基本ファイル名を指定します。最初の名前の名前スペースがいっぱいになると、2 番目の名前が使用され、以下 3 番目、4 番目と続きます。

エクスポート操作中に XML ファイルを作成するときに、まずこのリストから現行パス (*xml-path* で指定されたパス) に現行のベース名を追加し、それに 3 桁のシーケンス番号を追加し、さらに 3 文字の ID xml を追加した

ファイル名が構成されます。たとえば、現行 XML パスがディレクトリー /u/foo/xml/path/ で、現行 XML ファイル名が bar の場合、XML ファイルは、/u/foo/xml/path/bar.001.xml、/u/foo/xml/path/bar.002.xml などのように作成されます。

XML TO *xml-path*
XML ファイルが保管されるディレクトリーを指す 1 つ以上のパスを指定します。 XML パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの XQuery データ・モデル (XDM) インスタンスが含まれることになります。 複数のパスが指定された場合、XDM インスタンスはそれらのパスに均等に分散されます。

XMLSAVESCHEMA
すべての XML 列について XML スキーマ情報を保管することを指定します。挿入時に XML スキーマに照らして妥当性検査されたエクスポート後の各 XML 文書に関しては、そのスキーマの完全修飾 SQL ID が、対応する XML Data Specifier (XDS) 内に (SCH) 属性として格納されます。エクスポート後の文書が XML スキーマに照らして妥当性検査されなかった場合や、スキーマ・オブジェクトがデータベース内にもう存在しない場合、対応する XDS に SCH 属性は組み入れられません。

SQL ID のスキーマと名前の各部分は、XML スキーマに対応する SYSCAT.XSROBJECTS カタログ表の行の ″OBJECTSCHEMA″ および ″OBJECTNAME″ の値として格納されます。

XMLSAVESCHEMA オプションには、整形式 XML 文書を生成しない XQuery シーケンスとの互換性がありません。

使用上の注意

- エクスポート操作を開始する前に、すべての表操作が完了し、すべてのロックがペンディング解除になっていることを確認してください。これは、WITH HOLD でオープンされた、すべてのカーソルをクローズした後で COMMIT または ROLLBACK を発行することによって行われます。
- SELECT ステートメントでは表の別名を使用できます。
- メッセージ・ファイルに置かれたメッセージには、メッセージ検索サービスから戻される情報が含まれています。各メッセージは新しい行から始まります。
- DEL フォーマット・ファイルへエクスポートするために 254 よりも長い文字データの列が選択されると、エクスポート・ユーティリティーは警告メッセージを生成します。
- PC/IXF インポートは、データベース間でデータを移動する場合に使用します。行区切り文字を含む文字データが区切り文字付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字を含むフィールドは長さが変わることがあります。
- ソースとターゲットのデータベースが両方とも同じクライアントからアクセス可能である場合、ファイルのコピーというステップは必要ありません。
- DB2 Connect を使用して、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA から表をエクスポートすることができます。PC/IXF エクスポートだけがサポートされています。

- IXF 形式にエクスポートするときに、ID が IXF 形式でサポートされている最大サイズを超えていると、エクスポートは成功しますが、生成されるデータ・ファイルは、後から CREATE モードでインポート操作を実行するときに使用できません。SQL27984W が戻されます。
- Windows でディスケットにエクスポートするときに、1 枚のディスケットの容量を超えるデータが表に含まれている場合は、もう 1 枚のディスケットを用意することを求めるプロンプトがシステムから出され、マルチパート PC/IXF ファイル (マルチボリューム PC/IXF ファイル、論理分割 PC/IXF ファイルともいう) がそれぞれのディスケットに生成され、格納されます。最後のファイルを除く各ファイルには、DB2 CONTINUATION RECORD (または簡略形で ″AC″ レコード) が書き込まれます。つまり、各ファイルが論理的に分割されていることと、次のファイルの検出場所を示すレコードです。それらのファイルは、AIX システムに転送して、インポート・ユーティリティーとロード・ユーティリティーで読み取ることができます。エクスポート・ユーティリティーは、 AIX システムから呼び出される場合、複数部分からなる PC/IXF ファイルを作成しません。詳しい使用法については、IMPORT コマンドまたは LOAD コマンドを参照してください。
- エクスポート・ユーティリティーは、提供される SELECT ステートメントが、`SELECT * FROM tablename` という形式である場合、 IXF ファイルの表の NOT NULL WITH DEFAULT 属性を保管します。
- 型付き表をエクスポートする場合、副選択ステートメントは、ターゲット表名と WHERE 節を指定することによってのみ表現することができます。階層をエクスポートするとき、全選択と *選択ステートメント* は指定できません。
- IXF 以外のファイル形式の場合は、階層の全探索の方法、およびエクスポートする副表とが DB2 に知らされるよう、全探索順序リストを指定することをお勧めします。このリストが指定されていないと、階層のすべての表がエクスポートされ、 OUTER 順序がデフォルトの順序になります。 OUTER 関数によって指定されるデフォルトの順序を使うこともできます。
- インポート操作時には、同じ全探索順序を使用してください。ロード・ユーティリティーでは、階層または副階層のロードはサポートされていません。
- 保護行のある表からデータをエクスポートする場合は、セッション許可 ID の保持する LBAC クリデンシャルのために、エクスポートされる行が制限されることがあります。セッション許可 ID に読み取りアクセスがない行はエクスポートされません。エラーも警告も出ません。
- セッション許可 ID の保持する LBAC クリデンシャルのために、エクスポートに含まれている 1 つ以上の保護列からの読み取りが許可されない場合、エクスポートは失敗し、エラー (SQLSTATE 42512) が戻されます。
- エクスポート・パッケージは `DATETIME ISO` フォーマットを使ってバインドされるので、ストリング表現へのキャスト時に、すべての日付/時刻/タイムスタンプが ISO フォーマットに変換されます。 CLP パッケージは、`DATETIME LOC` フォーマット (ロケール固有のフォーマット) を使ってバインドされるので、CLP DATETIME フォーマットが ISO と異なる場合は、CLP とエクスポートの間に動作上の不整合が見られることがあります。 たとえば、以下の SELECT ステートメントは、期待される結果を戻します。

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
   COL2
   ----------
```

```
05/10/2005
05/10/2005
05/10/2005
3 record(s) selected.
```

しかし、次のような、同じ select 節を使用した export コマンドは、そのような結果を戻しません。

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
   Number of rows exported: 0
```

次に、以下のように、LOCALE 日付フォーマットを ISO フォーマットに置き換えると、期待される結果が得られます。

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
   Number of rows exported: 3
```

エクスポート・ユーティリティーのファイル・タイプ修飾子

表 7. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式

修飾子	説明
lobsinfile	*lob-path* では、LOB データが含まれているファイルのパスを指定します。 各パスには、データ・ファイルの LOB ロケーション指定子 (LLS) によって参照されている LOB が少なくとも 1 つ入っているファイルが 1 つ以上含まれています。LLS は、LOB ファイル・パスに格納されているファイルの LOB の位置を示したストリング表記です。LLS の形式は、*filename.ext.nnn.mmm/* になります (*filename.ext* は、LOB が含まれているファイルの名前、*nnn* は、そのファイルに入っている LOB のオフセット (バイト単位)、*mmm* は、その LOB の長さ (バイト単位) です)。例えば、データ・ファイルにストリング db2exp.001.123.456/ が格納されている場合は、ファイル db2exp.001 のオフセット 123 に LOB が配置されていて、その長さは 456 バイトということになります。 EXPORT 使用時に lobsinfile 修飾子を指定すると、LOB データは、LOBS TO 節に指定されている位置に配置されます。そうでなければ、LOB データは、データ・ファイル・ディレクトリーに送信されます。LOBS TO 節では、LOB ファイルを格納するディレクトリーのパスを 1 つ以上指定します。LOB パスごとに少なくとも 1 つのファイルが存在し、各ファイルには少なくとも 1 つの LOB が入ります。 LOBS TO オプションまたは LOBFILE オプションを指定すると、LOBSINFILE の動作が暗黙的にアクティブになります。 NULL LOB を指定する場合は、サイズとして -1 を入力します。サイズとして 0 を指定すると、長さ 0 の LOB として処理されます。長さ -1 の NULL LOB の場合は、オフセットとファイル名が無視されます。例えば、NULL LOB の LLS は、db2exp.001.7.-1/ のようになります。
xmlinsepfiles	各 XQuery データ・モデル (XDM) インスタンスが別のファイルに書き込まれます。デフォルトでは、同じファイルの中で複数の値が連結されます。
lobsinsepfiles	各 LOB 値が別のファイルに書き込まれます。デフォルトでは、同じファイルの中で複数の値が連結されます。
xmlnodeclaration	XDM インスタンスが XML 宣言タグなしで書き込まれます。デフォルトでは、XDM インスタンスのエクスポート時に、エンコード属性を指定した XML 宣言タグが先頭に組み込まれます。

表 7. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
xmlchar	XDM インスタンスが文字コード・ページで書き込まれます。文字コード・ページは、codepage ファイル・タイプ修飾子で指定されている値か、その修飾子が指定されていない場合はアプリケーション・コード・ページになります。デフォルトでは、XDM インスタンスは、Unicode で書き込まれます。
xmlgraphic	EXPORT コマンドで xmlgraphic 修飾子を指定すると、アプリケーション・コード・ページまたは codepage ファイル・タイプ修飾子にかかわりなく、エクスポート XML 文書は、UTF-16 コード・ページでエンコードされます。

表 8. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式

修飾子	説明

chardel*x*	*x* は、単一文字ストリング区切りです。デフォルト値は、二重引用符 (") です。文字ストリングを囲む二重引用符の代わりに指定の文字を使用します。[2] 文字ストリング区切りとして二重引用符を明示的に指定する場合は、以下のように指定します。
	`modified by chardel""`
	文字ストリング区切りとして単一引用符 (') を指定することもできます。その場合は、以下のようにします。
	`modified by chardel''`

codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、出力データ・セットに含まれているデータのコード・ページとして解釈されます。エクスポート操作の実行中に、文字データは、アプリケーション・コード・ページからこのコード・ページに変換されます。 DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。codepage 修飾子を lobsinfile 修飾子と一緒に使用することはできません。

coldel*x*	*x* は、単一文字列区切りです。デフォルト値は、コンマ (,) です。列の終わりを示すコンマの代わりに指定の文字を使用します。[2]
	以下の例では coldel; を指定しているので、エクスポート・ユーティリティーは、セミコロン文字 (;) をエクスポート・データの列区切りとして使用します。
	`db2 "export to temp of del modified by coldel;`
	`select * from staff where dept = 20"`

decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) の代わりにブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進値の接頭部として正符号を使用します。
decpt*x*	*x* は、小数点文字としてピリオドの代わりに使用する単一文字です。デフォルト値は、ピリオド (.) です。小数点文字として、ピリオドの代わりに指定の文字を使用します。[2]

表 8. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式 (続き)

修飾子	説明
nochardel	列データが文字区切りで囲まれなくなります。DB2 を使用してデータのインポートまたはロードを実行する場合は、このオプションを指定しないでください。このオプションは、文字区切りのないベンダー・データ・ファイルをサポートするために用意されています。正しくない使い方をすると、データが失われたり破損したりする可能性があります。 このオプションを chardelx または nodoubledel と一緒に指定することはできません。これらは、相互に排他的なオプションです。
nodoubledel	二重文字区切りの認識を抑止します。[2]

striplzeros	エクスポートするすべての 10 進数列から先行ゼロを除去します。
	
	次のような例を考察してみます。
	
	`db2 create table decimalTable ( c1 decimal( 31, 2 ) )`
	`db2 insert into decimalTable values ( 1.1 )`
	
	`db2 export to data of del select * from decimalTable`
	
	`db2 export to data of del modified by STRIPLZEROS`
	`select * from decimalTable`
	
	最初のエクスポート操作では、エクスポート・ファイル・データの内容が +0000000000000000000000000000001.10 になります。 2 番目の操作は、striplzeros 修飾子以外は最初の操作と同じですが、この場合は、エクスポート・ファイル・データの内容が +1.10 になります。

表 8. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式 (続き)

修飾子	説明
timestampformat="x"	x は、ソース・ファイルのタイム・スタンプの形式です。[4] 有効なタイム・スタンプ・エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (01 から 12 の 2 桁の数。 M および MMM とは相互に排他的) MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月のエレメントとは相互に排他的) H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 1 桁または 2 桁の数。) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数。 H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M (分) とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的) UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999990 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) タイム・スタンプ形式の例を以下に示します。 "YYYY/MM/DD HH:MM:SS.UUUUUU" MMM エレメントは、以下の値を生成します。 「Jan」、「Feb」、「Mar」、「Apr」、「May」、「Jun」、「Jul」、「Aug」、「Sep」、「Oct」、「Nov」、および「Dec」。「Jan」は 1 月と等しく、「Dec」は 12 月と等しいです。 ユーザー定義のタイム・スタンプ形式が含まれているデータを 'schedule' という表からエクスポートする例を以下に示します。 db2 export to delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" select * from schedule

表 9. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: *IXF* ファイル形式

修飾子	説明
codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、出力データ・セットに含まれているデータのコード・ページとして解釈されます。エクスポート操作の実行中に、文字データは、このコード・ページからアプリケーション・コード・ページに変換されます。 DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。`codepage` 修飾子を `lobsinfile` 修飾子と一緒に使用することはできません。

表 10. エクスポート・ユーティリティーの有効なファイル・タイプ修飾子: *WSF* ファイル形式

修飾子	説明
1	Lotus 1-2-3 リリース 1 または Lotus 1-2-3 リリース 1a との互換性がある WSF ファイルを作成します。[5] これがデフォルトです。
2	Lotus Symphony リリース 1.0 との互換性がある WSF ファイルを作成します。[5]
3	Lotus 1-2-3 バージョン 2 または Lotus Symphony リリース 1.1 との互換性がある WSF ファイルを作成します。[5]
4	DBCS 文字が含まれている WSF ファイルを作成します。

注:

1. MODIFIED BY オプションでサポートされていないファイル・タイプを使用しようとしても、エクスポート・ユーティリティーからは警告が生成されません。その場合は、エクスポート操作が失敗し、エラー・コードが戻されます。
2. 区切り文字のオーバーライドとして使用できる文字に適用される制約事項については、『*データ移動のための区切り文字の制約事項*』を参照してください。
3. 通常、エクスポート・ユーティリティーは、以下のような形式で書き込みを行います。
   - 日付データを *YYYYMMDD* 形式で書き込みます。
   - char(date) データを *YYYY-MM-DD* 形式で書き込みます。
   - 時間データを *HH.MM.SS* 形式で書き込みます。
   - タイム・スタンプ・データを *YYYY-MM-DD-HH. MM.SS.uuuuuu* 形式で書き込みます。

   エクスポート操作の SELECT ステートメントで指定する日時列に含まれているデータも、このような形式になります。
4. タイム・スタンプ形式の場合は、月の記述子と分の記述子の間であいまいさが残らないように注意する必要があります。どちらも、M という文字を使用するからです。月のフィールドは、他の日付フィールドと隣接している必要があります。分のフィールドは、他の時刻フィールドと隣接している必要があります。あいまいなタイム・スタンプ形式の例を以下に示します。

```
"M" (月または分のどちらにもとれる)
"M:M" (月と分の区別がつかない)
"M:YYYY:M" (両方とも月と解釈される)
"S:M:YYYY" (時刻値と日付値の両方に隣接している)
```

あいまいな場合は、ユーティリティーによってエラー・メッセージが生成され、操作は失敗します。

あいまいでないタイム・スタンプ形式の例を以下に示します。

```
"M:YYYY" (M (月))
"S:M" (M (分))
"M:YYYY:S:M" (M (月)....M (分))
"M:H:YYYY:M:D" (M (分)....M (月))
```

5. これらのファイルの出力先として、特定の製品を指定することもできます。*filetype-mod* パラメーター・ストリングに、Lotus 1-2-3 の場合は L、Symphony の場合は S をそれぞれ指定します。指定できるのは、1 つの値または製品指定子だけです。
6. XML 列では、WSF ファイル形式はサポートされていません。
7. XMLFILE 節や XML TO 節を指定しなくても、すべての XDM インスタンスは、メイン・データ・ファイルとは別の XML ファイルに書き込まれます。デフォルトでは、エクスポート・データ・ファイルのパスに XML ファイルも書き込まれます。XML ファイルのデフォルトのベース名は、エクスポート・データ・ファイルの名前に拡張子 ″.xml″ を追加した形になります。
8. XMLNODECLARATION ファイル・タイプ修飾子を指定しない限り、すべての XDM インスタンスが書き込まれるときには、エンコード属性を指定した XML 宣言が先頭に組み込まれます。
9. ファイル・タイプ修飾子 XMLCHAR または XMLGRAPHIC を指定しない限り、デフォルトでは、すべての XDM インスタンスが Unicode で書き込まれます。
10. XML データと LOB データのデフォルトのパスは、メイン・データ・ファイルのパスです。XML ファイルのデフォルトのベース名は、メイン・データ・ファイルです。LOB ファイルのデフォルトのベース名は、メイン・データ・ファイルです。例えば、メイン・データ・ファイルが

    ```
    /mypath/myfile.del
    ```

    の場合、XML データおよび LOB データのデフォルト・パスは

    ```
    /mypath"
    ```

    となり、デフォルトの XML ファイルのベース名は

    ```
    myfile.del
    ```

    となり、デフォルトの LOB ファイルのベース名は

    ```
    myfile.del
    ```

    .

    LOB ファイルを生成するには、LOBSINFILE ファイル・タイプ修飾子を指定する必要があります。
11. エクスポート・ユーティリティーは、それぞれの LOB ファイルまたは XML ファイルに数値 ID を追加します。この ID は、0 を埋め込んだ 3 桁のシーケンス値 (つまり、

    ```
    .001
    ```

) から始まります。999 番目の LOB ファイルまたは XML ファイルの後は、ID にゼロが埋め込まれなくなります (例えば、1000 番目の LOG ファイルまたは XML ファイルの拡張子は、

```
.1000
```

になります)。数値 ID の後に、データ・タイプを示す 3 桁の文字タイプ ID (

```
.lob
```

または

```
.xml
```

) が追加されます。例えば、生成される LOB ファイルの名前は

```
myfile.del.001.lob
```

という形式、生成される XML ファイルの名前は

```
myfile.del.001.xml
```

という形式になります。

12. エクスポート・ユーティリティーでは、整形式でない文書の XDM インスタンスでも、XQuery を指定することによってエクスポートすることが可能です。ただし、そのエクスポート文書を XML 列に直接インポートしたりロードしたりすることはできません。XML 列には完全な文書しか組み込めないからです。

db2Export - データベースからのデータのエクスポート

データベースから、いくつかある外部ファイル形式のいずれかにデータをエクスポートします。ユーザーは、SQL SELECT ステートメントによって、または型付き表の階層情報によってエクスポートするデータを指定します。

許可

以下のいずれか。

- sysadm
- dbadm

または、関係するそれぞれの表またはビューに対する CONTROL または SELECT 特権この関数に対しては、ラベル・ベースのアクセス制御 (LBAC) が実施されます。エクスポートするデータは、LBAC で保護されている場合、呼び出し元の LBAC 信用証明情報によって制限を受けることがあります。

必要な接続

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。

API 組み込みファイル

```
db2ApiDf.h
```

API およびデータ構造構文

```
SQL_API_RC SQL_API_FN
  db2Export (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2ExportStruct;

typedef SQL_STRUCTURE db2ExportIn
{
   db2Uint16 *piXmlSaveSchema;
} db2ExportIn;

typedef SQL_STRUCTURE db2ExportOut
{
   db2Uint64 oRowsExported;
} db2ExportOut;

SQL_API_RC SQL_API_FN
  db2gExport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2gExportStruct;
```

db2Export API パラメーター

versionNumber

入力。 2 番目のパラメーター pParmStruct として渡される、構造のバージョンとリリース・レベルを指定します。

pParmStruct
入力。 db2ExportStruct 構造を指すポインター。

pSqlca 出力。sqlca 構造を指すポインター。

db2ExportStruct データ構造パラメーター

piDataFileName
入力。データがエクスポートされるパスおよび外部ファイル名を含むストリングを指定します。

piLobPathList
入力。 media_type フィールドを SQLU_LOCAL_MEDIA に設定された sqlu_media_list 構造、および LOB ファイルが保管されるクライアント上のパスをリストするその sqlu_media_entry 構造を指すポインター。エクスポートした LOB データは、sqlu_media_entry 構造にリストされているすべてのパスに均一に分散されます。

piLobFileList
入力。 media_type フィールドを SQLU_CLIENT_LOCATION に設定された sqlu_media_list 構造と、基本ファイル名の入ったその sqlu_location_entry 構造を指すポインター。

このリスト内の最初の名前を使用しているネーム・スペースが使い尽くされると、 API は 2 番目の名前を使用し、以下同様に続きます。エクスポート操作での LOB ファイルの作成時には、現行パス (piLobPathList からの) にこのリストから現行ベース名を追加し、その後に 3 桁のシーケンス番号と .lob 拡張子を追加した形のファイル名が構成されます。例えば、現行の LOB パスが /u/foo/lob/path ディレクトリー、現行の LOB ファイル名が bar、そして LOBSINSEPFILES ファイル・タイプ修飾子が設定済みの場合、作成される LOB ファイルは、/u/foo/LOB/path/bar.001.lob、/u/foo/LOB/path/bar.002.lob、などとなります。 LOBSINSEPFILES ファイル・タイプ修飾子を設定しない場合、すべての LOB 文書が連結されて、/u/foo/lob/path/bar.001.lob という 1 つのファイルに書き込まれます。

piDataDescriptor
入力。出力ファイルの列名を指定する sqldcol 構造を指すポインター。dcolmeth フィールドの値によって、このパラメーターに提供される残りの情報をエクスポート・ユーティリティーがどのように解釈するかが判別されます。このパラメーターの有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQL_METH_N
名前。出力ファイルで使用する列名を指定します。

SQL_METH_D
デフォルト。表の既存の列の名前が、出力ファイルで使用されます。この場合、列数および列指定配列は、どちらも無視されます。列名は、piActionString に指定された SELECT ステートメントの出力から派生します。

piActionString
入力。有効な動的 SQL SELECT ステートメントの入った sqllob 構造を指すポインター。この構造には、4 バイトの長さフィールドと、 SELECT ス

テートメントを構成する文字が順に含まれます。 SELECT ステートメントは、データベースからデータを取り出し、外部ファイルに書き込むことを指定します。

(piDataDescriptor からの) 外部ファイルの列と、 SELECT ステートメントからのデータベース列とは、それぞれのリストまたは構造における位置に従って対応付けられます。データベースから選択されたデータの最初の列は、外部ファイルの最初の列に置かれ、その列名は外部列配列の最初のエレメントから取られます。

piFileType

入力。外部ファイル内のデータの形式を示すストリングを指定します。サポートされている外部ファイルの形式 (sqlutil ヘッダー・ファイルで定義) は以下のとおりです。

SQL_DEL

区切り文字付き ASCII。これは dBase プログラム、BASIC プログラム、 IBM® パーソナル・デシジョン・シリーズ・プログラム、およびその他の多数のデータベース・マネージャー/ファイル・マネージャーとの交換のための形式です。

SQL_WSF

ワークシート形式。 Lotus Symphony および 1-2-3 プログラムとの交換のための形式です。

SQL_IXF

IXF (統合交換フォーマット、PC バージョン)。表からデータをエクスポートする場合の推奨方式です。このファイル形式にエクスポートされたデータは、後で同じ表または別のデータベース・マネージャー表にインポートまたはロードすることができます。

piFileTypeMod

入力。 2 バイトの長さフィールドと、1 つ以上の処理オプションを指定する文字の配列を収容する sqldcol 構造を指すポインターです。このポインターが NULL であるか、このポインターが指す構造に 1 文字も入っていない場合、このアクションはデフォルトの指定が選択されたものとして解釈されます。

サポートされるすべてのファイル・タイプに、すべてのオプションを使用できるわけではありません。以下の『エクスポート・ユーティリティーのファイル・タイプ修飾子』の関連リンクを参照してください。

piMsgFileName

入力。このユーティリティーが戻すエラー、警告、および情報メッセージの宛先を含むストリングを指定します。オペレーティング・システム・ファイルまたは標準装置のパスおよび名前を指定できます。ファイルが既に存在する場合は、その情報が付加されます。存在していない場合は、新たに作成されます。

iCallerAction

入力。呼び出し側が要求するアクションを示します。有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQLU_INITIAL
最初の呼び出し。この値は、API への最初の呼び出しの際には必ず使用してください。最初の呼び出しまたは後続の呼び出しのいずれかが戻され、要求されたエクスポート操作が完了する前に呼び出し側のアプリケーションが何らかのアクションを行うことが必要な場合、呼び出し側のアクションを以下のどちらかに設定しなければなりません。

SQLU_CONTINUE
処理の継続。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (例えば、テープの終わり条件への応答) を要求した後で、 API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが完了したら、ユーティリティーが最初の要求の処理を続行するよう指定するものです。

SQLU_TERMINATE
処理の終了。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (例えば、テープの終わり条件への応答) を要求した後で、 API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが実行されなかった場合、ユーティリティーが最初の要求の処理を中断するよう指定するものです。

poExportInfoOut
db2ExportOut 構造を指すポインター。

piExportInfoIn
入力。 db2ExportIn 構造を指すポインター。

piXmlPathList
入力。 media_type フィールドを SQLU_LOCAL_MEDIA に設定された sqlu_media_list 構造、および XML ファイルが保管されるクライアント上のパスをリストするその sqlu_media_entry 構造を指すポインター。エクスポートした XML データは、sqlu_media_entry 構造にリストされているすべてのパスに均一に分散されます。

piXmlFileList
入力。 media_type フィールドを SQLU_CLIENT_LOCATION に設定された sqlu_media_list 構造と、基本ファイル名の入ったその sqlu_location_entry 構造を指すポインター。

このリスト内の最初の名前を使用しているネーム・スペースが使い尽くされると、 API は 2 番目の名前を使用し、以下同様に続きます。エクスポート操作での XML ファイルの作成時には、現行パス (piXmlFileList からの) にこのリストから現行ベース名を追加し、その後に 3 桁のシーケンス番号と .xml 拡張子を追加した形のファイル名が構成されます。例えば、現行の XML パスが /u/foo/xml/path ディレクトリー、現行の XML ファイル名が bar、そして XMLINSEPFILES ファイル・タイプ修飾子が設定済みの場合、作成される XML ファイルは /u/foo/xml/path/bar.001.xml、/u/foo/xml/path/bar.002.xml、などとなります。 XMLINSEPFILES ファイル・タイプ修飾子を設定しない場合、すべての XML 文書が連結されて、/u/foo/xml/path/bar.001.xml という 1 つのファイルに書き込まれます。

db2ExportIn データ構造パラメーター

piXmlSaveSchema
入力。エクスポートされた各 XML 文書を検証するのに使用される XML スキーマの SQL ID を、エクスポートされたデータ・ファイルに保管する必要があることを指示します。指定可能な値は TRUE および FALSE です。

db2ExportOut データ構造パラメーター

oRowsExported
出力。ターゲット・ファイルにエクスポートされたレコードの数を戻します。

db2gExportStruct データ構造固有パラメーター

iDataFileNameLen
入力。データ・ファイル名の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

iFileTypeLen
入力。ファイル・タイプの長さを示す 2 バイトの符号なし整数 (バイト単位) です。

iMsgFileNameLen
入力。メッセージ・ファイル名の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

使用上の注意

エクスポート操作を開始する前に、以下の 2 つの方法のいずれかで、すべての表操作を完了し、すべてのロックを解放する必要があります。

- WITH HOLD 節を使って定義したすべてのオープン・カーソルをクローズし、COMMIT ステートメントを実行して、データの変更をコミットします。
- ROLLBACK ステートメントを実行して、データ変更をロールバックします。

SELECT ステートメントでは表の別名を使用できます。

メッセージ・ファイルに置かれたメッセージには、メッセージ検索サービスから戻される情報が含まれています。各メッセージは新しい行から始まります。

エクスポート・ユーティリティーから警告が出された場合、そのメッセージは、メッセージ・ファイルに書き込まれますが、そのファイルを指定していない場合は、標準出力に書き込まれます。

外部列名配列 piDataDescriptor の列数 (sqldcol 構造の dcolnum フィールド) が、SELECT ステートメントによって生成された列数と同じでない場合には、警告メッセージが出されます。この場合、外部ファイルに書き込まれる列数はそれらのうち小さい方の数になります。出力ファイルを生成するために、余分のデータベース列または外部列名が使用されることはありません。

db2uexpm.bnd モジュールまたは配布された他の .bnd ファイルを手動でバインドする場合には、バインド・プログラムでフォーマット・オプションを使用しないでください。

DB2 Connect を使用して、DB2 for z/OS® and OS/390、DB2 for VM and VSE、および DB2 for System i® などの DRDA サーバーから表をエクスポートすることができます。PC/IXF エクスポートのみサポートされます。

PC/IXF インポートは、データベース間でデータを移動する場合に使用します。行区切り文字を含む文字データが区切り文字付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字を含むフィールドは長さが変わることがあります。

エクスポート・ユーティリティーは、AIX システムから呼び出されたときには複数部から成る PC/IXF ファイルを作成しません。

表の索引定義が PC/IXF ファイルに組み込まれるのは、単一のデータベース表の内容が、SELECT * FROM tablename で始まる piActionString パラメーターを指定して PC/IXF ファイルにエクスポートされ、piDataDescriptor パラメーターにデフォルト名が指定されているときです。ビューの索引は保管されません。piActionString の SELECT 節に結合が入っている場合も同様です。 piActionString パラメーターの WHERE 節、GROUP BY 節、または HAVING 節は索引の保管を妨げません。どの場合も、型付き表からのエクスポート時に、階層全体をエクスポートする必要があります。

提供された SELECT ステートメントが SELECT * FROM tablename の形式である場合には、エクスポート・ユーティリティーは表の NOT NULL WITH DEFAULT 属性を IXF ファイルに保管します。

型付き表をエクスポートする場合、副選択ステートメントは、ターゲット表名と WHERE 節を指定することによってのみ表現することができます。階層をエクスポートする場合、全選択と select-statement は指定できません。

IXF 以外のファイル形式の場合は、階層の全探索の方法、およびエクスポートする副表とが DB2 に知らされるよう、全探索順序リストを指定することをお勧めします。このリストが指定されていないと、階層のすべての表がエクスポートされ、OUTER 順序がデフォルトの順序になります。 OUTER 関数によって指定されるデフォルトの順序を使うこともできます。

注: インポート操作時には、同じ全探索順序を使用してください。ロード・ユーティリティーでは、階層または副階層のロードはサポートされていません。

REXX™ API 構文

```
EXPORT :stmt TO datafile OF filetype
[MODIFIED BY :filetmod] [USING :dcoldata]
MESSAGES msgfile [ROWS EXPORTED :number]


CONTINUE EXPORT


STOP EXPORT
```

REXX API パラメーター

stmt 有効な動的 SQL SELECT ステートメントを含む REXX ホスト変数。このステートメントにより、データベースから取り出すデータが指定されます。

datafile
データのエクスポート先となるファイルの名前。

filetype
エクスポート・ファイルのデータの形式。サポートされているファイル形式は、以下のとおりです。

DEL 区切り文字付き ASCII

WSF ワークシート形式

IXF 統合交換フォーマットの PC バージョン。

filetmod
追加の処理オプションを含むホスト変数。

dcoldata
エクスポート・ファイルで使用する列名を含むコンパウンド REXX ホスト変数。以下の項目において、XXX はホスト変数の名前を表しています。

XXX.0 列数 (残りの変数内のエレメントの数)

XXX.1 最初の列名。

XXX.2 2 番目の列名。

XXX.3 以降、3 番目、4 番目 ... と続きます。

このパラメーターが NULL の場合、または dcoldata に値が指定されていない場合、ユーティリティーはデータベース表からの列名を使用します。

msgfile
エラーおよび警告メッセージが送られるファイル、パス、または装置の名前。

number
エクスポートされた行の数が入れられるホスト変数。

エクスポート・セッション - CLP の例

例 1
次の例は、SAMPLE データベースの中の STAFF 表から myfile.ixf へ、IXF フォーマットの出力で情報をエクスポートする方法を示しています。ユーザーはすでにデータベースに接続していることが必要です。データベース接続が DB2 Connect を介していない場合、索引定義が存在するならそれは出力ファイルに格納されます。そうでなければ、データだけが格納されます。

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

例 2
次の例は、SAMPLE データベースの中の STAFF 表から awards.ixf へ、部署 (dept) 20 の従業員に関する情報を IXF フォーマットの出力でエクスポートする方法を示しています。ユーザーはすでにデータベースに接続していることが必要です。

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
   where dept = 20
```

例 3次の例は LOB を DEL ファイルにエクスポートする方法を示しています。

```
db2 export to myfile.del of del lobs to mylobs/
   lobfile lobs1, lobs2 modified by lobsinfile
   select * from emp_photo
```

例 4
次の例は LOB を DEL ファイルにエクスポートする方法を示しています。ここでは、最初のディレクトリーにファイルを入れることができない場合のために 2 番目のディレクトリーを指定しています。

```
db2 export to myfile.del of del
   lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
   select * from emp_photo
```

例 5 次の例はデータを DEL ファイルにエクスポートする方法を示しています。ここでは、単一引用符をストリング区切り文字として使用し、セミコロンを列の区切り文字として使用し、コンマを小数点として使用します。データを再びデータベースにインポートする場合、これと同じ規則を使用する必要があります。

```
db2 export to myfile.del of del
   modified by chardel'' coldel; decpt,
   select * from staff
```

第 3 章 インポート・ユーティリティー

インポートの概要

インポート・ユーティリティーは、SQL INSERT ステートメントを使用して、表、型付き表、ビューにデータを取り込むためのユーティリティーです。インポート・データを受け取る表またはビューにすでにデータが入っている場合は、既存のデータを入力データで置き換えるか、既存のデータに入力データを追加するかのいずれかを選択できます。

エクスポートと同じように、インポートも比較的シンプルなデータ移動ユーティリティーです。このユーティリティーを実行するには、コントロール・センターを使用するか、CLP コマンドを実行するか、ADMIN_CMD ストアード・プロシージャーを呼び出すか、ユーザー・アプリケーションで db2Import API を呼び出します。

インポートでサポートされているデータ・フォーマットや、インポートで使用できるフィーチャーがいくつかあります。

- インポートのデータ・フォーマットとしては、IXF、WSF、ASC、DEL がサポートされています。
- インポート操作をカスタマイズするために、ファイル・タイプ修飾子を一緒に使用できます。
- インポートを使用して、階層データや型付き表を移動することもできます。
- インポートは、すべてのアクティビティーをログに記録し、索引を更新し、制約を検証し、トリガーを起動します。
- インポートでは、データの挿入先になる表またはビューの列の名前を指定できます。
- インポートは、DB2 Connect でも使用できます。

インポート・モード

インポートには、データのインポート方式を示す 5 つのモードがあります。最初の 3 つ (INSERT、INSERT_UPDATE、REPLACE) は、ターゲット表がすでに存在する場合に使用します。その 3 つとも、データ・フォーマットとして、
IXF、WSF、ASC、DEL をサポートしています。ただし、ニックネームで使用できるのは、INSERT と INSERT_UPDATEだけです。

表 11. INSERT、INSERT_UPDATE、および REPLACE インポート・モードの概要

モード	最良事例の使用法
INSERT	入力データをターゲット表に挿入します (既存のデータは変更しません)
INSERT_UPDATE	主キー値が一致する行を入力行の値で更新します 一致する行がない場合は、インポート行を表にそのまま挿入します

表 11. INSERT、INSERT_UPDATE、および REPLACE インポート・モードの概要 (続き)

モード	最良事例の使用法
REPLACE	既存のデータをすべて削除し、インポート・データを挿入します (表と索引の定義はそのまま維持します)

他の 2 つのモード (REPLACE_CREATE、CREATE) は、ターゲット表が存在しない場合に使用します。これらのモードを使用できるのは、入力ファイルが PC/IXF 形式であり、作成する表の構造記述がそのファイルに入っている場合に限られます。オブジェクト表にそれ自体以外の従属表がある場合は、これらのモードでインポートを実行できません。

注: インポートの CREATE モードと REPLACE_CREATE モードは、非推奨になります。代わりに、db2look ユーティリティーを使用してください。

表 12. REPLACE_CREATE および CREATE インポート・モードの概要

モード	最良事例の使用法
REPLACE_CREATE	既存のデータをすべて削除し、インポート・データを挿入します (表と索引の定義はそのまま維持します) ターゲット表と索引が存在しない場合は、ターゲット表と索引を作成します
CREATE	ターゲット表と索引を作成します 新しい表を作成する表スペースの名前を指定できます

インポートのしくみ

インポートに必要なステップの数と時間の長さは、移動するデータの量と指定するオプションによって異なります。インポート操作の各ステップは、次のような流れになります。

1. 表をロッキングします
   既存のターゲット表で並行アクセスを許可するかどうかに応じて、インポートは、そのターゲット表の排他 (X) ロックまたは非排他 (IX) ロックを取得します。
2. データを検出して取り出します
   インポートは、FROM 節を使用して、入力データを検出します。XML データや LOB データが存在することをコマンドで指定していれば、インポートは、その種のデータも検出します。
3. データを挿入します
   インポートは、既存のデータを置き換えるか、新しいデータ行を表に追加します。
4. 制約をチェックして、トリガーを起動します
   インポートは、データを書き込むときに、挿入するそれぞれの行が、ターゲット表で定義されている制約に準拠しているかどうかを確認します。リジェクトされた行の情報は、メッセージ・ファイルに書き込まれます。インポートは、既存のトリガーも起動します。

5. 操作をコミットします
   インポートは、変更内容を保管して、ターゲット表のロックを解放します。インポートの処理中に、コミットを周期的に実行するように指定することも可能です。

基本的なインポート操作で必須の項目は、以下のとおりです。

- 入力ファイルのパスと名前
- ターゲット表またはターゲット・ビューの名前または別名
- 入力ファイル内のデータのフォーマット
- データをインポートする方式
- トラバース順序 (階層データをインポートする場合)
- 副表リスト (型付き表をインポートする場合)

追加のオプション

インポート操作をカスタマイズするために使用できるオプションがいくつかあります。MODIFIED BY 節にファイル・タイプ修飾子を指定すれば、データ・フォーマットを変更したり、インポート・ユーティリティーによるデータ処理の方法を指定したり、パフォーマンスを改善したりすることが可能になります。

インポート・ユーティリティーのデフォルトの動作では、正常なインポートが終了するまでコミットは実行されません (ただし、一部の ALLOW WRITE ACCESS インポートの場合は例外です)。このデフォルトの動作では、インポートの速度は上がりますが、並行性、リスタートのしやすさ、アクティブ・ログのスペースなどの考慮事項からすると、インポートの処理中にコミットを実行するように指定する方が望ましい場合もあります。そのためには、**COMMITCOUNT** パラメーターを「automatic」に設定するという方法があります。この場合、インポートは、コミットを実行するタイミングを内部で決定します。あるいは、**COMMITCOUNT** に特定の数値を設定することも可能です。その場合、インポートは、指定の数のレコードがインポートされるたびにコミットを実行します。

インポートのパフォーマンスを改善するための方法もいくつかあります。インポート・ユーティリティーは、組み込み SQL アプリケーションであり、内部で SQL フェッチを実行するので、SQL 操作に当てはまる最適化の手法は、インポートにも当てはまります。インポートでは、1 行ずつ挿入していくのがデフォルトの動作ですが、`compound` ファイル・タイプ修飾子を使用すれば、指定の数の行を一挙に挿入することが可能になります。インポートの実行時に多数の警告が生成される (つまり、それだけ操作の速度が低下する) ことが予想される場合は、`norowwarnings` ファイル・タイプ修飾子を指定して、リジェクトされた行に関する警告を抑制することもできます。

メッセージ・ファイル

インポートの実行時には、その操作に関連したエラー・メッセージ、警告メッセージ、通知メッセージを格納した標準の ASCII テキスト・メッセージ・ファイルが書き出されます。アプリケーション・プログラミング・インターフェース (API) db2Import でユーティリティーを呼び出す場合は、そのファイルの名前を事前に **MESSAGES** パラメーターで指定しておく必要がありますが、それ以外の場合にファイル名を指定するかどうかは任意です。メッセージ・ファイルは、インポートの進行状況をモニターするための便利な方法です。メッセージ・ファイルには、イン

ポートの進行中にもアクセスできるからです。インポート操作が失敗した場合は、メッセージ・ファイルを使用して、正常にインポートされた最後の行を確認することによって、リスタート・ポイントを判別できます。

注: リモート・データベースに対するインポート操作で生成される出力メッセージの量が 60 KB を超えると、ユーティリティーは、最初の 30 KB と最後の 30 KB を維持します。

インポートを使用するために必要な特権と権限

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャーの保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスが制御されます。

ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

SYSADM 権限または DBADM 権限があれば、どんなタイプのインポート操作でも実行できます。インポートのタイプごとに、インポートにかかわるそれぞれの表、ビュー、ニックネームで必要になる他の権限を以下の表にまとめておきます。

表 13. インポート操作を実行するために必要な権限

モード	必要な権限
INSERT	CONTROL または INSERT、SELECT
INSERT_UPDATE	CONTROL または INSERT、SELECT、UPDATE、および DELETE
REPLACE	CONTROL または INSERT、SELECT、および DELETE
REPLACE_CREATE	ターゲット表が存在する場合: CONTROL または INSERT、SELECT、および DELETE ターゲット表が存在しない場合: CREATETAB (データベース)、USE (表スペース)、および スキーマが存在しない場合: IMPLICIT_SCHEMA (データベース) または スキーマが存在する場合: CREATEIN (スキーマ)
CREATE	CREATETAB (データベース)、USE (表スペース)、および スキーマが存在しない場合: IMPLICIT_SCHEMA (データベース) または スキーマが存在する場合: CREATEIN (スキーマ)

注: IMPORT コマンドの **CREATE** オプションと **REPLACE_CREATE** オプションは非推奨で、将来のリリースでは廃止される可能性があります。
さらに、**REPLACE** オプションまたは **REPLACE_CREATE** オプションを使用するには、表をドロップするための権限がセッション許可 ID に必要です。

階層にインポートする場合は、必要な権限がモードによっても変わってきます。階層がすでに存在している場合に **REPLACE** 操作を実行するには、階層内のすべての副表に関する CONTROL 特権が必要です。階層が存在しない場合に **REPLACE_CREATE** 操作を実行するには、階層内のすべての副表に関する CONTROL 特権のほかに CREATETAB と USE が必要です。

さらに、LBAC (ラベル・ベースのアクセス制御) セキュリティー・ラベルが定義されている表にインポートする場合も、いくつかの考慮事項があります。保護列を持つ表にデータをインポートするには、表内のすべての保護列への書き込みアクセスを可能にする LBAC 信用証明情報がセッション許可 ID に必要です。保護行を持つ表にデータをインポートするには、表を保護するセキュリティー・ポリシーの一部である、書き込みアクセス用のセキュリティー・ラベルが、セッション許可 ID に対して付与されていなければなりません。

データのインポート

インポート・ユーティリティーは、サポートされているファイル・フォーマットを用いて、外部ファイルから表、階層、ビュー、またはニックネームにデータを挿入します。ロード・ユーティリティーは、高速な代替方法ですが、ロード・ユーティリティーでは、階層レベルのデータのロードはサポートされていません。

インポート・ユーティリティーを起動するには、その前にデータのインポート先となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。 DB2 for Linux、UNIX、または Windows クライアントから DB2 for Linux、UNIX、または Windows データベース・サーバーへのユーティリティーのアクセスは、DB2 Connect ゲートウェイまたはループバック環境を介してではなく、エンジンからの直接接続でなければなりません。このユーティリティーは COMMIT または ROLLBACK ステートメントを発行するため、インポートの起動前に COMMIT ステートメントまたは ROLLBACK 操作を実行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておいてください。

注: IMPORT コマンドの CREATE オプションと REPLACE_CREATE オプションは非推奨で、将来のリリースでは廃止される可能性があります。

インポート・ユーティリティーには、以下の制約事項が適用されます。

- 既存の表が、従属表の外部キーから参照される主キーを備えた親表である場合、そのデータは置換できず、追加だけが可能です。
- REFRESH IMMEDIATE モードで定義されたマテリアライズ照会表の基礎表へのインポート置換操作は実行できません。
- システム表、サマリー表、または構造化タイプ列を持つ表にデータをインポートすることはできません。
- 宣言された一時表にデータをインポートすることはできません。
- インポート・ユーティリティーを介してビューを作成することはできません。
- PC/IXF ファイルから表を作成する場合、参照制約と外部キー定義は保存されません。 (主キー定義は、データが前に SELECT * を使ってエクスポートされた場合、保存されます。)

- インポート・ユーティリティーは独自の SQL ステートメントを生成するので、場合によっては最大ステートメント・サイズの 2 MB を超えることがあります。
- CREATE または REPLACE_CREATE インポート・オプションを使用してパーティション表またはマルチディメンション・クラスタリング表 (MDC) を再作成することはできません。
- XML 列を含む表を再作成することはできません。
- 暗号化されたデータはインポートできません。
- インポート置換操作では、Not Logged Initially 節は考慮されません。IMPORT コマンドの REPLACE オプションは、CREATE TABLE ステートメントの NOT LOGGED INITIALLY (NLI) 節や ALTER TABLE ステートメントの ACTIVATE NOT LOGGED INITIALLY 節を考慮しません。NLI が呼び出される CREATE TABLE ステートメントまたは ALTER TABLE ステートメントと同じトランザクション内で REPLACE アクションを伴うインポートを実行する場合、そのインポートでは NLI 節は考慮されません。挿入はすべてログに記録されます。

  対処策 1: DELETE ステートメントを使用して表の内容を削除してから、INSERT ステートメントによってインポートを呼び出します。

  対処策 2: 表をドロップしてから再作成した後、INSERT ステートメントによってインポートを呼び出します。

インポート・ユーティリティーに適用される制限: リモート・データベースに対するインポート操作で生成された出力メッセージの量が 60 KB を超える場合、このユーティリティーは最初の 30 KB と最後の 30 KB を維持します。

インポート・ユーティリティーを起動するには、コマンド行プロセッサー (CLP) を使用するか、コントロール・センターの「表のインポート」ノートブックを使用するか、クライアント・アプリケーションからアプリケーション・プログラミング・インターフェース (API) の db2Import を呼び出します。

「表のインポート」ノートブックの使用

1. コントロール・センターから、「表」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
2. 「表」フォルダーをクリックします。ウィンドウの右側部分 (コンテンツ・ペイン) に、既存の表がすべて表示されます。
3. コンテンツ・ペイン内で対象となる表を右クリックし、ポップアップ・メニューから「インポート」を選択します。「表のインポート」がオープンします。

「表のインポート」についての詳細情報は、コントロール・センターのオンライン・ヘルプ機能を通して提供されます。

CLP による IMPORT コマンドの実行

シンプルなインポート操作の場合に指定する必要があるのは、入力ファイル、ファイル形式、インポート・モード、ターゲット表 (または作成する表の名前) だけです。

例えば、CLP からデータをインポートするには、以下の IMPORT コマンドを入力します。

```
db2 import from filename of fileformat import_mode into table
```

filename は、インポートするデータが含まれている入力ファイルの名前、ixf はファイル形式、insert はモード、table は、データの挿入先の表の名前です。

ただし、警告メッセージやエラー・メッセージを書き込むメッセージ・ファイルを指定することも可能です。そのためには、**MESSAGES** パラメーターとメッセージ・ファイル名を追加します。そのコマンドは、次のようになります。

```
db2 import from filename of fileformat messages messagefile import_mode into table
```

構文と使用法の詳細については、「IMPORT コマンド」を参照してください。

XML データのインポート

インポート・ユーティリティーを使用することにより、 DB2 Database for Linux, UNIX, and Windows ソース・データ・オブジェクトの表名またはニックネームを使用して XML データを XML 表列にインポートできます。

データを XML 表列にインポートするとき、XML FROM オプションを使用して入力 XML データ (1 つまたは複数) のパスを指定できます。例えば、以前にエクスポートされた XML ファイル "/home/user/xmlpath/xmldocs.001.xml" では、以下のコマンドを使用してデータを表にインポートして戻すことができます。

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

スキーマに対して挿入された文書を検証する

XMLVALIDATE オプションにより、XML 文書がインポートされる際に、それらが XML スキーマに準拠しているかどうかを検証できます。次の例では、XML 文書がエクスポートされた時点で保存されたスキーマ情報に準拠しているかどうかに関して、着信 XML 文書が検証されます。

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
     USING XDS INSERT INTO USER.T1
```

構文解析オプションの指定

XMLPARSE オプションを使用して、インポートされた XML 文書の空白文字を保存するかストリップするかを指定できます。次の例では、XML 文書がエクスポートされたときに保管された XML スキーマ情報に対してすべてのインポートされた XML 文書が検証されて、これらの文書は空白文字を保存しながら構文解析されます。

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
WHITESPACE XMLVALIDATE USING XDS INSERT INTO USER.T1
```

インポート表の再作成

インポート・ユーティリティーの CREATE モードを使用すれば、エクスポート・ユーティリティーによって保管した表を再作成できます。ただし、このプロセスにはいくつかの制約があり、入力表の属性の多くが保持されません。

インポート時に表を再作成するには、エクスポート操作でいくつかの要件を満たしておく必要があります。まず、元の表を IXF ファイルにエクスポートしなければなりません。ファイル形式 DEL または ASC のファイルをエクスポートした場合

は、出力ファイルにレコード・データは組み込まれますが、ターゲット表の記述は組み込まれません。このようなファイル・フォーマットのデータを持つ表を再作成するには、ターゲット表を作成してから、ロードまたはインポート・ユーティリティーを使用して、これらのファイルから表にデータを入れます。元の表定義をキャプチャーし、対応するデータ定義言語 (DDL) を生成するには、db2look ユーティリティーを使用します。さらに、エクスポート時に使用する SELECT ステートメントには、特定のアクション・ストリングしか組み込めません。例えば、SELECT 節では列名を使用できません。使用できるのは、SELECT * だけです。

注: インポートの CREATE モードは、非推奨になります。表のキャプチャーと再作成には、db2look ユーティリティーを使用してください。

保持される属性

元の表の属性のうち、表の再作成時に保持される属性は、以下のとおりです。

- 主キーの名前および定義
- 以下のような列情報
  - 列名
  - 列データ・タイプ。ユーザー定義の特殊タイプはその基本タイプとして保存されます。
  - ID プロパティー
  - 長さ (lob_file タイプの場合を除く)
  - コード・ページ (該当する場合)
  - ID オプション
  - 列が NULL 可能または不可のどちらで定義されているか
  - 定数のデフォルト値 (ある場合)。ただし他のタイプのデフォルト値は該当しません。
- 以下のような索引情報
  - 索引名
  - 索引の作成者名
  - 列名と、昇順または降順のどちらで各列がソートされるか
  - 索引がユニーク索引として定義されているかどうか
  - 索引がクラスター化されているかどうか
  - 索引で反転スキャンが可能かどうか
  - PCTFREE 値
  - MINPCTUSED 値

注: 索引の列名に文字 - または + が含まれていると、索引情報は保持されません。その場合は、SQL27984W が返されます。

失われる属性

元の表の属性のうち、表の再作成時に保持されない属性は、以下のとおりです。

- ソースが、通常の表、マテリアライズ照会表 (MQT)、ビュー、またはこれらのソースの全部または一部から取られた列セットのうちのどれであったか
- ユニーク制約およびその他のタイプの制約、またはトリガー (主キー制約を含まない)

- 以下のような表情報
  - MQT 定義 (該当する場合)
  - MQT オプション (該当する場合)
  - 表スペース・オプション。ただしこの情報は、IMPORT コマンドを使って指定することができます。
  - マルチディメンション・クラスタリング (MDC) ディメンション
  - パーティション表ディメンション
  - 表パーティション・キー
  - NOT LOGGED INITIALLY プロパティー
  - チェック制約
  - 表のコード・ページ
  - 保護表のプロパティー
  - 表または値の圧縮オプション
- 以下のような列情報
  - 定数値以外の任意のデフォルト値
  - LOB オプション (ある場合)
  - XML プロパティー
  - CREATE TABLE ステートメントの references 節 (ある場合)
  - 参照制約 (ある場合)
  - チェック制約 (ある場合)
  - 生成列オプション (ある場合)
  - データベースの有効範囲シーケンスに依存する列
- 以下のような索引情報
  - INCLUDE 列 (存在する場合)
  - 索引が主キー索引である場合は、索引名
  - 索引が主キー索引である場合は、降順のキー (デフォルトは昇順)
  - 索引列名に 16 進値 0x2B または 0x2D が含まれる
  - コード・ページ変換後に 128 バイトを超える索引名
  - PCTFREE2 値
  - ユニーク制約

注: このリストですべてを取り上げているわけではないので、このリストを使用するときには注意が必要です。

インポートが失敗し、SQL3311N が返された場合でも、ファイル・タイプ修飾子 `forcecreate` を使用すれば、表を再作成できます。この修飾子を使用すると、情報が欠落したまま、限られた情報だけで表が作成されます。

型付き表のインポートに関する考慮事項

インポート・ユーティリティーを使用すれば、型付き表どうしの間で既存のデータ階層を保持しながらデータを移動できます。必要に応じて、インポート時に表階層と型階層を作成することも可能です。

ある階層構造の型付き表から別の階層構造の型付き表にデータを移動するには、エクスポート操作で特定のトラバース順序を指定し、中間のフラット・ファイルを作成します。一方、インポート・ユーティリティーでは、CREATE、INTO *table-name*、UNDER、AS ROOT TABLE の各パラメーターによって、移動する階層のサイズと配置を制御します。さらに、インポートでは、ターゲット・データベースに格納する内容も制御します。例えば、それぞれの副表名の最後に属性リストを指定することによって、ターゲット・データベースに移動する属性を制限できます。属性リストを使用しない場合は、それぞれの副表のすべての列が移動されます。

表の再作成
実行できるインポートのタイプは、入力ファイルのファイル形式によって異なります。ASC、DEL、WSF のデータを操作する場合は、データをインポートする前にターゲット表または階層が存在している必要があります。一方、PC/IXF ファイルのデータは、表または階層が存在していない場合でも、CREATE のインポート操作を指定することによってインポートできます。ただし、CREATE オプションを指定しても、インポート時に副表の定義を変更することはできません。

トラバース順序
入力ファイルに含まれているトラバース順序によって、データ階層を維持することが可能になります。したがって、エクスポート・ユーティリティーを起動する場合とインポート・ユーティリティーを起動する場合では、同じトラバース順序を使用しなければなりません。

PC/IXF ファイル形式の場合は、ターゲットの副表の名前を指定する必要があるだけで、トラバース順序については、ファイルに格納されているデフォルトのトラバース順序が使用されます。

型付き表で CREATE 以外のオプションを使用する場合は、トラバース順序リストによってトラバース順序を指定できます。このユーザー指定のトラバース順序は、エクスポート操作時に使用されたトラバース順序と一致していなければなりません。インポート・ユーティリティーは、以下の条件が満たされていれば、ターゲット・データベースにデータを正確に移動できます。

- ソースとターゲットの両方のデータベースで、副表の定義が同じである。
- ソースとターゲットの両方のデータベースで、副表間の階層リレーションシップが同じである。
- トラバース順序が同じである。

トラバース順序を定義する場合、開始点と階層を下るパスを決定できますが、それぞれの分岐の最後までトラバースが終わってからでないと、階層内の次の分岐のトラバースを開始できません。インポート・ユーティリティーは、指定されたトラバース順序がこの条件に違反していないかどうかを検査します。

例

この項の例は、以下の 4 つの有効なトラバース順序がある階層構造に基づいています。

- Person、Employee、Manager、Architect、Student
- Person、Student、Employee、Manager、Architect

- Person、Employee、Architect、Manager、Student
- Person、Student、Employee、Architect、Manager

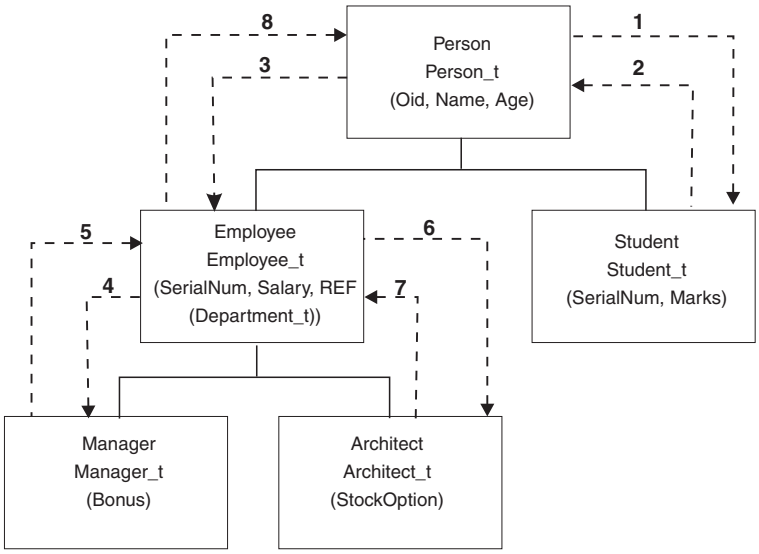


図 2. 階層の例

例 1

インポートを使用して階層 (前のエクスポート操作で作成したデータ・ファイル entire_hierarchy.ixf に含まれている階層) 全体を再作成するには、以下のコマンドを入力します。

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.ixf OF IXF CREATE INTO
   HIERARCHY STARTING Person AS ROOT TABLE
```

階層内のそれぞれの型が存在しない場合は、それが作成されます。型がすでに存在している場合は、ターゲット・データベースとソース・データベースとで同じ定義でなければなりません。もし同じでなければ、SQL エラー (SQL20013N) が戻されます。ここでは新たに階層を作成しているので、ターゲット・データベース (Target_db) に移動されるデータ・ファイルで定義されている副表がその中に存在していてはなりません。ソース・データベース階層のすべての表が新たに作成されます。ソース・データベースから移されるデータは、ターゲット・データベース内の適切な副表にインポートされます。

例 2

ソース・データベースの階層全体を再作成し、その階層をターゲット・データベースにインポートするときに、選択したデータだけを維持するには、以下のコマンドを入力します。

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.del OF DEL INSERT INTO (Person,
   Employee(Salary), Architect) IN HIERARCHY (Person, Employee,
   Manager, Architect, Student)
```

ターゲット表 PERSON、EMPLOYEE、および ARCHITECT はすでに存在していなければなりません。データは副表 PERSON、EMPLOYEE、および ARCHITECT の中にインポートされます。つまり、次のようにインポートされます。

- PERSON 内のすべての列を PERSON へ
- PERSON のすべての列、および EMPLOYEE 内の SALARY を EMPLOYEE へ
- PERSON のすべての列、EMPLOYEE 内の SALARY、および ARCHITECT のすべての列を ARCHITECT へ

SerialNum 列と REF(Employee_t) 列は、EMPLOYEE またはその副表 (つまり、データがインポートされている唯一の副表である ARCHITECT) にインポートされません。

注: ARCHITECT は EMPLOYEE の副表であり、EMPLOYEE に指定されている唯一のインポート列は SALARY なので、ARCHITECT にインポートされる EMPLOYEE 固有の列は SALARY だけになります。つまり、SerialNum 列と REF(Employee_t) 列はいずれも、EMPLOYEE 行や ARCHITECT 行にインポートされません。
表 MANAGER と表 STUDENT のデータはインポートされません。

例 3
次の例では、正規の表からエクスポートした後、ある階層内の 1 つの副表としてインポートします。 EXPORT コマンドが正規の表 (非型付き表) で実行されるので、データ・ファイルの中に Type_id 列はありません。これはファイル・タイプ修飾子 no_type_id を指定して示します。そうすれば、インポート・ユーティリティーは、最初の列が Type_id 列であることを予期しなくなります。

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO Student_sub_table.del OF DEL SELECT * FROM
   Regular_Student
DB2 CONNECT TO Target_db
DB2 IMPORT FROM Student_sub_table.del OF DEL METHOD P(1,2,3,5,4)
   MODIFIED BY NO_TYPE_ID INSERT INTO HIERARCHY (Student)
```

この例では、ターゲット表 STUDENT がすでに存在していなければなりません。STUDENT は副表なので、最初の列に Type_id が存在しないことを示すために、修飾子 no_type_id を使用します。しかし、Object_id 列、および STUDENT 表内の他のすべての属性が存在することを確認する必要があります。 STUDENT 表にインポートされるどの行でも、その最初の列は Object-id であると見なされます。
METHOD 節によって、最後の 2 つの属性の順序が逆転します。

LBAC で保護されたデータのインポートに関する考慮事項

保護行が含まれている表へのインポート操作を正常に実行するには、LBAC (ラベル・ベースのアクセス制御) 信用証明情報が必要です。さらに、ターゲット表に関連付けられているセキュリティー・ポリシーで有効なセキュリティー・ラベル、または有効なセキュリティー・ラベルに変換できるセキュリティー・ラベルを用意することも必要です。

有効な LBAC 信用証明情報がなければ、インポートは失敗し、エラー (SQLSTATE 42512) が返されます。入力データにセキュリティー・ラベルが含まれていない場合や、セキュリティー・ラベルが内部のバイナリー・フォーマットでない場合は、いくつかのファイル・タイプ修飾子を使用して、インポート操作を進めることができます。

保護行が含まれている表にデータをインポートする場合は、ターゲット表にデータ・タイプ DB2SECURITYLABEL の列が 1 つ含まれています。入力データ行にその列の値が含まれていなければ、その行はリジェクトされます。ただし、インポート・コマンドに usedefaults ファイル・タイプ修飾子が指定されている場合は例外です。その場合に、その表を保護しているセキュリティー・ポリシーで書き込みアクセスが認められているセキュリティー・ラベルがあれば、そのセキュリティー・ラベルが使用されます。書き込みアクセスが認められているセキュリティー・ラベルがなければ、その行はリジェクトされ、処理が次の行に移ります。

保護行が含まれている表にデータをインポートする場合に、入力データにデータ・タイプ DB2SECURITYLABEL の列の値が含まれていれば、その表にデータを挿入する場合と同じ規則が当てはまります。インポート対象の行を保護しているセキュリティー・ラベル (データ・ファイル内のその行にあるセキュリティー・ラベル) が書き込み可能であれば、その行の保護のためにそのセキュリティー・ラベルが使用されます。(つまり、そのセキュリティー・ラベルがデータ・タイプ DB2SECURITYLABEL の列に書き込まれます。)そのセキュリティー・ラベルで保護されている行への書き込みができない場合の動作は、ソース表を保護しているセキュリティー・ポリシーの作成方法によって異なります。

- ポリシーを作成した CREATE SECURITY POLICY ステートメントにオプション RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL が含まれていた場合には、挿入は失敗し、エラーが戻されます。
- CREATE SECURITY POLICY ステートメントにそのオプションが組み込まれていなかった場合や、 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL オプションが代わりに組み込まれていた場合は、データ・ファイルに含まれているその行のためのセキュリティー・ラベルは無視されます。書き込みアクセスが認められているセキュリティー・ラベルがあれば、その行の保護のためにそのセキュリティー・ラベルが使用されます。この場合、エラーや警告は出されません。書き込みアクセスが認められているセキュリティー・ラベルがなければ、その行はリジェクトされ、処理が次の行に移ります。

区切り文字に関する考慮事項

データ・タイプ DB2SECURITYLABEL の列にデータをインポートする場合、デフォルトでは、データ・ファイル内の値は、そのセキュリティー・ラベルの内部表記を構成する実際のバイトと見なされます。ただし、生データには改行文字が含まれている場合があり、その場合、IMPORT コマンドは、その改行文字を行の区切り文字と誤解してしまう可能性があります。この問題がある場合は、行区切り文字よりもストリング区切り文字を優先するために、delprioritychar ファイル・タイプ修飾子を使用します。delprioritychar を使用すると、ストリング区切り文字の中にあるレコード区切り文字または列区切り文字は、区切り文字と見なされなくなります。delprioritychar ファイル・タイプ修飾子は、改行文字が含まれている値が存在しない場合に使用しても差し支えありません。ただし、インポートの速度は少し低下します。

インポート対象のデータが ASC 形式の場合は、インポートされたセキュリティー・ラベルやセキュリティー・ラベル名に末尾空白が組み込まれる事態を回避するために、追加の手順を実行できます。ASCII フォーマットでは、列位置が区切りと

して使用されるので、可変長フィールドにインポートするときにこの問題が発生する可能性があります。末尾ブランク・スペースを切り捨てるには、striptblanks ファイル・タイプ修飾子を使用します。

非標準のセキュリティー・ラベル値
セキュリティー・ラベルの値が、セキュリティー・ラベルの各コンポーネントの値を含んだストリングになっている場合、例えば、*S:(ALPHA,BETA)* のようになっている場合でも、データ・ファイルをインポートできます。そのためには、ファイル・タイプ修飾子 seclabelchar を使用する必要があります。 seclabelchar を使用すると、データ・タイプ DB2SECURITYLABEL の列の値は、セキュリティー・ラベル用のストリング・フォーマットで記述されたセキュリティー・ラベルを含む、ストリング定数と見なされます。ストリングが正しい形式でなければ、その行は挿入されず、警告 (SQLSTATE 01H53) が返されます。そのストリングが、表を保護しているセキュリティー・ポリシーに組み込まれている有効なセキュリティー・ラベルを記述したものでない場合も、その行は挿入されず、警告 (SQLSTATE 01H53) が返されます。

セキュリティー・ラベル列の値がセキュリティー・ラベル名になっている場合も、データ・ファイルをインポートできます。この種のファイルをインポートするには、ファイル・タイプ修飾子 seclabelname を使用する必要があります。seclabelname を使用すると、データ・タイプ DB2SECURITYLABEL の列のすべての値は、既存のセキュリティー・ラベルの名前を含んだストリング定数と見なされます。表を保護しているセキュリティー・ポリシーでその名前のセキュリティー・ラベルが存在しなければ、その行は挿入されず、警告 (SQLSTATE 01H53) が返されます。

例

すべての例で、入力データ・ファイル myfile.del は DEL 形式になっています。すべての場合のデータのインポート先は、以下のステートメントで作成された REPS という名前の表です。

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

次の例の入力ファイルには、デフォルト形式のセキュリティー・ラベルが含まれていると見なされます。

```
db2 import from myfile.del of del modified by delprioritychar insert into reps
```

次の例の入力ファイルには、セキュリティー・ラベル・ストリング・フォーマットのセキュリティー・ラベルが含まれていると見なされます。

```
db2 import from myfile.del of del modified by seclabelchar insert into reps
```

次の例の入力ファイルには、セキュリティー・ラベル列のセキュリティー・ラベル名が含まれていると見なされます。

```
db2 import from myfile.del of del modified by seclabelname insert into reps
```

バッファー挿入インポート

パーティション・データベース環境では、インポート・ユーティリティーでバッファー挿入を使用可能にすることができます。そうすることにより、データのインポート時に発生するメッセージングが少なくなり、パフォーマンスが向上します。

バッファー挿入オプションを有効にすると、失敗したバッファー挿入に関する詳細が返されなくなるので、そのオプションを有効にするのは、エラー・レポートに関する心配がない場合に限ってください。

バッファー挿入を使用すると、インポートではデフォルトの **WARNINGCOUNT** 値が 1 に設定されます。そのため、行が 1 つでもリジェクトされると、操作は失敗します。レコードがリジェクトされた場合、ユーティリティーは現在のトランザクションをロールバックします。コミット済みのレコード数を調べれば、どのレコードが正常にデータベースに挿入されたかを判別することができます。コミット済みのレコード数がゼロ以外になりうるのは、COMMITCOUNT オプションを指定していた場合のみです。

別の **WARNINGCOUNT** 値を IMPORT コマンドに明示的に指定していた場合にいずれかの行がリジェクトされると、ユーティリティーからの行サマリー出力は誤っていることがあります。その原因は、バッファー挿入で使用される非同期エラー・レポートと、行グループの挿入時にエラーが検出されたことに起因したそのグループのすべての取り消し行が組み合わさったことにあります。どの入力レコードがリジェクトされたかに関するユーティリティーからのレポートは信頼できないので、どのレコードがコミット済みで、どのレコードをデータベースに再挿入する必要があるかを判別するのは困難になります。

バッファー挿入を要求するには、DB2 バインド・ユーティリティーを使用します。INSERT BUF オプションを使用して、データベースに対してインポート・パッケージ db2uimpm.bnd を再バインドする必要があります。以下に例を示します。

```
db2 connect to your_database
db2 bind db2uimpm.bnd insert buf
```

バッファー挿入フィーチャーと INSERT_UPDATE モードのインポート操作を併用することはできません。バインド・ファイル db2uImpInsUpdate.bnd は、この制限を課します。 INSERT BUF オプションを指定してこのファイルをバインドするべきではありません。そのようにすると、INSERT_UPDATE モードのインポート操作は失敗します。ただし、INSERT、REPLACE、REPLACE_CREATE のいずれかのモードのインポート操作は、この新しいファイルのバインドによる影響を受けません。

ID 列のインポートに関する考慮事項

インポート・ユーティリティーでは、入力データに ID 列の値があるかどうかにかかわりなく、ID 列が含まれている表にデータをインポートできます。

ID 関連のファイル・タイプ修飾子が使用されない場合、このユーティリティーは次のような規則に従って動作します。

- ID 列が GENERATED ALWAYS の場合、入力ファイル内の対応する行の中に ID 列用の値がないか、または NULL 値が明示的に指定されているときに、表の行用の ID 値が生成されます。 ID 列に非 NULL 値を指定すると、その行はリジェクトされます (SQL3550W)。

- ID 列が GENERATED BY DEFAULT の場合、ユーザー提供値が指定されていれば、インポート・ユーティリティーはその値を使用します。データが欠落しているかまたは明示的に NULL であれば、値が生成されます。

インポート・ユーティリティーは、ユーザー提供の ID 値に関して、 ID 列のデータ・タイプ (SMALLINT、INT、BIGINT、または DECIMAL) の値に対して通常行う以外の余分な妥当性検査を行いません。値が重複していても報告されません。しかも、 ID 列を持つ表へのデータのインポート時には compound=*x* 修飾子を使用できません。

ID 列が含まれている表にデータをインポートする操作を簡略化するための 2 つの方法があります。つまり、ファイル・タイプ修飾子として、identitymissing と identityignore を使用できます。

ID 列のないデータのインポート

identitymissing 修飾子は、入力データ・ファイルに ID 列の値が入っていない (NULL すらない) 場合に、 ID 列を持つ表のインポートを容易にします。例えば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 char(30),
                     c2 int generated by default as identity,
                     c3 real,
                     c4 char(1))
```

ユーザーが、データをファイル (import.del) から TABLE1 にインポートするとします。このとき、このデータは、 ID 列を持たない表からエクスポートされたものであると想定します。以下に、このようなファイルの例を示します。

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

このファイルをインポートする 1 つの方法は、次のように IMPORT コマンドを使用して、インポートする列を明示的にリストすることです。

```
db2 import from import.del of del replace into table1 (c1, c3, c4)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをインポートする別の方法は、次のように identitymissing ファイル・タイプ修飾子を使うことです。

```
db2 import from import.del of del modified by identitymissing
   replace into table1
```

ID 列のあるデータのインポート

identityignore 修飾子は、ある意味では identitymissing 修飾子の反対の役割を持ちます。この修飾子を指定すると、インポート・ユーティリティーは、入力データ・ファイルに ID 列用の値が入っていても、そのデータを無視して、各行ごとに ID 値を生成します。例えば、上記で定義したように、ユーザーが次のようなデータをファイル (import.del) から TABLE1 にインポートするとします。

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

ユーザー指定値の 1、2、および 3 が ID 列に使われない場合、ユーザーは次のような IMPORT コマンドを使うことができます。

```
db2 import from import.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように identityignore 修飾子を使用すると、構文が単純化されます。

```
db2 import from import.del of del modified by identityignore
   replace into table1
```

ID 列を持つ表を IXF ファイルにエクスポートするときには、 IMPORT コマンドの REPLACE_CREATE および CREATE オプションを使用して、 ID 列プロパティーを備えた表を再作成することができます。タイプ GENERATED ALWAYS の ID 列の入った表からこのような IXF ファイルが作成されている場合、identityignore 修飾子を指定するのが、データ・ファイルのインポートを正常に完了する唯一の方法です。このようにしなければ、すべての行がリジェクトされます (SQL3550W)。

注: IMPORT コマンドの CREATE オプションと REPLACE_CREATE オプションは非推奨で、将来のリリースでは廃止される可能性があります。

生成列のインポートに関する考慮事項

インポート・ユーティリティーでは、入力データに生成列の値があるかどうかにかかわりなく、ID 列以外の生成列が含まれている表にデータをインポートできます。

生成列関連のファイル・タイプ修飾子が使用されない場合、インポート・ユーティリティーは次のような規則に従って動作します。

- 生成列の値が生成されるのは、入力ファイル内の対応する行にその列の値が存在しない場合か、NULL 値が明示的に指定されている場合です。生成列に非 NULL 値を指定すると、その行はリジェクトされます (SQL3550W)。
- サーバーが NULL 可能でない生成列に NULL 値を生成すると、そのフィールドが属するデータ行はリジェクトされます (SQL0407N)。これが起きるのは、例えば、NULL 可能列でない生成列が 2 つの表の列の合計として定義されていて、それらの表の列に入力ファイル内で NULL 値が指定された場合です。

生成列が含まれている表にデータをインポートする操作を簡略化するための 2 つの方法があります。つまり、ファイル・タイプ修飾子として、generatedmissing と generatedignore を使用できます。

生成列のないデータのインポート

generatedmissing 修飾子は、表内に存在する生成列の値が入力データ・ファイル内に入っていない (NULL すらない) 場合に、生成列を持つ表へのデータ・インポートを容易にします。例えば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 int,
                     c2 int,
                     g1 int generated always as (c1 + c2),
                     g2 int generated always as (2 * c1),
                     c3 char(1))
```

ユーザーが、データをファイル (load.del) から TABLE1 にインポートするとします。このとき、このデータは、生成列を持たない表からエクスポートされたものであると想定します。以下に、このようなファイルの例を示します。

```
1, 5, J
2, 6, K
3, 7, I
```

このファイルをインポートする 1 つの方法は、次のように IMPORT コマンドを使用して、インポートする列を明示的にリストすることです。

```
db2 import from import.del of del replace into table1 (c1, c2, c3)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。このファイルをインポートするための別の方法は、generatedmissing ファイル・タイプ修飾子を以下のようにして使用することです。

```
db2 import from import.del of del modified by generatedmissing
   replace into table1
```

生成列のあるデータのインポート

generatedignore 修飾子は、ある意味では generatedmissing 修飾子の反対の役割を持ちます。この修飾子を指定すると、インポート・ユーティリティーは、すべての生成列用のデータが入力データ・ファイルに入っていても、そのデータを無視して、各行ごとに値を生成します。例えば、上記で定義したように、ユーザーが次のようなデータをファイル (import.del) から TABLE1 にインポートするとします。

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

ユーザー提供の非 NULL 値 10、11、12 (g1 の場合)、および 15、16、17 (g2 の場合) により、行はリジェクトされます (SQL3550W)。リジェクトされないようにするには、次のような IMPORT コマンドを発行します。

```
db2 import from import.del of del method P(1, 2, 5)
   replace into table1 (c1, c2, c3)
```

ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように generatedignore 修飾子を使用すると、構文が単純化されます。

```
db2 import from import.del of del modified by generatedignore
   replace into table1
```

INSERT_UPDATE の場合、生成列が主キー列でもあり、generatedignore 修飾子が指定されていれば、IMPORT コマンドは、generatedignore 修飾子を有効と見なします。 IMPORT コマンドが UPDATE ステートメントの WHERE 節でこの列をユーザー提供値に置換することはありません。

LOB のインポートに関する考慮事項

インポート・ユーティリティーでは、1 つの列値のサイズが 32 KB までに制限されているので、LOB をインポートするときには、追加の考慮事項について検討する必要があります。

デフォルトの動作では、インポート・ユーティリティーは、入力ファイル内のデータを、列にロードするデータとして処理します。ただし、メインの入力データ・フ

ファイルにラージ・オブジェクト (LOB) データが格納されていると、データのサイズが 32 KB に制限されます。したがって、データを失わないようにするには、LOB データをメインのデータ・ファイルとは別の場所に格納し、LOB をインポートするときに lobsinfile ファイル・タイプ修飾子を指定する必要があります。

LOBS FROM 節を指定すると、lobsinfile が暗黙的にアクティブ化されます。LOBS FROM 節によって、データのインポート時に LOB ファイルを検索するパスのリストをインポート・ユーティリティーに渡します。LOBS FROM オプションが指定されない場合、インポートする LOB ファイルは、入力リレーショナル・データ・ファイルと同じパスにあると見なされます。

LOB データの格納場所の指定
LOB ロケーション指定子 (LLS) を使用すれば、LOB 情報のインポート時に複数の LOB を 1 つのファイルに格納できます。lobsinfile を指定すると、エクスポート・ユーティリティーは、LOB データの格納場所を示す指定子を生成して、エクスポート出力ファイルに格納します。MODIFIED BY lobsinfile オプションを指定したデータをインポートする場合、データベースは、対応する LOB 列ごとに LLS が存在することを予期します。 LLS 以外のものが LOB 列にある場合には、データベースはこれを LOB ファイルと見なし、ファイル全体を LOB としてロードします。

CREATE モードのインポートでは、LONG IN 節を使用して、LOB データを作成して別の表スペースに格納するように指定することも可能です。

次の例では、LOB が別のファイルに格納されている状態で DEL ファイルをインポートする方法を示します。

```
IMPORT FROM inputfile.del OF DEL
 LOBS FROM /tmp/data
 MODIFIED BY lobsinfile
 INSERT INTO newtable
```

ユーザー定義特殊タイプのインポートに関する考慮事項

インポート・ユーティリティーは、ユーザー定義特殊タイプ (UDT) をそれと類似の基本データ・タイプに自動的にキャストします。それにより、UDT を基本データ・タイプに明示的にキャストする手間が省けます。キャストによって、SQL で UDT と基本データ・タイプとの間の比較が可能になります。

インポートに関する追加の考慮事項

クライアント/サーバー環境とインポート

リモート・データベースにファイルをインポートする場合は、ストアード・プロシージャーを呼び出してサーバー上でインポートを実行することができます。

ストアード・プロシージャーは、以下の場合には呼び出されません。

- アプリケーションとデータベースとでコード・ページが違っている場合。
- インポートされるファイルが、複数に分かれた PC/IXF ファイルである場合。
- データのインポートに使用される方式が列名か相対列位置のいずれかである場合。

- 指定されたターゲット列リストが 4 KB を超えている場合。
- LOBS FROM 節または lobsinfile 修飾子が指定されている場合。
- ASC ファイルに NULL INDICATORS 節が指定されている場合。

インポートでストアード・プロシージャーを使用する場合は、サーバーにインストールされているデフォルト言語を使ってメッセージ・ファイル内にメッセージが作成されます。クライアントとサーバーとで言語が同じ場合、メッセージはアプリケーションの言語になります。

インポート・ユーティリティーは、 sqllib ディレクトリー (または **DB2INSTPROF** レジストリー変数が指定されている場合はそれによって示されるディレクトリー) の tmp サブディレクトリーに、2 つの一時ファイルを作成します。 1 つのファイルはデータ用、もう 1 つのファイルはインポート・ユーティリティーが生成するメッセージ用です。

サーバー上でのデータの書き込みまたはオープンに関してエラーが出された場合は、以下の点を確認してください。

- ディレクトリーが存在する。
- それらのファイル用の十分のディスク・スペースがある。
- インスタンス所有者にそのディレクトリーでの書き込み許可がある。

インポート時の表のロッキング

インポート・ユーティリティーは、2 つの表ロッキング・モードをサポートしています。つまり、オフラインの ALLOW NO ACCESS モードとオンラインの ALLOW WRITE ACCESS モードです。

ALLOW NO ACCESS モードでは、複数の同時アプリケーションから表データにアクセスできないようになります。ALLOW WRITE ACCESS モードでは、複数の同時アプリケーションからインポート・ターゲット表へ読み取りおよび書き込みアクセスすることができます。モードを明示的に指定しない場合は、デフォルト・モードの ALLOW NO ACCESS でインポートが実行されます。さらに、インポート・ユーティリティーは、デフォルトで、分離レベル RS (読み取り固定) でデータベースにバインドされます。

オフライン・インポート (ALLOW NO ACCESS)

ALLOW NO ACCESS モードのインポートは、行を挿入する前にターゲット表の排他 (X) ロックを取得します。表のロックを保持することには、2 つの意味があります。

- 第 1 に、他のアプリケーションがインポート・ターゲット表に対する表ロックまたは行ロックを保持していると、インポート・ユーティリティーは、そのようなアプリケーションが変更内容をコミットするかロールバックするまで待機します。
- 第 2 に、インポートの実行中に、ロックを要求している他のすべてのアプリケーションは、そのインポート操作の完了を待機します。

注: locktimeout 値を指定すれば、インポート・ユーティリティーをはじめとするアプリケーションがロックをいつまでも待機しつづける事態を回避できます。

インポートは、操作の開始時に排他ロックを要求することによって、他のアプリケーションが処理中に同じターゲット表に対する行ロックを保持した場合に発生するデッドロックを回避します。

オンライン・インポート (ALLOW WRITE ACCESS)

ALLOW WRITE ACCESS モードのインポート・ユーティリティーは、ターゲット表の非排他 (IX) ロックを取得します。表に対してこのロックをかけることには 2 つの意義があります。

- 非互換の表ロックをかけている他のアプリケーションがあると、そのようなアプリケーションがすべて変更内容をコミットまたはロールバックするまでインポート・ユーティリティーはデータの挿入を開始しません。
- インポートが実行されているかぎり、非互換の表ロックを要求している他のすべてのアプリケーションは、そのインポートが現在のトランザクションをコミットするかまたはロールバックするまで待機します。インポートでの表ロックは、次のトランザクションに持ち越されることはないことに注意してください。これを受けて、オンライン・インポートでは、各コミットの完了ごとに表ロックの要求と、おそらくは待機を行う必要があります。
- 非互換の行ロックをかけている他のアプリケーションがある場合、そのようなアプリケーションがすべて変更内容をコミットまたはロールバックするまでインポート・ユーティリティーはデータの挿入を停止します。
- インポートの実行中に、非互換の行ロックを要求している他のすべてのアプリケーションは、そのインポート操作が現在のトランザクションをコミットするかロールバックするまで待機します。

ALLOW WRITE ACCESS インポートは、オンライン・プロパティーを維持しながら、デッドロックの可能性を減らすために、現在のトランザクションを周期的にコミットして行ロックをすべて解放することで、排他表ロックへのエスカレーションが発生しないようにしています。コミット頻度が明示的に設定されていなければ、インポートは、COMMITCOUNT AUTOMATIC が指定されている場合と同じようにコミットを実行します。 COMMITCOUNT が 0 に設定されていると、コミットは実行されません。

ALLOW WRITE ACCESS モードには、以下のものとの互換性はありません。

- REPLACE、CREATE、REPLACE_CREATE のいずれかのモードのインポート
- バッファー挿入を使用したインポート
- ターゲット・ビューへのインポート
- 階層表へのインポート
- ロックの細分度が (ALTER TABLE ステートメントの LOCKSIZE パラメーターで) 表レベルに設定されている表へのインポート

リファレンス - インポート

IMPORT

外部ファイルのデータを、サポートされているファイル・フォーマットで表、階層、ビュー、またはニックネームに挿入します。 LOAD はより高速な代替方法です。しかしロード・ユーティリティーでは、階層レベルのデータのロードはサポートされていません。

89 ページの『インポート・ユーティリティーのファイル・タイプ修飾子』へのクイック・リンク。

許可

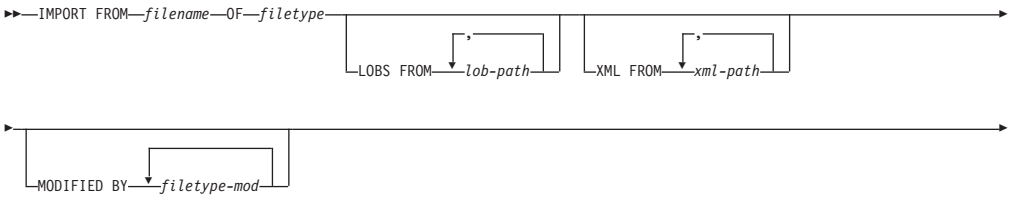
- INSERT オプションを使用して **IMPORT** コマンドを実行する場合、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 関係するそれぞれの表、ビュー、またはニックネームに対する CONTROL 特権
  - 関係するそれぞれの表またはビューに対する INSERT および SELECT 特権
- INSERT_UPDATE オプションを使用して既存の表に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 関係するそれぞれの表、ビュー、またはニックネームに対する CONTROL 特権
  - 関係するそれぞれの表またはビューに対する INSERT、 SELECT、UPDATE、および DELETE 特権
- REPLACE または **REPLACE_CREATE** オプションを使用して既存の表に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 表またはビューに対する CONTROL 特権
  - 表またはビューに対する INSERT、SELECT、および DELETE 特権
- CREATE または **REPLACE_CREATE** オプションを使用して新規の表に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - データベースに対する CREATETAB 権限、表スペースに対する USE 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 (表の暗黙または明示スキーマ名がない場合)
    - 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す場合)

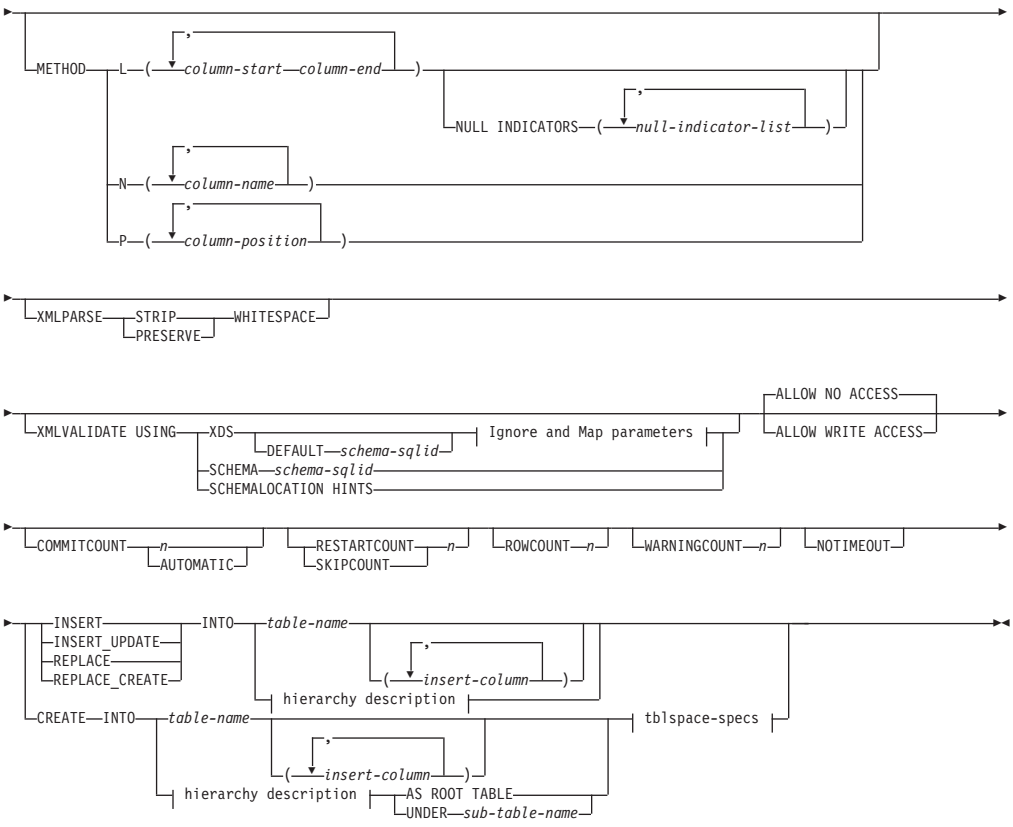
- CREATE または **REPLACE_CREATE** オプションを使って、存在しない階層に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - データベースに対する CREATETAB 権限および表スペースに対する USE 特権と、以下のいずれか。
    - データベースに対する IMPLICIT_SCHEMA 権限 (表のスキーマ名が存在しない場合)
    - スキーマに対する CREATEIN 特権 (表のスキーマが存在する場合)
    - 階層全体に対して **REPLACE_CREATE** オプションが使用されている場合は、階層内のすべての副表に対する CONTROL 特権
- REPLACE オプションを使用して既存の階層に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 階層内のすべての副表に対する CONTROL 特権
- 保護列を持つ表にデータをインポートするには、表内のすべての保護列への書き込みアクセスを可能にする LBAC 信用証明情報がセッション許可 ID に必要です。そうでない場合、インポートは失敗し、エラー (SQLSTATE 42512) が戻されます。
- 保護されている行のある表にデータをインポートするには、セッション許可 ID に、以下の基準を満たす LBAC クリデンシャルが必要です。
  - 表を保護しているセキュリティー・ポリシーの一部である
  - 書き込みアクセスに関して、セッション許可 ID に付与された

  挿入する行上のラベル、ユーザーの LBAC 信用証明情報、セキュリティー・ポリシーの定義、および LBAC 規則によって、行上のラベルが決まります。
- **REPLACE** または **REPLACE_CREATE** オプションが指定された場合、セッション許可 ID には、その表をドロップするための権限が付与されていなければなりません。

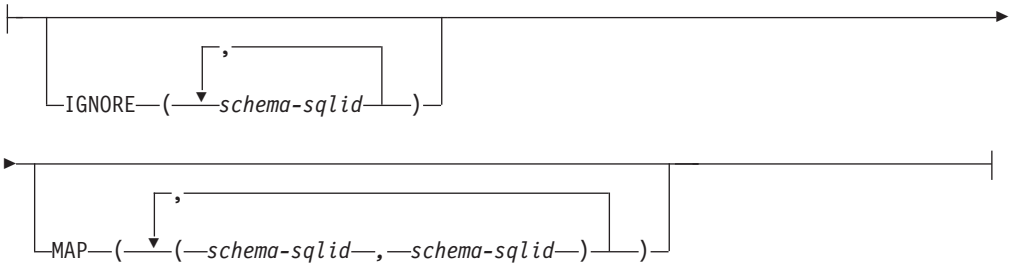
必要な接続

コマンド構文

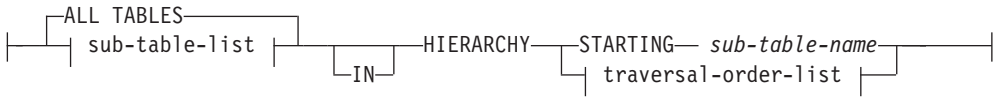




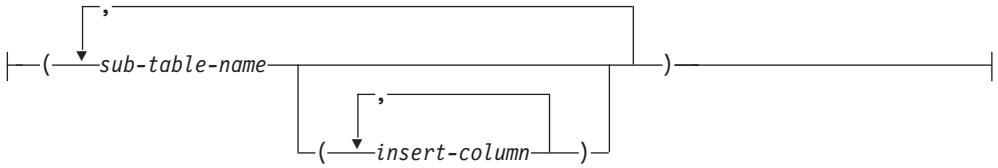
Ignore and Map parameters:



hierarchy description:

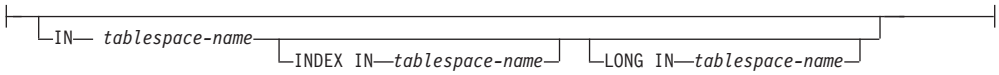


sub-table-list:



traversal-order-list:

tblspace-specs:



コマンド・パラメーター

ALL TABLES

階層専用の暗黙キーワード。階層をインポートする場合、走査順序で指定されるすべての表をインポートすることがデフォルトです。

ALLOW NO ACCESS

オフライン・モードでインポートを実行します。行の挿入の前には常に、ターゲット表に排他 (X) ロックがかけられます。これで、同時アプリケーションは表データにアクセスできなくなります。これがデフォルトのインポート動作です。

ALLOW WRITE ACCESS

オンライン・モードでインポートを実行します。最初の行の挿入時には、ターゲット表に意図的排他 (IX) ロックがかけられます。これで、表データへの同時の読み取りおよび書き出しアクセスが可能になります。オンライン・モードには、**REPLACE**、**CREATE**、または **REPLACE_CREATE** インポート・オプションとの互換性はありません。 オンライン・モードとバッファー挿入との連携はサポートされません。インポート操作によって挿入後のデータが定期的にコミットされるので、表ロックへのロック・エスカレーションが削減され、アクティブなログ・スペースが使い果たされることはなくなります。このようなコミットは、**COMMITCOUNT** オプションを使わなくても実行されます。 各コミットごとに、インポートでは IX 表ロックが外されるので、コミットの完了後に再びロックの設定が試みられます。ニックネームにインポートするときにはこのパラメーターが必要で、有効な数値を使って **COMMITCOUNT** を指定する必要があります (AUTOMATIC は有効なオプションとは見なされません)。

AS ROOT TABLE

1 つ以上の副表を、独立した表階層として作成します。

COMMITCOUNT *n* **| AUTOMATIC**

n 個のレコードがインポートされるたびに COMMIT を実行します。数 *n* を指定すると、インポートでは *n* 個のレコードのインポートの後にそのつど COMMIT が実行されます。コンパウンド挿入を使用した場合、ユーザー指定のコミット頻度 *n* は、そのコンパウンド・カウント値に最も近い整数の倍数に切り上げられます。 AUTOMATIC を指定すると、コミットの必要時期はインポート操作で内部的に判別されます。 次の 2 つのうちのいずれかの理由で、このユーティリティーはコミットを行います。

- アクティブ・ログ・スペースを使いきらないようにするため。

- ロックが行レベルから表レベルにエスカレーションしないようにするため。

ALLOW WRITE ACCESS オプションを指定した場合に **COMMITCOUNT** オプションを指定しないと、インポート・ユーティリティーは、**COMMITCOUNT** AUTOMATIC が指定されたものとしてコミットを実行します。

インポート操作で、アクティブ・ログ・スペースを使いきらないようにできるかどうかは、DB2 レジストリー変数 **DB2_FORCE_APP_ON_MAX_LOG** によって影響を受けます。

- **DB2_FORCE_APP_ON_MAX_LOG** が FALSE に設定され、**COMMITCOUNT** AUTOMATIC コマンド・オプションが指定されている場合、インポート・ユーティリティーは自動的に、アクティブ・ログ・スペースを使いきらないようにすることができます。
- **DB2_FORCE_APP_ON_MAX_LOG** が FALSE に設定され、**COMMITCOUNT** *n* コマンド・オプションが指定されている場合、インポート・ユーティリティーは、レコードの挿入または更新中に SQL0964C (トランザクション・ログがいっぱい) が発生した場合に、ログ満杯状態を解決しようと試みます。インポート・ユーティリティーは、無条件コミットを実行してから、レコードの挿入または更新を再試行します。それでも問題が解決しない場合 (ログの満杯状態がデータベース上の他のアクティビティーが原因で起きている場合など)、IMPORT コマンドは予期したとおりに失敗します。ただし、コミットされた行の数は、**COMMITCOUNT** *n* の値の倍数になっていない可能性があります。インポート操作の再試行時に、既にコミットされている行を処理しないようにするには、**RESTARTCOUNT** または **SKIPCOUNT** コマンド・パラメーターを使用します。
- **DB2_FORCE_APP_ON_MAX_LOG** が TRUE (デフォルト) に設定されている場合、レコードの挿入または更新中に SQL0964C が発生すると、インポート操作は失敗します。これは、**COMMITCOUNT** AUTOMATIC または **COMMITCOUNT** *n* のどちらを指定するかに関係なく発生します。

  アプリケーションは強制的にデータベースから切断され、現行作業単位はロールバックされます。インポート操作の再試行時に、既にコミットされている行を処理しないようにするには、**RESTARTCOUNT** または **SKIPCOUNT** コマンド・パラメーターを使用します。

CREATE

注: CREATE パラメーターは推奨されておらず、今後のリリースで除去される可能性があります。詳細については、『推奨されなくなった IMPORT コマンド・オプション **CREATE** および **REPLACE_CREATE**』を参照してください。

データベースのコード・ページで表の定義と行の内容を作成します。 DB2 の表、副表、または階層からエクスポートされたデータの場合、索引も作成されます。このオプションが階層に対するものである場合に、DB2 からデータがエクスポートされると、タイプ階層も作成されます。 このオプションは、IXF ファイルの場合にのみ使用することができます。

ニックネームにインポートするときには、このパラメーターは無効です。

注: データが MVS™ ホスト・データベースからエクスポートされたもので、ページ・サイズで計算した長さが 254 より大きい LONGVAR フィールドを含んでいる場合、**CREATE** は行が長過ぎるために失敗します。 制約事項のリストについては、『『インポート済みの表の再作成』』を参照してください。この場合、その表は手動で作成します。そして、IMPORT に **INSERT** を指定して呼び出すか、または LOAD コマンドを使用してください。

DEFAULT *schema-sqlid*
このオプションは、**USING XDS** パラメーターを指定した場合にのみ使用できます。 **DEFAULT** 節で指定されたスキーマは、インポート対象 XML 文書の XML Data Specifier (XDS) に XML スキーマを指定する SCH 属性が含まれていない場合に、妥当性検査のために使用するスキーマとなります。

DEFAULT 節は、**IGNORE** 節および **MAP** 節よりも優先されます。 XDS が **DEFAULT** 節を満たすなら、**IGNORE** と **MAP** の指定は無視されます。

FROM *filename*

HIERARCHY
階層データをインポートするよう指定します。

IGNORE *schema-sqlid*
このオプションは、**USING XDS** パラメーターを指定した場合にのみ使用できます。 **IGNORE** 節は、SCH 属性によって指定されていても無視するスキーマとして、1 つ以上のスキーマのリストを指定します。 インポートする XML 文書の XML Data Specifier の中に SCH 属性が存在し、その SCH 属性によって指定されるスキーマが無視するスキーマ・リストに含まれている場合には、インポートするその XML 文書についてスキーマ妥当性検査は実行されません。

あるスキーマが **IGNORE** 節の中で指定されている場合、**MAP** 節のスキーマ・ペアの左辺にそれを含めることはできません。

IGNORE 節は XDS にのみ適用されます。 あるスキーマが **IGNORE** 節によって指定されていても、それが **MAP** 節によってマップされているなら、それ以降そのスキーマが無視されることはありません。

IN *tablespace-name*
表を作成する表スペースを指定します。表スペースは存在している必要があり、REGULAR 表スペースでなければなりません。他の表スペースを指定しない場合、すべての表パーツはこの表スペースに保管されます。この節を指定しない場合、表は許可 ID によって作成された表スペース中に作成されます。何も検出されない場合、その表はデフォルト表スペースの USERSPACE1 に入れられます。 USERSPACE1 がドロップされていた場合、表作成は失敗します。

INDEX IN *tablespace-name*
表の索引を作成する表スペースを指定します。このオプションは、**IN** 節で指定される PRIMARY 表スペースが DMS 表スペースである場合のみ使用

できます。 指定した表スペースは存在している必要があり、かつ REGULAR または LARGE DMS 表スペースでなければなりません。

注: どの表スペースに索引を配置するかは、表を作成するときにのみ指定できます。

insert-column
データの挿入先となる表またはビュー内の列名を指定します。

INSERT
既存の表データを変更することなく、インポートされたデータを表に追加します。

INSERT_UPDATE
インポートしたデータ行をターゲット表に追加するか、または主キーが一致するものがあればターゲット表の既存行を更新します。

INTO *table-name*
データのインポート先となるデータベース表を指定します。この表として、システム表、宣言一時表、またはサマリー表は指定できません。

以前のサーバーの場合を除き、**INSERT**、**INSERT_UPDATE**、または **REPLACE** オプションには、完全修飾または非修飾の表名を使用しなければならないようなときでも、別名を使用することができます。修飾子付き表名は、*schema.tablename* の形式です。 *schema* には、表作成時のユーザー名が入ります。

LOBS FROM *lob-path*
LOB データ・ファイルの名前は、メイン・データ・ファイル (ASC、DEL、または IXF) の、LOB 列にロードされる列内に保管されます。指定できるパスの最大数は 999 です。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

ニックネームにインポートするときには、このパラメーターは無効です。

LONG IN *tablespace-name*
ロング列の値 (LONG VARCHAR、LONG VARGRAPHIC、LOB データ・タイプ、またはソース・タイプとしてこれらが指定されている特殊タイプ) を保管する表スペースを指定します。このオプションは、 **IN** 節で指定した PRIMARY 表スペースが DMS 表スペースである場合のみ使用できます。指定した表スペースは存在している必要があり、LARGE DMS 表スペースでなければなりません。

MAP *schema-sqlid*
このオプションは、**USING XDS** パラメーターを指定した場合にのみ使用できます。 **MAP** 節は、インポートする各 XML 文書について XML Data Specifier (XDS) の SCH 属性によって指定されるスキーマの代わりに使用する代替スキーマを指定するのに使用します。 **MAP** 節には、それぞれがあるスキーマから別のスキーマへのマッピングを表すスキーマ・ペアを 1 つ以上列挙したリストを指定します。 ペア中の最初のスキーマは、XDS 内の SCH 属性によって参照されるスキーマを表します。ペア中の 2 番目のスキーマは、スキーマ検証の実行で使用する必要のあるスキーマを表します。

あるスキーマが **MAP** 節のスキーマ・ペアの左辺で指定されている場合、**IGNORE** 節でさらにそれを指定することはできません。

スキーマ・ペアのマッピングが適用されたなら、その結果は最終的なものです。マッピング操作は推移的ではないため、選択されたスキーマが、それ以降に別のスキーマ・ペアのマッピングに適用されることはありません。

スキーマを複数回マップすることはできません。つまり、複数のペアの左辺に指定することはできません。

METHOD

L データのインポートを開始する列および終了する列の番号を指定します。列の番号は、データの行の先頭からのバイト単位のオフセットです。この番号は 1 から始まります。

注: このメソッドは、ASC ファイルの場合にのみ使用することができ、そのファイル・タイプに対してのみ有効なオプションです。

N インポートするデータ・ファイルの中の列の名前を指定します。これらの列名の大文字小文字の区別は、システム・カタログ内の対応する名前の大文字小文字の区別と一致しなければなりません。NULL 可能ではない各表の列には、**METHOD N** リスト内に対応する項目が必要です。例えば、データ・フィールドが F1、 F2、 F3、 F4、 F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、method N (F2, F1, F4, F3) は有効な要求ですが、 method N (F2, F1) は無効です。

注: この方式は、IXF ファイルの場合にのみ使用することができます。

P インポートする入力データ・フィールドのフィールド番号を指定します。

注: この方式は、IXF または DEL ファイルの場合にのみ使用でき、DEL ファイル・タイプに対してのみ有効なオプションです。

MODIFIED BY *filetype-mod*

ファイル・タイプ修飾子オプションを指定します。 89 ページの『インポート・ユーティリティーのファイル・タイプ修飾子』を参照してください。

NOTIMEOUT

インポート・ユーティリティーがロックの待機中にタイムアウトしないことを指定します。このオプションのほうが、 **locktimeout** データベース構成パラメーターより優先されます。他のアプリケーションは影響を受けません。

NULL INDICATORS *null-indicator-list*

このオプションは、**METHOD L** パラメーターを指定した場合にのみ使用できます。つまり、入力ファイルが ASC ファイルの場合です。 NULL 標識リストは、コンマで区切られた正の整数のリストで、各 NULL 標識フィールドの列の番号を指定します。列の番号は、データの行の先頭からのバイト単位の、各 NULL 標識フィールドのオフセットです。 NULL 標識リストには、**METHOD L** パラメーターで定義された各データ・フィールドに

対する 1 つの項目がなければなりません。列の番号がゼロであることは、対応するデータ・フィールド内に必ずデータがあることを示します。

NULL 標識列中の Y の値は、その列データが NULL であることを指定します。 NULL 標識列に Y 以外 の文字を指定した場合は、列データが NULL ではなく、**METHOD L** オプションで指定された列データがインポートされることを指定することになります。

nullindchar ファイル・タイプ修飾子を指定した **MODIFIED BY** オプションを使用すれば、NULL 標識文字を変更することができます。

OF *filetype*

入力ファイル内のデータのフォーマットを指定します。

- ASC (区切りなし ASCII フォーマット)
- DEL (区切り文字付き ASCII フォーマット)。さまざまなデータベース・マネージャーやファイル・マネージャー・プログラムで使用します
- WSF (ワークシート・フォーマット)。以下のプログラムで使用します。
  - Lotus 1-2-3
  - Lotus Symphony
- IXF (統合交換フォーマット、PC バージョン) は、DB2 専用のバイナリー・フォーマットです。

ニックネームにインポートするときには、WSF ファイル・タイプはサポートされません。

REPLACE

データ・オブジェクトを切り捨てることによって表内の既存のデータすべてを削除してから、インポートしたデータを挿入します。表定義および索引定義は変更されません。表がない場合は、このオプションを使用できません。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

ニックネームにインポートするときには、このパラメーターは無効です。

このオプションでは、CREATE TABLE ステートメントの NOT LOGGED INITIALLY (NLI) 節、あるいは ALTER TABLE ステートメントの ACTIVE NOT LOGGED INITIALLY 節は考慮されません。

NLI 節が呼び出される CREATE TABLE または ALTER TABLE ステートメントと同じトランザクションの中で、**REPLACE** オプションの指定されたインポートが実行された場合、インポートにおいてその NLI 節は考慮されません。挿入はすべてログに記録されます。

予備手段 1

DELETE ステートメントを使用して表の内容を削除した後、INSERT ステートメントによりインポートを呼び出す

予備手段 2

表をドロップしてからそれを再作成した後、INSERT ステートメントによってインポートを呼び出す

この制限は、DB2® Universal Database™ バージョン 7 および DB2 UDB バージョン 8 に適用されます。

REPLACE_CREATE

注: REPLACE_CREATE パラメーターは推奨されておらず、今後のリリースで除去される可能性があります。さらに詳しくは、『IMPORT コマンドの推奨されなくなったオプション CREATE および REPLACE_CREATE』を参照してください。

表がすでにある場合には、データ・オブジェクトを切り捨てることによって表内の既存のデータすべてを削除し、表定義や索引定義は変えることなく、インポートしたデータを挿入します。

表がまだない場合には、データベースのコード・ページで、表と索引の定義と行の内容を作成します。制約事項のリストについては、『*インポート済みの表の再作成*』を参照してください。

このオプションは、IXF ファイルの場合にのみ使用することができます。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

ニックネームにインポートするときには、このパラメーターは無効です。

RESTARTCOUNT *n*

n + 1 の位置のレコードからインポート操作を開始するよう指定します。最初の *n* 個のレコードはスキップされます。このオプションは機能的には **SKIPCOUNT** と同等です。 **RESTARTCOUNT** と **SKIPCOUNT** は相互に排他的です。

ROWCOUNT *n*

インポート (挿入または更新) するファイル内の物理レコードの数 *n* を指定します。ユーザーは、**SKIPCOUNT** または **RESTARTCOUNT** オプションで指示されたレコードから始めて、ファイルの *n* 行だけをインポートすることができます。 **SKIPCOUNT** または **RESTARTCOUNT** オプションの指定がないと、最初の *n* 行がインポートされます。 **SKIPCOUNT** *m* または **RESTARTCOUNT** *m* を指定すると、行 *m*+1 から *m*+*n* がインポートされます。コンパウンド挿入を使用した場合、ユーザー指定の **ROWCOUNT** *n* は、そのコンパウンド・カウント値に最も近い整数の倍数に切り上げられます。

SKIPCOUNT *n*

n + 1 の位置のレコードからインポート操作を開始するよう指定します。最初の *n* 個のレコードはスキップされます。このオプションは機能的には **RESTARTCOUNT** と同等です。 **SKIPCOUNT** と **RESTARTCOUNT** は相互に排他的です。

STARTING *sub-table-name*

階層専用キーワード。 *sub-table-name* から始まるデフォルト順を要求します。 PC/IXF ファイルの場合、デフォルト順は入力ファイルに保管されている順です。 PC/IXF ファイル・フォーマットの場合、デフォルト順は有効な唯一の順序です。

sub-table-list

型付き表で **INSERT** または **INSERT_UPDATE** オプションを指定した場合、データのインポート先副表を指定するために副表名のリストが使われます。

traversal-order-list
型付き表で **INSERT**、**INSERT_UPDATE**、または **REPLACE** オプションを指定した場合、インポートする階層内の副表のトラバーサル順序を指定するために副表名のリストを使います。

UNDER *sub-table-name*
1 つ以上の副表を作成する場合に親表を指定します。

WARNINGCOUNT *n*
n 個の警告後に、インポート操作を停止します。このパラメーターは、警告は予期されないが、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。インポート・ファイルまたはターゲット表が不適切に指定されると、インポート対象の各行ごとにインポート・ユーティリティーによって警告が生成され、このためにインポートが失敗する可能性があります。 *n* をゼロにした場合や、このオプションを指定しない場合、発行された警告の回数に関係なくインポート操作は続行します。

XML FROM *xml-path*
XML ファイルが含まれているパスを 1 つ以上指定します。

XMLPARSE
XML 文書の解析方法を指定します。このオプションが指定されていない場合、XML 文書の解析の動作は、CURRENT XMLPARSE OPTION 特殊レジスターの値によって決まります。

STRIP WHITESPACE
XML 文書の解析時に空白文字を除去することを指定します。

PRESERVE WHITESPACE
XML 文書の解析時に空白文字を除去しないことを指定します。

XMLVALIDATE
該当する場合に、XML 文書がスキーマに準拠しているかどうかの妥当性検査を実行することを指定します。

USING XDS
メイン・データ・ファイル内の XML Data Specifier (XDS) で識別される XML スキーマに照らし合わせて、XML 文書が妥当性検査されます。デフォルトでは、**USING XDS** 節によって **XMLVALIDATE** オプションが呼び出された場合、妥当性検査実行のために使用されるスキーマは、その XDS の SCH 属性によって決まります。 XDS の中で SCH 属性が指定されていない場合、**DEFAULT** 節によってデフォルト・スキーマが指定されているのでない限り、スキーマ妥当性検査は実行されません。

DEFAULT、**IGNORE**、および **MAP** 節を使用することにより、スキーマ決定の動作を変更することができます。これら 3 つの節はオプションであり、相互に適用されるのではなく XDS の指定に直接適用されます。例えば、**DEFAULT** 節で指定されているためにあるスキーマが選択された場合、それが **IGNORE** 節で指定されていたとしても無視されることはありません。同じように、**MAP** 節のペアの最初の部分で指定されているためにあるスキーマが選択された場合、それが別の **MAP** 節のペアの 2 番目の部分で指定されていたとしても再びマップされることはありません。

USING SCHEMA *schema-sqlid*
指定されている SQL ID の XML スキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。この場合、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

USING SCHEMALOCATION HINTS
ソース XML 文書の中で XML スキーマ・ロケーション・ヒントによって指定されているスキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。その XML 文書の中に schemaLocation 属性が指定されていない場合、妥当性検査は実行されません。 **USING SCHEMALOCATION HINTS** 節が指定されているなら、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

以下に示す **XMLVALIDATE** オプションの例を参照してください。

使用上の注意

インポート操作を開始する前に、すべての表操作が完了し、すべてのロックがペンディング解除になっていることを確認してください。これは、WITH HOLD でオープンされた、すべてのカーソルをクローズした後で COMMIT または ROLLBACK を発行することによって行われます。

インポート・ユーティリティーは、 SQL INSERT ステートメントを使用してターゲット表に行を追加します。このユーティリティーは、入力ファイル中の各行のデータにつき 1 つずつ INSERT ステートメントを発行します。 INSERT ステートメントが失敗した場合、以下の 2 通りの結果のいずれかになります。

- 後続の INSERT ステートメントが成功すると予測される場合には、警告メッセージがメッセージ・ファイルに書き込まれ、処理が継続されます。
- 後続の INSERT ステートメントが失敗すると予測され、データベースが損傷する可能性がある場合には、エラー・メッセージがメッセージ・ファイルに書き込まれ、処理が停止されます。

ユーティリティーは、**REPLACE** または **REPLACE_CREATE** 操作中に、古い行が削除された後、自動 COMMIT を実行します。したがって、表オブジェクトが切り捨てられた後、システムに障害が起こったり、アプリケーションがデータベース・マネージャーに割り込んだりすると、元のデータがすべて失われてしまいます。これらのオプションを使用する前に、元のデータがもはや必要ないことを確認してください。

CREATE、**REPLACE**、または **REPLACE_CREATE** 操作時にログが満杯になると、このユーティリティーは挿入されたレコードに対して自動 COMMIT を実行します。自動 COMMIT の後に、システムに障害が起こるか、またはアプリケーションがデータベース・マネージャーに割り込むと、部分的にデータの挿入された表はデータベース内に残ります。 **REPLACE** または **REPLACE_CREATE** オプションを使用してインポート操作全体をやり直すか、または正常にインポートされる行の数に設定した **RESTARTCOUNT** パラメーターを指定して **INSERT** を使用してください。

デフォルトでは、自動 COMMIT は **INSERT** または **INSERT_UPDATE** オプションでは実行されません。しかし、**COMMITCOUNT** パラメーターがゼロでない場合は実行されます。自動の COMMIT が実行されない場合にログが満杯になると、ROLLBACK が実行されます。

以下のいずれかの条件が真であると、オフライン・インポートでは自動の COMMIT は実行されません。

- ターゲットは表ではなくビューである。
- コンパウンド挿入を使用している。
- バッファー挿入を使用している。

デフォルトでは、オンライン・インポートは自動 COMMIT を実行して、アクティブ・ログ・スペースとロック・リストを両方とも解放します。自動 COMMIT が実行されないのは、ゼロの **COMMITCOUNT** 値を指定した場合のみです。

インポート・ユーティリティーが COMMIT を実行するたびに、 2 つのメッセージがメッセージ・ファイルに書き込まれます。一方は、コミットされるレコードの数を示し、もう一方は、 COMMIT の成功後に書き込まれます。障害の後にインポート操作を再開するときには、スキップするレコードの数 (最後の正常なコミットから判別される) を指定してください。

インポート・ユーティリティーでは、多少の非互換性問題がある入力データは受け入れられます (例えば、文字データは埋め込みまたは切り捨てを用いてインポートできます。数値データは異なる数値データ・タイプを用いてインポートできます)。しかし、大きな非互換性問題のあるデータは受け入れられません。

それ自体以外への依存があるオブジェクト表や、基本表に何らかの依存 (それ自体も含めて) があるオブジェクト・ビューを、 **REPLACE** または **REPLACE_CREATE** することはできません。そのような表またはビューを置換するには、以下のとおりに行ってください。

1. その表が親となっているすべての外部キーをドロップします。
2. インポート・ユーティリティーを実行します。
3. 表を変更して、外部キーを再作成します。

外部キーの再作成中にエラーが発生する場合、参照整合性を保守するためにデータを変更してください。

参照制約および外部キー定義は、PC/IXF ファイルから表を再作成する場合は保存されません。 (主キー定義は、データが前に `SELECT *` を使ってエクスポートされた場合、*保存されます*。)

リモート・データベースへのインポートでは、サーバーに、入力データ・ファイルのコピー、出力メッセージ・ファイル、およびデータベースのサイズ拡大を見込んだ十分なディスク・スペースが必要とされます。

インポート操作がリモート・データベースに対して実行され、出力メッセージ・ファイルが非常に長い (60 KB より長い) 場合、クライアント上でユーザーに戻されるメッセージ・ファイルがインポート操作中に欠落することがあります。メッセージ情報の最初の 30 KB と最後の 30 KB は、常に保持されます。

PC/IXF ファイルのリモート・データベースへのインポートは、 PC/IXF ファイルがディスケットにあるときよりも、ハード・ディスクにあるときの方がより速く行うことができます。

ASC、**DEL**、または **WSF** のファイル形式のデータをインポートするためには、それ以前にデータベース表または階層がすでに存在していなければなりません。ただし、表がまだ存在していない場合でも、IMPORT **CREATE** または IMPORT **REPLACE_CREATE** を使えば、PC/IXF ファイルからデータをインポートする際に表が作成されます。型付き表の場合、IMPORT **CREATE** はタイプ階層と表階層も作成することができます。

PC/IXF インポートは、データベース間でデータ (階層データも含む) を移動する場合に使用します。行区切り文字を含む文字データが区切り文字付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字を含むフィールドは長さが変わることがあります。ソースとターゲットのデータベースが両方とも同じクライアントからアクセス可能である場合、ファイルのコピーというステップは必要ありません。

ASC および DEL ファイルのデータは、インポートを実行するクライアント・アプリケーションのコード・ページであると仮定されます。異なるコード・ページのデータをインポートする場合は、異なるコード・ページを使用することのできる PC/IXF ファイルをお勧めします。 PC/IXF ファイルとインポート・ユーティリティーが同じコード・ページである場合は、通常のアプリケーションの場合のように処理が行われます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されている場合、インポート・ユーティリティーは、PC/IXF ファイルのデータのコード・ページと、インポートを実行中のアプリケーションのコード・ページが同じであると見なします。この処理は、それら 2 つのコード・ページ用の変換テーブルが存在する場合であっても行われます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されておらず、変換テーブルが存在する場合、PC/IXF ファイルのすべてのデータは、そのファイルのコード・ページからアプリケーションのコード・ページに変換されます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されておらず、変換テーブルが存在しない場合、インポート操作は失敗します。これが該当するのは、AIX オペレーティング・システムの DB2 クライアント上の PC/IXF ファイルの場合だけです。

8 KB ページ上の表オブジェクトの量が 1012 列の制限に近い場合、 PC/IXF データ・ファイルをインポートすると、 SQL ステートメントの最大サイズを超過するため、DB2 はエラーを戻します。この状態が発生する可能性があるのは、列が CHAR、 VARCHAR、または CLOB タイプの場合だけです。 **DEL** または **ASC** ファイルのインポートでは、この制限は当てはまりません。 PC/IXF ファイルを使って新しい表を作成している場合、別の方法として、db2look を使って表を作成した DDL ステートメントをダンプしてから、そのステートメント CLP から発行する、という方法があります。

DB2 Connect は、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA サーバーにデータをインポートするために使用できます。PC/IXF インポート (**INSERT** オプション) だけがサポートされています。**RESTARTCOUNT** パラメーターもサポートされていますが、**COMMITCOUNT** パラメーターはサポートされていません。

型付き表に対して **CREATE** オプションを使うと、PC/IXF ファイルの中で定義されているすべての副表が作成されます。副表定義は変更されません。型付き表に対して **CREATE** 以外のオプションを使うと、トラバーサル順序リストによって、トラバース順序を指定できます。その場合、トラバーサル順序リストはエクスポート操作で使用されたものと一致していなければなりません。 PC/IXF ファイル形式の場合は、ターゲット副表の名前を指定して、ファイルに格納されている全探索順序を使用するだけです。

インポート・ユーティリティーは、以前 PC/IXF ファイルにエクスポートされた表をリカバリーする場合に使用できます。その表は、エクスポート時の状態に戻ります。

システム表、宣言された一時表、またはサマリー表にデータをインポートすることはできません。

インポート・ユーティリティーを介してビューを作成することはできません。

マルチパート PC/IXF ファイルの個々のパートを Windows システムから AIX システムにコピーするインポート操作もサポートされています。IMPORT コマンドには、最初のファイルの名前だけを指定する必要があります。例えば、`IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1` のように記述します。 data.002 などのファイルも、data.ixf と同じディレクトリーに入れておく必要があります。

Windows オペレーティング・システムの場合は、以下のとおりです。

- 論理分割された PC/IXF ファイルのインポートはサポートされていません。
- 不正な形式の PC/IXF または WSF ファイルのインポートは、サポートされていません。

内部形式のセキュリティー・ラベルには、改行文字が含まれている可能性があります。 DEL ファイル形式を使用してファイルをインポートする場合、それらの改行文字が間違って区切りと解釈される可能性があります。この問題が起きた場合は、IMPORT コマンドで `delprioritychar` ファイル・タイプ修飾子を指定することによって、区切り文字に以前のデフォルト優先順位を使用してください。

フェデレーテッドに関する考慮事項

IMPORT コマンドで **INSERT**、**UPDATE**、または **INSERT_UPDATE** コマンド・パラメーターを使用するときには、関係するニックネームに対する CONTROL 特権があることを確認してください。インポート操作で使用するニックネームがすでに存在することを確認する必要があります。そのほかにも、IMPORT コマンド・パラメーターのセクションに記載されているようないくつかの制約事項に注意する必要があります。

一部のデータ・ソース (ODBC など) では、ニックネームへのインポートがサポートされていません。

インポート・ユーティリティーのファイル・タイプ修飾子

表 14. インポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式

修飾子	説明
compound=*x*	*x* は、1 から 100 までの (両端を含む) 数値です。非アトミックのコンパウンド SQL を使用してデータを挿入し、毎回 *x* 個のステートメントを試行します。 この修飾子を指定した場合に、トランザクション・ログに十分な大きさがないと、インポート操作は失敗します。トランザクション・ログは、**COMMITCOUNT** で指定されている行数か、**COMMITCOUNT** が指定されていなければデータ・ファイルに含まれている行数に対応できるだけの大きさでなければなりません。したがって、トランザクション・ログのオーバーフローを回避するために、**COMMITCOUNT** オプションを指定することをお勧めします。 この修飾子は、**INSERT_UPDATE** モードや階層表とは互換性がありません。さらに、`usedefaults`、`identitymissing`、`identityignore`、`generatedmissing`、`generatedignore` の各修飾子とも互換性がありません。
generatedignore	この修飾子を指定すると、インポート・ユーティリティーは、データ・ファイルに入っている、すべての生成済み列のデータを無視するようになります。その結果、生成済み列のすべての値がユーティリティーによって生成されます。この修飾子を `generatedmissing` 修飾子と一緒に使用することはできません。
generatedmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに生成済み列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この修飾子を `generatedignore` 修飾子と一緒に使用することはできません。
identityignore	この修飾子を指定すると、インポート・ユーティリティーは、データ・ファイルに入っている、ID 列のデータを無視するようになります。その結果、すべての IDENTITY 値がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を `identitymissing` 修飾子と一緒に使用することはできません。
identitymissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ID 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。この修飾子を `identityignore` 修飾子と一緒に使用することはできません。
lobsinfile	*lob-path* では、LOB データが含まれているファイルのパスを指定します。 各パスには、データ・ファイルの LOB ロケーション指定子 (LLS) によって参照されている LOB が少なくとも 1 つ入っているファイルが 1 つ以上含まれています。LLS は、LOB ファイル・パスに格納されているファイルの LOB の位置を示したストリング表記です。LLS の形式は、*filename.ext.nnn.mmm/* になります (*filename.ext* は、LOB が含まれているファイルの名前、*nnn* は、そのファイルに入っている LOB のオフセット (バイト単位)、*mmm* は、その LOB の長さ (バイト単位) です)。例えば、データ・ファイルにストリング `db2exp.001.123.456/` が格納されている場合は、ファイル db2exp.001 のオフセット 123 に LOB が配置されていて、その長さは 456 バイトということになります。 lobsinfile 修飾子を使用するときには、LOB ファイルの配置場所を **LOBS FROM** 節で指定します。**LOBS FROM** 節を指定すると、LOBSINFILE の動作が暗黙的にアクティブになります。**IMPORT** ユーティリティーは、データをインポートするときに、LOB ファイルを検索するためのパスのリストを LOBS FROM 節から受け取ります。 NULL LOB を指定する場合は、サイズとして -1 を入力します。サイズとして 0 を指定すると、長さ 0 の LOB として処理されます。長さ -1 の NULL LOB の場合は、オフセットとファイル名が無視されます。例えば、NULL LOB の LLS は、db2exp.001.7.-1/ のようになります。
no_type_id	1 つの副表にインポートする場合にのみ有効です。通常の表からエクスポートしたデータについて、この修飾子を使用してインポート操作を起動し、データを 1 つの副表に変換する、というのが典型的な使用法です。
nodefaults	ターゲット表の列のソース列を明示的に指定しない場合に、その表列が NULL 可能でなければ、デフォルト値はロードされません。このオプションを指定しない状態で、ターゲット表のいずれかの列のソース列を明示的に指定しない場合は、以下のいずれか動作が発生します。 • 列のデフォルト値を指定できる場合は、そのデフォルト値がロードされます。 • 列が NULL 可能で、その列のデフォルト値を指定できない場合は、NULL がロードされます。 • 列が NULL 可能ではなく、デフォルト値も指定できない場合は、エラーが戻され、ユーティリティーが処理を停止します。
norowwarnings	リジェクトされた行についてのすべての警告を抑止します。

表 14. インポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
rowchangetimestampignore	この修飾子を指定すると、インポート・ユーティリティーは、データ・ファイルに入っている、ROW CHANGE TIMESTAMP 列のデータを無視するようになります。その結果、すべての ROW CHANGE TIMESTAMP がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を `rowchangetimestampmissing` 修飾子と一緒に使用することはできません。
rowchangetimestampmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ROW CHANGE TIMESTAMP 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。この修飾子を `rowchangetimestampignore` 修飾子と一緒に使用することはできません。
seclabelchar	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、ストリング・フォーマットのセキュリティー・ラベル値であることを指定します。IMPORT は、ロード時に各セキュリティー・ラベルを内部形式に変換します。ストリングが正しい形式になっていないと、行はロードされず、警告 (SQLSTATE 01H53) が戻されます。ストリングが、表を保護するセキュリティー・ポリシーの一部である有効なセキュリティー・ラベルに対応していなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3243W) が戻されます。 seclabelname 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、インポートは失敗し、エラー (SQLCODE SQL3525N) が戻されます。
seclabelname	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、名前で示されていることを指定します。IMPORT は、その名前に対応する適切なセキュリティー・ラベルがあれば、その名前をそのセキュリティー・ラベルに変換します。表を保護するセキュリティー・ポリシーに、その名前に対応するセキュリティー・ラベルが存在しなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3244W) が戻されます。 seclabelchar 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、インポートは失敗し、エラー (SQLCODE SQL3525N) が戻されます。 **注:** ファイル・タイプが ASC の場合、セキュリティー・ラベル名の後のスペースは、名前の一部と解釈されます。そのような動作を避けるには、`striptblanks` ファイル・タイプ修飾子を使用して、スペースを除去するようにします。
usedefaults	ターゲット表の列のソース列が指定されていても、1 つ以上の行インスタンスでその列にデータが入っていない場合は、デフォルト値がロードされます。欠落データの例を以下に示します。 • DEL ファイル: 列の値として、2 つの隣接した列区切り (",,") や、任意の数のスペースで分離した 2 つの列区切り (", ,") が指定されている場合。 • DEL/ASC/WSF ファイル: 十分な数の列がない行や、元の指定に対応した十分な長さがない行。 **注:** ASC ファイルの場合、NULL 列値は、明示的な欠落とは見なされず、NULL 列値の代わりにデフォルトが入ることもありません。数値、日付、時刻、タイム・スタンプの列では、全桁スペース文字で NULL 列値を表記します。また、どのタイプの列でも、NULL INDICATOR を使用すれば、その列が NULL であることを示せます。 このオプションを指定しない場合に、行インスタンスのソース列にデータが入っていないと、以下のいずれかの動作が発生します。 • DEL/ASC/WSF ファイル: 列が NULL 可能であれば、NULL がロードされます。列が NULL 可能でなければ、ユーティリティーによって行がリジェクトされます。

表 15. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL)

修飾子	説明
codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、入力データ・セットに含まれているデータのコード・ページとして解釈されます。インポート操作の実行時に、文字データは、このコード・ページからアプリケーション・コード・ページに変換されます。 以下の規則が適用されます。 • ピュア DBCS (GRAPHIC)、混合 DBCS、EUC では、区切りが x00 から x3F の範囲 (両端を含む) に限定されています。 • nullindchar では、標準の ASCII セットのコード・ポイント x20 から x7F の範囲 (両端を含む) に含まれているシンボルを指定する必要があります。この修飾子では、ASCII のシンボルとコード・ポイントを参照します。 **注:** 1. codepage 修飾子を lobsinfile 修飾子と一緒に使用することはできません。 2. コード・ページをアプリケーション・コード・ページからデータベース・コード・ページに変換するときに、データの拡張が発生すると、データが切り捨てられ、データが失われる可能性があります。
dateformat="*x*"	*x* は、ソース・ファイルの日付の形式です。[2] 有効な日付エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (1 から 12 の範囲の 2 桁の数。 M とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月エレメントとは 相互に排他的) 指定されていないそれぞれのエレメントには、デフォルト値の 1 が割り当てられます。日付形式の例を以下に示します。 "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
implieddecimal	暗黙の小数点の位置が列定義によって決まるようになり、値の末尾という想定がなくなります。例えば、値 12345 は DECIMAL(8,2) 列に 12345.00 としてではなく、123.45 としてロードされます。
timeformat="*x*"	*x* は、ソース・ファイルの時刻の形式です。[2] 有効な時刻エレメントは、以下のとおりです。 H - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数) HH - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 2 桁の数; H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) 指定されていないそれぞれのエレメントには、デフォルト値の 0 が割り当てられます。時刻形式の例を以下に示します。 "HH:MM:SS" "HH.MM TT" "SSSSS"

表 15. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明

timestampformat="*x*"	*x* は、ソース・ファイルのタイム・スタンプの形式です。[2] 有効なタイム・スタンプ・エレメントは、以下のとおりです。
	
	YYYY - 年 (0000 から 9999 の範囲の 4 桁の数)
	M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数)
	MM - 月 (01 から 12 の 2 桁の数。
	M および MMM とは相互に排他的)
	MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。
	M と MM とは相互に排他的)
	D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数)
	DD - 日 (1 から 31 の範囲の 2 桁の数。
	D とは相互に排他的)
	DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。
	他の日または月のエレメントとは相互に排他的)
	H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の
	範囲の 1 桁または 2 桁の数。)
	HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の
	範囲の 2 桁の数。
	H と相互に排他的)
	M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数)
	MM - 分 (0 から 59 の範囲の 2 桁の数。
	M (分) とは相互に排他的)
	S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数)
	SS - 秒 (0 から 59 の範囲の 2 桁の数。
	S と相互に排他的)
	SSSSS - 夜中の 12 時から数えた秒数
	(00000 から 86399 の範囲の 5 桁の数。
	他の時刻エレメントとは相互に排他的)
	UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。
	他のマイクロ秒エレメントとは相互に排他的)
	UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。
	000000 から 999990 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数。
	000000 から 999900 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。
	000000 から 999000 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数。
	000000 から 990000 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。
	000000 から 900000 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	TT - 午前/午後の指定子 (AM または PM)
	
	指定されていない YYYY、M、MM、D、DD、DDD のいずれかのエレメントには、デフォルト値の 1 が割り当てられます。指定されていない MMM エレメントには、デフォルト値の 'Jan' が割り当てられます。指定されていない他のすべてのエレメントには、デフォルト値の 0 が割り当てられます。タイム・スタンプ形式の例を以下に示します。
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM エレメントの有効な値は、
	'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov'、'dec' です。これらの値では、大/小文字は区別されません。
	
	ユーザー定義の日付と時刻の形式が含まれているデータを schedule という表にインポートする例を以下に示します。
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

表 15. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
usegraphiccodepage	usegraphiccodepage を指定すると、グラフィックまたは 2 バイト文字のラージ・オブジェクト (DBCLOB) データ・フィールドにインポートするデータは、グラフィック・コード・ページのデータであるという想定で、処理が行われます。残りのデータは、文字コード・ページのデータであるという想定になります。グラフィック・コード・ページは、文字コード・ページに関連付けられています。IMPORT は、codepage 修飾子が指定されていればその修飾子によって、codepage 修飾子が指定されていなければアプリケーションのコード・ページによって、文字コード・ページを判別します。 ドロップ済み表のリカバリーで生成される区切り付きデータ・ファイルとこの修飾子を併用するのは、リカバリーする表にグラフィック・データが入っている場合に限られます。 **制約事項** EXPORT ユーティリティーで作成される DEL ファイルでは、usegraphiccodepage 修飾子を指定しないでください。そのファイルには、1 つのコード・ページでエンコードされたデータだけが入っているからです。usegraphiccodepage 修飾子は、ファイルに含まれている 2 バイト文字ラージ・オブジェクト (DBCLOB) でも無視されます。
xmlchar	XML 文書が文字コード・ページでエンコードされていることを指定します。 指定の文字コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードが文字コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。文字コード・ページは、codepage ファイル・タイプ修飾子で指定されている値か、その修飾子が指定されていない場合はアプリケーション・コード・ページになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。
xmlgraphic	XML 文書が指定のグラフィック・コード・ページでエンコードされていることを指定します。 指定のグラフィック・コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードがグラフィック・コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。グラフィック・コード・ページは、codepage ファイル・タイプ修飾子で指定されている値のグラフィック・コンポーネントか、その修飾子が指定されていない場合はアプリケーション・コード・ページのグラフィック・コンポーネントになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。 **注:** IMPORT コマンドで xmlgraphic 修飾子を指定する場合は、インポート対象の XML 文書のエンコードが UTF-16 コード・ページになっている必要があります。そうでない場合は、XML 文書が構文解析エラーでリジェクトされるか、表にインポートされてもデータ破損が生じる可能性があります。

表 16. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASC (区切りなし ASCII) ファイル形式

修飾子	説明
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のインポートが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にインポートできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。
nullindchar=*x*	*x* は、単一文字です。NULL 値を示す文字を *x* に変更します。*x* のデフォルト値は、Y です。[3] この修飾子は、EBCDIC データ・ファイルでは大文字と小文字の区別があります。ただし、文字が英字の場合は例外です。例えば、NULL 標識文字が N に指定されている場合は、n も NULL 標識として認識されます。

表 16. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASC (区切りなし ASCII) ファイル形式 (続き)

修飾子	説明
reclen=*x*	*x* は、最大値 32,767 の整数です。各行では *x* 個の文字が読み取られ、行の終わりを示す改行文字は使用されません。

striptblanks	可変長フィールドにデータをロードするときに、末尾ブランク・スペースを切り捨てます。このオプションを指定しなければ、ブランク・スペースは維持されます。
	
	以下の例では striptblanks を指定しているので、インポート・ユーティリティーは、末尾ブランク・スペースを切り捨てます。
	`db2 import from myfile.asc of asc`
	`  modified by striptblanks`
	`  method l (1 10, 12 15) messages msgs.txt`
	`  insert into staff`
	
	このオプションを striptnulls と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの t の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。

striptnulls	可変長フィールドにデータをロードするときに、末尾 NULL (0x00 文字) を切り捨てます。このオプションを指定しなければ、NULL は維持されます。 このオプションを striptblanks と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの padwithzero の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。

表 17. インポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式

修飾子	説明

chardel*x*	*x* は、単一文字ストリング区切りです。デフォルト値は、二重引用符 (") です。文字ストリングを囲む二重引用符の代わりに指定の文字を使用します。[34] 文字ストリング区切りとして二重引用符を明示的に指定する場合は、以下のように指定します。
	`modified by chardel""`
	
	文字ストリング区切りとして単一引用符 (') を指定することもできます。以下の例では chardel'' を指定しているので、インポート・ユーティリティーは、検出する単一引用符 (') を文字ストリング区切りとして解釈します。
	`db2 "import from myfile.del of del`
	`  modified by chardel''`
	`  method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x* は、単一文字列区切りです。デフォルト値は、コンマ (,) です。列の終わりを示すコンマの代わりに指定の文字を使用します。[34]
	
	以下の例では coldel; を指定しているので、インポート・ユーティリティーは、検出するセミコロン (;) を列区切りとして解釈します。
	`db2 import from myfile.del of del`
	`  modified by coldel;`
	`  messages msgs.txt insert into staff`

表 17. インポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式 (続き)

修飾子	説明
decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) の代わりにブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進値の接頭部として正符号を使用します。

decpt*x*	*x* は、小数点文字としてピリオドの代わりに使用する単一文字です。デフォルト値は、ピリオド (.) です。小数点文字として、ピリオドの代わりに指定の文字を使用します。[34]
	
	以下の例では `decpt;` を指定しているので、インポート・ユーティリティーは、検出するセミコロン (;) を小数点として解釈します。
	
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`decpt; messages msgs.txt insert into staff"`

delprioritychar	区切り文字に関する現在のデフォルトの優先順位は、レコード区切り、文字区切り、列区切り、という順序になっています。この修飾子を指定すると、区切り文字の優先順位が、文字区切り、レコード区切り、列区切り、という順序に戻されるので、古い優先順位に依存する既存のアプリケーションが保護されます。構文:
	
	`db2 import ... modified by delprioritychar ...`
	
	例えば、以下の DEL データ・ファイルがあるとします。
	
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	
	`delprioritychar` 修飾子を指定しているので、このデータ・ファイルは、2 行だけになります。1 番目と 3 番目の <row delimiter> は、実際のレコード区切りとして解釈されますが、2 番目の <row delimiter> は、第 2 行の最初のデータ列の一部として解釈されるからです。この修飾子を指定しなければ、それぞれの <row delimiter> が区切り文字として解釈され、このデータ・ファイルは 3 行になります。

keepblanks	タイプ CHAR、VARCHAR、LONG VARCHAR、CLOB の各フィールドで前後のブランクを保持します。このオプションを指定しないと、文字区切りの内側にない前後のすべてのブランクが除去され、表のすべてのブランク・フィールドに NULL が挿入されます。
nochardel	インポート・ユーティリティーは、列区切りの間で検出するすべてのバイトを列のデータの一部と見なします。文字区切りも、列データの一部として解析されます。DB2 でエクスポートしたデータについては、このオプションを指定しないでください (ただし、エクスポート時に nochardel を指定していた場合は例外です)。このオプションは、文字区切りのないベンダー・データ・ファイルをサポートするために用意されています。正しくない使い方をすると、データが失われたり破損したりする可能性があります。 このオプションを `chardelx`、`delprioritychar`、`nodoubledel` のいずれかと一緒に指定することはできません。これらは、相互に排他的なオプションです。
nodoubledel	二重文字区切りの認識を抑止します。

表 18. インポート・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式

修飾子	説明
forcein	ユーティリティーは、コード・ページの不一致があってもデータを受け入れ、コード・ページの変換を抑止します。 固定長ターゲット・フィールドについては、データを収容するだけの大きさがあるかどうかのチェックが行われます。nochecklengths を指定すると、チェックなしで各行のインポートが試行されます。
indexixf	ユーティリティーは、既存の表に現在定義されているすべての索引をドロップし、PC/IXF ファイルの索引定義から新しい索引を作成します。このオプションを使用できるのは、表の内容を置き換える場合に限られます。ビューで使用することはできません。*insert-column* を指定した場合も、使用できません。
indexschema=*schema*	索引作成時に、索引名として指定の *schema* を使用します。 *schema* を指定しない場合に、キーワード indexschema が指定されていれば、接続ユーザー ID が使用されます。そのキーワードが指定されていなければ、IXF ファイルのスキーマが使用されます。
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のインポートが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にインポートできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。
forcecreate	インポート操作時に情報が欠落していたり、限定されていたりする場合でも、SQL3311N を戻してから、表を作成することを指定します。

表 19. codepage と usegraphiccodepage を使用する場合の IMPORT の動作

codepage=N	usegraphiccodepage	IMPORT の動作
なし	なし	ファイル内のすべてのデータは、アプリケーション・コード・ページのデータであるという想定になります。
あり	なし	ファイル内のすべてのデータは、コード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにインポートすると、グラフィック・データが破損します。
なし	あり	ファイル内の文字データは、アプリケーション・コード・ページのデータであるという想定になります。グラフィック・データは、アプリケーション・グラフィック・データのコード・ページのデータであるという想定になります。 アプリケーション・コード・ページが 1 バイトの場合は、すべてのデータがアプリケーション・コード・ページのデータであるという想定になります。 **警告:** アプリケーション・コード・ページが 1 バイトの場合に、グラフィック・データをデータベースにインポートすると、データベースにグラフィック列が含まれていても、グラフィック・データは破損します。

表 19. *codepage* と *usegraphiccodepage* を使用する場合の *IMPORT* の動作 (続き)

codepage=N	usegraphiccodepage	IMPORT の動作
あり	あり	文字データは、コード・ページ N のデータであるという想定になります。グラフィック・データは、N のグラフィック・コード・ページのデータであるという想定になります。 N が 1 バイトまたは 2 バイトのコード・ページの場合は、すべてのデータがコード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにインポートすると、グラフィック・データが破損します。

注:

1. **MODIFIED BY** オプションでサポートされていないファイル・タイプを使用しようとしても、インポート・ユーティリティーからは警告が生成されません。その場合は、インポート操作が失敗し、エラー・コードが戻されます。
2. 日付形式ストリングを二重引用符で囲むのは、必須です。フィールド区切り文字には、a から z、A から Z、0 から 9 を組み込めません。フィールド区切り文字として、DEL ファイル形式の文字区切りまたはフィールド区切りと同じ文字を使用することはできません。エレメントの開始位置と終了位置があいまいでない場合は、フィールド区切り文字はオプションになります。修飾子によっては、項目が可変長の場合に D、H、M、S などのエレメントを使用することがあり、そのような場合は、開始位置と終了位置があいまいになることがあります。

   タイム・スタンプ形式の場合は、月の記述子と分の記述子の間であいまいさが残らないように注意する必要があります。どちらも、M という文字を使用するからです。月のフィールドは、他の日付フィールドと隣接している必要があります。分のフィールドは、他の時刻フィールドと隣接している必要があります。あいまいなタイム・スタンプ形式の例を以下に示します。

   ```
   "M" (月または分のどちらにもとれる)
   "M:M" (月と分の区別がつかない)
   "M:YYYY:M" (両方とも月と解釈される)
   "S:M:YYYY" (時刻値と日付値の両方に隣接している)
   ```

   あいまいな場合は、ユーティリティーによってエラー・メッセージが生成され、操作は失敗します。

   あいまいでないタイム・スタンプ形式の例を以下に示します。

   ```
   "M:YYYY" (M (月))
   "S:M" (M (分))
   "M:YYYY:S:M" (M (月)....M (分))
   "M:H:YYYY:M:D" (M (分)....M (月))
   ```

   二重引用符や円記号など、いくつかの文字の前ではエスケープ文字を使用する必要があります (¥ など)。

3. ファイル・タイプ修飾子 chardel、coldel、decpt に指定する文字値は、ソース・データのコード・ページに指定されている文字値でなければなりません。

   文字コード・ポイント (文字シンボルではない) を指定する場合は、xJJ または 0xJJ という構文を使用できます (JJ は、コード・ポイントの 16 進表記です)。例えば、列区切りとして # 文字を指定する場合は、以下のいずれかを使用します。

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

4. 区切り文字のオーバーライドとして使用できる文字に適用される制約事項については、『*データ移動のための区切り文字の制約事項*』を参照してください。
5. ニックネームにインポートする場合、以下のファイル・タイプ修飾子は使用できません。
   - indexixf
   - indexschema
   - dldelfiletype
   - nodefaults
   - usedefaults
   - no_type_idfiletype
   - generatedignore
   - generatedmissing
   - identityignore
   - identitymissing
   - lobsinfile
6. XML 列では、**WSF** ファイル形式はサポートされていません。
7. XML 列では、**CREATE** モードはサポートされていません。
8. すべての XML データをメイン・データ・ファイルとは別の XML ファイルに配置する必要があります。メイン・データ・ファイルの各 XML 列に XML Data Specifier (XDS) (または NULL 値) が入っている必要があります。
9. ファイル・タイプ修飾子 XMLCHAR または XMLGRAPHIC が指定されている場合を除き、XML 文書は、Unicode 形式であるか、エンコード属性の宣言タグが含まれているという前提で処理が行われます。
10. 整形式でない文書が含まれている行はリジェクトされます。
11. **XMLVALIDATE** オプションを指定した場合、対応するスキーマによる妥当性検査に成功した文書には、挿入時にスキーマ情報の注釈が付けられます。対応するスキーマによる妥当性検査に失敗した文書が含まれている行はリジェクトされます。妥当性検査を正常に実行するには、インポートを起動するユーザーの特権に、少なくとも以下のいずれかの権限が含まれている必要があります。
    - SYSADM または DBADM 権限
    - 妥当性検査に使用する XML スキーマに対する USAGE 特権
12. 暗黙的な非表示設定になっている Row Change Timestamp 列が含まれている表にインポートする場合は、その列の暗黙的な非表示のプロパティーが適用されません。したがって、インポートするデータに列のデータが含まれていない場

合に、明示的な列リストも存在しなければ、インポート・コマンドで rowchangetimestampmissing ファイル・タイプ修飾子を指定することが必要です。

IMPORT コマンド (ADMIN_CMD プロシージャーを使用)

外部ファイルのデータを、サポートされているファイル・フォーマットで表、階層、ビュー、またはニックネームに挿入します。 LOAD はより高速な代替方法です。しかしロード・ユーティリティーでは、階層レベルのデータのロードはサポートされていません。

114 ページの『インポート・ユーティリティーのファイル・タイプ修飾子』へのクイック・リンク。

許可

- INSERT オプションを使用して **IMPORT** コマンドを実行する場合、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 関係するそれぞれの表、ビュー、またはニックネームに対する CONTROL 特権
  - 関係するそれぞれの表またはビューに対する INSERT および SELECT 特権
- INSERT_UPDATE オプションを使用して既存の表に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 関係するそれぞれの表、ビュー、またはニックネームに対する CONTROL 特権
  - 関係するそれぞれの表またはビューに対する INSERT、 SELECT、UPDATE、および DELETE 特権
- REPLACE または **REPLACE_CREATE** オプションを使用して既存の表に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 表またはビューに対する CONTROL 特権
  - 表またはビューに対する INSERT、SELECT、および DELETE 特権
- CREATE または **REPLACE_CREATE** オプションを使用して新規の表に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - データベースに対する CREATETAB 権限、表スペースに対する USE 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 (表の暗黙または明示スキーマ名がない場合)

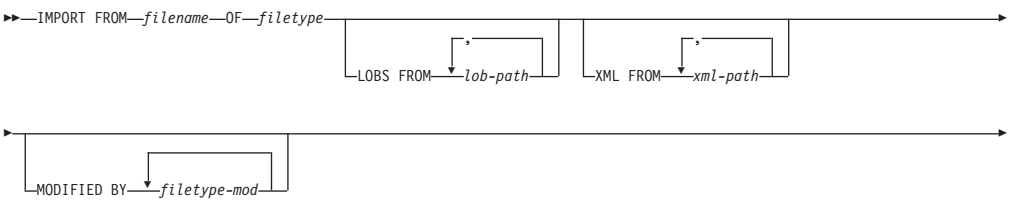
- スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

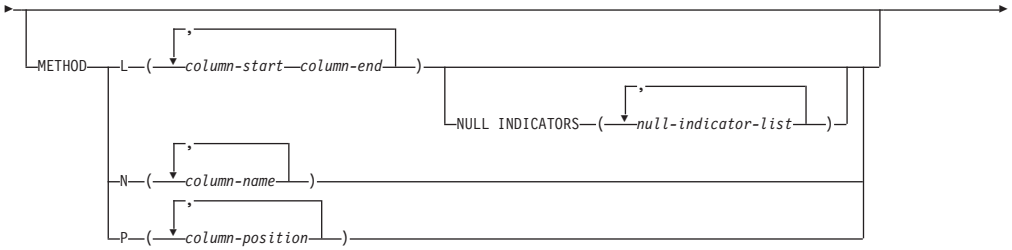
- CREATE または **REPLACE_CREATE** オプションを使って、存在しない階層に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - データベースに対する CREATETAB 権限および表スペースに対する USE 特権と、以下のいずれか。
    - データベースに対する IMPLICIT_SCHEMA 権限 (表のスキーマ名が存在しない場合)
    - スキーマに対する CREATEIN 特権 (表のスキーマが存在する場合)
    - 階層全体に対して **REPLACE_CREATE** オプションが使用されている場合は、階層内のすべての副表に対する CONTROL 特権
- REPLACE オプションを使用して既存の階層に **IMPORT** するには、以下のいずれかが必要です。
  - *sysadm*
  - *dbadm*
  - 階層内のすべての副表に対する CONTROL 特権
- 保護列を持つ表にデータをインポートするには、表内のすべての保護列への書き込みアクセスを可能にする LBAC 信用証明情報がセッション許可 ID に必要です。そうでない場合、インポートは失敗し、エラー (SQLSTATE 42512) が戻されます。
- 保護されている行のある表にデータをインポートするには、セッション許可 ID に、以下の基準を満たす LBAC クリデンシャルが必要です。
  - 表を保護しているセキュリティー・ポリシーの一部である
  - 書き込みアクセスに関して、セッション許可 ID に付与された

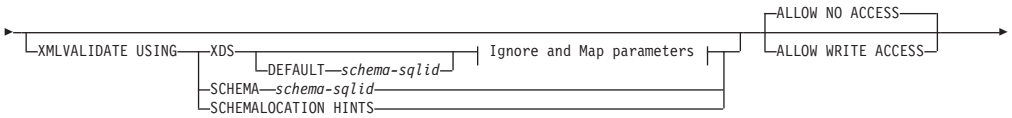
  挿入する行上のラベル、ユーザーの LBAC 信用証明情報、セキュリティー・ポリシーの定義、および LBAC 規則によって、行上のラベルが決まります。
- **REPLACE** または **REPLACE_CREATE** オプションが指定された場合、セッション許可 ID には、その表をドロップするための権限が付与されていなければなりません。

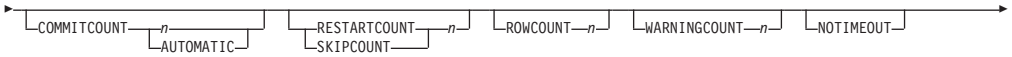
必要な接続

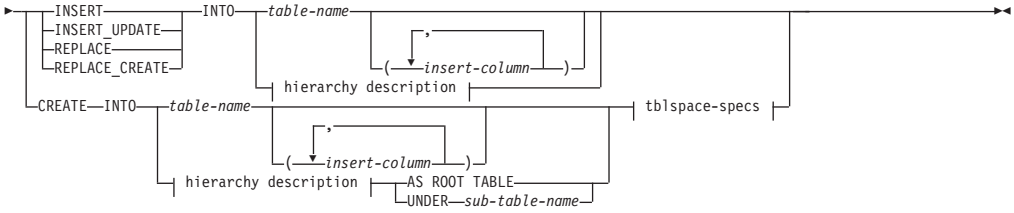
コマンド構文



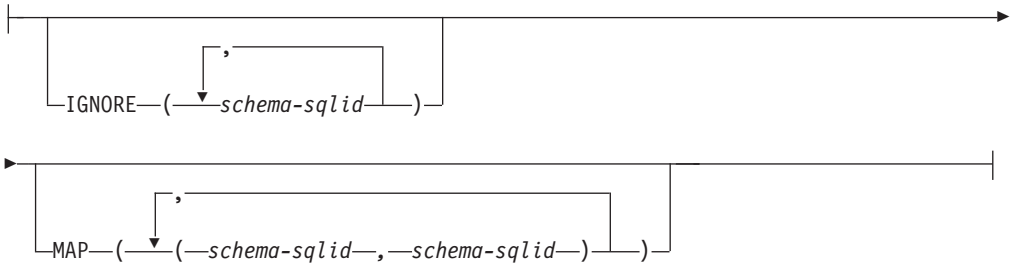






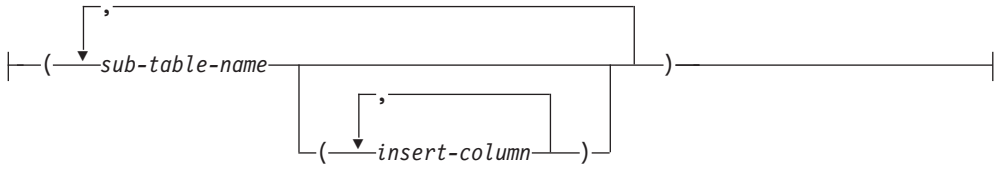


Ignore and Map parameters:



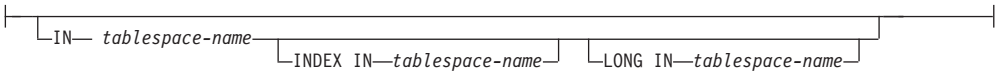
hierarchy description:

sub-table-list:



traversal-order-list:

tblspace-specs:



コマンド・パラメーター

ALL TABLES

階層専用の暗黙キーワード。階層をインポートする場合、走査順序で指定されるすべての表をインポートすることがデフォルトです。

ALLOW NO ACCESS

オフライン・モードでインポートを実行します。行の挿入の前には常に、ターゲット表に排他 (X) ロックがかけられます。これで、同時アプリケーションは表データにアクセスできなくなります。これがデフォルトのインポート動作です。

ALLOW WRITE ACCESS

オンライン・モードでインポートを実行します。最初の行の挿入時には、ターゲット表に意図的排他 (IX) ロックがかけられます。これで、表データへの同時の読み取りおよび書き出しアクセスが可能になります。オンライン・モードには、**REPLACE**、**CREATE**、または **REPLACE_CREATE** インポート・オプションとの互換性はありません。 オンライン・モードとバッファー挿入との連携はサポートされません。インポート操作によって挿入後のデータが定期的にコミットされるので、表ロックへのロック・エスカレーションが削減され、アクティブなログ・スペースが使い果たされることはなくなります。このようなコミットは、**COMMITCOUNT** オプションを使わなくても実行されます。 各コミットごとに、インポートでは IX 表ロックが外されるので、コミットの完了後に再びロックの設定が試みられます。ニックネームにインポートするときにはこのパラメーターが必要で、有効な数値を使って **COMMITCOUNT** を指定する必要があります (AUTOMATIC は有効なオプションとは見なされません)。

AS ROOT TABLE

1 つ以上の副表を、独立した表階層として作成します。

COMMITCOUNT *n* **| AUTOMATIC**

n 個のレコードがインポートされるたびに COMMIT を実行します。数 *n* を指定すると、インポートでは *n* 個のレコードのインポートの後にそのつど COMMIT が実行されます。コンパウンド挿入を使用した場合、ユーザー指定のコミット頻度 *n* は、そのコンパウンド・カウント値に最も近い整数の倍数に切り上げられます。 AUTOMATIC を指定すると、コミットの必要時期はインポート操作で内部的に判別されます。 次の 2 つのうちのいずれかの理由で、このユーティリティーはコミットを行います。

- アクティブ・ログ・スペースを使いきらないようにするため。

- ロックが行レベルから表レベルにエスカレーションしないようにするため。

ALLOW WRITE ACCESS オプションを指定した場合に **COMMITCOUNT** オプションを指定しないと、インポート・ユーティリティーは、**COMMITCOUNT** AUTOMATIC が指定されたものとしてコミットを実行します。

インポート操作で、アクティブ・ログ・スペースを使いきらないようにできるかどうかは、DB2 レジストリー変数 **DB2_FORCE_APP_ON_MAX_LOG** によって影響を受けます。

- **DB2_FORCE_APP_ON_MAX_LOG** が FALSE に設定され、**COMMITCOUNT** AUTOMATIC コマンド・オプションが指定されている場合、インポート・ユーティリティーは自動的に、アクティブ・ログ・スペースを使いきらないようにすることができます。
- **DB2_FORCE_APP_ON_MAX_LOG** が FALSE に設定され、**COMMITCOUNT** *n* コマンド・オプションが指定されている場合、インポート・ユーティリティーは、レコードの挿入または更新中に SQL0964C (トランザクション・ログがいっぱい) が発生した場合に、ログ満杯状態を解決しようと試みます。インポート・ユーティリティーは、無条件コミットを実行してから、レコードの挿入または更新を再試行します。それでも問題が解決しない場合 (ログの満杯状態がデータベース上の他のアクティビティーが原因で起きている場合など)、IMPORT コマンドは予期したとおりに失敗します。ただし、コミットされた行の数は、**COMMITCOUNT** *n* の値の倍数になっていない可能性があります。インポート操作の再試行時に、既にコミットされている行を処理しないようにするには、**RESTARTCOUNT** または **SKIPCOUNT** コマンド・パラメーターを使用します。
- **DB2_FORCE_APP_ON_MAX_LOG** が TRUE (デフォルト) に設定されている場合、レコードの挿入または更新中に SQL0964C が発生すると、インポート操作は失敗します。これは、**COMMITCOUNT** AUTOMATIC または **COMMITCOUNT** *n* のどちらを指定するかに関係なく発生します。

  アプリケーションは強制的にデータベースから切断され、現行作業単位はロールバックされます。インポート操作の再試行時に、既にコミットされている行を処理しないようにするには、**RESTARTCOUNT** または **SKIPCOUNT** コマンド・パラメーターを使用します。

CREATE

注: CREATE パラメーターは推奨されておらず、今後のリリースで除去される可能性があります。詳細については、『推奨されなくなった IMPORT コマンド・オプション **CREATE** および **REPLACE_CREATE**』を参照してください。

データベースのコード・ページで表の定義と行の内容を作成します。 DB2 の表、副表、または階層からエクスポートされたデータの場合、索引も作成されます。このオプションが階層に対するものである場合に、DB2 からデータがエクスポートされると、タイプ階層も作成されます。 このオプションは、IXF ファイルの場合にのみ使用することができます。

ニックネームにインポートするときには、このパラメーターは無効です。

注: データが MVS ホスト・データベースからエクスポートされたもので、ページ・サイズで計算した長さが 254 より大きい LONGVAR フィールドを含んでいる場合、**CREATE** は行が長過ぎるために失敗します。 制約事項のリストについては、『『インポート済みの表の再作成』』を参照してください。この場合、その表は手動で作成します。そして、IMPORT に **INSERT** を指定して呼び出すか、または LOAD コマンドを使用してください。

DEFAULT *schema-sqlid*
このオプションは、**USING XDS** パラメーターを指定した場合にのみ使用できます。 **DEFAULT** 節で指定されたスキーマは、インポート対象 XML 文書の XML Data Specifier (XDS) に XML スキーマを指定する SCH 属性が含まれていない場合に、妥当性検査のために使用するスキーマとなります。

DEFAULT 節は、**IGNORE** 節および **MAP** 節よりも優先されます。 XDS が **DEFAULT** 節を満たすなら、**IGNORE** と **MAP** の指定は無視されます。

FROM *filename*

HIERARCHY
階層データをインポートするよう指定します。

IGNORE *schema-sqlid*
このオプションは、**USING XDS** パラメーターを指定した場合にのみ使用できます。 **IGNORE** 節は、SCH 属性によって指定されていても無視するスキーマとして、1 つ以上のスキーマのリストを指定します。 インポートする XML 文書の XML Data Specifier の中に SCH 属性が存在し、その SCH 属性によって指定されるスキーマが無視するスキーマ・リストに含まれている場合には、インポートするその XML 文書についてスキーマ妥当性検査は実行されません。

あるスキーマが **IGNORE** 節の中で指定されている場合、**MAP** 節のスキーマ・ペアの左辺にそれを含めることはできません。

IGNORE 節は XDS にのみ適用されます。 あるスキーマが **IGNORE** 節によって指定されていても、それが **MAP** 節によってマップされているなら、それ以降そのスキーマが無視されることはありません。

IN *tablespace-name*
表を作成する表スペースを指定します。表スペースは存在している必要があり、REGULAR 表スペースでなければなりません。他の表スペースを指定しない場合、すべての表パーツはこの表スペースに保管されます。この節を指定しない場合、表は許可 ID によって作成された表スペース中に作成されます。何も検出されない場合、その表はデフォルト表スペースの USERSPACE1 に入れられます。 USERSPACE1 がドロップされていた場合、表作成は失敗します。

INDEX IN *tablespace-name*
表の索引を作成する表スペースを指定します。このオプションは、**IN** 節で指定される PRIMARY 表スペースが DMS 表スペースである場合のみ使用

できます。 指定した表スペースは存在している必要があり、かつ REGULAR または LARGE DMS 表スペースでなければなりません。

注: どの表スペースに索引を配置するかは、表を作成するときにのみ指定できます。

insert-column
データの挿入先となる表またはビュー内の列名を指定します。

INSERT
既存の表データを変更することなく、インポートされたデータを表に追加します。

INSERT_UPDATE
インポートしたデータ行をターゲット表に追加するか、または主キーが一致するものがあればターゲット表の既存行を更新します。

INTO *table-name*
データのインポート先となるデータベース表を指定します。この表として、システム表、宣言一時表、またはサマリー表は指定できません。

以前のサーバーの場合を除き、**INSERT**、**INSERT_UPDATE**、または **REPLACE** オプションには、完全修飾または非修飾の表名を使用しなければならないようなときでも、別名を使用することができます。修飾子付き表名は、*schema.tablename* の形式です。 *schema* には、表作成時のユーザー名が入ります。

LOBS FROM *lob-path*
LOB データ・ファイルの名前は、メイン・データ・ファイル (ASC、DEL、または IXF) の、LOB 列にロードされる列内に保管されます。指定できるパスの最大数は 999 です。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

ニックネームにインポートするときには、このパラメーターは無効です。

LONG IN *tablespace-name*
ロング列の値 (LONG VARCHAR、LONG VARGRAPHIC、LOB データ・タイプ、またはソース・タイプとしてこれらが指定されている特殊タイプ) を保管する表スペースを指定します。このオプションは、 **IN** 節で指定した PRIMARY 表スペースが DMS 表スペースである場合のみ使用できます。指定した表スペースは存在している必要があり、LARGE DMS 表スペースでなければなりません。

MAP *schema-sqlid*
このオプションは、**USING XDS** パラメーターを指定した場合にのみ使用できます。 **MAP** 節は、インポートする各 XML 文書について XML Data Specifier (XDS) の SCH 属性によって指定されるスキーマの代わりに使用する代替スキーマを指定するのに使用します。 **MAP** 節には、それぞれがあるスキーマから別のスキーマへのマッピングを表すスキーマ・ペアを 1 つ以上列挙したリストを指定します。 ペア中の最初のスキーマは、XDS 内の SCH 属性によって参照されるスキーマを表します。ペア中の 2 番目のスキーマは、スキーマ検証の実行で使用する必要のあるスキーマを表します。

あるスキーマが **MAP** 節のスキーマ・ペアの左辺で指定されている場合、**IGNORE** 節でさらにそれを指定することはできません。

スキーマ・ペアのマッピングが適用されたなら、その結果は最終的なものです。マッピング操作は推移的ではないため、選択されたスキーマが、それ以降に別のスキーマ・ペアのマッピングに適用されることはありません。

スキーマを複数回マップすることはできません。つまり、複数のペアの左辺に指定することはできません。

METHOD

L データのインポートを開始する列および終了する列の番号を指定します。列の番号は、データの行の先頭からのバイト単位のオフセットです。この番号は 1 から始まります。

注: このメソッドは、ASC ファイルの場合にのみ使用することができ、そのファイル・タイプに対してのみ有効なオプションです。

N インポートするデータ・ファイルの中の列の名前を指定します。これらの列名の大文字小文字の区別は、システム・カタログ内の対応する名前の大文字小文字の区別と一致しなければなりません。NULL 可能ではない各表の列には、**METHOD N** リスト内に対応する項目が必要です。例えば、データ・フィールドが F1、 F2、 F3、 F4、 F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、method N (F2, F1, F4, F3) は有効な要求ですが、 method N (F2, F1) は無効です。

注: この方式は、IXF ファイルの場合にのみ使用することができます。

P インポートする入力データ・フィールドのフィールド番号を指定します。

注: この方式は、IXF または DEL ファイルの場合にのみ使用でき、DEL ファイル・タイプに対してのみ有効なオプションです。

MODIFIED BY *filetype-mod*

ファイル・タイプ修飾子オプションを指定します。 114 ページの『インポート・ユーティリティーのファイル・タイプ修飾子』を参照してください。

NOTIMEOUT

インポート・ユーティリティーがロックの待機中にタイムアウトしないことを指定します。このオプションのほうが、 **locktimeout** データベース構成パラメーターより優先されます。他のアプリケーションは影響を受けません。

NULL INDICATORS *null-indicator-list*

このオプションは、**METHOD L** パラメーターを指定した場合にのみ使用できます。つまり、入力ファイルが ASC ファイルの場合です。 NULL 標識リストは、コンマで区切られた正の整数のリストで、各 NULL 標識フィールドの列の番号を指定します。列の番号は、データの行の先頭からのバイト単位の、各 NULL 標識フィールドのオフセットです。 NULL 標識リストには、**METHOD L** パラメーターで定義された各データ・フィールドに

対する 1 つの項目がなければなりません。列の番号がゼロであることは、対応するデータ・フィールド内に必ずデータがあることを示します。

NULL 標識列中の Y の値は、その列データが NULL であることを指定します。 NULL 標識列に Y *以外* の文字を指定した場合は、列データが NULL ではなく、**METHOD L** オプションで指定された列データがインポートされることを指定することになります。

nullindchar ファイル・タイプ修飾子を指定した **MODIFIED BY** オプションを使用すれば、NULL 標識文字を変更することができます。

OF *filetype*

入力ファイル内のデータのフォーマットを指定します。

- ASC (区切りなし ASCII フォーマット)
- DEL (区切り文字付き ASCII フォーマット)。さまざまなデータベース・マネージャーやファイル・マネージャー・プログラムで使用します
- WSF (ワークシート・フォーマット)。以下のプログラムで使用します。
  - Lotus 1-2-3
  - Lotus Symphony
- IXF (統合交換フォーマット、PC バージョン) は、DB2 専用のバイナリー・フォーマットです。

ニックネームにインポートするときには、WSF ファイル・タイプはサポートされません。

REPLACE

データ・オブジェクトを切り捨てることによって表内の既存のデータすべてを削除してから、インポートしたデータを挿入します。表定義および索引定義は変更されません。表がない場合は、このオプションを使用できません。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

ニックネームにインポートするときには、このパラメーターは無効です。

このオプションでは、CREATE TABLE ステートメントの NOT LOGGED INITIALLY (NLI) 節、あるいは ALTER TABLE ステートメントの ACTIVE NOT LOGGED INITIALLY 節は考慮されません。

NLI 節が呼び出される CREATE TABLE または ALTER TABLE ステートメントと同じトランザクションの中で、**REPLACE** オプションの指定されたインポートが実行された場合、インポートにおいてその NLI 節は考慮されません。挿入はすべてログに記録されます。

予備手段 1

DELETE ステートメントを使用して表の内容を削除した後、INSERT ステートメントによりインポートを呼び出す

予備手段 2

表をドロップしてからそれを再作成した後、INSERT ステートメントによってインポートを呼び出す

この制限は、DB2 Universal Database バージョン 7 および DB2 UDB バージョン 8 に適用されます。

REPLACE_CREATE

注: REPLACE_CREATE パラメーターは推奨されておらず、今後のリリースで除去される可能性があります。さらに詳しくは、『IMPORT コマンドの推奨されなくなったオプション CREATE および REPLACE_CREATE』を参照してください。

表がすでにある場合には、データ・オブジェクトを切り捨てることによって表内の既存のデータすべてを削除し、表定義や索引定義は変えることなく、インポートしたデータを挿入します。

表がまだない場合には、データベースのコード・ページで、表と索引の定義と行の内容を作成します。制約事項のリストについては、『*インポート済みの表の再作成*』を参照してください。

このオプションは、IXF ファイルの場合にのみ使用することができます。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

ニックネームにインポートするときには、このパラメーターは無効です。

RESTARTCOUNT *n*

n + 1 の位置のレコードからインポート操作を開始するよう指定します。最初の *n* 個のレコードはスキップされます。このオプションは機能的には **SKIPCOUNT** と同等です。 **RESTARTCOUNT** と **SKIPCOUNT** は相互に排他的です。

ROWCOUNT *n*

インポート (挿入または更新) するファイル内の物理レコードの数 *n* を指定します。ユーザーは、**SKIPCOUNT** または **RESTARTCOUNT** オプションで指示されたレコードから始めて、ファイルの *n* 行だけをインポートすることができます。 **SKIPCOUNT** または **RESTARTCOUNT** オプションの指定がないと、最初の *n* 行がインポートされます。 **SKIPCOUNT** *m* または **RESTARTCOUNT** *m* を指定すると、行 *m*+1 から *m*+*n* がインポートされます。コンパウンド挿入を使用した場合、ユーザー指定の **ROWCOUNT** *n* は、そのコンパウンド・カウント値に最も近い整数の倍数に切り上げられます。

SKIPCOUNT *n*

n + 1 の位置のレコードからインポート操作を開始するよう指定します。最初の *n* 個のレコードはスキップされます。このオプションは機能的には **RESTARTCOUNT** と同等です。 **SKIPCOUNT** と **RESTARTCOUNT** は相互に排他的です。

STARTING *sub-table-name*

階層専用キーワード。 *sub-table-name* から始まるデフォルト順を要求します。 PC/IXF ファイルの場合、デフォルト順は入力ファイルに保管されている順です。 PC/IXF ファイル・フォーマットの場合、デフォルト順は有効な唯一の順序です。

sub-table-list

型付き表で **INSERT** または **INSERT_UPDATE** オプションを指定した場合、データのインポート先副表を指定するために副表名のリストが使われます。

traversal-order-list
型付き表で **INSERT**、**INSERT_UPDATE**、または **REPLACE** オプションを指定した場合、インポートする階層内の副表のトラバーサル順序を指定するために副表名のリストを使います。

UNDER *sub-table-name*
1 つ以上の副表を作成する場合に親表を指定します。

WARNINGCOUNT *n*
n 個の警告後に、インポート操作を停止します。このパラメーターは、警告は予期されないが、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。インポート・ファイルまたはターゲット表が不適切に指定されると、インポート対象の各行ごとにインポート・ユーティリティーによって警告が生成され、このためにインポートが失敗する可能性があります。 *n* をゼロにした場合や、このオプションを指定しない場合、発行された警告の回数に関係なくインポート操作は続行します。

XML FROM *xml-path*
XML ファイルが含まれているパスを 1 つ以上指定します。

XMLPARSE
XML 文書の解析方法を指定します。このオプションが指定されていない場合、XML 文書の解析の動作は、CURRENT XMLPARSE OPTION 特殊レジスターの値によって決まります。

STRIP WHITESPACE
XML 文書の解析時に空白文字を除去することを指定します。

PRESERVE WHITESPACE
XML 文書の解析時に空白文字を除去しないことを指定します。

XMLVALIDATE
該当する場合に、XML 文書がスキーマに準拠しているかどうかの妥当性検査を実行することを指定します。

USING XDS
メイン・データ・ファイル内の XML Data Specifier (XDS) で識別される XML スキーマに照らし合わせて、XML 文書が妥当性検査されます。デフォルトでは、**USING XDS** 節によって **XMLVALIDATE** オプションが呼び出された場合、妥当性検査実行のために使用されるスキーマは、その XDS の SCH 属性によって決まります。 XDS の中で SCH 属性が指定されていない場合、**DEFAULT** 節によってデフォルト・スキーマが指定されているのでない限り、スキーマ妥当性検査は実行されません。

DEFAULT、**IGNORE**、および **MAP** 節を使用することにより、スキーマ決定の動作を変更することができます。これら 3 つの節はオプションであり、相互に適用されるのではなく XDS の指定に直接適用されます。例えば、**DEFAULT** 節で指定されているためにあるスキーマが選択された場合、それが **IGNORE** 節で指定されていたとしても無視されることはありません。同じように、**MAP** 節のペアの最初の部分で指定されているためにあるスキーマが選択された場合、それが別の **MAP** 節のペアの 2 番目の部分で指定されていたとしても再びマップされることはありません。

USING SCHEMA *schema-sqlid*
指定されている SQL ID の XML スキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。この場合、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

USING SCHEMALOCATION HINTS
ソース XML 文書の中で XML スキーマ・ロケーション・ヒントによって指定されているスキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。その XML 文書の中に schemaLocation 属性が指定されていない場合、妥当性検査は実行されません。 **USING SCHEMALOCATION HINTS** 節が指定されているなら、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

以下に示す **XMLVALIDATE** オプションの例を参照してください。

使用上の注意

インポート操作を開始する前に、すべての表操作が完了し、すべてのロックがペンディング解除になっていることを確認してください。これは、WITH HOLD でオープンされた、すべてのカーソルをクローズした後で COMMIT または ROLLBACK を発行することによって行われます。

インポート・ユーティリティーは、 SQL INSERT ステートメントを使用してターゲット表に行を追加します。このユーティリティーは、入力ファイル中の各行のデータにつき 1 つずつ INSERT ステートメントを発行します。 INSERT ステートメントが失敗した場合、以下の 2 通りの結果のいずれかになります。

- 後続の INSERT ステートメントが成功すると予測される場合には、警告メッセージがメッセージ・ファイルに書き込まれ、処理が継続されます。
- 後続の INSERT ステートメントが失敗すると予測され、データベースが損傷する可能性がある場合には、エラー・メッセージがメッセージ・ファイルに書き込まれ、処理が停止されます。

ユーティリティーは、**REPLACE** または **REPLACE_CREATE** 操作中に、古い行が削除された後、自動 COMMIT を実行します。したがって、表オブジェクトが切り捨てられた後、システムに障害が起こったり、アプリケーションがデータベース・マネージャーに割り込んだりすると、元のデータがすべて失われてしまいます。これらのオプションを使用する前に、元のデータがもはや必要ないことを確認してください。

CREATE、**REPLACE**、または **REPLACE_CREATE** 操作時にログが満杯になると、このユーティリティーは挿入されたレコードに対して自動 COMMIT を実行します。自動 COMMIT の後に、システムに障害が起こるか、またはアプリケーションがデータベース・マネージャーに割り込むと、部分的にデータの挿入された表はデータベース内に残ります。 **REPLACE** または **REPLACE_CREATE** オプションを使用してインポート操作全体をやり直すか、または正常にインポートされる行の数に設定した **RESTARTCOUNT** パラメーターを指定して **INSERT** を使用してください。

デフォルトでは、自動 COMMIT は **INSERT** または **INSERT_UPDATE** オプションでは実行されません。しかし、**COMMITCOUNT** パラメーターがゼロでない場合は実行されます。自動の COMMIT が実行されない場合にログが満杯になると、ROLLBACK が実行されます。

以下のいずれかの条件が真であると、オフライン・インポートでは自動の COMMIT は実行されません。

- ターゲットは表ではなくビューである。
- コンパウンド挿入を使用している。
- バッファー挿入を使用している。

デフォルトでは、オンライン・インポートは自動 COMMIT を実行して、アクティブ・ログ・スペースとロック・リストを両方とも解放します。自動 COMMIT が実行されないのは、ゼロの **COMMITCOUNT** 値を指定した場合のみです。

インポート・ユーティリティーが COMMIT を実行するたびに、 2 つのメッセージがメッセージ・ファイルに書き込まれます。一方は、コミットされるレコードの数を示し、もう一方は、 COMMIT の成功後に書き込まれます。障害の後にインポート操作を再開するときには、スキップするレコードの数 (最後の正常なコミットから判別される) を指定してください。

インポート・ユーティリティーでは、多少の非互換性問題がある入力データは受け入れられます (例えば、文字データは埋め込みまたは切り捨てを用いてインポートできます。数値データは異なる数値データ・タイプを用いてインポートできます)。しかし、大きな非互換性問題のあるデータは受け入れられません。

それ自体以外への依存があるオブジェクト表や、基本表に何らかの依存 (それ自体も含めて) があるオブジェクト・ビューを、 **REPLACE** または **REPLACE_CREATE** することはできません。そのような表またはビューを置換するには、以下のとおりに行ってください。

1. その表が親となっているすべての外部キーをドロップします。
2. インポート・ユーティリティーを実行します。
3. 表を変更して、外部キーを再作成します。

外部キーの再作成中にエラーが発生する場合、参照整合性を保守するためにデータを変更してください。

参照制約および外部キー定義は、PC/IXF ファイルから表を再作成する場合は保存されません。 (主キー定義は、データが前に `SELECT *` を使ってエクスポートされた場合、保存されます。)

リモート・データベースへのインポートでは、サーバーに、入力データ・ファイルのコピー、出力メッセージ・ファイル、およびデータベースのサイズ拡大を見込んだ十分なディスク・スペースが必要とされます。

インポート操作がリモート・データベースに対して実行され、出力メッセージ・ファイルが非常に長い (60 KB より長い) 場合、クライアント上でユーザーに戻されるメッセージ・ファイルがインポート操作中に欠落することがあります。メッセージ情報の最初の 30 KB と最後の 30 KB は、常に保持されます。

PC/IXF ファイルのリモート・データベースへのインポートは、 PC/IXF ファイルがディスケットにあるときよりも、ハード・ディスクにあるときの方がより速く行うことができます。

ASC、**DEL**、または **WSF** のファイル形式のデータをインポートするためには、それ以前にデータベース表または階層がすでに存在していなければなりません。ただし、表がまだ存在していない場合でも、IMPORT **CREATE** または IMPORT **REPLACE_CREATE** を使えば、PC/IXF ファイルからデータをインポートする際に表が作成されます。型付き表の場合、IMPORT **CREATE** はタイプ階層と表階層も作成することができます。

PC/IXF インポートは、データベース間でデータ (階層データも含む) を移動する場合に使用します。行区切り文字を含む文字データが区切り文字付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字を含むフィールドは長さが変わることがあります。ソースとターゲットのデータベースが両方とも同じクライアントからアクセス可能である場合、ファイルのコピーというステップは必要ありません。

ASC および DEL ファイルのデータは、インポートを実行するクライアント・アプリケーションのコード・ページであると仮定されます。異なるコード・ページのデータをインポートする場合は、異なるコード・ページを使用することのできる PC/IXF ファイルをお勧めします。 PC/IXF ファイルとインポート・ユーティリティーが同じコード・ページである場合は、通常のアプリケーションの場合のように処理が行われます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されている場合、インポート・ユーティリティーは、PC/IXF ファイルのデータのコード・ページと、インポートを実行中のアプリケーションのコード・ページが同じであると見なします。この処理は、それら 2 つのコード・ページ用の変換テーブルが存在する場合であっても行われます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されておらず、変換テーブルが存在する場合、PC/IXF ファイルのすべてのデータは、そのファイルのコード・ページからアプリケーションのコード・ページに変換されます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されておらず、変換テーブルが存在しない場合、インポート操作は失敗します。これが該当するのは、AIX オペレーティング・システムの DB2 クライアント上の PC/IXF ファイルの場合だけです。

8 KB ページ上の表オブジェクトの量が 1012 列の制限に近い場合、 PC/IXF データ・ファイルをインポートすると、 SQL ステートメントの最大サイズを超過するため、DB2 はエラーを戻します。この状態が発生する可能性があるのは、列が CHAR、 VARCHAR、または CLOB タイプの場合だけです。 **DEL** または **ASC** ファイルのインポートでは、この制限は当てはまりません。 PC/IXF ファイルを使って新しい表を作成している場合、別の方法として、db2look を使って表を作成した DDL ステートメントをダンプしてから、そのステートメント CLP から発行する、という方法があります。

DB2 Connect は、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA サーバーにデータをインポートするために使用できます。PC/IXF インポート (**INSERT** オプション) だけがサポートされています。**RESTARTCOUNT** パラメーターもサポートされていますが、**COMMITCOUNT** パラメーターはサポートされていません。

型付き表に対して **CREATE** オプションを使うと、PC/IXF ファイルの中で定義されているすべての副表が作成されます。副表定義は変更されません。型付き表に対して **CREATE** 以外のオプションを使うと、トラバーサル順序リストによって、トラバース順序を指定できます。その場合、トラバーサル順序リストはエクスポート操作で使用されたものと一致していなければなりません。 PC/IXF ファイル形式の場合は、ターゲット副表の名前を指定して、ファイルに格納されている全探索順序を使用するだけです。

インポート・ユーティリティーは、以前 PC/IXF ファイルにエクスポートされた表をリカバリーする場合に使用できます。その表は、エクスポート時の状態に戻ります。

システム表、宣言された一時表、またはサマリー表にデータをインポートすることはできません。

インポート・ユーティリティーを介してビューを作成することはできません。

マルチパート PC/IXF ファイルの個々のパートを Windows システムから AIX システムにコピーするインポート操作もサポートされています。IMPORT コマンドには、最初のファイルの名前だけを指定する必要があります。例えば、`IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1` のように記述します。 data.002 などのファイルも、data.ixf と同じディレクトリーに入れておく必要があります。

Windows オペレーティング・システムの場合は、以下のとおりです。

- 論理分割された PC/IXF ファイルのインポートはサポートされていません。
- 不正な形式の PC/IXF または WSF ファイルのインポートは、サポートされていません。

内部形式のセキュリティー・ラベルには、改行文字が含まれている可能性があります。 DEL ファイル形式を使用してファイルをインポートする場合、それらの改行文字が間違って区切りと解釈される可能性があります。この問題が起きた場合は、IMPORT コマンドで `delprioritychar` ファイル・タイプ修飾子を指定することによって、区切り文字に以前のデフォルト優先順位を使用してください。

フェデレーテッドに関する考慮事項

IMPORT コマンドで **INSERT**、**UPDATE**、または **INSERT_UPDATE** コマンド・パラメーターを使用するときには、関係するニックネームに対する CONTROL 特権があることを確認してください。インポート操作で使用するニックネームがすでに存在することを確認する必要があります。そのほかにも、IMPORT コマンド・パラメーターのセクションに記載されているようないくつかの制約事項に注意する必要があります。

一部のデータ・ソース (ODBC など) では、ニックネームへのインポートがサポートされていません。

インポート・ユーティリティーのファイル・タイプ修飾子

表 20. インポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式

修飾子	説明
compound=*x*	*x* は、1 から 100 までの (両端を含む) 数値です。非アトミックのコンパウンド SQL を使用してデータを挿入し、毎回 *x* 個のステートメントを試行します。 この修飾子を指定した場合に、トランザクション・ログに十分な大きさがないと、インポート操作は失敗します。トランザクション・ログは、**COMMITCOUNT** で指定されている行数か、**COMMITCOUNT** が指定されていなければデータ・ファイルに含まれている行数に対応できるだけの大きさでなければなりません。したがって、トランザクション・ログのオーバーフローを回避するために、**COMMITCOUNT** オプションを指定することをお勧めします。 この修飾子は、**INSERT_UPDATE** モードや階層表とは互換性がありません。さらに、`usedefaults`、`identitymissing`、`identityignore`、`generatedmissing`、`generatedignore` の各修飾子とも互換性がありません。
generatedignore	この修飾子を指定すると、インポート・ユーティリティーは、データ・ファイルに入っている、すべての生成済み列のデータを無視するようになります。その結果、生成済み列のすべての値がユーティリティーによって生成されます。この修飾子を `generatedmissing` 修飾子と一緒に使用することはできません。
generatedmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに生成済み列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この修飾子を `generatedignore` 修飾子と一緒に使用することはできません。
identityignore	この修飾子を指定すると、インポート・ユーティリティーは、データ・ファイルに入っている、ID 列のデータを無視するようになります。その結果、すべての IDENTITY 値がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を `identitymissing` 修飾子と一緒に使用することはできません。
identitymissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ID 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。この修飾子を `identityignore` 修飾子と一緒に使用することはできません。
lobsinfile	*lob-path* では、LOB データが含まれているファイルのパスを指定します。 各パスには、データ・ファイルの LOB ロケーション指定子 (LLS) によって参照されている LOB が少なくとも 1 つ入っているファイルが 1 つ以上含まれています。LLS は、LOB ファイル・パスに格納されているファイルの LOB の位置を示したストリング表記です。LLS の形式は、*filename.ext.nnn.mmm/* になります (*filename.ext* は、LOB が含まれているファイルの名前、*nnn* は、そのファイルに入っている LOB のオフセット (バイト単位)、*mmm* は、その LOB の長さ (バイト単位) です)。例えば、データ・ファイルにストリング `db2exp.001.123.456/` が格納されている場合は、ファイル db2exp.001 のオフセット 123 に LOB が配置されていて、その長さは 456 バイトということになります。 lobsinfile 修飾子を使用するときには、LOB ファイルの配置場所を **LOBS FROM** 節で指定します。**LOBS FROM** 節を指定すると、LOBSINFILE の動作が暗黙的にアクティブになります。**IMPORT** ユーティリティーは、データをインポートするときに、LOB ファイルを検索するためのパスのリストを LOBS FROM 節から受け取ります。 NULL LOB を指定する場合は、サイズとして -1 を入力します。サイズとして 0 を指定すると、長さ 0 の LOB として処理されます。長さ -1 の NULL LOB の場合は、オフセットとファイル名が無視されます。例えば、NULL LOB の LLS は、db2exp.001.7.-1/ のようになります。
no_type_id	1 つの副表にインポートする場合にのみ有効です。通常の表からエクスポートしたデータについて、この修飾子を使用してインポート操作を起動し、データを 1 つの副表に変換する、というのが典型的な使用法です。
nodefaults	ターゲット表の列のソース列を明示的に指定しない場合に、その表列が NULL 可能でなければ、デフォルト値はロードされません。このオプションを指定しない状態で、ターゲット表のいずれかの列のソース列を明示的に指定しない場合は、以下のいずれか動作が発生します。 • 列のデフォルト値を指定できる場合は、そのデフォルト値がロードされます。 • 列が NULL 可能で、その列のデフォルト値を指定できない場合は、NULL がロードされます。 • 列が NULL 可能ではなく、デフォルト値も指定できない場合は、エラーが戻され、ユーティリティーが処理を停止します。
norowwarnings	リジェクトされた行についてのすべての警告を抑止します。

表 20. インポート・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
rowchangetimestampignore	この修飾子を指定すると、インポート・ユーティリティーは、データ・ファイルに入っている、ROW CHANGE TIMESTAMP 列のデータを無視するようになります。その結果、すべての ROW CHANGE TIMESTAMP がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を `rowchangetimestampmissing` 修飾子と一緒に使用することはできません。
rowchangetimestampmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ROW CHANGE TIMESTAMP 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。この修飾子を `rowchangetimestampignore` 修飾子と一緒に使用することはできません。
seclabelchar	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、ストリング・フォーマットのセキュリティー・ラベル値であることを指定します。IMPORT は、ロード時に各セキュリティー・ラベルを内部形式に変換します。ストリングが正しい形式になっていないと、行はロードされず、警告 (SQLSTATE 01H53) が戻されます。ストリングが、表を保護するセキュリティー・ポリシーの一部である有効なセキュリティー・ラベルに対応していなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3243W) が戻されます。 seclabelname 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、インポートは失敗し、エラー (SQLCODE SQL3525N) が戻されます。
seclabelname	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、名前で示されていることを指定します。IMPORT は、その名前に対応する適切なセキュリティー・ラベルがあれば、その名前をそのセキュリティー・ラベルに変換します。表を保護するセキュリティー・ポリシーに、その名前に対応するセキュリティー・ラベルが存在しなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3244W) が戻されます。 seclabelchar 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、インポートは失敗し、エラー (SQLCODE SQL3525N) が戻されます。 **注:** ファイル・タイプが ASC の場合、セキュリティー・ラベル名の後のスペースは、名前の一部と解釈されます。そのような動作を避けるには、`striptblanks` ファイル・タイプ修飾子を使用して、スペースを除去するようにします。
usedefaults	ターゲット表の列のソース列が指定されていても、1 つ以上の行インスタンスでその列にデータが入っていない場合は、デフォルト値がロードされます。欠落データの例を以下に示します。 • DEL ファイル: 列の値として、2 つの隣接した列区切り (″,,″) や、任意の数のスペースで分離した 2 つの列区切り (″, ,″) が指定されている場合。 • DEL/ASC/WSF ファイル: 十分な数の列がない行や、元の指定に対応した十分な長さがない行。 **注:** ASC ファイルの場合、NULL 列値は、明示的な欠落とは見なされず、NULL 列値の代わりにデフォルトが入ることもありません。数値、日付、時刻、タイム・スタンプの列では、全桁スペース文字で NULL 列値を表記します。また、どのタイプの列でも、NULL INDICATOR を使用すれば、その列が NULL であることを示せます。 このオプションを指定しない場合に、行インスタンスのソース列にデータが入っていないと、以下のいずれかの動作が発生します。 • DEL/ASC/WSF ファイル: 列が NULL 可能であれば、NULL がロードされます。列が NULL 可能でなければ、ユーティリティーによって行がリジェクトされます。

表 21. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL)

修飾子	説明
codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、入力データ・セットに含まれているデータのコード・ページとして解釈されます。インポート操作の実行時に、文字データは、このコード・ページからアプリケーション・コード・ページに変換されます。 以下の規則が適用されます。 • ピュア DBCS (GRAPHIC)、混合 DBCS、EUC では、区切りが x00 から x3F の範囲 (両端を含む) に限定されています。 • nullindchar では、標準の ASCII セットのコード・ポイント x20 から x7F の範囲 (両端を含む) に含まれているシンボルを指定する必要があります。この修飾子では、ASCII のシンボルとコード・ポイントを参照します。 **注:** 1. codepage 修飾子を lobsinfile 修飾子と一緒に使用することはできません。 2. コード・ページをアプリケーション・コード・ページからデータベース・コード・ページに変換するときに、データの拡張が発生すると、データが切り捨てられ、データが失われる可能性があります。
dateformat="*x*"	*x* は、ソース・ファイルの日付の形式です。[2] 有効な日付エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (1 から 12 の範囲の 2 桁の数。 M とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月エレメントとは 相互に排他的) 指定されていないそれぞれのエレメントには、デフォルト値の 1 が割り当てられます。日付形式の例を以下に示します。 "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
implieddecimal	暗黙の小数点の位置が列定義によって決まるようになり、値の末尾という想定がなくなります。例えば、値 12345 は DECIMAL(8,2) 列に 12345.00 としてではなく、123.45 としてロードされます。
timeformat="*x*"	*x* は、ソース・ファイルの時刻の形式です。[2] 有効な時刻エレメントは、以下のとおりです。 H - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数) HH - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 2 桁の数; H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) 指定されていないそれぞれのエレメントには、デフォルト値の 0 が割り当てられます。時刻形式の例を以下に示します。 "HH:MM:SS" "HH.MM TT" "SSSSS"

表 21. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明

timestampformat="x"	x は、ソース・ファイルのタイム・スタンプの形式です。[2] 有効なタイム・スタンプ・エレメントは、以下のとおりです。
	
	YYYY - 年 (0000 から 9999 の範囲の 4 桁の数)
	M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数)
	MM - 月 (01 から 12 の 2 桁の数。
	M および MMM とは相互に排他的)
	MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。
	M と MM とは相互に排他的)
	D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数)
	DD - 日 (1 から 31 の範囲の 2 桁の数。
	D とは相互に排他的)
	DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。
	他の日または月のエレメントとは相互に排他的)
	H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の
	範囲の 1 桁または 2 桁の数。)
	HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の
	範囲の 2 桁の数。
	H と相互に排他的)
	M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数)
	MM - 分 (0 から 59 の範囲の 2 桁の数。
	M (分) とは相互に排他的)
	S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数)
	SS - 秒 (0 から 59 の範囲の 2 桁の数。
	S と相互に排他的)
	SSSSS - 夜中の 12 時から数えた秒数
	(00000 から 86399 の範囲の 5 桁の数。
	他の時刻エレメントとは相互に排他的)
	UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。
	他のマイクロ秒エレメントとは相互に排他的)
	UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。
	000000 から 999990 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数。
	000000 から 999900 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。
	000000 から 999000 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数。
	000000 から 990000 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。
	000000 から 900000 の範囲にマップされる。
	他のマイクロ秒エレメントとは相互に排他的)
	TT - 午前/午後の指定子 (AM または PM)
	
	指定されていない YYYY、M、MM、D、DD、DDD のいずれかのエレメントには、デフォルト値の 1 が割り当てられます。指定されていない MMM エレメントには、デフォルト値の 'Jan' が割り当てられます。指定されていない他のすべてのエレメントには、デフォルト値の 0 が割り当てられます。タイム・スタンプ形式の例を以下に示します。
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM エレメントの有効な値は、
	'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov'、'dec' です。これらの値では、大/小文字は区別されません。
	
	ユーザー定義の日付と時刻の形式が含まれているデータを schedule という表にインポートする例を以下に示します。
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

表 21. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
usegraphiccodepage	usegraphiccodepage を指定すると、グラフィックまたは 2 バイト文字のラージ・オブジェクト (DBCLOB) データ・フィールドにインポートするデータは、グラフィック・コード・ページのデータであるという想定で、処理が行われます。残りのデータは、文字コード・ページのデータであるという想定になります。グラフィック・コード・ページは、文字コード・ページに関連付けられています。IMPORT は、codepage 修飾子が指定されていればその修飾子によって、codepage 修飾子が指定されていなければアプリケーションのコード・ページによって、文字コード・ページを判別します。 ドロップ済み表のリカバリーで生成される区切り付きデータ・ファイルとこの修飾子を併用するのは、リカバリーする表にグラフィック・データが入っている場合に限られます。 **制約事項** EXPORT ユーティリティーで作成される DEL ファイルでは、usegraphiccodepage 修飾子を指定しないでください。そのファイルには、1 つのコード・ページでエンコードされたデータだけが入っているからです。usegraphiccodepage 修飾子は、ファイルに含まれている 2 バイト文字ラージ・オブジェクト (DBCLOB) でも無視されます。
xmlchar	XML 文書が文字コード・ページでエンコードされていることを指定します。 指定の文字コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードが文字コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。文字コード・ページは、codepage ファイル・タイプ修飾子で指定されている値か、その修飾子が指定されていない場合はアプリケーション・コード・ページになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。
xmlgraphic	XML 文書が指定のグラフィック・コード・ページでエンコードされていることを指定します。 指定のグラフィック・コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードがグラフィック・コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。グラフィック・コード・ページは、codepage ファイル・タイプ修飾子で指定されている値のグラフィック・コンポーネントか、その修飾子が指定されていない場合はアプリケーション・コード・ページのグラフィック・コンポーネントになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。 **注:** IMPORT コマンドで xmlgraphic 修飾子を指定する場合は、インポート対象の XML 文書のエンコードが UTF-16 コード・ページになっている必要があります。そうでない場合は、XML 文書が構文解析エラーでリジェクトされるか、表にインポートされてもデータ破損が生じる可能性があります。

表 22. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASC (区切りなし ASCII) ファイル形式

修飾子	説明
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のインポートが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にインポートできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。
nullindchar=*x*	*x* は、単一文字です。NULL 値を示す文字を *x* に変更します。*x* のデフォルト値は、Y です。[3] この修飾子は、EBCDIC データ・ファイルでは大文字と小文字の区別があります。ただし、文字が英字の場合は例外です。例えば、NULL 標識文字が N に指定されている場合は、n も NULL 標識として認識されます。

表 22. インポート・ユーティリティーの有効なファイル・タイプ修飾子: ASC (区切りなし ASCII) ファイル形式 (続き)

修飾子	説明
reclen=*x*	*x* は、最大値 32,767 の整数です。各行では *x* 個の文字が読み取られ、行の終わりを示す改行文字は使用されません。

striptblanks	可変長フィールドにデータをロードするときに、末尾ブランク・スペースを切り捨てます。このオプションを指定しなければ、ブランク・スペースは維持されます。
	
	以下の例では striptblanks を指定しているので、インポート・ユーティリティーは、末尾ブランク・スペースを切り捨てます。
	
	`db2 import from myfile.asc of asc`
	`  modified by striptblanks`
	`  method l (1 10, 12 15) messages msgs.txt`
	`  insert into staff`
	
	このオプションを striptnulls と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの t の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。

striptnulls	可変長フィールドにデータをロードするときに、末尾 NULL (0x00 文字) を切り捨てます。このオプションを指定しなければ、NULL は維持されます。 このオプションを striptblanks と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの padwithzero の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。

表 23. インポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式

修飾子	説明

chardel*x*	*x* は、単一文字ストリング区切りです。デフォルト値は、二重引用符 (") です。文字ストリングを囲む二重引用符の代わりに指定の文字を使用します。[34] 文字ストリング区切りとして二重引用符を明示的に指定する場合は、以下のように指定します。
	
	`modified by chardel""`
	
	文字ストリング区切りとして単一引用符 (') を指定することもできます。以下の例では chardel'' を指定しているので、インポート・ユーティリティーは、検出する単一引用符 (') を文字ストリング区切りとして解釈します。
	
	`db2 "import from myfile.del of del`
	`  modified by chardel''`
	`  method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x* は、単一文字列区切りです。デフォルト値は、コンマ (,) です。列の終わりを示すコンマの代わりに指定の文字を使用します。[34]
	
	以下の例では coldel; を指定しているので、インポート・ユーティリティーは、検出するセミコロン (;) を列区切りとして解釈します。
	
	`db2 import from myfile.del of del`
	`  modified by coldel;`
	`  messages msgs.txt insert into staff`

表 23. インポート・ユーティリティーの有効なファイル・タイプ修飾子: DEL (区切り付き ASCII) ファイル形式 (続き)

修飾子	説明
decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) の代わりにブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進値の接頭部として正符号を使用します。

decpt*x*	*x* は、小数点文字としてピリオドの代わりに使用する単一文字です。デフォルト値は、ピリオド (.) です。小数点文字として、ピリオドの代わりに指定の文字を使用します。[34]
	
	以下の例では `decpt;` を指定しているので、インポート・ユーティリティーは、検出するセミコロン (;) を小数点として解釈します。
	
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`decpt; messages msgs.txt insert into staff"`

delprioritychar	区切り文字に関する現在のデフォルトの優先順位は、レコード区切り、文字区切り、列区切り、という順序になっています。この修飾子を指定すると、区切り文字の優先順位が、文字区切り、レコード区切り、列区切り、という順序に戻されるので、古い優先順位に依存する既存のアプリケーションが保護されます。構文:
	
	`db2 import ... modified by delprioritychar ...`
	
	例えば、以下の DEL データ・ファイルがあるとします。
	
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	
	`delprioritychar` 修飾子を指定しているので、このデータ・ファイルは、2 行だけになります。1 番目と 3 番目の <row delimiter> は、実際のレコード区切りとして解釈されますが、2 番目の <row delimiter> は、第 2 行の最初のデータ列の一部として解釈されるからです。この修飾子を指定しなければ、それぞれの <row delimiter> が区切り文字として解釈され、このデータ・ファイルは 3 行になります。

keepblanks	タイプ CHAR、VARCHAR、LONG VARCHAR、CLOB の各フィールドで前後のブランクを保持します。このオプションを指定しないと、文字区切りの内側にない前後のすべてのブランクが除去され、表のすべてのブランク・フィールドに NULL が挿入されます。
nochardel	インポート・ユーティリティーは、列区切りの間で検出するすべてのバイトを列のデータの一部と見なします。文字区切りも、列データの一部として解析されます。DB2 でエクスポートしたデータについては、このオプションを指定しないでください (ただし、エクスポート時に nochardel を指定していた場合は例外です)。このオプションは、文字区切りのないベンダー・データ・ファイルをサポートするために用意されています。正しくない使い方をすると、データが失われたり破損したりする可能性があります。 このオプションを `chardelx`、`delprioritychar`、`nodoubledel` のいずれかと一緒に指定することはできません。これらは、相互に排他的なオプションです。
nodoubledel	二重文字区切りの認識を抑止します。

表 24. インポート・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式

修飾子	説明
forcein	ユーティリティーは、コード・ページの不一致があってもデータを受け入れ、コード・ページの変換を抑止します。 固定長ターゲット・フィールドについては、データを収容するだけの大きさがあるかどうかのチェックが行われます。nochecklengths を指定すると、チェックなしで各行のインポートが試行されます。
indexixf	ユーティリティーは、既存の表に現在定義されているすべての索引をドロップし、PC/IXF ファイルの索引定義から新しい索引を作成します。このオプションを使用できるのは、表の内容を置き換える場合に限られます。ビューで使用することはできません。*insert-column* を指定した場合も、使用できません。
indexschema=*schema*	索引作成時に、索引名として指定の *schema* を使用します。 *schema* を指定しない場合に、キーワード indexschema が指定されていれば、接続ユーザー ID が使用されます。そのキーワードが指定されていなければ、IXF ファイルのスキーマが使用されます。
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のインポートが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にインポートできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。
forcecreate	インポート操作時に情報が欠落していたり、限定されていたりする場合でも、SQL3311N を戻してから、表を作成することを指定します。

表 25. *codepage* と *usegraphiccodepage* を使用する場合の *IMPORT* の動作

codepage=N	usegraphiccodepage	IMPORT の動作
なし	なし	ファイル内のすべてのデータは、アプリケーション・コード・ページのデータであるという想定になります。
あり	なし	ファイル内のすべてのデータは、コード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにインポートすると、グラフィック・データが破損します。
なし	あり	ファイル内の文字データは、アプリケーション・コード・ページのデータであるという想定になります。グラフィック・データは、アプリケーション・グラフィック・データのコード・ページのデータであるという想定になります。 アプリケーション・コード・ページが 1 バイトの場合は、すべてのデータがアプリケーション・コード・ページのデータであるという想定になります。 **警告:** アプリケーション・コード・ページが 1 バイトの場合に、グラフィック・データをデータベースにインポートすると、データベースにグラフィック列が含まれていても、グラフィック・データは破損します。

表 25. *codepage* と *usegraphiccodepage* を使用する場合の *IMPORT* の動作 (続き)

codepage=N	usegraphiccodepage	IMPORT の動作
あり	あり	文字データは、コード・ページ N のデータであるという想定になります。グラフィック・データは、N のグラフィック・コード・ページのデータであるという想定になります。 N が 1 バイトまたは 2 バイトのコード・ページの場合は、すべてのデータがコード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにインポートすると、グラフィック・データが破損します。

注:

1. **MODIFIED BY** オプションでサポートされていないファイル・タイプを使用しようとしても、インポート・ユーティリティーからは警告が生成されません。その場合は、インポート操作が失敗し、エラー・コードが戻されます。
2. 日付形式ストリングを二重引用符で囲むのは、必須です。フィールド区切り文字には、a から z、A から Z、0 から 9 を組み込めません。フィールド区切り文字として、DEL ファイル形式の文字区切りまたはフィールド区切りと同じ文字を使用することはできません。エレメントの開始位置と終了位置があいまいでない場合は、フィールド区切り文字はオプションになります。修飾子によっては、項目が可変長の場合に D、H、M、S などのエレメントを使用することがあり、そのような場合は、開始位置と終了位置があいまいになることがあります。

   タイム・スタンプ形式の場合は、月の記述子と分の記述子の間であいまいさが残らないように注意する必要があります。どちらも、M という文字を使用するからです。月のフィールドは、他の日付フィールドと隣接している必要があります。分のフィールドは、他の時刻フィールドと隣接している必要があります。あいまいなタイム・スタンプ形式の例を以下に示します。

   ```
   "M" (月または分のどちらにもとれる)
   "M:M" (月と分の区別がつかない)
   "M:YYYY:M" (両方とも月と解釈される)
   "S:M:YYYY" (時刻値と日付値の両方に隣接している)
   ```

   あいまいな場合は、ユーティリティーによってエラー・メッセージが生成され、操作は失敗します。

   あいまいでないタイム・スタンプ形式の例を以下に示します。

   ```
   "M:YYYY" (M (月))
   "S:M" (M (分))
   "M:YYYY:S:M" (M (月)....M (分))
   "M:H:YYYY:M:D" (M (分)....M (月))
   ```

   二重引用符や円記号など、いくつかの文字の前ではエスケープ文字を使用する必要があります (¥ など)。

3. ファイル・タイプ修飾子 chardel、coldel、decpt に指定する文字値は、ソース・データのコード・ページに指定されている文字値でなければなりません。

   文字コード・ポイント (文字シンボルではない) を指定する場合は、xJJ または 0xJJ という構文を使用できます (JJ は、コード・ポイントの 16 進表記です)。例えば、列区切りとして # 文字を指定する場合は、以下のいずれかを使用します。

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

4. 区切り文字のオーバーライドとして使用できる文字に適用される制約事項については、『*データ移動のための区切り文字の制約事項*』を参照してください。
5. ニックネームにインポートする場合、以下のファイル・タイプ修飾子は使用できません。
   - indexixf
   - indexschema
   - dldelfiletype
   - nodefaults
   - usedefaults
   - no_type_idfiletype
   - generatedignore
   - generatedmissing
   - identityignore
   - identitymissing
   - lobsinfile
6. XML 列では、**WSF** ファイル形式はサポートされていません。
7. XML 列では、**CREATE** モードはサポートされていません。
8. すべての XML データをメイン・データ・ファイルとは別の XML ファイルに配置する必要があります。メイン・データ・ファイルの各 XML 列に XML Data Specifier (XDS) (または NULL 値) が入っている必要があります。
9. ファイル・タイプ修飾子 XMLCHAR または XMLGRAPHIC が指定されている場合を除き、XML 文書は、Unicode 形式であるか、エンコード属性の宣言タグが含まれているという前提で処理が行われます。
10. 整形式でない文書が含まれている行はリジェクトされます。
11. **XMLVALIDATE** オプションを指定した場合、対応するスキーマによる妥当性検査に成功した文書には、挿入時にスキーマ情報の注釈が付けられます。対応するスキーマによる妥当性検査に失敗した文書が含まれている行はリジェクトされます。妥当性検査を正常に実行するには、インポートを起動するユーザーの特権に、少なくとも以下のいずれかの権限が含まれている必要があります。
    - SYSADM または DBADM 権限
    - 妥当性検査に使用する XML スキーマに対する USAGE 特権
12. 暗黙的な非表示設定になっている Row Change Timestamp 列が含まれている表にインポートする場合は、その列の暗黙的な非表示のプロパティーが適用されません。したがって、インポートするデータに列のデータが含まれていない場

合に、明示的な列リストも存在しなければ、インポート・コマンドで rowchangetimestampmissing ファイル・タイプ修飾子を指定することが必要です。

db2Import - 表、階層、ニックネーム、ビューへのデータのインポート

サポートされているファイル形式を用いて、外部ファイルから表、階層、ニックネーム、またはビューにデータを挿入します。ロード・ユーティリティーのほうが、この関数より高速です。ただし、ロード・ユーティリティーは、階層レベルでのデータのロードや、ニックネームへのロードをサポートしていません。

許可

- INSERT オプションを使用して **IMPORT** コマンドを実行する場合、以下のいずれかが必要です。
  - sysadm
  - dbadm
  - 関係するそれぞれの表、ビュー、またはニックネームに対する CONTROL 特権
  - 関係するそれぞれの表またはビューに対する INSERT および SELECT 特権
- INSERT_UPDATE オプションを使用して既存の表に **IMPORT** するには、以下のいずれかが必要です。
  - sysadm
  - dbadm
  - 表、ビュー、またはニックネームに対する CONTROL 特権
  - 関係するそれぞれの表またはビューに対する INSERT、 SELECT、UPDATE、および DELETE 特権
- REPLACE または **REPLACE_CREATE** オプションを使用して既存の表に **IMPORT** するには、以下のいずれかが必要です。
  - sysadm
  - dbadm
  - 表またはビューに対する CONTROL 特権
  - 表またはビューに対する INSERT、SELECT、および DELETE 特権
- CREATE または **REPLACE_CREATE** オプションを使用して新規の表に **IMPORT** するには、以下のいずれかが必要です。
  - sysadm
  - dbadm
  - データベースに対する CREATETAB 権限、表スペースに対する USE 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 (表の暗黙または明示スキーマ名がない場合)
    - 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す場合)

- CREATE または **REPLACE_CREATE** オプションを使って、存在しない表または階層に **IMPORT** するには、以下のいずれかが必要です。
  - sysadm
  - dbadm
  - データベースに対する CREATETAB 権限と、次のいずれか
    - データベースに対する IMPLICIT_SCHEMA 権限 (表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 (表のスキーマが存在する場合)
    - 階層全体に対して **REPLACE_CREATE** オプションが使用されている場合は、階層内のすべての副表に対する CONTROL 特権
- REPLACE オプションを使用して既存の階層に **IMPORT** するには、以下のいずれかが必要です。
  - sysadm
  - dbadm
  - 階層内のすべての副表に対する CONTROL 特権

必要な接続

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。

API 組み込みファイル

```
db2ApiDf.h
```

API およびデータ構造構文

```
SQL_API_RC SQL_API_FN
  db2Import (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ImportIn *piImportInfoIn;
   struct db2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   struct sqllob *piLongActionString;
} db2ImportStruct;

typedef SQL_STRUCTURE db2ImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
```

```
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2ImportIn;

typedef SQL_STRUCTURE db2ImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2ImportOut;

typedef SQL_STRUCTURE db2DMUXmlMapSchema
{
   struct db2Char                       iMapFromSchema;
   struct db2Char                       iMapToSchema;
} db2DMUXmlMapSchema;

typedef SQL_STRUCTURE db2DMUXmlValidateXds
{
   struct db2Char *piDefaultSchema;
   db2Uint32 iNumIgnoreSchemas;
   struct db2Char *piIgnoreSchemas;
   db2Uint32 iNumMapSchemas;
   struct db2DMUXmlMapSchema *piMapSchemas;
} db2DMUXmlValidateXds;

typedef SQL_STRUCTURE db2DMUXmlValidateSchema
{
   struct db2Char *piSchema;
} db2DMUXmlValidateSchema;

typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16 iUsing;
   struct db2DMUXmlValidateXds *piXdsArgs;
   struct db2DMUXmlValidateSchema *piSchemaArgs;
} db2DMUXmlValidate;

SQL_API_RC SQL_API_FN
  db2gImport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2gImportIn *piImportInfoIn;
   struct dbg2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct sqllob *piLongActionString;
} db2gImportStruct;
```

```
typedef SQL_STRUCTURE db2gImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2gImportIn;

typedef SQL_STRUCTURE db2gImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2gImportOut;
```

db2Import API パラメーター

versionNumber
入力。 2 番目のパラメーター **pParmStruct** として渡される構造のバージョンとリリースのレベルを指定します。

pParmStruct
入出力。db2ImportStruct 構造を指すポインター。

pSqlca 出力。sqlca 構造を指すポインター。

db2ImportStruct データ構造パラメーター

piDataFileName
入力。データがインポートされるパスおよび外部入力ファイル名を含むストリングを指定します。

piLobPathList
入力。 media_type フィールドを SQLU_LOCAL_MEDIA に設定された sqlu_media_list、および LOB ファイルが置かれているクライアント上のパスをリストするその sqlu_media_entry 構造を指すポインター。ニックネームにインポートするときには、このパラメーターは無効です。

piDataDescriptor
入力。外部ファイルからインポートするよう選択された列に関する情報を収めた sqldcol 構造を指すポインターです。 dcolmeth フィールドの値によって、このパラメーターに提供される残りの情報をインポート・ユーティリティーがどのように解釈するかが判別されます。このパラメーターの有効な値は、次のとおりです。

SQL_METH_N
名前。外部入力ファイルの列は、列名によって選択されます。

SQL_METH_P
位置。外部入力ファイルの列は、列の位置によって選択されます。

SQL_METH_L

ロケーション。外部入力ファイルの列は、列のロケーションによって選択されます。以下のいずれかの条件のために無効であるロケーションのペアを指定したインポート呼び出しは、データベース・マネージャーによってリジェクトされます。

- 開始または終了ロケーションが有効な範囲 (1 から符号付き 2 バイト整数の最大値) に入っていない場合。
- 終了ロケーションが開始ロケーションよりも小さい場合。
- ロケーションの対により定義された入力列幅に、ターゲット列のタイプおよび長さとの互換性がない場合。

開始ロケーションと終了ロケーションの対がゼロに等しい場合は、NULL 可能列が NULL で埋め込まれることを示します。

SQL_METH_D

デフォルト。 **piDataDescriptor** が NULL の場合、または SQL_METH_D に設定されている場合には、外部入力ファイルの列のデフォルト選択が実行されます。この場合、列数および列指定配列は、どちらも無視されます。 DEL、IXF、または WSF ファイルの場合、外部入力ファイルにある最初の n 個の列のデータは、そのままの順序で取り出されます。 n は、データがインポートされるデータベース列の数です。

piActionString

非推奨。**piLongActionString** に換わりました。

piLongActionString

入力。 4 バイト長のフィールドが含まれる sqllob 構造を指すポインターと、それに続いて表に影響するアクションを指定する文字の配列。

文字配列の形式は、以下のようになります。

```
{INSERT | INSERT_UPDATE | REPLACE | CREATE | REPLACE_CREATE}
INTO {tname[(tcolumn-list)] |
[{ALL TABLES | (tname[(tcolumn-list)][, tname[(tcolumn-list)]])}]
[IN] HIERARCHY {STARTING tname | (tname[, tname])}
[UNDER sub-table-name | AS ROOT TABLE]}
```

INSERT

既存の表データを変更することなく、インポートされたデータを表に追加します。

INSERT_UPDATE

主キー値が表にない場合はインポートした行を追加し、主キー値がある場合はそれらの行を更新に使用します。このオプションは、ターゲット表に主キーがあり、指定された (または暗黙指定された) インポートされるターゲット列のリストに、主キーのすべての列が組み込まれているときのみ有効です。このオプションは、ビューには適用されません。

REPLACE

表オブジェクトを切り捨てることによって表から既存データをすべて削除し、インポートされたデータを挿入します。表定義および索引定義は変更されません。 (indexixf が FileTypeMod に入ってい

て、 FileType が SQL_IXF である場合、索引は削除および置換されます。) 表がまだ定義されていない場合には、エラーが戻されます。

注: 既存のデータを削除した後にエラーが発生した場合、そのデータは失われてしまいます。
ニックネームにインポートするときには、このパラメーターは無効です。

CREATE

注: CREATE パラメーターは推奨されておらず、今後のリリースで除去される可能性があります。さらに詳しくは、『IMPORT コマンドの推奨されなくなったオプション CREATE および REPLACE_CREATE』を参照してください。

指定された表が定義されていない場合に、指定された PC/IXF ファイルにある情報を使用して表定義と行の内容が作成されます。 DB2 によりファイルが事前にエクスポートされている場合、索引も作成されます。指定された表が既に存在している場合、エラーが戻されます。このオプションは、PC/IXF ファイル形式にのみ有効です。
ニックネームにインポートするときには、このパラメーターは無効です。

REPLACE_CREATE

注: REPLACE_CREATE パラメーターは推奨されておらず、今後のリリースで除去される可能性があります。さらに詳しくは、『IMPORT コマンドの推奨されなくなったオプション CREATE および REPLACE_CREATE』を参照してください。

指定された表が定義されている場合に、 PC/IXF ファイルにある PC/IXF 行情報を使用して表の内容が置き換えられます。表がまだ定義されていない場合は、指定された PC/IXF ファイルにある情報を使用して表定義と行の内容が作成されます。 DB2 により PC/IXF ファイルが事前にエクスポートされている場合、索引も作成されます。このオプションは、PC/IXF ファイル形式にのみ有効です。

注: 既存のデータを削除した後にエラーが発生した場合、そのデータは失われてしまいます。
ニックネームにインポートするときには、このパラメーターは無効です。

tname データが挿入される表、型付き表、ビュー、またはオブジェクト・ビューの名前。以前のサーバーの場合を除き、REPLACE、INSERT_UPDATE、または INSERT には、修飾または非修飾の名前を使わなければならないようなときでも、別名を指定することができます。ビューの場合、読み取り専用ビューにすることはできません。

tcolumn-list
データが挿入される先の表またはビュー内にある列名のリスト。列名は、コンマで区切らなければなりません。列名が指定されない場

合、 CREATE TABLE または ALTER TABLE ステートメントで定義された列名が使用されます。型付き表に指定されている列のリストがない場合、それぞれの副表のすべての列にデータが挿入されます。

sub-table-name
CREATE オプションで 1 つ以上の副表を作成する際に、親表を指定します。

ALL TABLES
階層専用の暗黙キーワード。階層をインポートする際、デフォルトでは全探索順序リストで指定されているすべての表がインポートされます。

HIERARCHY
階層データをインポートするよう指定します。

STARTING
階層専用のキーワード。指定された副表の名前から開始して、デフォルト順序を使用するよう指定します。

UNDER
階層および CREATE 専用のキーワード。新しい階層、副階層、または副表を、指定された副表の下に作成するよう指定します。

AS ROOT TABLE
階層および CREATE 専用のキーワード。新しい階層、副階層、または副表を、独立型の階層として作成するよう指定します。

tname および tcolumn-list パラメーターは、 SQL INSERT ステートメントの tablename および colname リストに対応し、同一の制限の下にあります。

tcolumn-list 内の列と、外部列 (指定または暗黙指定された) とは、リストまたは構造における位置に従って対応付けられます (sqldcol 構造で指定された最初の列からのデータは、 tcolumn-list の最初のエレメントに対応する表またはビュー・フィールドに挿入されます)。

異なる数の列が指定された場合、実際に処理される列の数は 2 つのうちの小さい方です。このことにより、エラーが発生する (一部の NULL 不可の表フィールドに入れるべき値がないため) か、または情報メッセージが表示される (一部の外部ファイル列が無視されるため) 可能性があります。

ニックネームにインポートするときには、このパラメーターは無効です。

piFileType
入力。外部ファイル内のデータの形式を示すストリングを指定します。サポートされている外部ファイル形式は、以下のとおりです。

SQL_ASC
区切り文字なし ASCII。

SQL_DEL
区切り文字付き ASCII。これは dBase プログラム、BASIC プログラム、 IBM パーソナル・デシジョン・シリーズ・プログラム、お

よびその他の多数のデータベース・マネージャー/ファイル・マネージャーとの交換のための形式です。

SQL_IXF
IXF (統合交換フォーマットの PC バージョン)。同じ表または別のデータベース・マネージャー表へ再インポートできるよう、表からデータをエクスポートする場合に推奨される方式です。

SQL_WSF
ワークシート形式。 Lotus Symphony および 1-2-3 プログラムとの交換のための形式です。 ニックネームにインポートするときには、WSF ファイル・タイプはサポートされません。

piFileTypeMod
入力。 2 バイトの長さフィールドと、 1 つ以上の処理オプションを指定する文字の配列を含む構造を指すポインターです。このポインターが NULL であるか、このポインターが指す構造に 1 文字も入っていない場合、このアクションはデフォルトの指定が選択されたものとして解釈されます。

サポートされるすべてのファイル・タイプに、すべてのオプションを使用できるわけではありません。以下の『インポート・ユーティリティーのファイル・タイプ修飾子』の関連リンクを参照してください。

piMsgFileName
入力。このユーティリティーが戻すエラー、警告、および情報メッセージの宛先を含むストリングを指定します。オペレーティング・システム・ファイルまたは標準装置のパスおよび名前を指定できます。ファイルが既に存在する場合は、そのファイルが付加されます。存在していない場合は、新たに作成されます。

iCallerAction
入力。呼び出し側が要求するアクションを示します。有効な値は以下のとおりです。

SQLU_INITIAL
最初の呼び出し。この値は、API への最初の呼び出しの際には必ず使用してください。最初の呼び出しまたは後続の呼び出しのいずれかが戻され、要求されたインポート操作が完了する前に呼び出し側のアプリケーションが何らかのアクションを行うことが必要な場合、呼び出し側のアクションを以下のどちらかに設定しなければなりません。

SQLU_CONTINUE
処理の継続。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (例えば、テープの終わり条件への応答) を要求した後で、 API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが完了したら、ユーティリティーが最初の要求の処理を続行するよう指定するものです。

SQLU_TERMINATE
処理の終了。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (例えば、テープの終わり条件への応答) を要求した後で、 API への後続呼び出しを出す場合

だけです。この値は、ユーティリティーが要求したユーザー・アクションが実行されなかった場合、ユーティリティーが最初の要求の処理を中断するよう指定するものです。

piImportInfoIn
入力。 db2ImportIn 構造を指すポインター。

poImportInfoOut
出力。db2ImportOut 構造を指すポインター。

piNullIndicators
入力。 ASC ファイルの場合にのみ使用します。列データが NULL 可能であるかどうかを示す整数の配列です。この配列内のエレメント数は、入力ファイル内の列数と一致していなければなりません。この配列のエレメントとデータ・ファイルからインポートされる列との間には、 1 対 1 の順序付けられた対応関係があります。このため、エレメントの数は、**piDataDescriptor** パラメーターの dcolnum フィールドと同じでなければなりません。配列の各エレメントには、NULL 標識フィールドとして使用される、データ・ファイル内の列を識別する数値、または表の列が NULL 可能ではないことを示すゼロが含まれます。エレメントがゼロでない場合には、データ・ファイル内の識別された列に Y または N が入っていなければなりません。Y は表列のデータが NULL であることを示し、N は表列のデータが NULL ではないことを示します。

piXmlPathList
入力。 media_type フィールドを SQLU_LOCAL_MEDIA に設定された sqlu_media_list、および XML ファイルが置かれているクライアント上のパスをリストするその sqlu_media_entry 構造を指すポインター。

db2ImportIn データ構造パラメーター

iRowcount
入力。ロードされる物理レコードの数。これを使用すると、ファイル内の最初の **iRowcount** 個の行だけをロードすることができます。 **iRowcount** が 0 の場合、インポートではファイルのすべての行が処理されます。

iRestartcount
入力。レコードを挿入または更新する前にスキップするレコードの数。機能上は **iSkipcount** パラメーターと同等です。 **iRestartcount** と **iSkipcount** パラメーターは相互に排他的です。

iSkipcount
入力。レコードを挿入または更新する前にスキップするレコードの数。機能上は **iRestartcount** と同等です。

piCommitcount
入力。データベースにコミットする前にインポートするレコードの数。**piCommitcount** 個のレコードがインポートされるたびに、コミットが実行されます。 NULL 値は、デフォルトのコミット・カウント値 (オフライン・インポートの場合はゼロ、オンライン・インポートの場合は AUTOMATIC) を指定します。 Commitcount AUTOMATIC は、DB2IMPORT_COMMIT_AUTO の値を渡すことによって指定されます。

iWarningcount

入力。 **iWarningcount** 個の警告後に、インポート操作を停止します。このパラメーターは、警告は予期されないが、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。インポート・ファイルまたはターゲット表が不適切に指定されると、インポート対象の各行ごとにインポート・ユーティリティーによって警告が生成され、このためにインポートが失敗する可能性があります。

iWarningcount が 0 の場合、またはこのオプションが指定されていない場合、警告が何度出されてもインポート操作は続行します。

iNoTimeout

入力。インポート・ユーティリティーがロックの待機中にタイムアウトしないことを指定します。このオプションのほうが、 **locktimeout** データベース構成パラメーターより優先されます。他のアプリケーションは影響を受けません。有効な値は以下のとおりです。

DB2IMPORT_LOCKTIMEOUT
locktimeout 構成パラメーターの値を優先することを示します。

DB2IMPORT_NO_LOCKTIMEOUT
タイムアウトしないことを示します。

iAccessLevel

入力。アクセス・レベルを指定します。有効な値は以下のとおりです。

- **SQLU_ALLOW_NO_ACCESS**
  インポート・ユーティリティーが表を排他ロックすることを指定します。

- **SQLU_ALLOW_WRITE_ACCESS**
  インポート中も、表内のデータに対する読み取りまたは書き込みアクセスが可能であることを指定します。

最初の行の挿入時には、ターゲット表に意図的排他 (IX) ロックがかけられます。これで、表データへの同時の読み取りおよび書き出しアクセスが可能になります。オンライン・モードには、**REPLACE**、**CREATE**、または **REPLACE_CREATE** インポート・オプションとの互換性はありません。オンライン・モードとバッファー挿入との連携はサポートされません。インポート操作によって挿入後のデータが定期的にコミットされるので、表ロックへのロック・エスカレーションが削減され、アクティブなログ・スペースが使い果たされることはなくなります。このようなコミットは、**piCommitCount** パラメーターを使わなくても実行されます。各コミットごとに、インポートでは IX 表ロックが外されるので、コミットの完了後に再びロックの設定が試みられます。ニックネームにインポートするときにはこのパラメーターが必要で、有効な数値を使って **piCommitCount** パラメーターを指定する必要があります (AUTOMATIC は有効なオプションとは見なされません)。

piXmlParse

入力。 XML 文書に対して行われる必要のある解析のタイプ。 include ディレクトリーに入っている db2ApiDf ヘッダー・ファイル内の有効値は、次のとおりです。

DB2DMU_XMLPARSE_PRESERVE_WS
空白が保持されます。

DB2DMU_XMLPARSE_STRIP_WS
空白が取り除かれます。

piXmlValidate
入力。 db2DMUXmlValidate 構造を指すポインター。 XML 文書の XML スキーマ検証を行う必要があることを示します。

db2ImportOut データ構造パラメーター

oRowsRead
出力。インポート中にファイルから読み取られたレコードの数。

oRowsSkipped
出力。挿入または更新を開始する前にスキップしたレコードの数。

oRowsInserted
出力。ターゲット表に挿入された行の数。

oRowsUpdated
出力。インポートされたレコード (主キーの値が既に表内に存在するレコード) からの情報によって更新された、ターゲット表内の行数。

oRowsRejected
出力。インポートできなかったレコードの数。

oRowsCommitted
出力。正常にインポートされ、データベースにコミット済みのレコード数。

db2DMUXmlMapSchema データ構造パラメーター

iMapFromSchema
入力。マップ元の XML スキーマの SQL ID。

iMapToSchema
入力。マップ先の XML スキーマの SQL ID。

db2DMUXmlValidateXds データ構造パラメーター

piDefaultSchema
入力。 XDS に SCH 属性が入っていない場合に検証で使用する必要のある XML スキーマの SQL ID。

iNumIgnoreSchemas
入力。 XML スキーマが、XDS 内の SCH 属性によって参照される場合に、XML スキーマ検証中に無視される XML スキーマの数。

piIgnoreSchemas
入力。 XML スキーマが、XDS 内の SCH 属性によって参照される場合に、XML スキーマ検証中に無視される XML スキーマのリスト。

iNumMapSchemas
入力。 XML スキーマ検証中にマップされる XML スキーマの数。スキーマ・マップのペア中の最初のスキーマは、XDS 内の SCH 属性によって参照されるスキーマを表します。ペア中の 2 番目のスキーマは、スキーマ検証の実行で使用する必要のあるスキーマを表します。

piMapSchemas

入力。 XML スキーマ・ペアのリスト。各ペアは、1 つのスキーマから別のものへのマッピングを表します。ペア中の最初のスキーマは、XDS 内の SCH 属性によって参照されるスキーマを表します。ペア中の 2 番目のスキーマは、スキーマ検証の実行で使用する必要のあるスキーマを表します。

db2DMUXmlValidateSchema データ構造パラメーター

piSchema

入力。使用する XML スキーマの SQL ID。

db2DMUXmlValidate データ構造パラメーター

iUsing 入力。 XML スキーマ検証の実行で何を使用するかの指定。 include ディレクトリーに入っている db2ApiDf ヘッダー・ファイル内の有効値は、次のとおりです。

- **DB2DMU_XMLVAL_XDS**
  検証は、XDS に従って行う必要があります。これは、CLP ″**XMLVALIDATE USING XDS**″ 節に対応します。
- **DB2DMU_XMLVAL_SCHEMA**
  検証は、指定されたスキーマに従って行う必要があります。これは、CLP ″**XMLVALIDATE USING SCHEMA**″ 節に対応します。
- **DB2DMU_XMLVAL_SCHEMALOC_HINTS**
  検証は、XML 文書内に入っている schemaLocation ヒントに従って行う必要があります。これは、″**XMLVALIDATE USING SCHEMALOCATION HINTS**″ 節に対応します。

piXdsArgs

入力。 CLP ″**XMLVALIDATE USING XDS**″ 節に対応する引数を表す db2DMUXmlValidateXds 構造を指すポインター。

このパラメーターが適用されるのは、同じ構造内の **iUsing** パラメーターを DB2DMU_XMLVAL_XDS に設定する場合のみです。

piSchemaArgs

入力。 CLP ″**XMLVALIDATE USING SCHEMA**″ 節に対応する引数を表す db2DMUXmlValidateSchema 構造を指すポインター。

このパラメーターが適用されるのは、同じ構造内の **iUsing** パラメーターを DB2DMU_XMLVAL_SCHEMA に設定する場合のみです。

db2gImportStruct データ構造固有のパラメーター

iDataFileNameLen

入力。 **piDataFileName** パラメーターの長さ (バイト単位) を指定します。

iFileTypeLen

入力。 **piFileType** パラメーターの長さ (バイト単位) を指定します。

iMsgFileNameLen

入力。 **piMsgFileName** パラメーターの長さ (バイト単位) を指定します。

使用上の注意

インポート操作を開始する前に、以下の 2 つの方法のいずれかで、すべての表操作を完了し、すべてのロックを解放する必要があります。

- WITH HOLD 節を使って定義したすべてのオープン・カーソルをクローズし、COMMIT ステートメントを実行して、データの変更をコミットします。
- ROLLBACK ステートメントを実行して、データ変更をロールバックします。

インポート・ユーティリティーは、 SQL INSERT ステートメントを使用してターゲット表に行を追加します。

このユーティリティーは、入力ファイル中の各行のデータにつき 1 つずつ INSERT ステートメントを発行します。 INSERT ステートメントが失敗した場合、以下の 2 通りの結果のいずれかになります。

- 後続の INSERT ステートメントが成功すると予測される場合には、警告メッセージがメッセージ・ファイルに書き込まれ、処理が継続されます。
- 後続の INSERT ステートメントが失敗すると予測され、データベースが損傷する可能性がある場合には、エラー・メッセージがメッセージ・ファイルに書き込まれ、処理が停止されます。

ユーティリティーは、**REPLACE** または **REPLACE_CREATE** 操作中に、古い行が削除された後、自動 COMMIT を実行します。したがって、表オブジェクトが切り捨てられた後、システムに障害が起こったり、アプリケーションがデータベース・マネージャーに割り込んだりすると、元のデータのすべてが失われてしまいます。これらのオプションを使用する前に、元のデータがもはや必要ないことを確認してください。

CREATE、**REPLACE**、または **REPLACE_CREATE** 操作時にログが満杯になると、このユーティリティーは挿入されたレコードに対して自動 COMMIT を実行します。自動 COMMIT の後に、システムに障害が起こるか、またはアプリケーションがデータベース・マネージャーに割り込むと、部分的にデータの挿入された表はデータベース内に残ります。 **REPLACE** または **REPLACE_CREATE** オプションを使用してインポート操作全体をやり直すか、または正常にインポートされる行の数に設定した **iRestartcount** パラメーターを指定して **INSERT** を使用してください。

デフォルトでは、自動 COMMIT は **INSERT** または **INSERT_UPDATE** オプションでは実行されません。ただし、***piCommitcount** パラメーターがゼロでない場合は実行されます。ログが満杯であれば、ROLLBACK が実行されます。

インポート・ユーティリティーが COMMIT を実行するたびに、 2 つのメッセージがメッセージ・ファイルに書き込まれます。一方は、コミットされるレコードの数を示し、もう一方は、 COMMIT の成功後に書き込まれます。障害の後にインポート操作を再開するときには、スキップするレコードの数 (最後の正常なコミットから判別される) を指定してください。

インポート・ユーティリティーでは、多少の非互換性問題がある入力データは受け入れられます (例えば、文字データは埋め込みまたは切り捨てを用いてインポート

できます。数値データは異なる数値データ・タイプを用いてインポートできます)。しかし、大きな非互換性問題のあるデータは受け入れられません。

オブジェクト表に何らかの従属 (それ自体への従属は除く) がある場合は、そのオブジェクト表を **REPLACE** または **REPLACE_CREATE** することはできません。また、オブジェクト・ビューの基本表に何らかの従属 (それ自体への従属を含む) がある場合は、そのオブジェクト・ビューを **REPLACE** または **REPLACE_CREATE** することはできません。そのような表またはビューを置換するには、以下のとおりに行ってください。

1. その表が親となっているすべての外部キーをドロップします。
2. インポート・ユーティリティーを実行します。
3. 表を変更して、外部キーを再作成します。

外部キーの再作成中にエラーが発生する場合、参照整合性を保守するためにデータを変更してください。

PC/IXF ファイルから表を作成する場合、参照制約と外部キー定義は保存されません。 (主キー定義は、データが前に SELECT * を使ってエクスポートされた場合、保存されます。)

リモート・データベースへのインポートでは、サーバーに、入力データ・ファイルのコピー、出力メッセージ・ファイル、およびデータベースのサイズ拡大を見込んだ十分なディスク・スペースが必要とされます。

インポート操作がリモート・データベースに対して実行され、出力メッセージ・ファイルが非常に長くなった (60 KB を超過) 場合、クライアントのユーザーに戻されるメッセージ・ファイルがインポート操作中に欠落する可能性があります。メッセージ情報の最初の 30 KB と最後の 30 KB は、常に保持されます。

piDataDescriptor でデフォルト以外の値を使用したり、 **piLongActionString** で明示的な表列のリストを指定したりすると、リモート・データベースへのインポート速度は遅くなります。

データベース表または階層が存在していないと、ASC、DEL、または WSF ファイル形式のデータをインポートできません。ただし、表が存在していない場合でも、IMPORT **CREATE** または IMPORT **REPLACE_CREATE** は、PC/IXF ファイルからデータをインポートする際に表を作成します。型付き表の場合、IMPORT **CREATE** はタイプ階層と表階層も作成することができます。

PC/IXF インポートは、データベース間でデータ (階層データも含む) を移動する場合に使用します。行区切り文字を含む文字データが区切り文字付き ASCII (DEL) ファイルにエクスポートされ、テキスト転送プログラムによって処理される場合、行区切り文字を含むフィールドは長さが変わることがあります。

ASC および DEL ファイルのデータは、インポートを実行するクライアント・アプリケーションのコード・ページであると仮定されます。異なるコード・ページのデータをインポートする場合は、異なるコード・ページを使用することのできる PC/IXF ファイルをお勧めします。 PC/IXF ファイルとインポート・ユーティリティーが同じコード・ページである場合は、通常のアプリケーションの場合のように処理が行われます。それぞれのコード・ページが異なっており、**FORCEIN** オプショ

ンが指定されている場合、インポート・ユーティリティーは、PC/IXF ファイルのデータのコード・ページと、インポートを実行中のアプリケーションのコード・ページが同じであると見なします。この処理は、それら 2 つのコード・ページ用の変換テーブルが存在する場合であっても行われます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されておらず、変換テーブルが存在する場合、PC/IXF ファイルのすべてのデータは、そのファイルのコード・ページからアプリケーションのコード・ページに変換されます。それぞれのコード・ページが異なっており、**FORCEIN** オプションが指定されておらず、変換テーブルが存在しない場合、インポート操作は失敗します。これが適用されるのは、DB2 for AIX クライアント上の PC/IXF ファイルの場合だけです。

8KB ページ上の表オブジェクトの量が 1012 列の制限に近い場合、 PC/IXF データ・ファイルをインポートすると、 SQL ステートメントの最大サイズを超過するため、DB2 はエラーを戻します。この状態が発生する可能性があるのは、列が CHAR、 VARCHAR、または CLOB タイプの場合だけです。 DEL または ASC ファイルのインポートには、この制限事項は適用されません。

DB2 Connect を使用して、DB2 for OS/390、DB2 for VM and VSE、および DB2 for OS/400 などの DRDA サーバーからデータをインポートできます。 PC/IXF インポート (**INSERT** オプション) だけがサポートされています。 **restartcnt** パラメーターもサポートされていますが、 **commitcnt** パラメーターはサポートされていません。

型付き表で CREATE オプションを使用するときは、 PC/IXF ファイルで定義されているすべての副表を作成してください。副表の定義は変更できません。型付き表で CREATE 以外のオプションを使用するときは、全探索順序リストによって全探索順序を指定できます。このため、全探索順序リストはエクスポート操作時に使用したものと一致する必要があります。 PC/IXF ファイル形式の場合は、ターゲット副表の名前を指定して、ファイルに格納されている全探索順序を使用するだけです。インポート・ユーティリティーは、以前 PC/IXF ファイルにエクスポートされた表をリカバリーする場合に使用できます。その表は、エクスポート時の状態に戻ります。

システム表、宣言された一時表、またはサマリー表にデータをインポートすることはできません。

インポート・ユーティリティーを介してビューを作成することはできません。

Windows オペレーティング・システムの場合は、以下のとおりです。

- 論理分割された PC/IXF ファイルのインポートはサポートされていません。
- 不正な形式の PC/IXF または WSF ファイルのインポートは、サポートされていません。

フェデレーテッドに関する考慮事項

db2Import API、および **INSERT**、**UPDATE**、または **INSERT_UPDATE** パラメーターを使用するときには、関係するニックネームに対する CONTROL 特権があることを確認してください。インポート操作で使用したいニックネームが既に存在することを確認する必要があります。

インポート・セッション - CLP の例

例 1

次に示すのは myfile.ixf から STAFF 表に情報をインポートする方法の例です。

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff

SQL3150N  The H record in the PC/IXF file has product "DB2    01.00", date
"19970220", and time "140848".

SQL3153N  The T record in the PC/IXF file has name "myfile",
qualifier "        ", and source "            ".

SQL3109N  The utility is beginning to load data from file "myfile".

SQL3110N  The utility has completed processing.  "58" rows were read from the
input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "58".

SQL3222W  ...COMMIT of any database changes was successful.

SQL3149N  "58" rows were processed from the input file.  "58" rows were
successfully inserted into the table.  "0" rows were rejected.
```

例 2

次の例では、ID 列が含まれている表にインポートする方法を示します。

TABLE1 には、以下の 4 つの列があります。

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 は、C2 が GENERATED ALWAYS ID 列であることを除き、 TABLE1 と同じです。

DATAFILE1 内のデータ・レコード (DEL フォーマット):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2 内のデータ・レコード (DEL フォーマット):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

以下のコマンドでは、行 1 および 2 の ID 値が生成されます。これは、DATAFILE1 内にこれらの行の ID 値が存在しないためです。ただし、行 3 および 4 には、ユーザー提供の ID 値である 100 および 101 がそれぞれ割り当てられます。

```
db2 import from datafile1.del of del replace into table1
```

DATAFILE1 を TABLE1 にインポートして、すべての行の ID 値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

```
db2 import from datafile1.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
db2 import from datafile1.del of del modified by identityignore
   replace into table1
```

DATAFILE2 を TABLE1 にインポートして、それぞれの行ごとに ID 値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
db2 import from datafile2.del of del modified by identitymissing
   replace into table1
```

識別に関係するファイル・タイプ修飾子を使用せずに DATAFILE1 を TABLE2 にインポートすると、行 1 と 2 は挿入されますが、行 3 と 4 はリジェクトされます。これは、行 3 と 4 では独自に非 NULL 値が提供されており、 ID 列が GENERATED ALWAYS であるためです。

例 3

次の例では、NULL 標識が含まれている表にインポートする方法を示します。

TABLE1 には以下に示す 5 つの列があります。

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 には以下に示す 6 つのエレメントがあります。

- ELE1、位置 01 から 20
- ELE2、位置 21 から 22
- ELE5、位置 23 から 23
- ELE3、位置 24 から 27
- ELE4、位置 28 から 31
- ELE6、位置 32 から 32
- ELE6、位置 33 から 40

データ・レコード:

```
1...5....10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    wxyzN
Test data 4,5 and 6 WWN6789    Y
```

以下のコマンドは、ASCFILE1 から TABLE1 にレコードをインポートします。

```
db2 import from ascfile1 of asc
method L (1 20, 21 22, 24 27, 28 31)
null indicators (0, 0, 23, 32)
insert into table1 (col1, col5, col2, col3)
```

注:

1. COL4 は入力ファイルにはないので、そのデフォルト値 (NOT NULL WITH DEFAULT と定義されている) を使用して TABLE1 に挿入されます。
2. 位置 23 および 32 は、TABLE1 の COL2 と COL3 に NULL をロードするかどうかを行ごとに示すために使用されます。特定のレコードのうち列の NULL 標識位置が Y なら、その列は NULL になります。それが N の場合は、入力レコードのその列のデータ位置 (L(.....) で定義) にあるデータ値が、その行の列データのソースとして使用されます。この例では、行 1 のどの列も NULL ではなく、行 2 の COL2 は NULL であり、行 3 の COL3 は NULL です。
3. この例では、COL1 と COL5 の NULL INDICATORS は 0 (ゼロ) として指定されますが、それはそのデータを NULL 不可能であることを示しています。
4. 特定の列に対する NULL INDICATOR は入力レコードのどの位置でも可能ですが、その位置は必ず指定しなければならず、 Y または N のいずれかの値が提供される必要があります。

第 4 章 ロード・ユーティリティー

ロードの概要

ロード・ユーティリティーを使用すれば、新しく作成した表やすでにデータが入っている表に、大量のデータを効率よく移動することができます。このユーティリティーは、XML、ラージ・オブジェクト (LOB)、ユーザー定義タイプ (UDT) などのほとんどのデータ・タイプを処理できます。インポート・ユーティリティーは SQL INSERT を実行するのに対し、ロード・ユーティリティーはフォーマット設定したページをデータベースに直接書き込むため、インポート・ユーティリティーよりも処理が高速です。ロード・ユーティリティーはトリガーを起動したり、参照制約や表制約のチェックを実行したりしません (索引の一意性の妥当性チェックを除く)。

ロードのプロセスは、以下に示す 4 つの明確なフェーズ (段階) で構成されています (図 3 を参照)。

1. **ロード**
   ロード・フェーズ中に、データは表にロードされ、必要に応じて索引キーと表統計が集められます。保管点または整合点は、LOAD コマンドの **SAVECOUNT** パラメーターによって指定されるインターバルで確立されます。保管点の時点で正常にロードされた入力行の数を示すメッセージが生成されます。
2. **構築**
   構築フェーズ中には、ロード・フェーズ中に収集した索引キーに基づいて索引が生成されます。索引キーはロード・フェーズ中にソートされ、索引統計が集められます (STATISTICS USE PROFILE オプションを指定してプロファイルが索引統計情報の収集を示す場合)。この統計データは、RUNSTATS コマンドによって収集される統計とよく似たものです。
3. **削除**
   削除フェーズ中に、ユニーク・キーまたは主キー違反の発生した行が表から除かれます。削除されるこれらの行はロード例外表に保管されます (表が指定された場合)。
4. **索引コピー**
   索引コピー・フェーズでは、SYSTEM TEMPORARY 表スペースから元の表スペースに索引データがコピーされます。これは、READ ACCESS オプションを指定したロード操作時に、索引作成に SYSTEM TEMPORARY 表スペースが指定されていた場合にのみ発生します。

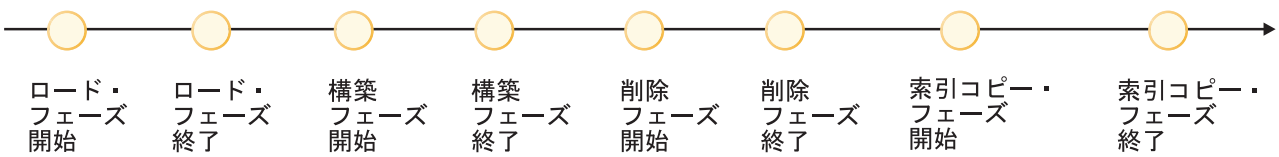


図 3. ロード・プロセスの 4 つのフェーズ (ロード、構築、削除、および索引コピー)。

注: ロード・ユーティリティーを呼び出した後で LIST UTILITIES コマンドを使って、ロード操作の進行状況をモニターすることができます。

データをロードするには、以下の情報が必要になります。

- 入力となるファイル、Named PIPE、または装置のパスと名前。
- ターゲット表の名前または別名。
- 入力ソースのフォーマット。このフォーマットは、DEL、ASC、PC/IXF、または CURSOR です。
- 入力データを表に追加するか、それとも表内の既存データを置換するか。
- アプリケーション・プログラミング・インターフェース (API) db2Load によってユーティリティーが起動される場合、メッセージ・ファイル名。

ロード・モード

- **INSERT**
  このモードのロードは、既存のデータに変更を加えずに表に対して入力データを付加します。
- **REPLACE**
  このモードのロードは、表から既存のデータを削除して、表に入力データを追加します。
- **RESTART**
  このモードでは、中断したロードが再開されます。ほとんどの場合、ロードは失敗したフェーズから再開されます。失敗したフェーズがロード・フェーズだった場合、ロードは最後に成功した整合点から再開されます。
- **TERMINATE**
  このモードでは、失敗したロード操作がロールバックされます。

指定できるオプションは次のとおりです。

- ロードするデータがクライアントに置かれていること (リモート接続されたクライアントからロード・ユーティリティーを起動する場合)。 CLIENT オプションを指定したとしても、XML および LOB データは常にサーバーから読み取られることに注意してください。
- データのロードに使用する方式: 列のロケーション、列名、または相対列ロケーション。
- ユーティリティーが整合点を確立する頻度。
- データの挿入先となる表の列の名前。
- ロード操作の進行中に、表の中に事前に存在するデータを照会できるかどうか。
- ロード操作で、他のユーティリティーまたはアプリケーションが表の使用を終了するのを待機するようにするか、あるいは続行する前にその他のアプリケーションからの接続を強制的に切断するか。
- 索引を作成する代替 SYSTEM TEMPORARY 表スペース。
- LOB が格納されている入力ファイルのパスと名前。

  注: ロード・ユーティリティーは COMPACT LOB オプションを考慮しません。
- メッセージ・ファイル名。ロード操作においては、メッセージ・ファイルを作成するよう指定できます。メッセージ・ファイルには、それらの操作に関連したエラー、警告、および通知の各メッセージが入れられます。これらのファイルの名前は、MESSAGES パラメーターで指定します。

注:

1. メッセージ・ファイルの内容を表示できるのは、操作が終了した後でのみです。ロード操作の実行中にロード・メッセージを表示するには、LOAD QUERY コマンドを使用できます。
2. メッセージ・ファイル内では、各メッセージはそれぞれ新たな行で始まり、DB2 メッセージ検索機能により提供される情報がそこに入れられます。

- ロードしている列値に暗黙の小数点があるかどうか。
- 表をロードした後、使用可能なフリー・スペースの大きさをユーティリティーが変更するかどうか。
- ロード・プロセス中に統計情報を収集するかどうか。このオプションは、ロード操作を REPLACE モードで実行する場合にのみサポートされます。統計は、表に定義されているプロファイルに従って収集されます。 LOAD コマンドの実行の前に、RUNSTATS コマンドを使用してこのプロファイルをあらかじめ作成しておく必要があります。プロファイルを作成していない場合に、プロファイルに従って統計を収集するようロード操作に指示すると、エラー・メッセージが戻されます。

  既存データが入っている表にロードする場合、統計情報は収集されません。追加がなされた表に関する現行の統計情報を収集するには、ロード・プロセスの完了後に RUNSTATS ユーティリティーを呼び出します。ユニーク索引のある表に関する統計情報を収集していて、削除フェーズ中に重複キーが削除された場合、それらのレコードの削除に合わせて統計情報が更新されることはありません。重複レコードの有効な数を入手する場合は、ロード操作中には統計情報を収集しないでください。その代わりに、ロード・プロセスの完了後に RUNSTATS ユーティリティーを呼び出すようにしてください。
- 変更部分のコピーを維持するかどうか。これにより、データベースのロールフォワード・リカバリーが可能になります。このオプションは、データベースのロールフォワード・リカバリーが無効になっている場合、つまりデータベース構成パラメーター *logarchmeth1* および *logarchmeth2* が OFF に設定されている場合にはサポートされません。コピーが作成されていないのにロールフォワード・リカバリーが有効になっていると、ロード操作の完了時に表スペースがバックアップ・ペンディング状態のままになります。

  データベースを完全にリカバリーできるようにするには、ログを記録する必要があります。ロード・ユーティリティーを利用した場合、データのロードに関連したロギングはほとんど不要です。ログを記録する代わりに、表のうちのロードした部分のコピーを作成することもできます。障害発生後にデータベースをリカバリーできるようになっているデータベース環境では、以下のいずれかを実行できます。

  – 表のうちのロードした部分のコピーを作成することを明示的に要求する。
  – 表の属する表スペースのバックアップを、ロード操作の完了直後に取る。

  データベース構成パラメーター *logindexbuild* を設定し、しかも COPY YES リカバリー可能性オプションと INCREMENTAL 索引作成オプションを指定してロード操作を起動した場合、そのロードでは索引の修正がすべてログ記録されます。これらのオプションを使用する利点は、このロードのログ・レコードを先へ順に

たどっていけば、索引のリカバリーも行われるという点にあります (これに対して、通常は、ロードで REBUILD 索引付けモードを使わないかぎり、索引はリカバリーされません)。

既存データが入っている表にロードする際、データベースがリカバリー可能でない場合は、ロード・ユーティリティーを呼び出す前に、そのデータベースあるいはロードする表の表スペースのバックアップ・コピーがあることを確認し、エラーがあってもリカバリーできるようにしておいてください。

リカバリー可能なデータベースで複数のロード操作を連続して実行する場合は、それぞれのロード操作をリカバリー不能に指定しておき、一連のロードの最後にバックアップを取ることにより、各ロード操作を COPY YES オプション付きで呼び出す場合よりも短い時間で一連の操作を実行できます。
NONRECOVERABLE オプションを使用すると、あるロード・トランザクションをリカバリー不能としてマークし、それ以降にロールフォワード操作を実行してもリカバリーできなくするよう指定できます。ロールフォワード・ユーティリティーはそのトランザクションをスキップし、データのロード先の表を ″invalid (無効)″ としてマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。ロールフォワードの完了後、このような表はドロップすることしかできません (図 4 を参照)。このオプションを指定すると、表スペースはロード操作後にバックアップ・ペンディング状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。

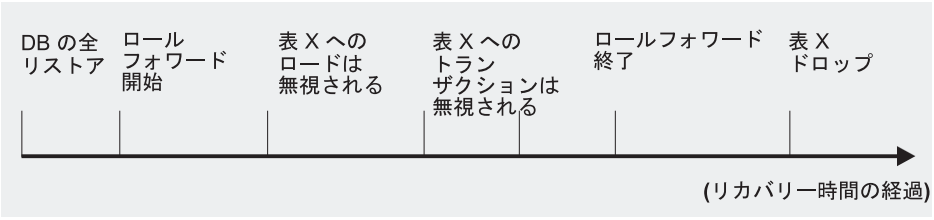


図 4. ロールフォワード操作時のリカバリー不能処理

- ロード操作中の一時ファイル作成時に使用する完全修飾パス。パス名は、LOAD コマンドの TEMPFILES PATH パラメーターで指定します。デフォルト値はデータベースのパスです。このパスはサーバー・マシン上にあり、 DB2 のインスタンスが排他的にアクセスします。そのため、このパラメーターに指定するパス名の修飾は、クライアントではなくサーバーのディレクトリー構造を反映していなければならず、 DB2 インスタンス所有者にはそのパスに対して読み取りと書き込みの両方の許可が必要です。

ロードを使用するために必要な特権と権限

ユーザーは、特権によってデータベース・リソースを作成したりアクセスしたりすることが可能になります。権限レベルは、特権をグループ化する手段となるものであり、さらに高水準のデータベース・マネージャーの保守およびユーティリティーのさまざまな操作を提供します。それらの働きにより、データベース・マネージャ

ーとそのデータベース・オブジェクトへのアクセスが制御されます。ユーザーは、適切な許可 (必要な特権または権限) が付与されているオブジェクトにしかアクセスできません。

データを表にロードするには、以下のいずれかが必要です。

- SYSADM 権限
- DBADM 権限
- データベースに対する LOAD 権限と次のいずれか
  - 表の INSERT 特権。INSERT モード、および (直前のロード INSERT 操作を終了するための) TERMINATE モード、もしくは (直前のロード INSERT 操作を再始動するための) RESTART モードでロード・ユーティリティーが呼び出される場合。
  - 表の INSERT および DELETE 特権。REPLACE モード、および (直前のロード REPLACE 操作を終了するための) TERMINATE モード、もしくは (直前のロード REPLACE 操作を再始動するための) RESTART モードでロード・ユーティリティーが呼び出される場合。
  - 例外表の INSERT 特権 (例外表をロード操作の一部として使用する場合)。
  - SYSCAT.TABLES の SELECT 特権は、LOAD がカタログ表を照会する場合に必要です。

すべてのロード・プロセス (および一般にすべての DB2 サーバー・プロセス) はインスタンス所有者が所有しており、これらのプロセスはすべてインスタンスの ID を使って必要なファイルにアクセスするため、インスタンス所有者にはデータ・ファイルを入力するために読み取りアクセスが必要です。だれがコマンドを呼び出すかに関係なく、これらの入力データ・ファイルはインスタンス所有者から読み取り可能になっていなければなりません。

REPLACE オプションを指定する場合、セッション許可 ID には表をドロップするための権限が必要です。

Windows、およびWindows.NET オペレーティング・システムで、DB2 が Windows サービスとして実行されている場合、ネットワーク・ドライブに常駐するファイルからデータをロードするには、これらのファイルに対する読み取りアクセスを持つユーザー・アカウントの下で実行するようにDB2 のサービスを構成する必要があります。

注:

- 保護列を持つ表にデータをインポートするには、表内のすべての保護列への書き込みアクセスを可能にする LBAC 信用証明情報がセッション許可 ID に必要です。
- 保護行を持つ表にデータをロードするには、表を保護するセキュリティー・ポリシーの一部である、書き込みアクセス用のセキュリティー・ラベルが、セッション許可 ID に対して付与されていなければなりません。

LOAD 権限

データベース・レベルの LOAD 権限、および表に対する INSERT 特権を持っているユーザーは、 LOAD コマンドを使用してデータを表にロードすることができます。

データベース・レベルの LOAD 権限、および表に対する INSERT 特権を持っているユーザーは、直前のロード操作でデータを挿入するロードを行った場合に、LOAD RESTART または LOAD TERMINATE を行うことができます。

データベース・レベルの LOAD 権限、および表に対する INSERT 特権と DELETE 特権を持っているユーザーは、 LOAD REPLACE コマンドを使用できます。

直前のロード操作でロード置換を行った場合、ユーザーは、DELETE 特権が付与されていないと、 LOAD RESTART または LOAD TERMINATE を行うことができません。

ロード操作の一部として例外表が使用される場合、ユーザーには、その例外表に対する INSERT 特権が必要です。

この権限を持っているユーザーは、 QUIESCE TABLESPACES FOR TABLE、RUNSTATS、および LIST TABLESPACES コマンドを実行することができます。

データのロード

ロード・ユーティリティーを使用すれば、新しく作成した表やすでにデータが入っている表に、大量のデータを効率よく移動することができます。

ロード・ユーティリティーを起動するには、その前にデータのロード先となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。ユーティリティーは COMMIT ステートメントを発行するため、ロード・ユーティリティーの呼び出し前に COMMIT または ROLLBACK ステートメントを発行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておかなければなりません。データは入力ファイルに示されている順序でロードされます。ただしマルチディメンション・クラスタリング (MDC) 表、パーティション表、または anyorder ファイル・タイプ修飾子を使用する場合を除きます。特定の順序を望む場合には、ロード操作を試行する前にデータをソートしてください。クラスタリングする必要がある場合は、ロードする前にクラスタリング索引に従ってデータをソートしておいてください。マルチディメンション・クラスタリング (MDC) 表にデータをロードする際には、ロード操作前のソートは必要ありません。データは MDC 表定義に従ってクラスター化されます。パーティション表にデータをロードする際には、ロード操作前のソートは必要ありません。データは表定義に従ってパーティション化されます。

ロード・ユーティリティーにはいくつかの制約事項が適用されます (このリストでそのすべてが挙げられているわけではありません)。

- ニックネームへのデータのロードはサポートされていません。
- 型付き表または構造化タイプ列をもつ表へのデータのロードはサポートされていません。
- 宣言された一時表へのデータのロードはサポートされていません。

- XML データはサーバー・サイドからのみ読み取ることができます。クライアントから XML ファイルを読み取る場合、インポート・ユーティリティーを使用します。
- バックアップ・ペンディング状態にある表スペースで表を作成またはドロップすることはできません。
- DB2 Connect、あるいは DB2 バージョン 2 より前のサーバー・レベルを経由してアクセスされるデータベースにデータをロードすることはできません。現行でのみ利用可能なオプションは、旧リリースのサーバーでは使用できません。
- LOAD REPLACE 操作中にエラーが発生すると、表のオリジナル・データは失われます。ロード操作を再開できるようにするには、入力データのコピーを作成しておいてください。
- 新しくロードした行ではトリガーが起動されません。トリガーに関連付けられたビジネス・ルールは、ロード・ユーティリティーによって適用されません。
- 暗号化されたデータのロードは、サポートされていません。

パーティション表にロードする際には、ロード・ユーティリティーにいくつかの制約事項が適用されます (このリストですべてを取り上げているわけではありません)。

- パーティション・エージェントが複数存在する場合、整合点はサポートされません。
- データ・パーティションのサブセットにデータをロードしている間、その他のデータ・パーティションを完全にオンラインにしておく機能はサポートされません。
- ロード操作または SET INTEGRITY ペンディング操作で使用される例外表は、パーティション化できません。
- ロード・ユーティリティーが挿入モードまたは再始動モードで実行されていて、デタッチされた従属データがロード・ターゲット表にある場合には、ユニーク索引を再作成することはできません。

ロード・ユーティリティーは、コマンド行プロセッサー (CLP)、コントロール・センターの「ロード」ウィザード、またはアプリケーション・プログラミング・インターフェース (API)、 db2Load から起動できます。

「ロード」ウィザードの使用

1. コントロール・センターから、「表」フォルダーが見つかるまでオブジェクト・ツリーを展開します。
2. 「表」フォルダーをクリックします。ウィンドウの右側部分 (コンテンツ・ペイン) に、既存の表がすべて表示されます。
3. コンテンツ・ペインで対象の表を右クリックし、ポップアップ・メニューから「ロード」を選択します。「ロード」ウィザードがオープンします。
4. データを正常にロードするには、ウィザードの各ページに必要な情報を指定します。

ロード・ウィザードについての詳細は、オンライン・ヘルプ機能によって提供されます。

CLP を使用した LOAD コマンドの発行

CLP によって発行する LOAD コマンドの例を以下に示します。

```
db2 load from stafftab.ixf of ixf messages staff.msgs
   insert into userid.staff copy yes use tsm data buffer 4000
```

この例の詳細は次のとおりです。

- 警告メッセージやエラー・メッセージはすべて staff.msgs ファイルに入れられます。
- 変更された部分のコピーは、Tivoli® Storage Manager (TSM) に保管されます。
- 4,000 ページ分のバッファー・スペースがロード操作中に使用されます。

CLP によって発行する LOAD コマンドの別の例を以下に示します。

```
db2 load from stafftab.ixf of ixf messages staff.msgs
   tempfiles path /u/myuser replace into staff
```

この例の詳細は次のとおりです。

- 表データは置換されます。
- TEMPFILES PATH パラメーターを使用して、一時ファイルを書き込むサーバー・パスとして /u/myuser を指定しています。

注: これらの例では、ロード用入力ファイルに相対パス名を使っています。相対パス名を使用できるのは、データベースと同じデータベース・パーティション上にあるクライアントから呼び出す場合だけです。できれば完全修飾パス名を使用してください。

ロード・ユーティリティーを呼び出した後で LIST UTILITIES コマンドを使って、ロード操作の進行状況をモニターすることができます。 INSERT モード、REPLACE モード、または RESTART モードで実行するロード操作の場合、進行状況の詳細モニタリングのサポートを利用することができます。現在のロード・フェーズに関する詳細情報を表示するには、SHOW DETAILS オプションを指定して LIST UTILITIES コマンドを出します。 TERMINATE モードで実行するロード操作では、詳細情報は得られません。 LIST UTILITIES コマンドは、ロード終了ユーティリティーが現在稼働中であることを示すだけです。

ロード操作は、ユニーク制約、パーティション表の範囲制約、生成される列、および LBAC セキュリティー規則を保守します。他のすべての制約では、表は、ロード操作の開始時点に「SET INTEGRITY ペンディング」状態に置かれます。ロード操作が完了したら、SET INTEGRITY ステートメントを使って、表の「SET INTEGRITY ペンディング」状態を終了しなければなりません。

XML データのロード

ロード・ユーティリティーを使用して、大量の XML データを表に効率的に移動できます。

データを XML 表列にロードする場合、XML FROM オプションを使用して入力 XML データ (1 つまたは複数) のパスを指定できます。例えば、データを XML ファイルから "/home/user/xmlpath/xmlfile1.xml" にロードするには、以下のコマンドを使用できます。

```
LOAD FROM data1.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

区切られた ASCII 入力ファイル ″data1.del″ には、ロードする XML データの場所を説明する XML データ指定子 (XDS) が含まれます。例えば、以下の XDS は、ファイル ″xmldata.ext″ 内のオフセット 123 バイトにある長さが 456 バイトの XML 文書について説明しています。

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' />
```

スキーマに対して挿入された文書を検証する

XMLVALIDATE オプションにより、 XML 文書がロードされるときに、それらを XML スキーマに対して妥当性検査できます。次の例では、区切られた ASCII 入力ファイル ″data2.del″ 内で XDS によって識別されるスキーマに対して、着信 XML 文書が検証されます。

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
     USING XDS INSERT INTO USER.T2
```

この場合、XDS には XML スキーマの完全修飾 SQL ID ″S1.SCHEMA_A″ のある SCH 属性が妥当性検査で使用するために含まれます。

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' SCH='S1.SCHEMA_A' />
```

構文解析オプションの指定

XMLPARSE オプションを使用して、ロードされた XML 文書の空白文字を保存するかストリップするかを指定できます。次の例では、SQL ID ″S2.SCHEMA_A″ のあるスキーマに対してすべてのロードされた XML 文書が検証されて、これらの文書は空白文字を保存しながら構文解析されます。

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
WHITESPACE XMLVALIDATE USING SCHEMA S2.SCHEMA_A INSERT INTO USER.T1
```

パーティション表におけるロードの考慮事項

以下の一般制約事項を除き、既存のすべてのロード・フィーチャーはターゲット表がパーティション化されている場合にサポートされます。

- パーティション・エージェントが複数存在する場合、整合点はサポートされません。
- データ・パーティションのサブセットにデータをロードしている間、その他のデータ・パーティションを完全にオンラインのままにしておく機能はサポートされません。
- ロード操作で使用される例外表は、パーティション化できません。
- ロード・ユーティリティーが挿入モードまたは再始動モードで実行されていて、デタッチされた従属データがロード・ターゲット表にある場合には、ユニーク索引を再作成することはできません。
- MDC 表のロードと同様、入力データ・レコードの厳密な順序は、パーティション表をロードする際には保持されません。順序はセルまたはデータ・パーティションの中のみで維持されます。
- 各データベース・パーティションで複数のフォーマッターを使用するロード操作では、入力レコードの大まかな順序のみを保持します。各データベース・パーティション上で単一のフォーマッターを実行すると、入力レコードがセルまたは表

パーティション・キーごとにグループ化されます。各データベース・パーティション上で単一のフォーマッターを実行するには、明示的に CPU_PARALLELISM に 1 を要求してください。

一般的なロードの動作

ロード・ユーティリティーは、データ・レコードを適切なデータ・パーティションに挿入します。ロードの前に入力データをパーティション化するための外部ユーティリティー (スプリッターなど) を使用する上での要件はありません。

ロード・ユーティリティーは、アタッチまたはデタッチされたデータ・パーティションにアクセスしません。データは可視のデータ・パーティションのみに挿入されます。可視のデータ・パーティションは、アタッチされたりデタッチされたりしません。また、ロード置換操作では、アタッチまたはデタッチされたデータ・パーティションを切り捨てることはありません。ロード・ユーティリティーではカタログ・システム表上のロックを獲得するため、ロード・ユーティリティーはコミットされていない ALTER TABLE トランザクションがあれば待機します。そのようなトランザクションは、カタログ表内の関連する行の排他ロックを獲得します。排他ロックを終了しなければロード操作は進行できません。これは、ロード操作の実行中は、コミットされていない ALTER TABLE ...ATTACH、DETACH、または ADD PARTITION トランザクションはありえないということを意味します。アタッチまたはデタッチされたデータ・パーティションに宛てられたすべての入力ソース・レコードはリジェクトされ、例外表が指定されている場合にはそこから取得できます。ターゲット表データ・パーティションの一部がアタッチまたはデタッチされた状態であったことを示すため、通知メッセージがメッセージ・ファイルに書き込まれます。ターゲット表に対応するカタログ表の関連する行のロックは、ロード・ユーティリティーの実行中に ALTER TABLE ...ATTACH、DETACH、または ADD PARTITION 操作を実行することによりユーザーがターゲット表のパーティションを変更することを防ぎます。

無効な行の処理

ロード・ユーティリティーで可視のデータ・パーティションのいずれにも属さないレコードが検出されると、そのレコードはリジェクトされ、ロード・ユーティリティーは処理を継続します。範囲制約違反のためにリジェクトされたレコードの数は明示的には表示されませんが、リジェクトされたレコードの全体数には含められます。範囲違反のためにレコードをリジェクトしても行の警告数は増加しません。範囲違反が検出されたものの、レコードごとのメッセージはログに記録されないということを示す単一のメッセージ (SQL0327N) がロード・ユーティリティーのメッセージ・ファイルに書き込まれます。例外表には、ターゲット表のすべての列に加えて、特定の行で発生した違反のタイプを記述する列が含まれます。無効データを含む行 (パーティション化できないデータを含む) は、ダンプ・ファイルに書き込まれます。

例外表への挿入は非効率であるため、どの制約違反を例外表に挿入するかを制御できます。例えば、ロード・ユーティリティーのデフォルトの動作は、範囲制約違反またはユニーク制約違反のためにリジェクトされた (その違反がなければ有効だった) 行を例外表に挿入することです。この動作は、FOR EXCEPTION 節を使用し、NORANGEEXC (範囲制約違反の場合) または NOUNIQUEEXC (ユニーク制約違反の場合) を指定することによってオフにすることができます。それらの制約違反を

例外表に挿入しないことを指定する場合、または例外表を指定しない場合、範囲制約またはユニーク制約に違反する行に関する情報は失われます。

履歴ファイル
ターゲット表がパーティション化されている場合、対応する履歴ファイルの項目は、ターゲット表により範囲を設定された表スペースのリストを含みません。操作対象のオブジェクト ID (「T」ではなく「R」) は、ロード操作がパーティション表に対して実行されたことを示します。

ロード操作の終了
ロード置換を終了すると、すべてのデータ・パーティションが完全に切り捨てられ、ロード挿入を終了すると、すべてのデータ・パーティションがロード前の長さに切り捨てられます。ロード・コピー・フェーズで失敗した ALLOW READ ACCESS ロード操作の終了中に索引は無効になります。索引にタッチした ALLOW NO ACCESS ロード操作を終了する時にも索引は無効になります (それが無効になるのは、索引付けモードが再作成されているか増分保守の間にキーが挿入されたために索引が不整合状態になっているためです)。データを複数のターゲットにロードしても、ロード・フェーズ中に取られた整合点からロード操作を再始動できない点を除き、ロード・リカバリー操作に何の影響もありません。この場合、ターゲット表がパーティション化されている場合には、SAVECOUNT ロード・オプションは無視されます。この動作は、MDC ターゲット表へのデータのロードと一貫しています。

生成列
生成される列がパーティション・キー、ディメンション・キー、または分散キーのいずれかにある場合、 `generatedoverride` ファイル・タイプ修飾子は無視され、ロード・ユーティリティーは `generatedignore` ファイル・タイプ修飾子が指定された場合のように値を生成します。ここで不正な生成列値をロードすると、レコードが不適切な物理ロケーション (例えば不適切なデータ・パーティション、MDC ブロック、またはデータベース・パーティション) に配置されてしまう可能性があります。例えば、あるレコードがいったん間違ったデータ・パーティションに置かれると、整合性の設定ではそのレコードを別の物理ロケーションに移動しなければなりませんが、それはオンラインでの整合性の設定操作中には行えません。

データの可用性
現行の ALLOW READ ACCESS ロード・アルゴリズムは、パーティション表に拡張されています。ALLOW READ ACCESS ロード操作の場合は、複数のリーダーが同時に表全体 (ロードするデータ・パーティションとロードしないものの両方を含む) にアクセスすることができます。

データ・パーティションの状態
ロードに成功した後、特定の条件下では、可視のデータ・パーティションの表の状態が「SET INTEGRITY ペンディング」または「読み取りアクセスのみ」のいずれかまたは両方に変更される場合があります。ロード操作で保守できない制約が表にある場合に、データ・パーティションはこれらの状態になる可能性があります。そのような制約には、チェック制約とデタッチされたマテリアライズ照会表が含まれる場合があります。ロード操作に失敗すると、すべての可視のデータ・パーティションの表の状態が「ロード・ペンディング」になります。

エラー分離
データ・パーティション・レベルでのエラー分離はサポートされていません。エラーを分離するとは、エラーにならなかったデータ・パーティションでロードを継続し、エラーになったデータ・パーティションで停止するということを意味します。エラーは異なるデータベース・パーティションの間で分離できますが、ロード・ユーティリティーは可視のデータ・パーティションのサブセット上でトランザクションをコミットしたり、残りの可視のデータ・パーティションをロールバックしたりすることはできません。

その他の考慮事項

- いずれかの索引が無効とマークされている場合には増分索引付けはサポートされません。索引の再作成が必要な場合、またはデタッチされた従属物が SET INTEGRITY ステートメントでの妥当性検査を必要としている場合には、索引は無効であると見なされます。
- 範囲別パーティション化、ハッシュによる分散、またはディメンションによる編成のいずれかのアルゴリズムの組み合わせを使用してパーティション化された表へのロードもサポートされています。
- ロードの影響を受けるオブジェクトと表スペース ID のリストが含まれるログ・レコードの場合、これらのログ・レコード (LOAD START および COMMIT (PENDING LIST)) のサイズは非常に大きくなる場合があり、そうなると、他のアプリケーションで使用できるアクティブ・ログ・スペースの量が減少してしまいます。
- 表がパーティション化されていて、かつ分散されている場合、パーティション・データベースのロードがすべてのデータベース・パーティションには影響を与えない場合があります。出力データベース・パーティション上のオブジェクトのみが変更されます。
- ロード操作中、パーティション表のメモリー使用量は表の数とともに増加します。増加の合計は直線的にならない点に注意してください。データ・パーティションの数に比例するのはメモリー所要量全体のうちのほんの僅かだからです。

LBAC で保護されたデータのロードに関する考慮事項

保護行が含まれている表へのロード操作を正常に実行するには、LBAC (ラベル・ベースのアクセス制御) 信用証明情報が必要です。さらに、ターゲット表に関連付けられているセキュリティー・ポリシーで有効なセキュリティー・ラベル、または有効なセキュリティー・ラベルに変換できるセキュリティー・ラベルを用意することも必要です。

有効な LBAC 信用証明情報がなければ、ロードは失敗し、エラー (SQLSTATE 42512) が返されます。入力データにセキュリティー・ラベルが含まれていない場合や、セキュリティー・ラベルが内部のバイナリー・フォーマットでない場合は、いくつかのファイル・タイプ修飾子を使用して、ロード操作を進めることができます。

保護行が含まれている表にデータをロードする場合は、ターゲット表にデータ・タイプ DB2SECURITYLABEL の列が 1 つ含まれています。入力データ行にその列の値が含まれていなければ、その行はリジェクトされます。ただし、ロード・コマンドに `usedefaults` ファイル・タイプ修飾子が指定されている場合は例外です。その

場合に、その表を保護しているセキュリティー・ポリシーで書き込みアクセスが認められているセキュリティー・ラベルがあれば、そのセキュリティー・ラベルが使用されます。書き込みアクセスが認められているセキュリティー・ラベルがなければ、その行はリジェクトされ、処理が次の行に移ります。

保護行が含まれている表にデータをロードする場合に、入力データにデータ・タイプ DB2SECURITYLABEL の列の値が含まれていれば、その表にデータを挿入する場合と同じ規則が当てはまります。ロード対象の行を保護しているセキュリティー・ラベル (データ・ファイル内のその行にあるセキュリティー・ラベル) が書き込み可能であれば、その行の保護のためにそのセキュリティー・ラベルが使用されます。(つまり、そのセキュリティー・ラベルがデータ・タイプ DB2SECURITYLABEL の列に書き込まれます。)そのセキュリティー・ラベルで保護されている行への書き込みができない場合の動作は、ソース表を保護しているセキュリティー・ポリシーの作成方法によって異なります。

- ポリシーを作成した CREATE SECURITY POLICY ステートメントにオプション RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL が含まれていた場合には、行はリジェクトされます。
- CREATE SECURITY POLICY ステートメントにそのオプションが組み込まれていなかった場合や、 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL オプションが代わりに組み込まれていた場合は、データ・ファイルに含まれているその行のためのセキュリティー・ラベルは無視されます。書き込みアクセスが認められているセキュリティー・ラベルがあれば、その行の保護のためにそのセキュリティー・ラベルが使用されます。この場合、エラーや警告は出されません。書き込みアクセスが認められているセキュリティー・ラベルがなければ、その行はリジェクトされ、処理が次の行に移ります。

区切り文字に関する考慮事項

データ・タイプ DB2SECURITYLABEL の列にデータをロードする場合、デフォルトでは、データ・ファイル内の値は、そのセキュリティー・ラベルの内部表記を構成する実際のバイトと見なされます。ただし、生データには改行文字が含まれている場合があり、その場合、LOAD コマンドは、その改行文字を行の区切り文字と誤解してしまう可能性があります。この問題がある場合は、行区切り文字よりもストリング区切り文字を優先するために、`delprioritychar` ファイル・タイプ修飾子を使用します。`delprioritychar` を使用すると、ストリング区切り文字の中にあるレコード区切り文字または列区切り文字は、区切り文字と見なされなくなります。`delprioritychar` ファイル・タイプ修飾子は、改行文字が含まれている値が存在しない場合に使用しても差し支えありません。ただし、ロードの速度は少し低下します。

ロード対象のデータが ASC 形式の場合は、ロードされたセキュリティー・ラベルやセキュリティー・ラベル名に末尾空白が組み込まれる事態を回避するために、追加の手順を実行する必要があります。ASCII フォーマットでは、列位置が区切りとして使用されるので、可変長フィールドにロードするときにこの問題が発生する可能性があります。末尾ブランク・スペースを切り捨てるには、`striptblanks` ファイル・タイプ修飾子を使用します。

非標準のセキュリティー・ラベル値

セキュリティー・ラベルの値が、セキュリティー・ラベルの各コンポーネントの値を含んだストリングになっている場合、例えば、*S:(ALPHA,BETA)* のようになってい

る場合でも、データ・ファイルをロードできます。そのためには、ファイル・タイプ修飾子 seclabelchar を使用する必要があります。 seclabelchar を使用すると、データ・タイプ DB2SECURITYLABEL の列の値は、セキュリティー・ラベル用のストリング・フォーマットで記述されたセキュリティー・ラベルを含む、ストリング定数と見なされます。ストリングが正しい形式でなければ、その行は挿入されず、警告 (SQLSTATE 01H53) が返されます。そのストリングが、表を保護しているセキュリティー・ポリシーに組み込まれている有効なセキュリティー・ラベルを記述したものでない場合も、その行は挿入されず、警告 (SQLSTATE 01H53) が返されます。

セキュリティー・ラベル列の値がセキュリティー・ラベル名になっている場合も、データ・ファイルをロードできます。この種のファイルをロードするには、ファイル・タイプ修飾子 seclabelname を使用する必要があります。seclabelname を使用すると、データ・タイプ DB2SECURITYLABEL の列のすべての値は、既存のセキュリティー・ラベルの名前を含んだストリング定数と見なされます。表を保護しているセキュリティー・ポリシーでその名前のセキュリティー・ラベルが存在しなければ、その行はロードされず、警告 (SQLSTATE 01H53) が返されます。

リジェクトされた行
ロード中にリジェクトされた行は、リジェクトされた理由に応じて、ダンプ・ファイルか例外表のいずれかに送られます (それらが LOAD コマンドで指定された場合)。構文解析エラーが原因でリジェクトされた行はダンプ・ファイルに送られます。セキュリティー・ポリシーを違反した行は例外表に送られます。

例

すべての例で、入力データ・ファイル myfile.del は DEL 形式になっています。すべての場合のデータのロード先は、以下のステートメントで作成された REPS という名前の表です。

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

次の例の入力ファイルには、デフォルト形式のセキュリティー・ラベルが含まれていると見なされます。

```
db2 load from myfile.del of del modified by delprioritychar insert into reps
```

次の例の入力ファイルには、セキュリティー・ラベル・ストリング・フォーマットのセキュリティー・ラベルが含まれていると見なされます。

```
db2 load from myfile.del of del modified by seclabelchar insert into reps
```

次の例の入力ファイルには、セキュリティー・ラベル列のセキュリティー・ラベル名が含まれていると見なされます。

```
db2 load from myfile.del of del modified by seclabelname insert into reps
```

ID 列のロードに関する考慮事項

ロード・ユーティリティーを使用すると、入力データに ID 列の値があるかどうかにかかわりなく、ID 列が含まれている表にデータをロードできます。

ID 関連のファイル・タイプ修飾子が使用されない場合、このユーティリティーは次のような規則に従って動作します。

- ID 列が GENERATED ALWAYS の場合、入力ファイル内の対応する行の中に ID 列用の値がないか、または NULL 値が明示的に指定されているときに、表の行用の ID 値が生成されます。 ID 列に非 NULL 値を指定すると、その行はリジェクトされます (SQL3550W)。
- ID 列が GENERATED BY DEFAULT の場合、ユーザー提供値が指定されていれば、ロード・ユーティリティーはその値を使用します。データが欠落しているかまたは明示的に NULL であれば、値が生成されます。

ロード・ユーティリティーは、ユーザー提供の ID 値に関して、 ID 列のデータ・タイプ (SMALLINT、INT、BIGINT、または DECIMAL) の値に対して通常以外の妥当性検査は行いません。値が重複していても報告されません。

ほとんどの場合、ロード・ユーティリティーは、行への ID 列値の割り当て順がデータ・ファイル内での出現順と同じになることを保証できません。 ID 列値の割り当てはロード・ユーティリティーが並列で管理するため、これらの値は任意の順に割り当てられます。これについての例外は、以下のとおりです。

- 単一パーティション・データベースでは、CPU_PARALLELISM が 1 に設定されているとき、行は並列に処理されません。この場合、ID 列値は暗黙のうちに、データ・ファイル・パラメーター内での行の出現順と同じ順序で割り当てられます。
- 複数パーティション・データベースでは、ID 列値が分散キーにある場合、および単一パーティション・エージェントがある場合 (つまり、複数パーティション・エージェントまたは `anyorder` ファイル・タイプ修飾子を指定しない場合)、ID 列値はデータ・ファイルでの行の出現順と同じ順序で割り当てられます。

表をパーティション・データベースにロードするときに、表のパーティション・キーに ID 列があり、`identityoverride` 修飾子が指定されていない場合、SAVECOUNT オプションを指定できません。パーティション・キーに ID 列があり、ID 値が生成されている場合、少なくとも 1 つのデータベース・パーティション上のロード・フェーズからロードを再始動するには、ロード・フェーズの最初からロード全体を再始動する必要があります。これは、整合点が見つからないことを意味します。

注: 以下の基準すべてが満たされている場合、ロード RESTART 操作は許可されません。

- ロードされている表がパーティション・データベース環境にあり、分散キーにある ID 列、または分散キーの一部となる生成列で参照される ID 列のいずれかを少なくとも 1 つその表に含む。
- `identityoverride` 修飾子が指定されていない。
- 失敗した前回のロード操作には、ロード・フェーズの後に失敗したデータベース・パーティションのロードが含まれていた。

代わりに LOAD TERMINATE または LOAD REPLACE 操作を実行してください。

ID 列が含まれている表にデータをロードする操作を簡略化するには、ファイル・タイプ修飾子として、identitymissing、 identityignore、および identityoverride の 3 つのうちのいずれかを使用する方法があります。これらの指定は相互に排他的です。

ID 列を持たないデータのロード

identitymissing 修飾子は、入力データ・ファイルに ID 列の値が入っていない (NULL すらない) 場合に、 ID 列を持つ表のロードを容易にします。例えば、次のような SQL ステートメントで定義された表があるとします。

```
create table table1 (c1 varchar(30),
                     c2 int generated by default as identity,
                     c3 decimal(7,2),
                     c4 char(1))
```

ID 列を持たない表からエクスポートされたファイル (load.del) からデータを TABLE1 にロードする場合、以下の例を参照してください。

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

このファイルをロードする 1 つの方法は、次のように LOAD コマンドを使用して、ロードする列を明示的にリストすることです。

```
db2 load from load.del of del replace into table1 (c1, c3, c4)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをロードする別の方法は、次のように identitymissing ファイル・タイプ修飾子を使うことです。

```
db2 load from load.del of del modified by identitymissing
   replace into table1
```

このコマンドを実行すると、データ・ファイルの 3 つの列が TABLE1 の c1、c3、および c4 にロードされます。値は c2 の各行に生成されます。

ID 列を持つデータのロード

identityignore 修飾子を指定すると、ロード・ユーティリティーは、入力データ・ファイルに ID 列用のデータが入っていても、そのデータを無視して、各行ごとに ID 値を生成します。例えば、上記で定義したように、ユーザーが次のようなデータの入ったデータ・ファイル (load.del) から TABLE1 をロードするとします。

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

ユーザー指定値の 1、2、および 3 が ID 列に使われない場合は、次のような LOAD コマンドを発行することができます。

```
db2 load from load.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように identityignore 修飾子を使用すると、構文が単純化されます。

```
db2 load from load.del of del modified by identityignore
   replace into table1
```

ユーザー指定値を持つデータのロード

identityoverride 修飾子は、 GENERATED ALWAYS ID 列をもつ表にユーザー指定値をロードするために使用します。これが非常に役立つのは、別のデータベース・システムからデータをマイグレーションするときに GENERATED ALWAYS として表を定義しなければならない場合、または ROLLFORWARD DATABASE コマンドで DROPPED TABLE RECOVERY オプションを使用してリカバリーしたデータから表をロードする場合です。この修飾子を使用した場合、 ID 列でデータ (または NULL データ) の入っていない行はリジェクトされます (SQL3116W)。この修飾子を使用すると、 GENERATED ALWAYS 列の固有性特性に違反する可能性があることも覚えておいてください。この違反が発生した場合にはまず LOAD TERMINATE 操作を実行し、その後、LOAD INSERT または LOAD REPLACE 操作を実行してください。

生成列のロードに関する考慮事項

入力データに生成列の値があるかどうかに関係なく、ID 列以外の生成列が含まれている表にデータをロードできます。ロード・ユーティリティーは列値を生成します。

生成列関連のファイル・タイプ修飾子が使用されない場合、ロード・ユーティリティーは次のような規則に従って動作します。

- データ・ファイルの対応する行に列の値が欠落している場合や NULL 値が提供されている場合には、生成列に値が作成されます。生成列に非 NULL 値を指定すると、その行はリジェクトされます (SQL3550W)。
- NULL 可能列でない生成列用に NULL 値が作成されると、データの行全体がリジェクトされます (SQL0407N)。これが起きるのは、例えば、NULL 可能列でない生成列が 2 つの表の列の合計として定義されていて、それらの表の列にデータ・ファイルに NULL 値が組み込まれた場合です。

生成列が含まれている表にデータをロードする操作を簡略化するには、ファイル・タイプ修飾子として、generatedmissing、 generatedignore、および generatedoverride の 3 つのうちのいずれかを使用する方法があります。これらの指定は相互に排他的です。

生成列のないデータのロード

generatedmissing 修飾子は、表内に存在する生成列の値が入力データ・ファイル内に入っていない (NULL すらない) 場合に、生成列を持つ表のロードを容易にします。例えば、次のような SQL ステートメントで定義された表があるとします。

```
CREATE TABLE table1 (c1 INT,
                     c2 INT,
                     g1 INT GENERATED ALWAYS AS (c1 + c2),
                     g2 INT GENERATED ALWAYS AS (2 * c1),
                     c3 CHAR(1))
```

生成列を持たない表からエクスポートされたファイル (load.del) からデータを TABLE1 にロードする場合、以下の例を参照してください。

```
1, 5, J
2, 6, K
3, 7, I
```

このファイルをロードする 1 つの方法は、次のように LOAD コマンドを使用して、ロードする列を明示的にリストすることです。

```
DB2 LOAD FROM load.del of del REPLACE INTO table1 (c1, c2, c3)
```

ただし、多くの列を持つ表の場合、この構文は扱いにくく、誤りを生じる可能性があります。ファイルをロードする別の方法は、次のように generatedmissing ファイル・タイプ修飾子を使うことです。

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedmissing
   REPLACE INTO table1
```

このコマンドを実行すると、データ・ファイルの 3 つの列が TABLE1 の c1、c2、および c3 にロードされます。 generatedmissing 修飾子により、TABLE1 の列 g1 と g2 の値が自動的に生成されます。それらの値はデータ・ファイル列にマップされません。

生成列のあるデータのロード

generatedignore 修飾子を指定すると、ロード・ユーティリティーは、ターゲット表にあるすべての生成列用のデータが入力データ・ファイルに入っていても、そのデータを無視して、生成される各列に計算された値をロードします。例えば、上記で定義したように、次のようなデータの入ったデータ・ファイル (load.del) から TABLE1 をロードするとします。

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

生成列に関連したファイル・タイプ修飾子が使用されない場合、ユーザー指定の非 NULL 値 10、11、12 (g1 の場合)、および 15、16、17 (g2 の場合) により、行はリジェクトされます (SQL3550W)。これを回避するには、次のような LOAD コマンドを発行します。

```
DB2 LOAD FROM load.del of del method P(1, 2, 5)
   REPLACE INTO table1 (c1, c2, c3)
```

ここでも、表に多くの列があると、このアプローチは扱いにくく、誤りを生じる可能性があります。次のように generatedignore 修飾子を使用すると、構文が単純化されます。

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedignore
   REPLACE INTO table1
```

このコマンドを実行すると、データ・ファイルの列が TABLE1 の c1 (データ 1、2、3 が入る)、c2 (データ 5、6、7 が入る)、および c3 (データ J、K、I が入る) にロードされます。 generatedignore 修飾子により TABLE1 の列 g1 と g2 の値が自動的に生成され、データ・ファイルの列 (10、11、12 および 15、16、17) は無視されます。

ユーザー指定値を持つデータのロード

generatedoverride 修飾子は、生成列をもつ表にユーザー指定値をロードするために使用します。これが役立つのは、別のデータベース・システムからデータをマイグレーションする場合、または ROLLFORWARD DATABASE コマンドの RECOVER DROPPED TABLE オプションを使用してリカバリーしたデータから表をロードする場合です。この修飾子を使用した場合、 NULL 不能の生成列でデータ (または NULL データ) の入っていない行はリジェクトされます (SQL3116W)。

この修飾子が使用される場合、ロード操作の後、表は SET INTEGRITY ペンディング状態になります。ユーザー指定値を検証せずに、表を SET INTEGRITY ペンディング状態から解放するには、以下のコマンドを発行します。

```
SET INTEGRITY FOR table-name GENERATED COLUMN IMMEDIATE
  UNCHECKED
```

表を SET INTEGRITY ペンディング状態から解放し、ユーザー指定値を強制的に検査するには、以下のコマンドを発行します。

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED
```

生成される列がパーティション・キー、ディメンション・キー、または分散キーのいずれかにある場合、 generatedoverride 修飾子は無視され、ロード・ユーティリティーは generatedignore 修飾子が指定された場合のように値を生成します。これは、ユーザー指定の生成列値がその生成列の定義と矛盾するようなシナリオを回避するために行われます。そのような矛盾があると、結果として生成されるレコードは不適切な物理ロケーション (例えば不適切なデータ・パーティション、MDC ブロック、またはデータベース・パーティション) に配置されてしまいます。

注: 列値の生成がロードでサポートされないケースが 1 つあります。生成された列式のうちの 1 つに、FENCED であるユーザー定義関数が入っている場合です。そのような表へのロードを試みると、ロード操作は失敗します。ただし、generatedoverride ファイル・タイプ修飾子を使えば、その種の生成列に自分独自の値を指定することができます。

バージョン 7 以前のクライアントとバージョン 8 以降のサーバーを同時に使用する場合の考慮事項

バージョン 7 以前のクライアントとバージョン 8 以降のサーバーとの間でロード操作を開始する場合、ロード・ユーティリティーは、生成列がある表を SET INTEGRITY ペンディング状態にします。バージョン 7 以前のクライアントが生成列のある表にデータをロードするのに使用されたために、表が SET INTEGRITY ペンディング状態になった場合には、以下のステートメントを発行してその状態を解除し、強制的に値を生成してください。

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED FORCE GENERATED;
```

CURSOR ファイル・タイプを使用したデータの移動

LOAD コマンドを使用する際に CURSOR ファイルを指定することにより、中間エクスポート・ファイルを作成しなくても、 SQL 照会の結果を直接ターゲット表にロードすることができます。

さらに、SQL 照会内でニックネームを参照するか、DECLARE CURSOR ステートメント内で DATABASE オプションを使用するか、または API インターフェースの使用時に sqlu_remotefetch_entry メディア項目を使用することによって、別のデータベースからデータをロードすることができます。

CURSOR ファイル・タイプを使ってデータを移動するための方法は 3 つあります。 1 つ目の方法はコマンド行プロセッサー (CLP) を使用する方法で、2 つ目は API を使用する方法、3 つ目は ADMIN_CMD プロシージャーを使用する方法です。 CLP と ADMIN_CMD プロシージャーの主な違いが以下の表で略述されています。

表 26. *CLP* と *ADMIN_CMD* プロシージャーの違い。

相違	CLP	ADMIN_CMD プロシージャー
構文	照会ステートメント、およびカーソルで使用されるソース・データベースは、LOAD コマンドの外部で、DECLARE CURSOR ステートメントを使って定義されます。	照会ステートメント、およびカーソルで使用されるソース・データベースは、LOAD コマンドの内部で、LOAD from (DATABASE database-alias query-statement) を使って定義されます。
別のデータベースにアクセスするためのユーザー許可	現在接続しているものとは別のデータベースにデータがある場合、DATABASE キーワードを DECLARE CURSOR ステートメント内で使用する必要があります。同ステートメントにユーザー ID とパスワードを指定することもできます。 DECLARE CURSOR ステートメントにユーザー ID およびパスワードを指定しない場合、ソース・データベースの接続に対して明示的に指定するユーザー ID およびパスワードがターゲット・データベースのアクセスに使用されます。	現在接続しているものとは別のデータベースにデータがある場合、照会ステートメントの前に LOAD コマンドで DATABASE キーワードを使用する必要があります。ターゲット・データベースにアクセスするには、ソース・データベースの接続に対して明示的に指定したユーザー ID とパスワードが必要です。ソース・データベースにユーザー ID またはパスワードを指定することはできません。そのため、ターゲット・データベースとの接続確立時にユーザー ID およびパスワードを指定しなかった場合、または指定されたユーザー ID とパスワードを使ってソース・データベースに対して認証を行えない場合、ADMIN_CMD プロシージャーを使ってロードを実行することはできません。

CLP から LOAD FROM CURSOR 操作を実行するには、まず SQL 照会に対してカーソルを宣言しなければなりません。これが宣言できたら、宣言されるカーソルの名前を *cursorname* に、また CURSOR をファイル・タイプにして、LOAD コマンドを発行できます。

以下に例を示します。

1. 以下の定義に従い、ソース表とターゲット表の両方が同じデータベースに存在すると仮定します。

   表 ABC.TABLE1 には次の 3 つの列があります。

   - ONE INT
   - TWO CHAR(10)
   - THREE DATE

表 ABC.TABLE2 には次の 3 つの列があります。

- ONE VARCHAR
- TWO INT
- THREE DATE

以下の CLP コマンドを実行すると、すべてのデータが ABC.TABLE1 から ABC.TABLE2 にロードされます。

```
DECLARE mycurs CURSOR FOR SELECT TWO, ONE, THREE FROM abc.table1
   LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

注: 上記の例では、CLP を介して SQL 照会からロードする方法を示します。ただし、db2Load API を介して SQL 照会からロードすることもできます。*sqlu_statement_entry* 構造および SQLU_SQL_STMT メディア・タイプを使用するには sqlu_media_list 構造の *piSourceList* を定義します。そして *piFileType* の値を SQL_CURSOR として定義します。

2. 以下の定義に従い、ソース表とターゲット表がそれぞれ異なるデータベースに存在すると仮定します。

データベース「dbsource」の表 ABC.TABLE1 には次の 3 つの列があります。

- ONE INT
- TWO CHAR(10)
- THREE DATE

データベース「dbtarget」の表 ABC.TABLE2 には次の 3 つの列があります。

- ONE VARCHAR
- TWO INT
- THREE DATE

フェデレーションを可能にし、データ・ソース ('dsdbsource') をカタログした場合は、次の例に示すとおり、ソース・データベースに対してニックネームを宣言した後、このニックネームに対してカーソルを宣言し、FROM CURSOR オプションを指定して LOAD コマンドを呼び出すことができます。

```
CREATE NICKNAME myschema1.table1 FOR dsdbsource.abc.table1
DECLARE mycurs CURSOR FOR SELECT TWO,ONE,THREE FROM myschema1.table1
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

あるいは以下の例で示されているように、DECLARE CURSOR ステートメントの DATABASE オプションを使用することもできます。

```
DECLARE mycurs CURSOR DATABASE dbsource USER dsciaraf USING mypasswd
FOR SELECT TWO,ONE,THREE FROM abc.table1
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

DECLARE CURSOR ステートメントの DATABASE オプション (ロード API 使用時の remotefetch メディア・タイプに相当する) を使用することには、ニックネームのアプローチに勝る利点があります。

パフォーマンス

remotefetch メディア・タイプを使ったデータのフェッチは、ロード操作と緊密に統合されています。ニックネームのアプローチと比べて、レコード・フェッチのための遷移の層が少なくなります。さらに、複数パーティション・データベースでソース表とターゲット表が全く均等に分散されている場合、ロード・ユーティリティーはデータのフェッチを並列的に処理します。これにより、パフォーマンスは向上します。

使いやすさ

フェデレーションの使用可能化、リモート・データ・ソースの定義、またはニックネームの宣言は不要です。必要なのは DATABASE オプション (および必要な場合は USER および USING オプション) の指定だけです。

このメソッドはカタログされているデータベースで使用できますが、ニックネームの使用は、簡単にカタログできない各種データ・ソースからのフェッチのための堅固な機構を提供します。

この remotefetch 機能をサポートするために、ロード・ユーティリティーは SOURCEUSEREXIT 機能をサポートするインフラストラクチャーを利用します。ロード・ユーティリティーは、アプリケーションとして実行されるプロセスを作成し、それによってソース・データベースへの接続を管理し、フェッチを実行します。このアプリケーションはロード・ユーティリティーが実行されるトランザクションではなく、固有のトランザクションと関連付けられます。

注:

1. 上記の例では、CLP を介して DECLARE CURSOR ステートメントの DATABASE オプションを使用することにより、カタログされているデータベースに対する SQL 照会からロードを行う方法を示しています。しかし、db2Load API を介して、カタログされているデータベースに対する SQL 照会からロードを行うこともできます。その場合は、*db2LoadStruct* 構造の *piSourceList* および *piFileTypevalues* を、それぞれ sqlu_remotefetch_entry メディア項目および SQLU_REMOTEFETCH メディア・タイプを使用するよう定義します。
2. 上記の例で示したとおり、SQL 照会のソース列タイプはターゲット列タイプと互換性がなければなりませんが、同一である必要はありません。

制約事項

DATABASE オプションを使用して定義されているカーソルからロードする場合 (db2Load API で sqlu_remotefetch_entry メディア項目を使用する場合も同様)、次の制約事項が適用されます。

1. SOURCEUSEREXIT オプションを並行して指定することはできません。
2. METHOD N オプションはサポートされません。
3. `usedefaults` ファイル・タイプ修飾子はサポートされません。

従属即時ステージング表の伝搬

ロードされる表が即時伝搬属性を持つステージング表の基礎表であり、ロード操作が挿入モードで実行される場合には、従属即時ステージング表への後続の伝搬は増分になります。

増分伝搬時には、基礎表にある追加行に対応する行がステージング表に追加されます。基礎表が大きく、追加データが少ない場合には、増分伝搬の速度はそれだけ速くなります。また、ステージング表を使用して、その従属据え置きマテリアライズ照会表をリフレッシュすると、パフォーマンスが改善されます。増分伝搬が許可されず、ステージング表に不完全というマークが付けられる場合もあります。つまり、CONST_CHECKED 列のステージング・バイトの値は F になります。この状態では、ステージング表を使用して、従属据え置きマテリアライズ照会表をリフレッシュすることはできず、マテリアライズ照会表の保守プロセスでフル・リフレッシュが必要になります。

表が不完全状態で、INCREMENTAL オプションが指定されているにもかかわらず、表の増分伝搬が実行できない場合、エラーが戻されます。以下のいずれかが発生すると、システムは即時データ伝搬をオフにし、表状態を不完全に設定します。

- ステージング表の基礎表でロード置換操作が実行されるか、または基礎表に対する最後の整合性チェックの後で NOT LOGGED INITIALLY WITH EMPTY TABLE オプションが活動化された場合。
- ステージング表の従属マテリアライズ照会表、またはステージング表が REPLACE または INSERT モードでロードされた場合。
- 整合性チェック時に FULL ACCESS オプションを使用することによってステージング表が伝搬される前に、基礎表が SET INTEGRITY ペンディング状態ではなくなった場合。
- ステージング表の基礎表で、整合性のチェックが非増分的に実行された場合。
- ステージング表またはその基礎表が入った表スペースがある時点でロールフォワードされており、ステージング表とその基礎表が異なる表スペースに常駐する場合。

ステージング表で、 SYSCAT.TABLES カタログの CONST_CHECKED 列に W 値があり、 NOT INCREMENTAL オプションが指定されていない場合、ステージング表への増分伝搬が実行され、 SYSCAT.TABLES の CONST_CHECKED 列には、すべてのデータをシステムがチェックしたわけではないことを示す U というマークが付けられます。

以下の例は、ステージング表 G1 の基礎表 UT1 およびその従属据え置きマテリアライズ照会表 AST1 へのロード挿入操作を示します。このシナリオでは、UT1 の整合性チェックと AST1 のリフレッシュの両方が増分的に処理されます。

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1,G1 IMMEDIATE CHECKED;

REFRESH TABLE AST1 INCREMENTAL;
```

従属即時マテリアライズ照会表のリフレッシュ

即時リフレッシュ・マテリアライズ照会表の基礎表が INSERT オプションを使用してロードされる場合には、 REFRESH IMMEDIATE で定義される従属マテリアライズ照会表で SET INTEGRITY ステートメントを実行すると、マテリアライズ照会表で増分リフレッシュが実行されます。

増分リフレッシュ時には、基礎表にある追加行に対応する行が更新され、マテリアライズ照会表に挿入されます。基礎表が大きく、追加データが少ない場合には、増

分リフレッシュの速度はそれだけ速くなります。増分リフレッシュが許可されず、フル・リフレッシュ (つまり、マテリアライズ照会表定義照会の再計算) が使用される場合もあります。

INCREMENTAL オプションを指定したにもかかわらず、マテリアライズ照会表の増分的な処理を実行できない場合、以下に示す条件が成立するならエラーが戻されます。

- マテリアライズ照会表の基礎表でロード置換操作が実行されるか、または基礎表に対する最後の整合性チェックの後で NOT LOGGED INITIALLY WITH EMPTY TABLE オプションが活動化された場合。
- マテリアライズ照会表がロードされている場合 (REPLACE または INSERT モードのいずれかで)。
- 整合性チェック時に FULL ACCESS オプションを使用することによってマテリアライズ照会表がリフレッシュされる前に、基礎表が SET INTEGRITY ペンディング状態ではなくなった場合。
- マテリアライズ照会表の基礎表で、整合性のチェックが非増分的に実行された場合。
- マテリアライズ照会表が、マイグレーション前の SET INTEGRITY ペンディング状態にあった場合。
- マテリアライズ照会表またはその基礎表が入った表スペースがある時点でロールフォワードされており、マテリアライズ照会表とその基礎表が異なる表スペースに常駐する場合。

マテリアライズ照会表で、SYSCAT.TABLES カタログの CONST_CHECKED 列に 1 つまたは複数の W 値がある場合で、 SET INTEGRITY ステートメントに NOT INCREMENTAL オプションが指定されていない場合、表は増分にリフレッシュされ、SYSCAT.TABLES の CONST_CHECKED 列には、すべてのデータをシステムがチェックしたわけではないことを示す U というマークが付けられます。

以下の例は、マテリアライズ照会表 AST1 の基礎表 UTI へのロード挿入操作を示します。 UT1 でデータ整合性がチェックされ、データ移動不可モードになります。 AST1 の増分リフレッシュが完了すると、UT1 はフル・アクセス状態に戻ります。このシナリオでは、UT1 の整合性チェックと AST1 のリフレッシュの両方が増分的に処理されます。

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1 IMMEDIATE CHECKED;
REFRESH TABLE AST1;
```

のマルチディメンション・クラスタリングの考慮事項

以下の制約事項はマルチディメンション・クラスタリング (MDC) 表へのデータのロード時に適用されます。

- LOAD コマンドの SAVECOUNT オプションはサポートされていません。
- これらの表は独自のフリー・スペースを管理するため、 totalfreespace ファイル・タイプ修飾子はサポートされていません。
- MDC 表には anyorder ファイル・タイプ修飾子が必要です。 anyorder 修飾子を使わないで MDC 表へのロードを実行すると、 anyorder 修飾子はユーティリティーによって暗黙的に使用可能になります。

MDC 表に LOAD コマンドを使用する場合には、ユニーク制約の違反は以下のように処理されます。

- ロードするデータと同じユニーク・キーを持つレコードが (ロード操作の開始前に既に) 表に存在する場合、元のレコードは残り、新規レコードが削除フェーズで削除されます。
- ロードするデータと同じユニーク・キーを持つレコードが (ロード操作の開始前には) 表に存在しない場合、ユニーク・キーとそれと重複する (同じユニーク・キーを持つ) レコードの両方が表にロードされる場合には、レコードのうち 1 つだけがロードされ、その他のレコードは削除フェーズで除去されます。

  注: どのレコードがロードされて、どのレコードが削除されるかを判別するための明示的な技法はありません。

パフォーマンスの考慮

MDC 表のロード時のロード・ユーティリティーのパフォーマンスを改善するには、*util_heap_sz* データベース構成パラメーター値を大きくしなければなりません。ユーティリティーで使用できるメモリーを増やすと、 mdc-load アルゴリズムのパフォーマンスが大きく向上します。こうすると、ロード・フェーズで実行されるデータのクラスタリング時に、ディスクの入出力を減らすことができます。 LOAD コマンドの DATA BUFFER オプションが指定される際には、この値も大きくしなければなりません。 LOAD コマンドを使用して複数の MDC 表を同時にロードする場合には、それに応じて、*util_heap_sz* の値を大きくしなければなりません。

すべての MDC 表にはブロック索引があるため、MDC ロード操作には常に構築フェーズがあります。

ロード・フェーズでは、ブロック・マップの保守のために余分のロギングが実行されます。割り振られるエクステントごとに、おおよそ 2 つの余分のログ・レコードがあります。パフォーマンスを良くするためには、このことを考慮に入れた値に *logbufsz* データベース構成パラメーターを設定する必要があります。

MDC 表にデータをロードするために、索引付きのシステム一時表が使用されます。表のサイズはロードされる個々のセルの数に比例します。表にあるそれぞれの行のサイズは MDC 次元キーのサイズに比例します。ロード操作時にこの表の操作によるディスク入出力を最小限に抑えるには、 TEMPORARY 表スペースのバッファー・プールの大きさが十分であることを確認してください。

カスタマイズしたアプリケーション (ユーザー出口) を使用したデータの移動

ロード SOURCEUSEREXIT オプションを使用すると、カスタマイズしたスクリプトまたは実行ファイル (ここではユーザー*出口*と呼びます) をロード・ユーティリティーが実行するための機構が提供されます。

ユーザー出口の目的は、1 つ以上の Named PIPE に、ロード・ユーティリティーによって同時に読み取られるデータを取り込むことです。複数パーティション・データベースでは、ユーザー出口の複数のインスタンスを同時に呼び出すことにより、入力データの並列処理を実現することができます。

図 5 で示されているとおり、ロード・ユーティリティーは 1 つ以上の Named PIPE を作成し、カスタマイズされた実行ファイルを実行するためのプロセスを作成します。ユーザー出口による Named PIPE へのデータのフィードとロード・ユーティリティーによる読み取りは並行して行われます。

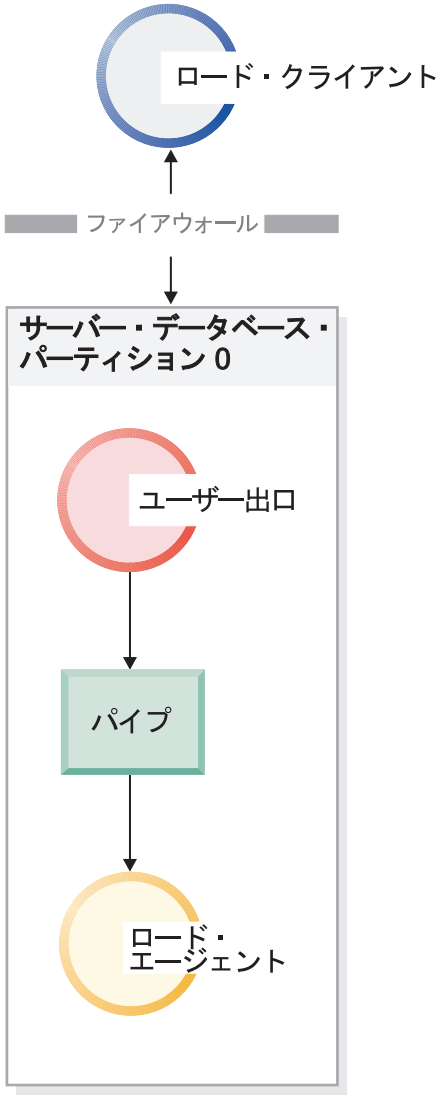


図 5. ロード・ユーティリティーは着信データをパイプから読み取って、処理します。

パイプにフィードされるデータは、指定されるロード・オプション (ファイル・タイプおよびファイル・タイプ修飾子を含む) を反映しなければなりません。ロード・ユーティリティーは、指定されるデータ・ファイルを直接的には読み取りません。代わりに、指定されるデータ・ファイルはユーザー出口に対する引数として、その実行時に渡されます。

ユーザー出口の呼び出し

ユーザー出口は DB2 インストール・ディレクトリー (多くの場合 sqllib) の bin サブディレクトリーになければなりません。ロード・ユーティリティーは、次のコマンド行引数を使ってユーザー出口の実行ファイルを呼び出します。

```
<base pipename> <number of source media>
<source media 1> <source media 2> ... <user exit ID>
<number of user exits> <database partition number>
```

詳細は次のとおりです。

<base pipename >
ロード・ユーティリティーによって作成され、データの読み取り元となる Named PIPE のベース名です。ユーティリティーは、LOAD コマンドに対して提供される各ソース・ファイルにつき 1 つのパイプを作成します。これらのパイプには .xxx が付加されます。xxx は、提供されるソース・ファイルの索引です。例えば、LOAD コマンドに 2 つのソース・ファイルが提供され、ユーザー出口に渡される <base pipename> 引数が pipe123 である場合、ユーザー出口がデータをフィードする 2 つの名前付きパイプは pipe123.000 および pipe123.001 となります。パーティション・データベース環境では、ロード・ユーティリティーは基本のパイプ名にデータベース・パーティション (DBPARTITION) 番号 .yyy を付加して、pipe123.xxx.yyy. というパイプ名になります。

<number of source media>
後に続くメディア引数の数です。

<source media 1> <source media 2> ...
LOAD コマンドで指定される 1 つ以上のソース・ファイルのリストです。各ソース・ファイルは二重引用符で囲まれます。

<user exit ID>
PARALLELIZE オプションを使用可能にした場合に使うことができる特殊値です。この整数値 (1 から N の範囲で、N は作成されるユーザー出口の総数) は、実行中のユーザー出口の特定のインスタンスを識別します。
PARALLELIZE オプションが使用可能でない場合、この値はデフォルトの 1 になります。

<number of user exits>
PARALLELIZE オプションを使用可能にした場合に使うことができる特殊値です。この値は、現在実行中のユーザー出口の総数を表します。
PARALLELIZE オプションが使用可能でない場合、この値はデフォルトの 1 になります。

<database partition number>
PARALLELIZE オプションを使用可能にした場合に使うことができる特殊値です。これはユーザー出口が実行しているデータベース・パーティション (DBPARTITION) 番号です。 PARALLELIZE オプションが使用可能でない場合、この値はデフォルトの 0 になります。

追加のオプションとフィーチャー

以下のセクションでは、SOURCEUSEREXIT 機能の追加オプションについて説明します。

REDIRECT
このオプションにより、ユーザー出口プロセスの STDIN ハンドルにデータを渡すか、または STDOUT および STDERR ハンドルからデータを取り込むことが可能になります。

INPUT FROM BUFFER <buffer>

これにより、ユーザー出口の STDIN 入力ストリームに直接情報を渡すことが可能になります。ユーザー出口を実行するプロセスを作成した後、ロード・ユーティリティーはこの新規プロセスの STDIN のファイル記述子を獲得し、指定されたバッファーに渡します。ユーザー出口は STDIN を読み取って情報を獲得します。ロード・ユーティリティーは STDIN を使用して <buffer> の内容をユーザー出口に送るだけで、その内容の解釈や変更は行いません。例えば、ユーザー出口が STDIN から 8 バイトのユーザー ID と 8 バイトのパスワードの 2 つの値を読み取るように設計されている場合、C で作成されたユーザー出口実行ファイルには次の行が含まれるかもしれません。

```
rc = read (stdin, pUserID, 8);
rc = read (stdin, pPasswd, 8);
```

ユーザーは、以下の LOAD コマンドに示すように、INPUT FROM BUFFER オプションを使ってこの情報を渡すことができます。

```
LOAD FROM myfile1 OF DEL INSERT INTO table1
SOURCEUSEREXIT myuserexit1 REDIRECT INPUT FROM BUFFER myuseridmypasswd
```

注: ロード・ユーティリティーは <buffer> のサイズを LOB 値の最大サイズに制限します。しかし、コマンド行プロセッサー (CLP) の場合は、<buffer> のサイズは CLP ステートメントの最大サイズに制限されます。また、CLP の場合、<buffer> に含める文字を従来の ASCII 文字のみにすることも勧められています。 db2Load API を使ってロード・ユーティリティーが呼び出される場合、あるいはその代わりに INPUT FROM FILE オプションが使用される場合、これらの問題を回避できます。

INPUT FROM FILE <filename>

クライアント・サイドのファイルの内容をユーザー出口の STDIN 入力ストリームに直接渡すことが可能になります。このオプションはほぼ INPUT FROM BUFFER オプションと同じですが、このオプションには潜在的 CLP の制限がありません。ファイル名はクライアント・サイドの完全修飾ファイルでなければならず、LOB 値の最大サイズ以下でなければなりません。

OUTPUT TO FILE <filename>

ユーザー出口プロセスの STDOUT および STDERR ストリームを取り込み、それをサーバー・サイドのファイルに入れることが可能になります。ユーザー出口実行可能ファイルを実行するプロセスを作成した後、ロード・ユーティリティーはこの新規プロセスの STDOUT および STDERR ハンドルを指定のファイル名にリダイレクトします。このオプションは、ユーザー出口内のエラーおよびアクティビティーのデバッグとロギングを行うのに便利です。ファイル名は、サーバー・サイドの完全修飾ファイルでなければなりません。 PARALLELIZE オプションが有効になっている場合は、ユーザー出口ごとに 1 つのファイルが存在し、それぞれのファイルには 3 桁の数値 ID が付加されます (例えば *filename.000*)。

PARALLELIZE

このオプションは、複数のユーザー出口プロセスを同時に呼び出すことにより、ロード・ユーティリティーに入るデータのスループットを向上させることができます。このオプションは、複数パーティション・データベースにのみ適用できます。ロード操作中にデータが複数のデータベース・パーティシ

ョンに分散される場合、呼び出されるユーザー出口インスタンスの数はパーティショニング・エージェントの数と同じになります。ロード操作中にデータが複数のデータベース・パーティションに分散されない場合は、ロードするエージェントの数と同じになります。

各ユーザー出口に渡される <user exit ID>、<number of user exits>、および <database partition number> 引数は、それぞれユーザー出口の固有 ID (1 から N)、総数 (N)、およびユーザー出口インスタンスが実行されているデータベース・パーティション (DBPARTITION) 番号を表します。各ユーザー出口プロセスが Named PIPE に書き出すどのデータも、他の並行プロセスによって複写されることがないようにしなければなりません。ユーザー出口アプリケーションがそうするための方法はたくさんありますが、これらの値はデータの複写を防ぐ上で役に立つかもしれません。例えば、各データのレコードに固有の整数の列値が含まれる場合、ユーザー出口アプリケーションは <user exit ID> および <number of user exits> の値を使用して、各ユーザー出口インスタンスが固有の結果セットを自分の名前付きパイプに戻すようにすることができます。ユーザー出口アプリケーションは、次のように **MODULUS** プロパティーを使用することができるかもしれません。

```
i = <user exit ID>
N = <number of user exits>

foreach record
{
   if ((unique-integer MOD N) == i)
   {
     write this record to my named-pipe
   }
}
```

作成されるユーザー出口プロセスの数は、データベース・パーティションに指定される分散モードによって異なります。

1. 172 ページの図 6 が示すように、PARTITION_AND_LOAD (デフォルト) または PARTITION_ONLY (PARALLEL なし) が指定されている場合は、各事前パーティショニング・エージェントごとに 1 つのユーザー出口プロセスが作成されます。 .

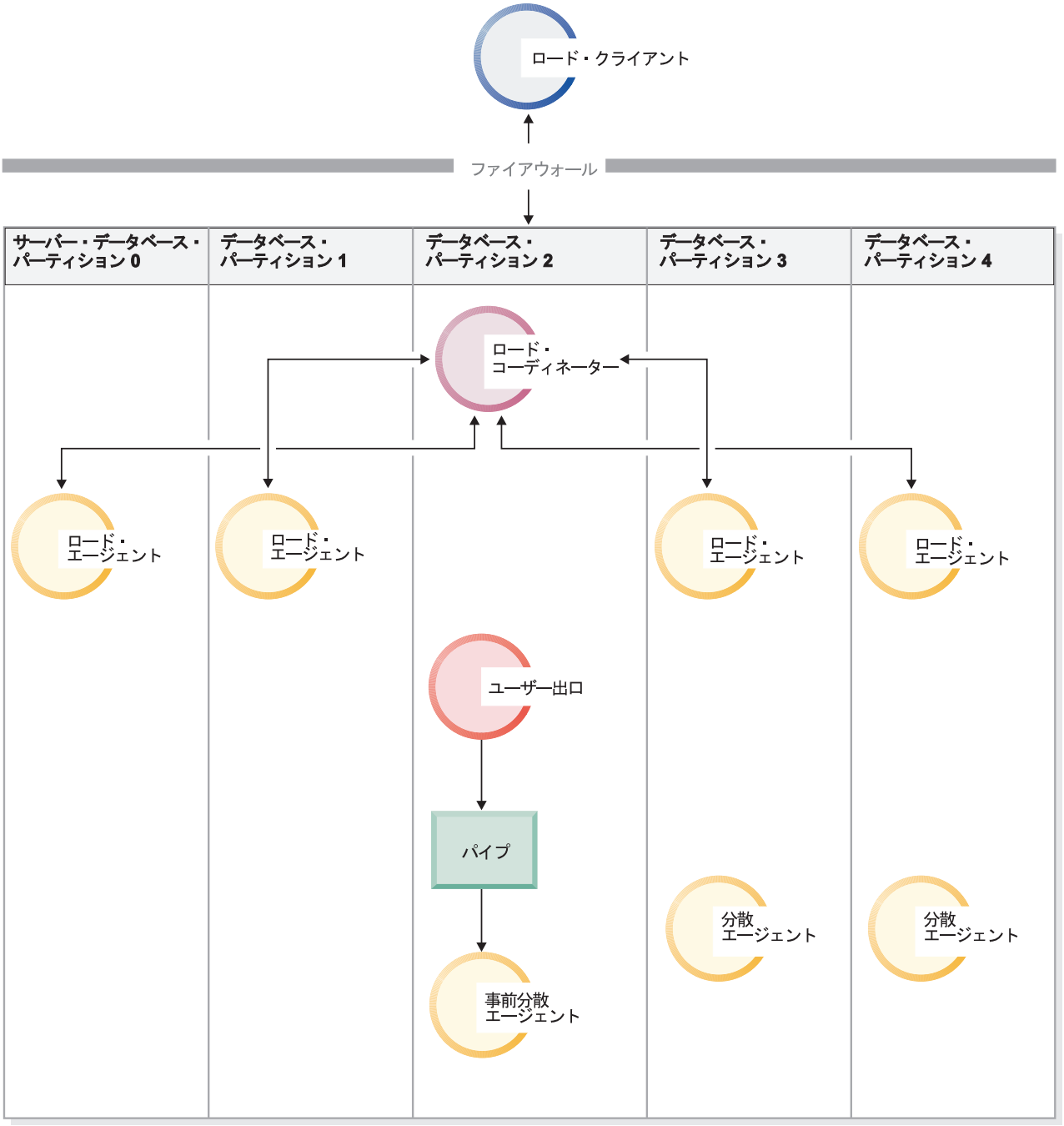


図 6. PARTITION_AND_LOAD (デフォルト) または PARTITION_ONLY (PARALLEL なし) が指定されている場合に実行される各種タスク。

2. 173 ページの図 7 が示すように、PARTITION_AND_LOAD (デフォルト) または PARTITION_ONLY (PARALLEL あり) が指定されている場合は、各パーティショニング・エージェントごとに 1 つのユーザー出口プロセスが作成されます。

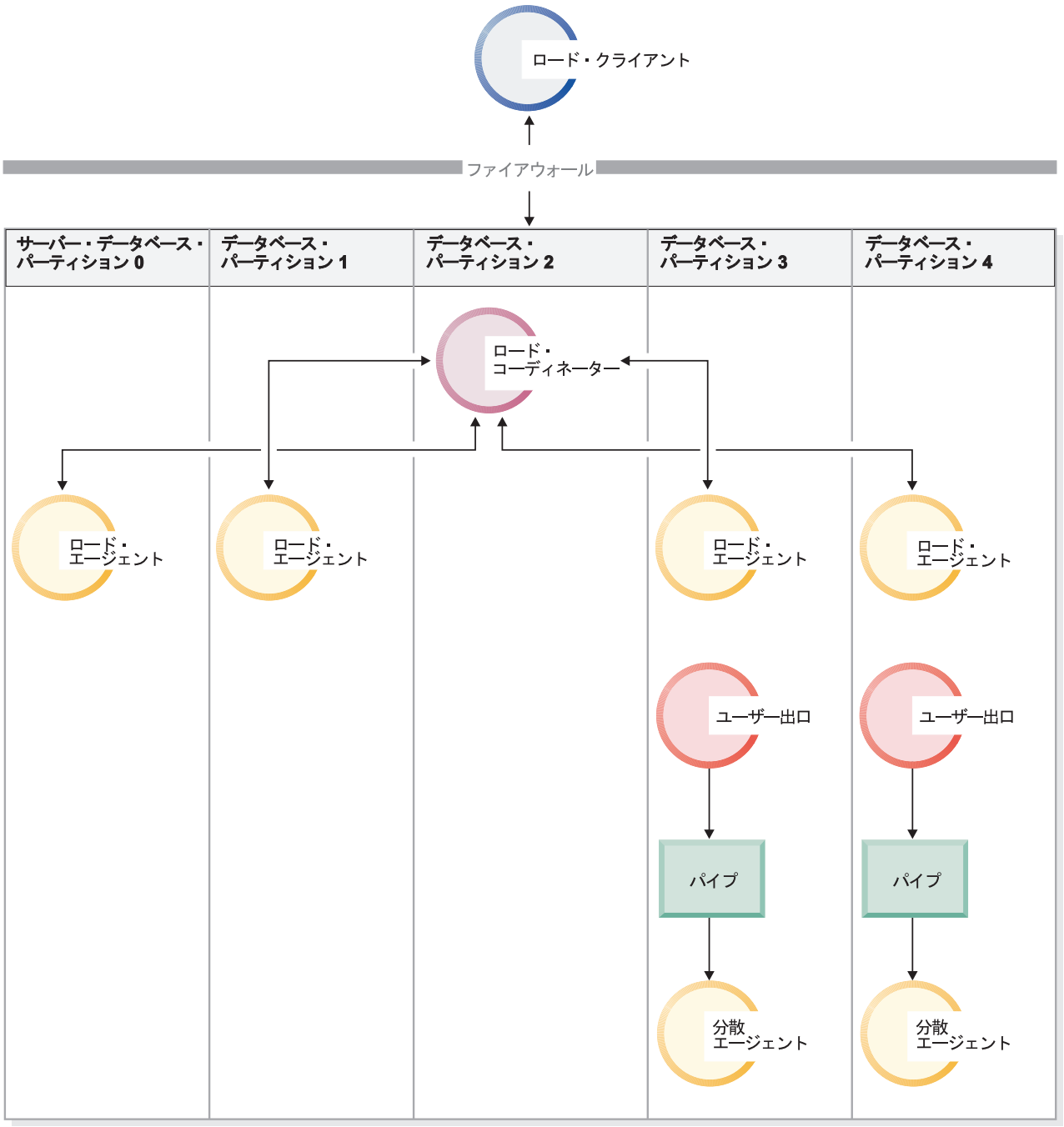


図 7. PARTITION_AND_LOAD (デフォルト) または PARTITION_ONLY (PARALLEL あり) が指定されている場合に実行される各種タスク。

3. 174 ページの図 8 が示すように、 LOAD_ONLY または LOAD_ONLY_VERIFY_PART が指定されている場合は、ロード・エージェントごとに 1 つのユーザー出口プロセスが作成されます。

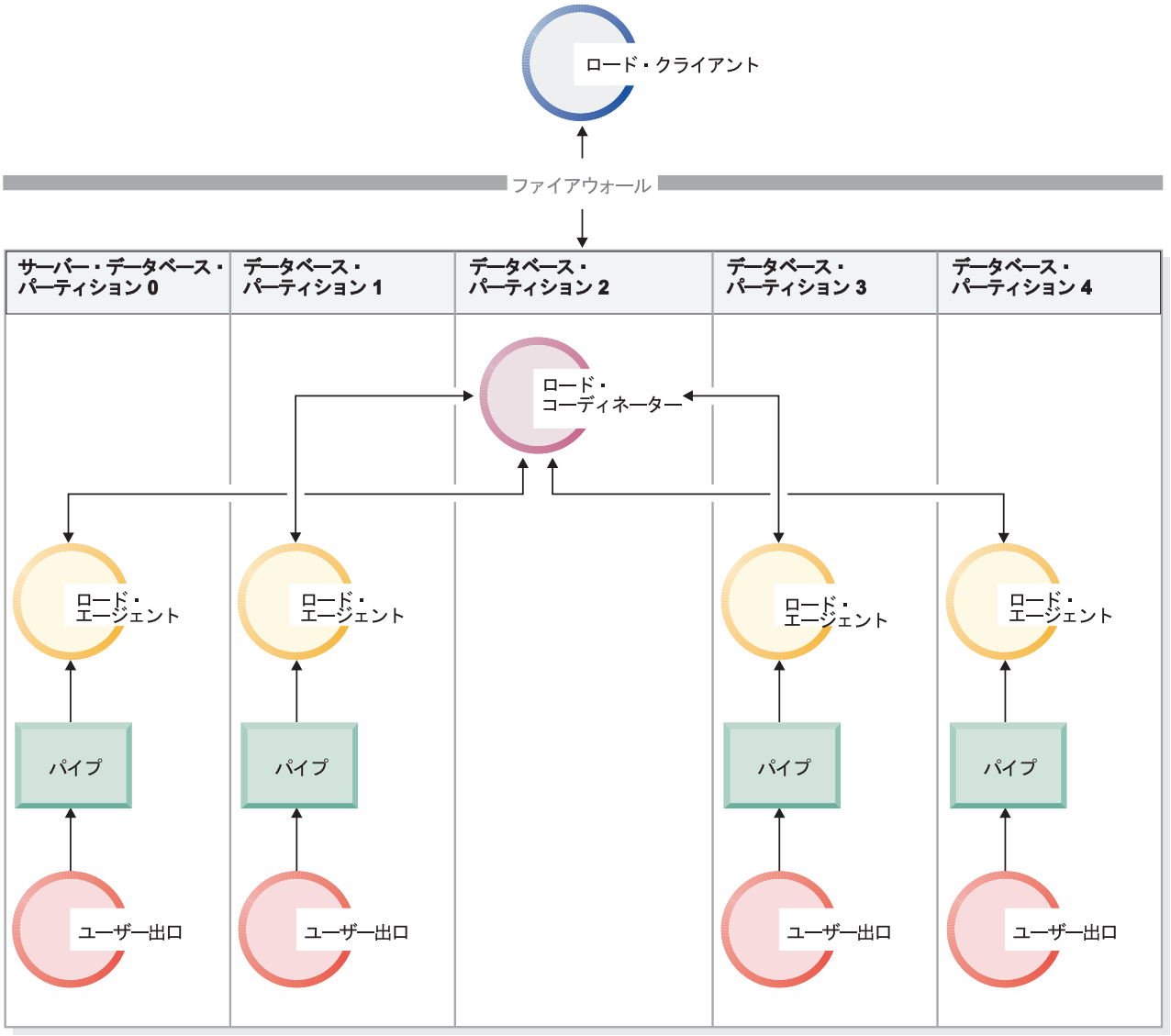


図 8. LOAD_ONLY または LOAD_ONLY_VERIFY_PART が指定される場合に実行される各種タスク。

制約事項

- SOURCEUSEREXIT PARALLELIZE オプションを指定しないと、 LOAD_ONLY および LOAD_ONLY_VERIFY_PART の partitioned-db-cfg モード・オプションはサポートされません。

ロードに関する追加の考慮事項

並列処理とロード

ロード・ユーティリティーは、複数のプロセッサーや複数の記憶装置が使用されているハードウェア構成 (対称マルチプロセッサー (SMP) 環境など) を利用します。

ロード・ユーティリティーを使って大容量データの並列処理を実行する方法はいくつかあります。その 1 つは複数の記憶装置を使用する方法であり、ロード操作中に入出力の並列処理が可能になります (175 ページの図 9 を参照)。別の方法は SMP

環境における複数のプロセッサーの使用が関係しており、パーティション内の並列処理が可能になります (図 10 を参照)。これらの両方の方法を併用すれば、データのロード時間をさらに短くすることができます。

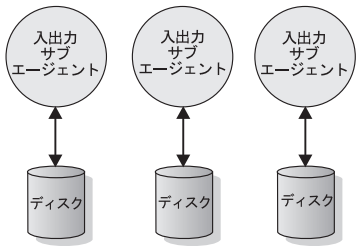


図 9. データ・ロード時に入出力の並列処理を利用する

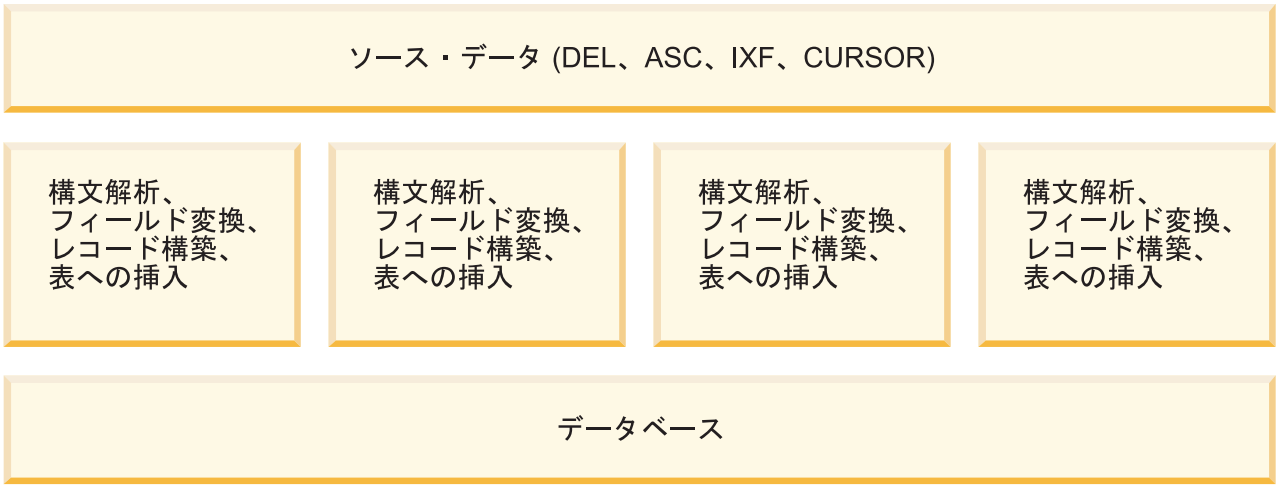


図 10. データ・ロード時のパーティション内の並列処理の利用

ロード操作時の索引作成

索引は、ロード操作の構築フェーズで作成されます。 LOAD コマンドで指定できる索引付けモードは 4 つあります。

1. REBUILD。すべての索引を再作成します。
2. INCREMENTAL。索引を新しいデータで拡張します。
3. AUTOSELECT。ロード・ユーティリティーが REBUILD モードか INCREMENTAL モードかを自動的に決定します。 AUTOSELECT がデフォルトです。ロード REPLACE 操作が実行される場合は REBUILD 索引付けモードが使用されます。ロード REPLACE 操作が実行されない場合、索引付けモードは、新しくロードされたデータの量に対する表内の既存のデータ量の比率に基づいて選択されます。比率が十分に大きい場合は INCREMENTAL 索引付けモードが選択されます。そうでない場合、REBUILD 索引付けモードが選択されます。
4. DEFERRED。このモードが指定されている場合、ロード・ユーティリティーは索引を作成しようとしません。索引には最新表示が必要であるというマークが付け

られ、最初にアクセスされるときに強制的に再作成されることがあります。次の状況では DEFERRED オプションを使用できません。

- ALLOW READ ACCESS オプションが指定される場合 (このオプションは索引を保守せず、索引スキャナーは有効な索引を必要とする)
- 表に対してユニーク索引が定義される場合
- XML データがロードされる場合 (XML Path 索引はユニーク索引であり、デフォルトでは表に XML 列が追加されるたびに作成される)

ALLOW READ ACCESS オプションを指定するロード操作では、選択した索引付けモードによっては、スペース使用量およびロギングに特に配慮する必要があります。 ALLOW READ ACCESS オプションが指定されると、ロード・ユーティリティーは、索引が再作成中であっても引き続きそれらを照会に使用できるようにします。

ALLOW READ ACCESS モードのロード操作で INDEXING MODE INCREMENTAL オプションを指定する際には、ロード・ユーティリティーは、索引ツリーの整合性を保護するログ・レコードを作成します。書き込まれるログ・レコードの数は、挿入されるキーの数の一部であり、同様の SQL 挿入操作で必要とされる数よりずっと少ない数です。 INDEXING MODE INCREMENTAL オプションを指定した ALLOW NO ACCESS モードのロード操作は、通常のスペース割り振りログのほかには、小さなログ・レコードしか作成しません。

注: COPY YES を指定せず、*logindexrebuild* 構成パラメーターを ON に設定した場合にのみこのことは当てはまります。

ALLOW READ ACCESS モードのロード操作で INDEXING MODE REBUILD オプションを指定すると、元の索引と同じ表スペースか、または SYSTEM TEMPORARY 表スペースのいずれかで、新しい索引がシャドー として作成されます。元の索引は変更されずにロード操作で使用することができ、表は排他ロックされたままでロード操作の終わりに新規索引に置き換えられるだけです。ロード操作に失敗してトランザクションがロールバックされる場合でも、元の索引は変更されません。

デフォルトでは、シャドー索引は、元の索引と同じ表スペースに作成されます。元の索引と新規索引の両方が同時に保守されるため、同時に両方の索引を保留できる十分な表スペースがなければなりません。ロード操作が打ち切られると、新規索引の作成に使用される余分のスペースが解放されます。ロード操作がコミットされると、元の索引に使用されるスペースが解放され、新規索引が現行の索引になります。元の索引と同じ表スペースに新規索引が作成されると、元の索引の置換がほとんど同時に行われます。

SMS 表スペースで索引が作成される場合、 `.IN1` 接尾部および `.INX` 接尾部のある表スペース・ディレクトリーで索引ファイルを見ることができます。これらの接尾部は、どれが元の索引で、どれがシャドー索引であるかを示しません。ただし、DMS 表スペースで索引が作成される場合、新規のシャドー索引は不可視になります。

索引作成パフォーマンスの改善

SYSTEM TEMPORARY 表スペースでの新規索引の作成
元の表スペースでスペースが不足しないようにするために、新規索引を SYSTEM TEMPORARY 表スペースに作成することができます。 USE <tablespace-name> オプションを使用すると、 INDEXING MODE REBUILD および ALLOW READ ACCESS オプションを使用する際に、 SYSTEM TEMPORARY 表スペースで索引を再作成できます。システム一時表は SMS 表スペースまたは DMS 表スペースのどちらでもかまいませんが、 SYSTEM TEMPORARY 表スペースのページ・サイズは、元の索引表スペースのページ・サイズに一致しなければなりません。

ロード操作が ALLOW READ ACCESS モードでない場合、または索引付けモードに互換性がない場合には、 USE <tablespace-name> オプションは無視されます。 USE <tablespace-name> オプションは、 INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT オプションでのみサポートされます。 INDEXING MODE AUTOSELECT オプションが指定されており、ロード・ユーティリティーが索引の増分保守を選択する場合には、 USE <tablespace-name> は無視されます。

ロード再開操作では、元のロード操作で代替表スペースを使用しなかった場合でも、代替表スペースを使用して索引を作成できます。元のロード操作が ALLOW READ ACCESS モードで発行されなかった場合には、 ALLOW READ ACCESS モードでロード再開操作を発行することはできません。ロード終了操作では索引を再作成しないため、 USE <tablespace-name> は無視されます。

ロード操作の構築フェーズでは、 SYSTEM TEMPORARY 表スペースに索引が作成されます。その後、索引コピー・フェーズで、 SYSTEM TEMPORARY 表スペースから元の索引表スペースに索引がコピーされます。元の索引表スペースに新規索引用の十分なスペースがあることを確認するには、構築フェーズで元の表スペースにスペースを割り振らなければなりません。したがって、ロード操作で索引スペースが不足する場合には、構築フェーズでこれを実行します。この場合、元の索引が消失することはありません。

索引コピー・フェーズは、構築および削除フェーズの後に実行されます。索引コピー・フェーズが始まる前に、表が排他的にロックされます。つまり、索引コピー・フェーズ全体に渡って、読み取りアクセスは使用不可になります。索引コピー・フェーズは物理コピーであるため、表はかなりの期間使用できなくなります。

注: SYSTEM TEMPORARY 表スペースまたは索引表スペースのどちらかが DMS 表スペースである場合、 SYSTEM TEMPORARY 表スペースの読み取りにより、SYSTEM TEMPORARY 表スペースでのランダム入出力が発生し、そのために遅延が生じる可能性があります。索引表スペースへの書き込みはこれまでどおり最適化され、DISK_PARALLELISM 値が使用されます。

大規模な索引に関する考慮事項
ロード時の大規模な索引作成のパフォーマンスの改善には、*sortheap* データベース構成パラメーターの調整が役に立つことがあります。 *sortheap* は、ロード操作中に索引キーのソート処理専用に割り振るメモリーの大きさを指定します。例えば、キーのソート処理で索引ごとに 4000 ページの主メモリーを使用するようロード・ユ

ーティリティーに指示するには、*sortheap* を 4000 ページに設定し、データベースからすべてのアプリケーションを切り離した後、LOAD コマンドを発行します。

索引が大きすぎてメモリー内でソートできないと、ソート・スピルあるいはオーバーフローが起こります。つまり、データは、複数の「ソート実行」で分割され、後で組み合わせられる TEMPORARY 表スペースに保管されます。ソート・スピルが発生したかどうかの判別を行うには、sort_overflows モニター・エレメントを使用します。 *sortheap* パラメーターのサイズを大きくしてもソート・スピルを避けられない場合、 TEMPORARY 表スペースのバッファー・プールを十分大きくして、スピルが原因で生じるディスク入出力量を最小化することを確認してください。さらに、複数のソート実行の組み合わせにおいて入出力並列処理を実現するため、TEMPORARY 表スペースの宣言時に、それぞれが異なるディスク装置に存在する複数のコンテナーを指定することをお勧めします。ロード操作はすべてのキーをメモリー内に維持するため、表に複数の索引が定義されている場合は、それに比例してメモリーの消費量が増加します。

索引作成の据え置き

一般的に言えば、索引作成を据え置くよりも、REBUILD または INCREMENTAL モードのいずれかを指定することによりロード操作中に索引を作成する方が効率的です。 図 11 で示すように、多くの場合、表はデータのロード、索引の構築、統計の収集という 3 つのステップによって構築されます。このため、ロード操作中、索引の作成中 (表ごとに複数の索引が可能)、および統計の収集中 (表データと索引すべてに対する入出力が発生) に、複数のデータ入出力が発生することになります。別の方法として、ロード・ユーティリティーがデータを一括してこれらの作業を実行するようにすれば、さらに時間が短くなります。ただし、ユニーク索引がある場合、重複データが検出されるとロードのパフォーマンスは低下します。

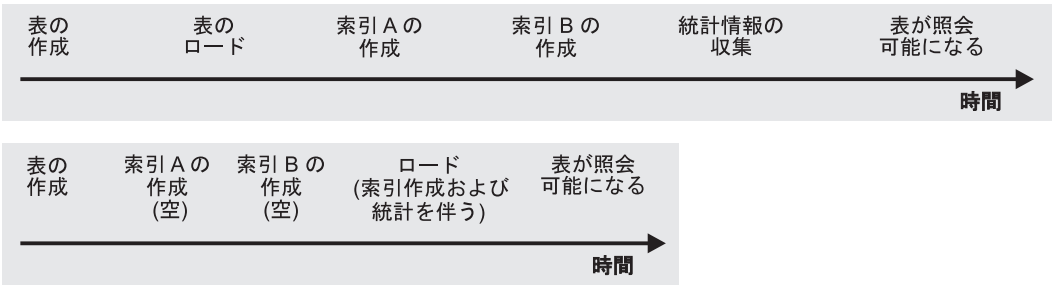


図 11. 索引作成と統計収集の並行処理によるロード・パフォーマンスの向上： 多くの場合、表はデータのロード、索引の構築、統計の収集という 3 つのステップによって構築されます。このため、ロード操作中、索引の作成中 (表ごとに複数の索引が可能)、および統計の収集中 (表データと索引すべてに対する入出力が発生) に、複数のデータ入出力が発生することになります。別の方法として、ロード・ユーティリティーがデータを一括してこれらの作業を実行するようにすれば、さらに時間が短くなります。

場合によっては、索引作成を据え置き、CREATE INDEX ステートメントを呼び出すことでパフォーマンスが改善されることもあります。索引再作成時のソートが使用するページの最大数は *sortheap* で指定されています。さらにスペースが必要な場合は、TEMP バッファー・プールが使用され、(最終的には) ディスクにスピルされます。ロードがスピルされ、結果としてパフォーマンスが低下する場合には、INDEXING MODE DEFERRED を使用して LOAD を実行し、索引を後で再作成し

たほうが賢明な場合があります。 CREATE INDEX は、一度に 1 つの索引を作成するので、表を何度もスキャンしてキーを収集するのに比べてメモリーの使用量を削減できます。

ロード操作と同時に索引を作成する代わりに、 CREATE INDEX ステートメントを使用して索引を作成する別の利点は、CREATE INDEX ステートメントでは、INTRA PARALLEL がオンになっているとキーのソートに複数のプロセス、つまりスレッドを使用できることにあります。実際の索引の構築は、並列で実行されません。

XML データのロード時の索引作成エラーの解決

索引付けエラーのために失敗するロード操作は、db2diag.log ログ・ファイルとインポート・ユーティリティーを一緒に使用して XML データの問題となる値を識別し、訂正することで解決できます。

ロード操作でエラー・メッセージ SQL20305N (sqlcode -20305) が返される場合、これは 1 つ以上の XML ノード値に索引付けできなかったことを示します。エラー・メッセージはエラーの理由コードを出力します。コマンド行プロセッサーに ? SQL20305N と入力して、対応する理由コードの説明とユーザー応答を検索します。

挿入操作中の索引付けの問題については、生成される XQuery ステートメントが db2diag.log ログ・ファイルに出力されて、文書内の失敗した XML ノード値を見つける上で役立ちます。失敗した XML ノード値を見つけるために XQuery ステートメントを使用する方法についての詳細は、『XML 索引付けの一般的な問題』を参照してください。

しかし、ロード操作中の索引付けの問題については、生成される XQuery ステートメントは db2diag.log ログ・ファイルに出力されません。これらの XQuery ステートメントを生成するには、ロードされなかった失敗した行に対してインポート・ユーティリティーを実行する必要があります。リジェクトされた行は表には存在しないため、XQuery ステートメントを失敗した文書に対して実行することはできません。この問題を解決するため、同じ定義を持つ新規表を、索引を付けずに作成する必要があります。その後、失敗した行を新規表にロードでき、さらに XQuery ステートメントを新規表に対して実行し、文書内の失敗した XML ノード値を見つけることができるようになります。

索引付けのエラーを解決するには、以下のステップを実行します。

1. 出力情報にあるレコード番号を使用して、ロード操作中にどの行がリジェクトされたかを調べます。
2. リジェクトされた行のみが含まれる .del ファイルを作成します。
3. 元の表 (T1) と同じ列を持つ新規表 (例えば T2) を作成します。新規表には索引を作成しないでください。
4. リジェクトされた行を新規表 T2 にロードします。
5. 元の表 T1 のそれぞれのリジェクトされた行について以下を行います。
   a. リジェクトされた行を T1 にインポートして、SQL20305N メッセージを受け取ります。エラーが最初に出されたところでインポートは停止します。
   b. db2diag.log ログ・ファイルを調べ、生成された XQuery ステートメントを見つけます。入力文書で失敗したノード値を検索するため、db2diag.log ログ・

ファイルでストリング「SQL20305N」を検索し、理由コード番号を突き合わせます。理由コードの後に一連の指示があり、続いてエラーの原因となった文書内の問題値を見つけるために使用できる、生成された XQuery ステートメントがあります。

c. 新規表 T2 を使用するように XQuery ステートメントを変更します。
d. T2 に対して XQuery ステートメントを実行し、文書内の問題値を見つけます。
e. 文書が含まれる .xml ファイルの中の問題値を修正します。
f. ステップ a に戻り、リジェクトされた行を再度 T1 にインポートします。インポートの停止の原因となった行は、今度は正常に挿入されるはずです。他にもリジェクトされた行が .del ファイルにある場合、インポート・ユーティリティーは次のエラーで停止し、別の SQL20305N メッセージが出力されます。インポートが正常に実行されるようになるまで、これらのステップを続けます。

例

以下の例では、索引 BirthdateIndex が *date* データ・タイプに対して作成されています。 REJECT INVALID VALUES オプションが指定されているため、*/Person/Confidential/Birthdate* の XML パターン値は *date* データ・タイプに対してすべて有効となる必要があります。このデータ・タイプにキャストできない XML パターン値がある場合、エラーが返されます。

以下の XML 文書を使用すると 5 つの行がロードされるはずですが、Birthdate 値を索引付けできないため、最初と 4 番目の行はリジェクトされます。ファイル person1.xml では、値 `March 16, 2002` が正しい日付形式ではありません。ファイル person4.xml では、値 `20000-12-09` の年にゼロが余分にあるため、有効な XML 日付値ではありますが、DB2 で許可される年 (0001 から 9999) の範囲外となっています。例を簡潔にするため、出力例の一部は編集されています。

ロードする 5 つの XML ファイルは以下のとおりです。

person1.xml (Birthdate 値は無効です)

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>March 16, 2002</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person>
```

person2.xml (Birthdate 値は有効です)

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
```

```
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>2002-03-16</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person>
```

person3.xml (Birthdate 値は有効です)

```
<?xml version="1.0"?>
<Person gender="Female">
  <Name>
    <Last>McCarthy</Last>
    <First>Laura</First>
  </Name>
  <Confidential>
    <Age unit="years">6</Age>
    <Birthdate>2001-03-12</Birthdate>
    <SS>444-55-6666</SS>
  </Confidential>
  <Address>5960 Daffodil Lane, San Jose, CA 95120</Address>
</Person>
```

person4.xml (Birthdate 値は無効です)

```
<?xml version="1.0"?>
<Person gender="Female">
  <Name>
    <Last>Wong</Last>
    <First>Teresa</First>
  </Name>
  <Confidential>
    <Age unit="years">7</Age>
    <Birthdate>20000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential>
  <Address>5960 Tulip Court, San Jose, CA 95120</Address>
</Person>
```

person5.xml (Birthdate 値は有効です)

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Smith</Last>
    <First>Chris</First>
  </Name>
  <Confidential>
    <Age unit="years">10</Age>
    <Birthdate>1997-04-23</Birthdate>
    <SS>666-77-8888</SS>
  </Confidential>
  <Address>5960 Dahlia Street, San Jose, CA 95120</Address>
</Person>
```

入力ファイル person.del には以下が含まれます。

```
1, <XDS FIL='person1.xml'/>
2, <XDS FIL='person2.xml'/>
3, <XDS FIL='person3.xml'/>
4, <XDS FIL='person4.xml'/>
5, <XDS FIL='person5.xml'/>
```

DDL および LOAD ステートメントは以下のとおりです。

```
CREATE TABLE T1 (docID INT, XMLDoc XML);

CREATE INDEX BirthdateIndex ON T1(xmlDoc)
   GENERATE KEY USING XMLPATTERN '/Person/Confidential/Birthdate' AS SQL DATE
   REJECT INVALID VALUES;

LOAD FROM person.del OF DEL INSERT INTO T1
```

上記の一連の XML ファイルのロード中に発生する索引付けエラーを解決するため、以下のステップを行います。

1. 出力情報にあるレコード番号を使用して、ロード操作中にどの行がリジェクトされたかを調べます。以下の出力では、レコード番号 1 およびレコード番号 4 はリジェクトされました。

   ```
   SQL20305N  An XML value cannot be inserted or updated because of an error
   detected when inserting or updating the index identified by "IID = 3" on table
   "LEECM.T1".  Reason code = "5".  For reason codes related to an XML schema the
   XML schema identifier = "*N" and XML schema data type = "*N".  SQLSTATE=23525

   SQL3185W  The previous error occurred while processing data from row "F0-1" of
   the input file.

   SQL20305N  An XML value cannot be inserted or updated because of an error
   detected when inserting or updating the index identified by "IID = 3" on table
   "LEECM.T1".  Reason code = "4".  For reason codes related to an XML schema the
   XML schema identifier = "*N" and XML schema data type = "*N".  SQLSTATE=23525

   SQL3185W  The previous error occurred while processing data from row "F0-4" of
   the input file.

   SQL3227W  Record token "F0-1" refers to user record number "1".

   SQL3227W  Record token "F0-4" refers to user record number "4".

   SQL3107W  There is at least one warning message in the message file.

   Number of rows read         = 5
   Number of rows skipped      = 0
   Number of rows loaded       = 3
   Number of rows rejected     = 2
   Number of rows deleted      = 0
   Number of rows committed    = 5
   ```

2. リジェクトされた行が含まれる新規ファイル reject.del を作成します。

   ```
   1, <XDS FIL='person1.xml'/>
   4, <XDS FIL='person4.xml'/>
   ```

3. 元の表 T1 と同じ列を持つ新規表 T2 を作成します。新規表には索引を作成しないでください。

   ```
   CREATE TABLE T2 LIKE T1
   ```

4. リジェクトされた行を新規表 T2 にロードします。

   ```
   LOAD FROM reject.del OF DEL INSERT INTO T2;
   ```

5. 元の表 T1 のリジェクトされた行 1 について以下を行います。

   a. リジェクトされた行を T1 にインポートして、-20305 メッセージを受け取ります。

      ```
      IMPORT FROM reject.del OF DEL INSERT INTO T1
      SQL3109N  The utility is beginning to load data from file "reject.del".

      SQL3306N  An SQL error "-20305" occurred while inserting a row into the
                table.
      ```

```
SQL20305N  An XML value cannot be inserted or updated because of an error
detected when inserting or updating the index identified by "IID = 3" on
table "LEECM.T1".  Reason code = "5".  For reason codes related to an XML
schema the XML schema identifier = "*N" and XML schema data type = "*N".
SQLSTATE=23525

SQL3110N  The utility has completed processing.  "1" rows were read from
the input file.
```

b. db2diag.log ログ・ファイルを調べ、生成された XQuery ステートメントを見つけます。

```
FUNCTION: DB2 UDB, Xml Storage and Index Manager, xmlsDumpXQuery, probe:608
DATA #1 : String, 36 bytes
SQL Code: SQL20305N ; Reason Code: 5
DATA #2 : String, 265 bytes
To locate the value in the document that caused the error, create a
table with one XML column and insert the failing document in the table.
Replace the table and column name in the query below with the created
table and column name and execute the following XQuery.
DATA #3 : String, 247 bytes
xquery for $i in db2-fn:xmlcolumn(
  "LEECM.T1.XMLDOC")[/*:Person/*:Confidential/*:Birthdate="March 16, 2002"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

c. 新規表 T2 を使用するように XQuery ステートメントを変更します。

```
xquery for $i in db2-fn:xmlcolumn(
  "LEECM.T2.XMLDOC")[/*:Person/*:Confidential/*:Birthdate="March 16, 2002"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

d. 表 T2 に対して XQuery ステートメントを実行し、文書内の問題値を見つけます。

```
<Result><ProblemDocument><Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
    <Age unit="years">5</Age>
    <Birthdate>March 16, 2002</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person></ProblemDocument><ProblemValue><Confidential>
    <Age unit="years">5</Age>
    <Birthdate>March 16, 2002</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential></ProblemValue></Result>
```

e. 文書が含まれるファイル person1.xml の中の問題値を修正します。 March 16, 2002 は正しい日付形式ではないため、2002-03-16 に変更されます。

```
<?xml version="1.0"?>
<Person gender="Male">
  <Name>
    <Last>Cool</Last>
    <First>Joe</First>
  </Name>
  <Confidential>
```

```
    <Age unit="years">5</Age>
    <Birthdate>2002-03-16</Birthdate>
    <SS>111-22-3333</SS>
  </Confidential>
  <Address>5224 Rose St. San Jose, CA 95123</Address>
</Person>
```

f. ステップ a. に戻り、リジェクトされた行を再度表 T1 にインポートします。

6. (ステップ 5 の最初の繰り返し)

a. リジェクトされた行を表 T1 にインポートします。インポート・ファイルから 2 つの行が読み取られたので、最初の行は正常にインポートされます。新しいエラーが 2 番目の行で発生します。

```
IMPORT FROM reject.del OF DEL INSERT INTO T1
SQL3109N  The utility is beginning to load data from file "reject.del".

SQL3306N  An SQL error "-20305" occurred while inserting a row into the
          table.

SQL20305N  An XML value cannot be inserted or updated because of an error
detected when inserting or updating the index identified by "IID = 3" on
table "LEECM.T1".  Reason code = "4".  For reason codes related to an XML
schema the XML schema identifier = "*N" and XML schema data type = "*N".
SQLSTATE=23525

SQL3110N  The utility has completed processing.  "2" rows were read from
the input file.
```

b. db2diag.log ログ・ファイルを調べ、生成された XQuery ステートメントを見つけます。

```
FUNCTION: DB2 UDB, Xml Storage and Index Manager, xmlsDumpXQuery, probe:608
DATA #1 : String, 36 bytes
SQL Code: SQL20305N ; Reason Code: 4
DATA #2 : String, 265 bytes
To locate the value in the document that caused the error, create a
table with one XML column and insert the failing document in the table.
Replace the table and column name in the query below with the created
table and column name and execute the following XQuery.
DATA #3 : String, 244 bytes
xquery for $i in db2-fn:xmlcolumn("LEECM.T1.XMLDOC")
   [/*:Person/*:Confidential/*:Birthdate="20000-12-09"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

c. 表 T2 を使用するように XQuery ステートメントを変更します。

```
xquery for $i in db2-fn:xmlcolumn("LEECM.T2.XMLDOC")
   [/*:Person/*:Confidential/*:Birthdate="20000-12-09"]
return
<Result>
 <ProblemDocument> {$i} </ProblemDocument>
 <ProblemValue>{$i/*:Person/*:Confidential/*:Birthdate/..} </ProblemValue>
</Result>;
```

d. XQuery ステートメントを実行し、文書内の問題値を見つけます。

```
<Result><ProblemDocument><Person gender="Female">
  <Name>
    <Last>Wong</Last>
    <First>Teresa</First>
  </Name>
  <Confidential>
    <Age unit="years">7</Age>
```

```
    <Birthdate>20000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential>
  <Address>5960 Tulip Court, San Jose, CA 95120</Address>
</Person></ProblemDocument><ProblemValue><Confidential>
    <Age unit="years">7</Age>
    <Birthdate>20000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential></ProblemValue></Result>
```

e. 文書が含まれるファイル person4.xml の中の問題値を修正します。値 20000-12-09 の年にゼロが余分にあるため、DB2 で許可される年 (0001 から 9999) の範囲外となっています。値が 2000-12-09 に変更されます。

```
<?xml version="1.0"?>
<Person gender="Female">
  <Name>
    <Last>Wong</Last>
    <First>Teresa</First>
  </Name>
  <Confidential>
    <Age unit="years">7</Age>
    <Birthdate>2000-12-09</Birthdate>
    <SS>555-66-7777</SS>
  </Confidential>
  <Address>5960 Tulip Court, San Jose, CA 95120</Address>
</Person>
```

f. ステップ a に戻り、リジェクトされた行を再度 T1 にインポートします。

7. (ステップ 5 の 2 回目の繰り返し)

a. リジェクトされた行を T1 にインポートします。

```
IMPORT FROM reject.del OF DEL INSERT INTO T1
SQL3109N  The utility is beginning to load data from file "reject.del".

SQL3110N  The utility has completed processing.  "2" rows were read from
the input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "2".

SQL3222W  ...COMMIT of any database changes was successful.

SQL3149N  "2" rows were processed from the input file.  "2" rows were
successfully inserted into the table.  "0" rows were rejected.

Number of rows read         = 2
Number of rows skipped      = 0
Number of rows inserted     = 2
Number of rows updated      = 0
Number of rows rejected     = 0
Number of rows committed    = 2
```

これで問題は解決されました。person.del のすべての行は、正常に表 T1 に挿入されます。

ロード操作時のコンプレッション・ディクショナリーの作成

一定の基準を満たすロード INSERT およびロード REPLACE 操作は、ディクショナリー自動作成 (ADC) のトリガーとなります。十分なデータが処理された後、COMPRESS 属性が有効になっており、かつコンプレッション・ディクショナリーを持たない表に対してロードが実行されると ADC が発生します。

データ行圧縮は、静的なディクショナリー・ベースの圧縮アルゴリズムを使用して、データを圧縮します。しかし、圧縮を行うにはまずディクショナリーが表に存在していなければなりません。ロード操作時のデフォルトの動作では (KEEPDICTIONARY オプションによって示される)、既存のディクショナリーを使用するか、ディクショナリーが存在しない場合はデータの特定のしきい値 (約 1 MB) をスキャンしてからディクショナリーを生成します。ロード・ユーティリティーは、ターゲット表に存在するデータを使用してディクショナリーを作成します (このデータはその表に保管されるデータの種類を代表するものであると想定)。ターゲット表の既存のデータが不足している場合、ロード・ユーティリティーはまず十分な入力データをサンプリングし、その後入力データと既存のデータの両方を使用してディクショナリーを作成します。

範囲パーティション表で ADC が発生すると、パーティションはそれぞれ個々の表のように扱われます。ディクショナリーがパーティション間で共有されることはありません。 ADC はすでにディクショナリーが存在するパーティションでは発生せず、各パーティションで生成されるディクショナリーはそのパーティションの既存のデータにのみ (必要な場合はロードされたデータも可) 基づきます。

INSERT モードで実行されるロードはすべて暗黙的に KEEPDICTIONARY の動作に従います。しかし、ロード REPLACE 操作に関しては、RESETDICTIONARY オプションというもう 1 つの選択肢があります。

KEEPDICTIONARY オプションを使用したロード REPLACE
KEEPDICTIONARY オプションを使用するロード REPLACE は既存のディクショナリーを維持し、ターゲット表の COMPRESS 属性が有効になっている限り、そのディクショナリーを使用して、ロードされたデータを圧縮します。ディクショナリーが存在しない場合、ロード・ユーティリティーは COMPRESS 属性が有効になっている表に対して新規のディクショナリーを生成します (ただし、表にロードされているデータがしきい値の 1 MB を超える場合に限ります)。ターゲット表のデータは置き換えられるので、ロード・ユーティリティーは入力データのみを使ってディクショナリーを作成します。ディクショナリーの作成後、そのディクショナリーは表に挿入され、ロード操作が継続します。

RESETDICTIONARY オプションを使用したロード REPLACE
COMPRESS 属性がオンになっている表にロードする際に RESETDICTIONARY オプションを使用することには 2 つの重要な含意があります。まず、LOAD REPLACE が完了した後で、ターゲット表に少しでもデータが存在していれば、ADC が発生します。つまり、1 行でもデータが存在していれば、それに基づいて新規のコンプレッション・ディクショナリーを作成できるということです。もう 1 つは、次のいずれかの状況が存在する場合、既存のディクショナリーは置換ではなく削除される (ターゲット表はコンプレッション・ディクショナリーを失うことになる) という含意です。

- COMPRESS 属性がオフになっている表に対して操作が実行される場合。
- 何もロードされない場合 (ゼロ行)。この場合、通知ログに ADM5591W が出力されます。

注: RESETDICTIONARY オプションを指定した LOAD REPLACE 後に LOAD TERMINATE 操作を実行する場合、既存のコンプレッション・ディクショナリーは置換ではなく削除されます。

パフォーマンスへの影響

ADC がロード操作のパフォーマンスに影響を与える要因として次のものがあります。

- 表データの初期スキャン
  LOAD INSERT では、コンプレッション・ディクショナリーを作成する前に、ADC の最小しきい値だけでなく既存の表データすべてをスキャンします。そのため、表のサイズが大きければ大きいほどこのスキャンに時間がかかります。
- コンプレッション・ディクショナリーを作成するための追加処理
  ディクショナリーの作成に実際にかかる時間は最小時間です。さらに、いったんディクショナリーが作成されると、ADC はデフォルトでオフになります。

ロード・パフォーマンスを改善するためのオプション

ロード・パフォーマンスを最適化するために使用できる各種コマンド・パラメーターがあります。さらに、ロード固有のファイル・タイプ修飾子もいくつかあり、それらは時としてユーティリティーのパフォーマンスを著しく向上させる場合があります。

コマンド・パラメーター

DISK_PARALLELISM、CPU_PARALLELISM、および DATA BUFFER の各パラメーターがユーザーによって指定されていない場合、ロード・ユーティリティーはこれらのパラメーターの最適な値を決定することにより、パフォーマンスを最大にするよう試みます。最適化は、ユーティリティー・ヒープにおける使用可能なサイズとフリー・スペースに基づいてなされます。これらのパラメーターを特定の必要に合わせて調整しようとする前に、オートノミック DISK_PARALLELISM および CPU_PARALLELISM 設定の使用を考慮してください。

以下に、ロード・ユーティリティーで利用可能な各種オプションとパフォーマンスとの関係を説明します。

ALLOW READ ACCESS

このオプションを使用すると、ロード操作の進行中に表を照会することができます。ロード操作前に表に存在していたデータのみ表示できます。
INDEXING MODE INCREMENTAL オプションも指定されているときに、ロード操作が失敗した場合には、後続のロード終了操作で、索引での矛盾を訂正する必要があります。これを行うには、相当量の入出力を伴う索引スキャンが必要になります。ロード終了操作に ALLOW READ ACCESS オプションも指定されている場合には、入出力にバッファー・プールが使用されます。

COPY YES または NO

このパラメーターは、ロード操作中に入力データのコピーを作成するかどうかを指定するのに使用します。 COPY YES (順方向リカバリーが有効になっている場合のみ適用可能) を指定すると、ロードするデータはすべてロード操作中にコピーされるため、ロードのパフォーマンスは低下します。入出力活動が増加すると、入出力制約のシステムにおけるロード時間が大きくなることがあります。複数の装置やディレクトリー (異なるディスク上にある) を指定することにより、この操作によって生じるパフォーマンス上の不利をいくらか相殺できます。 COPY NO (順方向リカバリーが有効になっている

場合のみ適用可能) を指定すると、ロード・パフォーマンスに影響はありません。ただし、ロードされる表に関連した表スペースはすべて「バックアップ・ペンディング」状態になり、それらの表スペースをバックアップしてからでないと表にアクセスできなくなります。

CPU_PARALLELISM

このパラメーターは、データベース・パーティションごとに実行されるプロセスの数を利用して (マシンが対応している場合) ロードのパフォーマンスを大幅に向上させたい場合に使用します。このパラメーターには、ロード・ユーティリティーがデータ・レコードの構文解析、変換、およびフォーマット設定に使用するプロセス数またはスレッド数を指定します。指定可能な最大数は 30 です。指定された値をサポートするためのメモリーが十分でない場合、ユーティリティーはこの値を調整します。このパラメーターを指定しない場合、ロード・ユーティリティーはシステムの CPU 数に基づくデフォルト値を選択します。

パラメーターの値に関係なく、ソース・データ内のレコードの順序は保持されます (図 12を参照)。ただし、次の条件を満たしている必要があります。

- `anyorder` ファイル・タイプ修飾子が指定されていない
- PARTITIONING_DBPARTNUMS オプション (パーティション化で複数のパーティションが使用される) が指定されていない

表に LOB または LONG VARCHAR のいずれかのデータが収められていると、CPU_PARALLELISM は 1 に設定されます。この場合、並列処理はサポートされません。

このパラメーターの使用は対称マルチプロセッサー (SMP) ハードウェアに限定されていませんが、非 SMP 環境でこのパラメーターを使っても、明確なパフォーマンスの向上は期待できません。

図 12. ロード操作中に、データベース・パーティションごとに実行されるプロセスの数を利用した場合にソース・データのレコード順序は保持される

DATA BUFFER

DATA BUFFER パラメーターは、ロード・ユーティリティーにバッファーとして割り当てるメモリーの合計を指定します (4 KB 単位)。このバッファーは、サイズに応じていくつかのエクステント にしておくことをお勧めします。データ・バッファーはユーティリティー・ヒープから割り当てられます。 DB2 ユーティリティーが使用するメモリーは、システム上の利用可能なストレージの大きさに応じてなるべく大きくするようにしてください。それに応じて、データベース構成パラメーター *util_heap_sz* (ユーティリティー・ヒープ・サイズ) を変更できます。 *util_heap_sz* のデフォルト値は 5,000 (4 KB) ページです。ロードはユーティリティー・ヒープからのメモリーを利用するいくつかのユーティリティーの 1 つにすぎないため、ロード・ユーティリティーから利用可能なページ数としてこのパラメーターで定義する数はなるべく 50% を超えないようにし、ユーティリティー・ヒープにも十分な大きさを定義するようにしてください。

DISK_PARALLELISM

DISK_PARALLELISM パラメーターは、ロード・ユーティリティーがデータ・レコードをディスクに書き込むのに使用するプロセス数またはスレッド数を指定します。このパラメーターは、データのロード時に使用可能なコンテナーを利用してロードのパフォーマンスを大幅に向上させたい場合に使用します。指定可能な最大数は、CPU_PARALLELISM 値 (ロード・ユーティリティーが実際に使用している値) の 4 倍か 50 のいずれか大きい方です。デフォルトでは、 DISK_PARALLELISM はロードする表のオブジェクトを備えたすべての表スペース上にある表スペース・コンテナーの合計数と等しい値です (この値が指定可能な最大数を超えていない場合)。

NONRECOVERABLE

順方向リカバリーが有効になっており、ロールフォワード時に表に対するロード・トランザクションをリカバリーする必要がない場合にこのパラメーターを使用します。 NONRECOVERABLE ロードのパフォーマンスと COPY NO ロードのパフォーマンスは同じです。しかし、データ損失の可能性に関しては大きな違いがあります。 NONRECOVERABLE ロードは、表に完全にアクセスできるようにする一方で、その表をロールフォワード・リカバリー不可能としてマークします。これによって問題が生じる場合があります。ロード操作によってロールフォワードする必要がある場合、ロードされたデータおよび表に対するそれ以降の更新がすべて失われる可能性があります。COPY NO ロードは、従属表スペースをすべて「バックアップ・ペンディング」状態にして、バックアップが実行されるまで表にアクセスできないようにします。その種のロードの後にはバックアップが強制されるため、ロードされたデータまたは表に対してそれ以降行われる更新が失われる危険はありません。つまり、COPY NO ロードを使用すると完全なリカバリーが行えます。

注: これ以後のリストアやロールフォワード・リカバリー操作の間にそれらのロード・トランザクションが検出された場合、表は更新されずに `invalid` としてマークされます。それ以降、この表に対する処理は無視されます。ロールフォワード操作の完了後は表のドロップのみが可能です。

SAVECOUNT

このパラメーターは、ロード操作の**ロード・フェーズ中** に整合点を確立するインターバルを設定するのに使用します。整合点を確立するために実行される活動を同期化するには時間がかかります。これをあまりに頻繁に実行すると、ロードのパフォーマンスがかなり低下します。大量の行をロードすることになっている場合は、SAVECOUNT 値を大きく指定することをお勧めします (例えば 1 億個のレコードが関係するロード操作の場合は値 10,000,000 など)。

ロード再開操作は、最後の整合点から自動的に続行します。ただし、ロード・フェーズから再開する場合に限ります。

STATISTICS USE PROFILE

表統計プロファイルで指定した統計を収集します。このパラメーターを使用すると、ロード操作の完了後に RUNSTATS ユーティリティーを呼び出す場合よりも効率よくデータ分散と索引統計情報を収集することができます。ただし、ロード操作そのもののパフォーマンスは低下します (特に DETAILED INDEXES ALL を指定した場合)。

最適なパフォーマンスを得るために、アプリケーションには入手可能な最大限のデータ分散と索引統計情報が必要です。統計情報が更新されると、アプリケーションではその最新の統計情報に基づいて表データへの新しいアクセス・パスを使用できます。表への新しいアクセス・パスは、BIND コマンドを使ってアプリケーション・パッケージを再バインドすることにより作成できます。表統計プロファイルは、SET PROFILE オプションを指定して RUNSTATS コマンドを実行することによって作成されます。

データを大規模な表にロードする場合は、*stat_heap_sz* (統計ヒープのサイズ) データベース構成パラメーターに指定する値をさらに大きくすることをお勧めします。

USE <tablespace-name>

ALLOW READ ACCESS ロードが行われており、索引付けモードが REBUILD である場合にこのパラメーターを使用すると、 SYSTEM TEMPORARY 表スペースで索引を再構築し、ロード操作の索引コピー・フェーズで索引表スペースにコピーし直すことができます。

デフォルトでは、完全に再構築された索引 (*シャドー索引とも呼ばれる*) は、元の索引と同じ表スペースに作成されます。この場合、元の索引とシャドー索引の両方が同じ表スペースに同時に置かれているため、リソース問題の原因となる可能性があります。シャドー索引が元の索引と同じ表スペースに作成される場合には、元の索引は即座にシャドーに置き換えられます。ただし、SYSTEM TEMPORARY 表スペースにシャドー索引が作成される場合には、ロード操作で索引コピー・フェーズが必要になります。このフェーズでは、SYSTEM TEMPORARY 表スペースから索引表スペースに索引をコピーします。このコピーに関連した入出力はかなり大きくなります。表スペースのいずれかが DMS 表スペースである場合、 SYSTEM TEMPORARY 表スペースの入出力の順序が変わる可能性があります。
DISK_PARALLELISM オプションにより指定される値は、索引コピー・フェーズで優先されます。

WARNINGCOUNT

このパラメーターには、ロード操作を強制終了するまでにユーティリティーが戻すことのできる警告の数の限界値を指定します。警告が少ししかない、あるいはまったくないことが予想される場合、 WARNINGCOUNT パラメーターを比較的低い数値に設定してください。 WARNINGCOUNT の数に達すると、ロード操作は停止します。これを指定することにより、ロード操作を完了する前に問題を訂正することが可能になります。

ファイル・タイプ修飾子

ANYORDER

デフォルトでは、ロード・ユーティリティーはソース・データのレコード順序を保持します。 SMP 環境でロードが行われるときにその順序を保持するには、並列処理間で同期する必要があります。

SMP 環境で `anyorder` ファイル・タイプ修飾子を指定すると、順序を保持しないという指示がロード・ユーティリティーに与えられます。これにより、その順序を保持するために必要な同期をしないですむため、効率が上がります。しかし、ロード

するデータがあらかじめソートされている場合、anyorder を指定するとその順序が崩れてしまい、あらかじめソートしておくことによるメリットがそれ以降の照会で失われてしまいます。

注: CPU_PARALLELISM が 1 の場合、anyorder ファイル・タイプ修飾子は何の影響も及ぼしません。また、この修飾子には SAVECOUNT オプションとの互換性がありません。

BINARYNUMERICS、ZONEDDECIMAL、および PACKEDDECIMAL
固定長の区切りなし ASCII (ASC) ソース・データの場合、数値データをバイナリーで表現するとロード時のパフォーマンスが向上します。 packeddecimal ファイル・タイプ修飾子が指定される場合、ロード・ユーティリティーは 10 進データをパック 10 進数フォーマット (バイトにつき 2 桁) として解釈します。 zoneddecimal ファイル・タイプ修飾子が指定される場合、ロード・ユーティリティーは 10 進データをゾーン 10 進フォーマット (バイトにつき 1 桁) として解釈します。それ以外の数値タイプの場合はすべて、binarynumerics ファイル・タイプ修飾子が指定されると、ロード・ユーティリティーはデータをバイナリー・フォーマットとして解釈します。

注:

- binarynumerics、packeddecimal、または zoneddecimal ファイル・タイプ修飾子が指定されると、プラットフォームに関係なく、数値データはビッグ・エンディアン (最上位のバイトから記録/送信する) フォーマットとして解釈されます。
- packeddecimal ファイル・タイプ修飾子と zoneddecimal ファイル・タイプ修飾子を同時に指定することはできません。
- packeddecimal および zoneddecimal ファイル・タイプ修飾子は 10 進ターゲット列に対してのみ適用され、バイナリー・データはターゲット列の定義と一致しなければなりません。
- binarynumerics、packeddecimal、または zoneddecimal ファイル・タイプ修飾子を指定するときは、reclen ファイル・タイプ修飾子も指定する必要があります。

FASTPARSE
注意して使用してください。ロードされているデータが有効であることが明確な場合は、注意が必要なデータのロードを行う場合に比べ、徹底した構文検査をロードの際に実行する必要はない可能性があります。実際、このステップの有効範囲を狭めることによって、約 10% から 20% ほどロードのパフォーマンスを向上させることができます。これは fastparse ファイル・タイプ修飾子を使用することによって行えます。この修飾子は ASC および DEL ファイルのユーザー指定の列値に対して実行されるデータ・チェックを簡略化します。

NOROWWARNINGS
ロード操作中にリジェクトされた行に関する警告メッセージは、指定されたファイルに書き込まれます。しかし、リジェクトされたレコード、無効なレコード、または切り捨てられたレコードをロード・ユーティリティーが大量に処理する必要がある場合は、ロードのパフォーマンスに対してマイナスの影響を与える可能性があります。多数の警告が予期されるような場合は、norowwarnings ファイル・タイプ修飾子を使用してこれらの警告の記録を抑制すると良いでしょう。

PAGEFREESPACE、**INDEXFREESPACE**、および **TOTALFREESPACE**
データを表に挿入し、更新していくうちに、表や索引の再編成が必要になります。1 つの解決策は、pagefreespace、indexfreespace、および totalfreespace を使って表および索引のフリー・スペースの量を増やすことです。最初の 2 つの修飾子は PCTFREE の値に優先し、フリー・スペースとして残されるデータおよび索引ページのパーセンテージを指定します。一方、totalfreespace はフリー・スペースとして表に追加される総ページ数のパーセンテージを指定します。

参照整合性を維持するためのロードのフィーチャー

一般的に、ロード・ユーティリティーはインポート・ユーティリティーより効率的なユーティリティーですが、ロードされる情報の参照整合性を保つためにさまざまなフィーチャーを必要とします。

- **表ロック**。これは並行性制御を提供し、ロード操作中、データ・アクセスが無制限に行われないようにします。
- **表の状態** および **表スペースの状態**。これを使用すると、データへのアクセスを制御するか、または特定のユーザー処置を引き出すことができます。
- **ロード例外表**。知らずに無効データの行が削除されることがないようにします。

ロード操作に続く整合性違反のチェック

以下のいずれかの状態が存在する場合は、ロード操作の後に、その表が READ または NO ACCESS モードで SET INTEGRITY ペンディング状態になっていることがあります。

- 表に表チェック制約または参照整合性制約が定義されている場合。
- この表に生成列があり、 V7 以前のクライアントを使用してロード操作が開始された場合。
- この表に従属する IMMEDIATE 指定のマテリアライズ照会表またはこの表を参照する IMMEDIATE 指定のステージング表がある場合。
- 表がステージング表またはマテリアライズ照会表の場合。

ロードした表の SET INTEGRITY ペンディング状態は、その表に対応する SYSCAT.TABLES 項目の STATUS フラグに示されます。 STATUS の値が N で、また ACCESS MODE の値が F である場合に、ロードした表が完全に使用可能となります。これは表が完全にアクセス可能であり、通常状態であることを示します。

ロードされる表に従属表がある場合には、SET INTEGRITY PENDING CASCADE パラメーターを指定して、ロードされる表の SET INTEGRITY ペンディング状態が即時に従属表にカスケードされるようにするかどうかを指示できます。

ロードされる表に、従属外部キー表、従属マテリアライズ照会表、および従属ステージング表と共に制約があるときに、すべての表がロード操作前に通常状態である場合には、指定されるロード・パラメーターに基づいて、以下のような結果になります。

INSERT、ALLOW READ ACCESS、および SET INTEGRITY PENDING CASCADE IMMEDIATE

ロードされる表、その従属マテリアライズ照会表、および従属ステージング表は、読み取りアクセスを持つ SET INTEGRITY ペンディング状態になります。

INSERT、ALLOW READ ACCESS、および SET INTEGRITY PENDING CASCADE DEFERRED

ロードされる表だけが読み取りアクセスを持つ SET INTEGRITY ペンディング状態に置かれます。従属外部キー表、従属マテリアライズ照会表、および従属ステージング表は元の状態のままです。

INSERT、ALLOW NO ACCESS、および SET INTEGRITY PENDING CASCADE IMMEDIATE

ロードされる表、その従属マテリアライズ照会表、および従属ステージング表は、アクセスを持たない SET INTEGRITY ペンディング状態になります。

INSERT または REPLACE、ALLOW NO ACCESS、および SET INTEGRITY PENDING CASCADE DEFERRED

ロードされる表だけが、アクセスを持たない SET INTEGRITY ペンディング状態になります。従属外部キー表、IMMEDIATE 指定の従属マテリアライズ照会表、および IMMEDIATE 指定の従属ステージング表は元の状態のままです。

REPLACE、ALLOW NO ACCESS、および SET INTEGRITY PENDING CASCADE IMMEDIATE

表およびそのすべての従属外部キー表、IMMEDIATE 指定の従属マテリアライズ照会表、および IMMEDIATE 指定の従属ステージング表は、アクセスを持たない SET INTEGRITY ペンディング状態になります。

注: ロード置換操作で ALLOW READ ACCESS オプションを指定すると、エラーが発生します。

SET INTEGRITY ペンディング状態を除去するには、SET INTEGRITY ステートメントを使用します。 SET INTEGRITY ステートメントは表をチェックして制約違反がないかどうかを調べ、その表の SET INTEGRITY ペンディング状態を終了します。すべてのロード操作が INSERT モードで実行される場合、SET INTEGRITY ステートメントを使用して制約を増分的に処理します (つまり表のうち追加された部分だけをチェックして、制約違反がないかどうかを調べます)。以下に例を示します。

```
db2 load from infile1.ixf of ixf insert into table1
db2 set integrity for table1 immediate checked
```

制約違反がないかどうかをチェックするのは TABLE1 のうち追加部分だけです。追加部分だけをチェックして制約違反がないかどうかを調べることにより、表全体をチェックするよりも時間が短くてすみます。これは、大きな表にデータを少しだけ追加した場合に特に有効です。

SET INTEGRITY PENDING CASCADE DEFERRED オプションを指定して表がロードされるときに、 SET INTEGRITY ステートメントを使用して整合性違反をチェ

ックする場合には、従属表はアクセスを持たない SET INTEGRITY ペンディング状態になります。表をこの状態から解除する場合には、明示的要求を発行する必要があります。

従属マテリアライズ照会表または従属ステージング表を持つ表が INSERT オプションを使用してロードされるときに、SET INTEGRITY ステートメントを使用して整合性違反をチェックする場合には、表は SET INTEGRITY ペンディング状態ではなくなり、No Data Movement モードに入れられます。これは、従属マテリアライズ照会表の後続の増分リフレッシュ、および従属ステージング表の増分伝搬を容易にするために実行されます。 No Data Movement 状態では、表内の行の移動の原因となるような操作は許可されません。

No Data Movement 状態は、SET INTEGRITY ステートメントを発行する際に FULL ACCESS オプションを指定することによりオーバーライドできます。表は完全にアクセス可能になりますが、従属マテリアライズ照会表の完全再計算が後続の REFRESH TABLE ステートメントで実行され、従属ステージング表は不完全な状態になります。

ロード操作で ALLOW READ ACCESS オプションが指定されている場合には、SET INTEGRITY ステートメントを使用して制約違反がチェックされるまで、表は読み取りアクセス状態のままです。ロード操作がコミットされたら、アプリケーションは表で、ロード操作前に存在したデータを照会できますが、SET INTEGRITY ステートメントが発行されるまでは、新しくロードされたデータを表示することはできません。

制約違反をチェックする前に、いくつかのロード操作を表で実行できます。ALLOW READ ACCESS モードですべてのロード操作が完了した場合には、最初のロード操作の前に表に存在していたデータだけが照会に使用できます。

表が 1 つでも複数でも、このステートメントを 1 回呼び出すだけでチェックできます。従属表を独自にチェックする場合、その親表が SET INTEGRITY ペンディング状態になっていてはなりません。そうしないと、親表と従属表の両方を同時にチェックしなければならなくなります。参照整合性が循環している場合、その循環に関係しているすべての表を 1 回の SET INTEGRITY ステートメント呼び出しに組み込む必要があります。従属表をロードしている間に、親表に制約違反がないかどうかをチェックするのがよいかもしれません。ただしこれが可能なのは、2 つの表が同じ表スペースにない場合だけです。

SET INTEGRITY ステートメントの発行時に INCREMENTAL オプションを指定すると、増分処理を明示的に要求することができます。しかし、ほとんどの場合 DB2 データベースは増分処理を選択するため、このオプションは不要です。増分処理を実行できない場合には、自動的に全処理が実行されます。 INCREMENTAL オプションを指定したにもかかわらず増分処理を実行できない場合、以下に示す条件が成立するならエラーが戻されます。

- 表が SET INTEGRITY ペンディング状態の間に、この表に新しく制約が追加される場合。
- 表に対する最後の整合性チェックの後で、ロード置換操作が行われるか、または NOT LOGGED INITIALLY WITH EMPTY TABLE オプションが活動化される場合。

- 親表が、ロード置換されるか、または増分ではない方法で整合性チェックされる場合。
- 表が、マイグレーション前の SET INTEGRITY ペンディング状態にある場合。マイグレーション後に最初に表が整合性チェックされるときは、完全処理が必要。
- 表またはその親表の入った表スペースがある時点でロールフォワードされ、表とその親が異なる表スペースに配置される場合。

表で SYSCAT.TABLES カタログの CONST_CHECKED 列に 1 つまたは複数の W 値があるときに、SET INTEGRITY ステートメントに NOT INCREMENTAL オプションが指定されていない場合、表は増分処理され、SYSCAT.TABLES の CONST_CHECKED 列には、すべてのデータをシステムがチェックしたわけではないことを示す U というマークが付けられます。

SET INTEGRITY ステートメントは、制約違反のある行を削除した結果として DELETE トリガーを起動することはありませんが、表が SET INTEGRITY ペンディング状態ではなくなるとトリガーが起動されます。そのため、例外表のデータを収集して、ロードした表に例外表の行を挿入すると、その表に定義されている INSERT トリガーが起動されます。これについては考慮が必要です。 1 つの選択肢は、INSERT トリガーをいったんドロップしてから例外表から行を挿入し、その後で INSERT トリガーを再作成することです。

SET INTEGRITY を使用した制約違反の検査

一般的に、表の整合性処理を手動で行う必要があるのは、次の 3 つの場合です。表にデータをロードした後、表に制約を追加して表を変更する場合、そして表を変更して生成された列を追加する場合の 3 つです。

- 表の制約のチェックおよび表上で整合性処理の実行をオンにするには、以下のいずれかを実行する必要があります。
  - SYSADM または DBADM 権限
  - チェックする表に対する CONTROL 権限、および、例外が 1 つ以上の表にポストされる場合は例外表に対する 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置かれるすべての直下の外部キー表上、直下の IMMEDIATE で定義されるマテリアライズ照会表上、および直下の IMMEDIATE で定義されるステージング表上の CONTROL 特権。
  - LOAD 権限、および例外が 1 つ以上の表にポストされる場合は、次のものが必要です。
    - チェックする各表に対する SELECT および DELETE 特権
    - 例外表に対する INSERT 特権
- 表上で整合性処理を実行せずに表の制約のチェックをオンにするには、以下のいずれかを実行する必要があります。
  - SYSADM または DBADM 権限
  - チェックする表に対する 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置かれる個々の直下の外部キー表上、直下の IMMEDIATE で定義されるマテリアライズ照会表上、および直下の IMMEDIATE で定義されるステージング表上の CONTROL 特権。

- 制約のチェック、即時リフレッシュ、または表に対する即時伝搬をオフにするには、以下のいずれかを実行する必要があります。
  - SYSADM または DBADM 権限
  - 表上、すべての子孫外部キー表上、直下のマテリアライズ照会表上、およびステートメントによって整合性チェックをオフにする直接のステージング表上の CONTROL 特権。
  - LOAD 権限

表に制約が定義されていたり、従属外部キー表、従属マテリアライズ照会表、従属ステージング表がある場合には、ロード操作により、表が自動的に SET INTEGRITY ペンディング状態になります。ロード操作が完了すると、ロードされたデータの整合性をチェックしたり、表の制約のチェックをオンにすることができます。 表に従属外部キー表、従属マテリアライズ照会表、従属ステージング表がある場合、それらは自動的に SET INTEGRITY ペンディング状態になります。「整合性の設定」ウィンドウを使用して、それらの各表の整合性処理を個別に行う必要があります。

外部キー、制約のチェックまたは生成された列を追加して表を変更する場合には、表を変更する前に、制約のチェックをオフにする必要があります。 制約を追加した後には、新しく追加した制約に対して既存のデータが違反していないかチェックして、制約のチェックをオンに戻す必要があります。 さらに、表にデータをロードする場合、表へのデータのロードが完了しないと、表でチェック制約を活動化することはできません。表にデータをインポートする場合、表にデータをインポートする前にチェック制約を活動化してください。

制約のチェックとは、制約違反、外部キー違反、および生成された列の違反のチェックを指します。整合性処理とは、制約のチェックの実行に加えて、ID および生成済み列の移植、マテリアライズ照会表のリフレッシュ、およびステージング表への伝搬を指します。

通常は、表の参照整合性およびチェック制約は自動的に強制され、マテリアライズ照会表は自動的に即時リフレッシュされ、ステージング表は自動的に伝搬されます。 状況によっては、この動作を手動で変更する必要がある場合があります。

コントロール・センターを使用して制約違反をチェックするには、以下のようにします。

1. 次のようにして、「整合性の設定」ウィンドウをオープンします。コントロール・センターで、「**表**」フォルダーが表示されるまでオブジェクト・ツリーを展開します。「**表**」フォルダーをクリックします。既存の表は、 ウィンドウの右側のペインに表示されます。使用する表を右クリックして、ポップアップ・メニューから「**整合性の設定**」を選択します。「整合性の設定」ウィンドウがオープンします。
2. 処理する表の「現行の整合性状況」を確認する。

3. 表の制約チェックをオンにして、表データをチェックしないようにするには、次のようにしてください。
   a. 「**検査せずに即時にペンディング状態から出る**」ラジオ・ボタンを選択する。
   b. オンにする整合性処理のタイプを指定する。
   c. 表に対してデータ移動操作を即時に行いたい場合 (再編成や再配布など) は、「**フル・アクセス権限**」ラジオ・ボタンを選択する。ただし、以降の従属マテリアライズ照会表のリフレッシュには、もっと長い時間がかかります。表に関連したマテリアライズ照会表がある場合は、マテリアライズ照会表をリフレッシュするために必要な時間を削減するために、このラジオ・ボタンを選択しないことをお勧めします。
4. 表の制約のチェックをオンにして、既存の表データをチェックするには、次のようにしてください。
   a. 「**検査して即時にペンディング状態から出る**」ラジオ・ボタンを選択する。
   b. 実行する整合性処理のタイプを選択する。「**現行の保全状況**」にマテリアライズ照会表のチェック制約値が不完全であると表示されている場合には、マテリアライズ照会表を増分リフレッシュできません。
   c. オプション: 整合性処理中に ID または生成された列を移植したい場合は、「**強制生成**」チェック・ボックスを選択する。
   d. 表がステージング表でない場合は、「**整理**」チェック・ボックスのチェックが外れていることを確認する。
   e. 表に対してデータ移動操作を即時に行いたい場合は、「**フル・アクセス権限**」ラジオ・ボタンを選択する。
   f. オプション: 例外表を指定する。参照あるいはチェック制約の違反が起きている行は、表から削除され、例外表にコピーされます。例外表を指定しない場合で制約違反が起きている場合、検出された最初の違反のみが返され、表は SET INTEGRITY ペンディング状態のままです。
5. 表に対する制約のチェック、即時リフレッシュ、即時伝搬をオフにするには、次のようにしてください。
   a. 「**オフ**」ラジオ・ボタンを選択する。表は SET INTEGRITY ペンディング状態となります。
   b. 「**カスケード**」オプションを使用して、即時にカスケードするかカスケードを遅らせるかを指定する。即時にカスケードする場合は、「**マテリアライズ照会表**」、「**外部キー表**」、「**ステージング表**」チェック・ボックスを使用して、カスケードする表を指示する。

      注: 親表の制約のチェックをオフにして、外部キー表への変更をカスケードすることを指定すると、そのすべての従属外部キー表の外部キー制約もオフになります。基礎表の制約のチェックをオフにして、チェック・ペンディング状態をマテリアライズ照会表にカスケードすることを指定すると、そのすべての従属マテリアライズ照会表の即時リフレッシュ・プロパティーもオフになります。基礎表の制約のチェックをオフにして、SET INTEGRITY ペンディング状態をステージング表にカスケードすることを指定すると、そのすべての従属ステージング表の即時伝搬プロパティーもオフになります。

コマンド行を使用して制約違反をチェックするには、SET INTEGRITY ステートメントを使用します。

トラブルシューティングのヒント

症状 表の制約のチェック、即時リフレッシュ、即時伝搬をオンにしようとすると、以下のエラー・メッセージを受け取ります。

DB2 メッセージ

親表または基礎表 TABLE2 が SET INTEGRITY ペンディング状態の場合や、SET INTEGRITY ステートメントによって SET INTEGRITY ペンディング状態になるような場合は、SET INTEGRITY ステートメントを使って従属表 TABLE1 をチェックできません。

TABLE1 とは、制約のチェック、即時リフレッシュ、即時伝搬をオンにしようとしている表で、 TABLE2 に従属しています。

考えられる原因

制約のチェック、即時リフレッシュ、または即時伝搬は、SET INTEGRITY ペンディング状態の親または基礎表をもつ表ではオンにできません。

アクション

表の制約のチェックをオンにして、親または基礎表を SET INTEGRITY ペンディング状態から解放してください。まず、DB2 メッセージで親または基礎表として示されている表から行います。その表が別の表に従属している場合には、従属性チェーンの上部にある表のトップダウン・アプローチで、制約のチェックをオンにする必要があります。

アテンション: 選択した表に 1 つ以上の表との循環参照制約のリレーションシップがある場合は、「整合性の設定」ウィンドウを使って制約のチェックをオンにすることはできません。この場合、SQL SET INTEGRITY コマンドを発行するには、コマンド・エディターを使用する必要があります。

ロード操作時の表のロッキング

大抵の場合、ロード・ユーティリティーは、表レベル・ロックを使用して、表へのアクセスを制限します。ロックのレベルは、ロード操作の段階、およびロード操作が読み取りアクセスを許可するように指定されているかどうかによって異なります。

ALLOW NO ACCESS モードのロード操作は、ロード中に表に対して超排他ロック (Z-lock) を使用します。

ALLOW READ ACCESS モードのロード操作を開始する前に、ロード・ユーティリティーは、ロード操作前に開始したすべてのアプリケーションが、ターゲット表に対するロックを解放するのを待機します。ロード操作の始めに、ロード・ユーティリティーは表に対する更新ロック (U ロック) を獲得します。これは、データがコミットされるまで、このロックを保留します。ロード・ユーティリティーが表に対する U ロックを獲得する時、ロード操作の開始前に表に対するロックを保留するすべてのアプリケーションがそれらのロック (互換性のあるロックでも) を解放するのを待機します。これは U ロックを Z ロックに一時的にアップグレードすることによって達成されます。ターゲット表に新しく表ロック要求が出されても、要求され

るロックがロード操作の U ロックと互換性のあるものである限り、Z ロックがこれと競合することはありません。データがコミットされるときに、ロード・ユーティリティーはロックを Z ロックにアップグレードするため、コミット時には、競合するロックを持つアプリケーションが終了するまでロード・ユーティリティーが待機することで、いくらかの遅延が発生する場合があります。

注: アプリケーションが表に対するロックを解放するのを待機する間に、ロード操作がロードを開始しないうちにタイムアウトになる可能性があります。ただし、データをコミットするために必要な Z ロックを待機している間に、ロード操作がタイムアウトになることはありません。

競合するロックを持つアプリケーション
LOAD コマンドの LOCK WITH FORCE オプションを使用すると、ターゲット表に対する競合するロックを保留するアプリケーションを強制的にオフにし、ロード操作を継続できるようにすることができます。 ALLOW READ ACCESS モードのロード操作を継続するためにまず、以下のロックを保留するアプリケーションが強制的にオフにされます。

- 表更新ロックと競合する表ロック (例: インポートまたは挿入)。
- ロード操作のコミット・フェーズで存在するすべての表ロック。

システム・カタログ表に対する競合するロックを保留するアプリケーションは、ロード・ユーティリティーによって強制的にオフにされることはありません。ロード・ユーティリティーによってアプリケーションが強制的にシステムからオフにされると、アプリケーションはそのデータベース接続を失い、エラーが戻されます (SQL1224N)。

リカバリー可能データベースのロード操作に COPY NO オプションが指定される場合、表スペースがバックアップ・ペンディング状態になるまで、ターゲット表スペースのすべてのオブジェクトが共用モードでロックされます。これは、アクセス・モードにかかわりなく発生します。 LOCK WITH FORCE オプションが指定されている場合には、共用ロックと競合する表スペースにあるオブジェクトに対してロックを保留するすべてのアプリケーションが強制的にオフになります。

読み取りアクセス・ロード操作

ロード・ユーティリティーは、他のアプリケーションがロード中の表に対して行うアクセスの量を制御する 2 つのオプションを提供します。 ALLOW NO ACCESS オプションは表を排他的にロックし、表のロード時には、表データへのアクセスを許可しません。

ALLOW NO ACCESS オプションはデフォルトの動作です。 ALLOW READ ACCESS オプションは、他のアプリケーションによる表へのすべての書き込みアクセスを行えないようにしますが、既存のデータへの読み取りアクセスは許可します。この項では、ALLOW READ ACCESS オプションを扱います。

ロード操作を開始する前に存在する表データおよび索引データは、ロード操作の進行中に照会で表示できます。次のような例を考察してみます。

1. 1 つの整数列のある表を作成します。

```
create table ED (ed int)
```

2. 3 つの行をロードします。

```
load from File1 of del insert into ED
...
Number of rows read         = 3
Number of rows skipped      = 0
Number of rows loaded       = 3
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 3
```

3. 表を照会します。

```
select * from ED

ED
-----------
          1
          2
          3

  3 record(s) selected.
```

4. ALLOW READ ACCESS オプションを指定してロード操作を実行し、さらに 2 つの行のデータをロードします。

```
load from File2 of del insert into ED allow read access
```

5. 同時に、他の接続ではロード操作の進行中に表を照会します。

```
select * from ED

ED
-----------
          1
          2
          3

  3 record(s) selected.
```

6. ロード操作が終了するのを待ってから、表を照会します。

```
select * from ED

ED
-----------
          1
          2
          3
          4
          5

  5 record(s) selected.
```

ALLOW READ ACCESS オプションは、ロード操作が進行中の場合やロード操作が失敗した後でも、ユーザーがいつでも表データにアクセスできるようにするため、大量のデータをロードする際に大変便利です。 ALLOW READ ACCESS モードのロード操作の動作は、アプリケーションの分離レベルに依存しません。つまり、どの分離レベルであっても、事前に存在するデータを常に読み取ることができますが、ロード操作が終了するまでは新しくロードされたデータを読み取ることはできません。

ロード操作の最初と最後という 2 つの場合を除いては、その操作中ずっと読み取りアクセスが提供されます。

最初に、ロード操作はそのセットアップ・フェーズの終わり近くの少しの間、特殊 Z ロックを獲得します。この特殊 Z ロックを要求するロード操作より先に、表に対する非互換のロックを保持するアプリケーションがあった場合、ロード操作はこの非互換のロックが解放されるのを一定期間だけ待ち、その後タイムアウトになって失敗します。時間は *locktimeout* データベース構成パラメーターによって決定されます。 LOCK WITH FORCE オプションを指定すると、ロード操作は他のアプリケーションを強制的にオフにし、タイムアウトになるのを回避します。ロード操作は特殊 Z ロックを獲得し、このフェーズをコミットし、このロックを解放した後、ロード・フェーズに移ります。 ALLOW READ ACCESS モードでロード操作を開始した後、何らかのアプリケーションが表を読み取ろうとしてロックを要求した場合、ロックは付与されます。これが、この特殊 Z ロックと競合することはありません。新しいアプリケーションがターゲット表から既存のデータを読み取ろうとする場合に、これが可能です。

次に、ロード操作の最後でデータをコミットする前に、ロード・ユーティリティーは表に対する超排他ロック (Z ロック) を獲得します。ロード・ユーティリティーは、表に対してロックを保持しているすべてのアプリケーションがこれらを解放するまで待機します。これは、データがコミットされる前の遅延の原因となることがあります。 LOCK WITH FORCE オプションを使用すると、競合するアプリケーションを強制的にオフにし、待機せずにロード操作が継続されます。通常、ALLOW READ ACCESS モードのロード操作は、短期間の排他ロックを獲得します。しかし、USE <tablespace-name> オプションが指定されている場合には、索引コピー・フェーズの期間ずっと排他ロックが続きます。

複数のデータベース・パーティションで定義されている表に対してロード・ユーティリティーが実行している間、ロード・プロセス・モデルはそれぞれのデータベース・パーティションで個別に実行します。つまり、他のデータベース・パーティションからは独立してロックが獲得および解放されます。したがって、照会その他の操作が並行して実行された場合、同じロックに対して競合するため、デッドロックの可能性があります。例えば、データベース・パーティション 0 の表ロックが操作 A に認可され、データベース・パーティション 1 の表ロックがロード操作に認可されたとします。ロード操作がデータベース・パーティション 0 の表ロックを待って待機している間、操作 A はデータベース・パーティション 1 の表ロックが認可されるまで待機するため、デッドロックが発生する可能性があります。この場合、デッドロック検出機能はいずれかの操作を任意にロールバックします。

注:

1. ロード操作が中断または失敗すると、このロード操作の発行時に指定されたものと同じアクセス・レベルで継続します。したがって、ALLOW NO ACCESS モードのロード操作が失敗すると、ロードが終了するか、またはロード再開が発行されるまで、表データはアクセス不可になります。 ALLOW READ ACCESS モードのロード操作が打ち切られても、既存の表データはこれまでどおり読み取りアクセスが可能です。
2. 中断または失敗したロード操作に ALLOW READ ACCESS オプションが指定されていた場合には、ロード再開操作またはロード終了操作にもこれを指定することができます。しかし、中断または失敗したロード操作に ALLOW NO ACCESS オプションが指定されていた場合には、ロード再開操作またはロード終了操作に ALLOW READ ACCESS オプションを指定することはできません。

以下の場合には、ALLOW READ ACCESS オプションはサポートされません。

- REPLACE オプションが指定されている場合。ロード置換操作では、新規データをロードする前に既存の表データを切り捨てるため、ロード操作の完了後まで照会できる既存のデータはありません。
- 索引に無効のマークが付き、再作成されるのを待機している場合。一部のロールフォワード・シナリオで、または db2dart コマンドの使用時に、索引に無効のマークが付く可能性があります。
- INDEXING MODE DEFERRED オプションが指定されている場合。このモードでは、索引に再作成が必要というマークが付けられます。
- ALLOW NO ACCESS ロード操作が再開処理中または終了処理中の場合。これが完全にオンラインになるまで、表に対して ALLOW READ ACCESS モードのロード操作を実行することはできません。
- ロード操作が、SET INTEGRITY PENDING NO ACCESS 状態の表で実行されている場合。これは、制約付きの表に対する複数のロード操作でも同じです。 SET INTEGRITY ステートメントが発行されるまで、表がオンラインになることはありません。

一般に、表データがオフラインになっている場合には、この表がオンラインになるまでロード操作時に読み取りアクセスを使用することはできません。

ロード操作時およびロード操作後の表スペースの状態

ロード・ユーティリティーは、表スペースの状態を使用して、ロード操作時のデータベースの整合性を保持します。これらの状態は、データへのアクセスを制御するかユーザー処置を引き出すことによって機能します。

ロード・ユーティリティーは、ロード操作に関係する表スペースを静止させる (表スペースに対して永続ロックを置く) ことはなく、COPY NO パラメーターが指定されているロード操作の場合にのみ、表スペース状態を使用します。

表スペースの状態は、LIST TABLESPACES コマンドを使用することによって検査できます。表スペースは同時に複数の状態になる場合もあります。 LIST TABLESPACES によって戻される状態は以下のとおりです。

正常
「正常」という状態は、表スペースが作成された後の最初の状態であり、現在、表スペースに表れるような (異常な) 状態がないことを示しています。

ロード進行中
「ロード進行中」という状態は、表スペースに進行中のロードがあることを示しています。この状態にあると、ロード操作時に従属表のバックアップを行えません。ロード・ユーティリティーは、リカバリー可能データベースで **COPY NO** パラメーターを指定した場合のみ表スペースを「ロード進行中」の状態にするため、表スペースの状態は (すべてのロード操作で使用される)「ロード進行中」の表の状態とは異なります。表スペースはロード操作の間はこの状態のままです。

バックアップ・ペンディング
リカバリー可能データベースに対してロード操作を実行し、**COPY NO** パラメーターを指定すると、表スペースは最初のコミット後、「バックアップ・ペンディン

グ」という表スペースの状態になります。「バックアップ・ペンディング」の状態にある表スペースは更新できません。表スペースの「バックアップ・ペンディング」状態を解除する唯一の方法は、表スペースをバックアップすることです。ロード操作をキャンセルしても、表スペースの状態はロード操作の開始時に変更されていてロールバックできないため、表スペースは引き続き「バックアップ・ペンディング」状態のままです。

リストア・ペンディング
COPY NO オプションを指定して正常なロード操作を実行し、データベースをリストアし、その操作によってロールフォワードを行った場合、関連する表スペースは「リストア・ペンディング」という状態になります。表スペースを「リストア・ペンディング」状態から解除するには、リストア操作を実行しなければなりません。

表スペースの状態の例

以下のようにして、入力ファイル (staffdata.del) を表 NEWSTAFF にロードするとします。

```
update db cfg for sample using logretain recovery;
backup db sample;
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff copy no;
connect reset;
```

さらに、別のセッションを開き、次のコマンドを発行します。

```
connect to sample;
list tablespaces;
connect reset;
```

USERSPACE1 (サンプル・データベースのデフォルトの表スペース) は「ロード進行中」の状態になり、最初のコミット後は「バックアップ・ペンディング」の状態にもなります。ロード操作の完了後、LIST TABLESPACES コマンドを実行することにより、USERSPACE1 が現在「バックアップ・ペンディング」の状態にあることが分かります。

```
Tablespace ID                        = 2
Name                                 = USERSPACE1
Type                                 = Database managed space
Contents                             = All permanent data. Large table space.
State                                = 0x0020
  Detailed explanation:
  Backup pending
```

ロード操作時およびロード操作後の表の状態

ロード・ユーティリティーは、表の状態を使用して、ロード操作時のデータベースの整合性を保持します。これらの状態は、データへのアクセスを制御するかユーザー処置を引き出すことによって機能します。

表の状態を判別するには、LOAD QUERY コマンドを発行します。このコマンドはロード操作の状態も検査します。表は同時に複数の状態になる場合もあります。LOAD QUERY によって戻される状態は以下のとおりです。

正常状態
「正常」という状態は、表が作成された後の最初の状態であり、現在、表に表れるような (異常な) 状態がないことを示しています。

読み取りアクセスのみ
ALLOW READ ACCESS オプションを指定すると、表は「読み取りアクセスのみ」の状態になります。ロード・コマンドを呼び出す前に存在した表のデータが、ロード操作時に読み取り専用モードで使用可能になります。 ALLOW READ ACCESS オプションを指定したときに、ロード操作が失敗した場合には、ロード操作の前に表に存在していたデータが、失敗後も読み取り専用モードで引き続き使用可能になります。

ロード進行中
「ロード進行中」という表の状態は、表に進行中のロードがあることを示しています。ロード・ユーティリティーは、ロードが正常に完了すると、この過渡状態を解除します。しかし、ロード操作が失敗または中断されると、表の状態は「ロード・ペンディング」に変わります。

再配分進行中
「再配分進行中」という表の状態は、表に進行中の再配分があることを示しています。再配分ユーティリティーは、表の処理が正常に完了すると、この過渡状態を解除します。しかし、再配分操作が失敗または中断されると、表の状態は「再配分ペンディング」に変わります。

ロード・ペンディング
「ロード・ペンディング」という表の状態は、ロード操作が失敗または中断されたことを示しています。以下のいずれかのステップを行うことにより、「ロード・ペンディング」状態を解除することができます。

- 失敗の原因に対処します。例えば、ロード・ユーティリティーがディスク・スペースを使い果たした場合、表スペースにコンテナーを追加します。その後、ロード操作を再開します。
- ロード操作を終了します。
- ロード操作が失敗したその同じ表に対し、LOAD REPLACE 操作を実行します。
- ロードする表の表スペースを、最新の表スペースまたはデータベース・バックアップを指定した RESTORE DATABASE コマンドを使ってリカバリーした後、それ以降のリカバリー処理を実行します。

再配分ペンディング
「再配分ペンディング」という表の状態は、再配分操作が失敗または中断されたことを示します。 REDISTRIBUTE CONTINUE または REDISTRIBUTE ABORT 操作を実行すると、「再配分ペンディング」の状態を解除できます。

ロード再始動不可
「ロード再始動不可」の状態の表は部分的にロードされ、ロード再開操作を行うことができません。表は次の 2 つの状況で「ロード再始動不可」の状態になります。

- 正常に再開または終了できなかった失敗ロード操作の後ロールフォワード操作を実行した場合

- 表の状態が「ロード進行中」または「ロード・ペンディング」になっている間に行われたオンライン・バックアップからリストア操作を実行した場合

表は「ロード・ペンディング」状態にもなります。表の「ロード再始動不可」の状態を解除するには、LOAD TERMINATE または LOAD REPLACE コマンドを発行します。

SET INTEGRITY ペンディング
「SET INTEGRITY ペンディング」という状態は、ロードされた表に未確認の制約があることを示しています。ロード・ユーティリティーは、制約のある表でロード操作を開始する際に、表をこの状態にします。表の SET INTEGRITY ペンディング状態を解除するには、SET INTEGRITY ステートメントを使用してください。

タイプ 1 索引
「タイプ 1 索引」という状態は、表が現在タイプ 1 索引を使用していることを示します。このタイプは現在非推奨となっています。この索引に対して REORG ユーティリティーを使用する場合、この索引は、CONVERT オプションを使用してタイプ 2 に変換できます。

使用不可
リカバリー不能のロード操作からロールフォワードを実行すると、表は「使用不可」状態になります。この状態の表は使用することができません。ドロップするか、バックアップからリストアする必要があります。

複数の状態にある表の例

以下のようにして、相当量のデータを持つ入力ファイル (staffdata.del) を表 NEWSTAFF にロードするとします。

```
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff allow read access;
connect reset;
```

さらに、別のセッションを開き、次のコマンドを発行します。

```
connect to sample;
load query table newstaff;
connect reset;
```

LOAD QUERY コマンドを実行すると、NEWSTAFF 表の状態が「読み取りアクセスのみ」と「ロード進行中」であることが分かります。

```
Tablestate:
 Load in Progress
 Read Access Only
```

ロード例外表

ロード例外表は、ロード操作時にユニーク索引規則、範囲制約、およびセキュリティー・ポリシーに違反したすべての行を 1 つにまとめたレポートです。ロード例外表は LOAD コマンドの FOR EXCEPTION 節を使用することによって指定します。

制約事項: 例外表には、ID 列も、他のどのタイプの生成列も入れることはできません。 1 次表内に ID 列があると、例外表内のそれに対応する列には、その列のタイ

プ、長さ、および NULL 可能性の属性しか入れることはできません。さらに、例外表をパーティション化したり、例外表にユニーク索引を入れたりすることもできません。

ロード・ユーティリティーで使用される例外表は、SET INTEGRITY ステートメントが使用する例外表と同一のものです。これはロード中の表の定義を反映するユーザー作成の表であり、追加の列がいくつか入っています。

ロード例外表は、ロードされている表の存在する表スペース、またはその他の表スペースに割り当てることができます。どちらにしても、ロード例外表およびロードされている表は同じデータベース・パーティション・グループに割り当て、どちらも同じ分散キーを使用するようにしてください。

例外表を使用する状況
ユニーク索引を含むデータ、および重複レコードが含まれている可能性のあるデータをロードするときは、例外表を使用します。例外表を指定していない場合に重複レコードが検出されると、ロード操作はそのまま継続してしまい、削除された重複レコードに関する警告メッセージだけが出されます。重複レコードはログに記録されません。

ロード操作の完了後、エラーになったデータを例外表の中にある情報を使って訂正することができます。その後、訂正したデータを表に挿入できます。

行は例外表の中の既存の情報に追加されます。表が空であることを確かめるための検査は行われないため、前回のロード操作で無効だった行に新規の情報が単に追加されます。現在のロード操作で無効な行だけが必要な場合には、ユーティリティーを呼び出す前に既存の行を削除しておくことができます。または、ロード操作を定義するときに、違反が発見された時刻、および違反した制約の名前を例外表に記録するように指定できます。

削除イベントは発生するたびに記録されるため、固有性の条件に違反するレコードが多数あると、ロードの削除フェーズ中にログが満杯になる可能性があります。

無効なデータが原因で索引の構築前にリジェクトされた行は、例外表に挿入されません。

失敗した、または不完全なロード

中断したロード操作の再始動

ロード操作中に失敗や中断が発生した場合は、ロード・ユーティリティーを使用して操作を終了するか、表を再ロードするか、またはロード操作を再始動することができます。

実在しないデータ・ファイルや無効な列名などのユーザー・エラーが原因でロード・ユーティリティーを開始することすらできない場合、その操作はターゲット表を通常の状態にしたまま終了します。

ロード操作を開始すると、ターゲット表の状態は「ロード進行中」になります。操作が失敗すると表の状態は「ロード・ペンディング」に変わります。表のこの状態

を解除するには、LOAD TERMINATE を発行して操作をロールバックするか、LOAD REPLACE を発行して表全体を再ロードするか、または LOAD RESTART を発行することができます。

一般的に、この状態における最良の選択はロード操作を再始動することです。ロード・ユーティリティーはロード操作を最初からではなく、その進行内の最後に正常に到達した点から再開するので、この選択は時間の節約になります。操作を再開する正確な場所は、元のコマンドで指定されたパラメーターによって異なります。SAVECOUNT オプションが指定されており、前回のロード操作が失敗した場所がロード・フェーズである場合、ロード操作は到達した最後の整合点から再開します。上記以外の場合、ロード操作は正常に到達した最後のフェーズ (ロード、構築、または削除フェーズ) の最初から再開します。

XML 文書をロードしている場合は、動作が若干異なります。SAVECOUNT オプションは XML データのロードではサポートされていないため、ロード・フェーズ中に失敗したロード操作は、操作の最初から再開します。他のデータ・タイプと同様に、構築フェーズ中にロードが失敗した場合、REBUILD モードでは索引が構築されるので、各行からすべての索引キーを取り出すために表がスキャンされます。しかし、各 XML 文書も索引キーを選択するためにスキャンされる必要があります。キーのために XML 文書をスキャンするこの処理には、文書を再び構文解析することが必要ですが、これはコストのかかる操作になります。さらに、内部の XML 索引 (領域の索引やパスの索引など) が、最初に再構築される必要があります。これには、XDA オブジェクトのスキャンも必要です。

ロード操作の失敗の原因となった状況を修正した後、ロード・コマンドを再発行します。元のコマンドと同じパラメーターを正確に指定することにより、必要な一時ファイルをロード・ユーティリティーが見つけられるようにしてください。この例外は、読み取りアクセスを許可しない場合です。 ALLOW READ ACCESS オプションを指定したロード操作は、ALLOW NO ACCESS オプションとして再開することも可能です。

注: ロード・ユーティリティーが作成した一時ファイルは、決して削除したり変更したりしないでください。

以下のコマンドの実行によりロード操作が発生し、そのロード操作が失敗したとします。

```
LOAD FROM file_name OF file_type
SAVECOUNT n
MESSAGES message_file
load_method
INTO target_tablename
```

この場合、指定したロード・メソッド (load_method) を RESTART メソッドで置き換えることによってこれを再開します。

```
LOAD FROM file_name OF file_type
SAVECOUNT n
MESSAGES message_file
RESTART
INTO target_tablename
```

失敗したロードのうち、再開できないもの

操作に関連した表の状態が「ロード再始動不可」になっている場合は、失敗または中断されたロード操作を再開することができません。表がその状態になるのには、以下の理由があります。

- 正常に再開または終了されなかった失敗ロード操作の後ロールフォワード操作が実行された
- 表の状態が「ロード進行中」または「ロード・ペンディング」になっている間に行われたオンライン・バックアップからリストア操作が実行された

LOAD TERMINATE または LOAD REPLACE コマンドのいずれかを発行してください。

ALLOW READ ACCESS ロード操作の再開または終了

ALLOW READ ACCESS オプションを指定するロード操作が中断またはキャンセルされた場合にも、ALLOW READ ACCESS オプションを使用してその操作を再開または終了することができます。 ALLOW READ ACCESS オプションを使用することにより、終了または再開操作の進行中に、他のアプリケーションは表データを照会することができます。 ALLOW READ ACCESS モードのロード操作と同様に、表はデータがコミットされるまで排他的にロックされます。

索引オブジェクトが使用できない場合や、無効のマークが付いている場合には、ALLOW READ ACCESS モードでロード再開または終了操作を実行することは許可されません。

索引コピー・フェーズで元のロード操作が中断またはキャンセルされた場合、索引が壊れている可能性があるために、 ALLOW READ ACCESS モードの再開操作は許可されません。

ロード・フェーズで ALLOW READ ACCESS モードのロード操作が中断またはキャンセルされた場合、ロード・フェーズで再開します。ロード・フェーズ以外のフェーズで中断またはキャンセルされた場合には、構築フェーズで再開します。元のロード操作が ALLOW NO ACCESS モードである場合、元のロード操作が削除フェーズに到達するときに索引が有効であれば、そのフェーズで再開操作が行われます。索引に無効のマークが付いている場合には、ロード・ユーティリティーは構築フェーズからロード操作を再開します。

注: INDEXING MODE INCREMENTAL オプションが指定されている場合でも、すべてのロード再開操作は REBUILD 索引付けモードを選択します。

LOAD TERMINATE コマンドを発行すると、通常、中断またはキャンセルされたロード操作が最小限の遅延でロールバックされます。ただし、ALLOW READ ACCESS および INDEXING MODE INCREMENTAL が指定されているロード操作で LOAD TERMINATE コマンドを発行すると、ロード・ユーティリティーが索引をスキャンし、矛盾があればそれを修正する際に、遅延が生じる可能性があります。この遅延の長さは、索引のサイズによって異なり、ロード終了操作に ALLOW READ ACCESS オプションが指定されているかどうかにかかわりなく発生します。構築フェーズの前に元のロード操作が失敗した場合には、この遅延は発生しません。

注: 索引での矛盾を修正することから発生する遅延は、索引に無効というマークを付けて、これらを再構築することによって発生する遅延よりもはるかに小さくなります。

ロード再始動操作は、ロード再始動不可の状態にある表には実行できません。ロールフォワード操作時には、この表はロード再始動不可の表状態になる可能性があります。これはロード操作の終了前のある時点までロールフォワードを実行する場合、あるいは中断またはキャンセルされたロード操作を介してロールフォワードを実行するものの、ロード終了操作またはロード再開操作の終わりまでロールフォワードしない場合に発生します。

ロード・コピー・ロケーション・ファイルを使ったデータのリカバリー

DB2LOADREC レジストリー変数は、ロード・コピーのロケーション情報の入っているファイルを示すのに使用します。このファイルは、ロールフォワード・リカバリー中にロード・コピーの位置情報として使用されます。

DB2LOADREC には以下に関する情報があります。

- メディアの種類
- 使用するメディア装置の数
- 表のロード操作中に生成されるロード・コピーのロケーション
- ロード・コピーのファイル名 (もしあれば)

ロケーション・ファイルが存在しない場合、あるいはファイル内に一致する項目がない場合は、ログ・レコードからの情報が使用されます。

ファイル内の情報は、ロールフォワード・リカバリーの実行前に上書きされることがあります。

注:

1. 複数パーティション・データベースでは、 db2set コマンドを使用して、すべてのデータベース・パーティション・サーバーに **DB2LOADREC** レジストリー変数を設定する必要があります。
2. 複数パーティション・データベースでは、それぞれのデータベース・パーティション・サーバーにロード・コピー・ファイルが存在しなければならず、ファイル名 (パスも含む) は同じでなければなりません。
3. **DB2LOADREC** レジストリー変数によって識別されるファイルの項目が有効ではない場合、無効な項目を置き換える情報を提供するために古いロード・コピー・ロケーション・ファイルが使用されます。

ロケーション・ファイルには、以下の情報が提供されます。最初の 5 つのパラメーターには有効な値を指定する必要があり、これらのパラメーターはロード・コピーを識別するために使用されます。全体の構造は記録されるロード・コピーごとに繰り返されます。以下に例を示します。

```
TIMestamp      19950725182542         * Time stamp generated at load time
DBPartition    0                      * DB Partition number (OPTIONAL)
SCHema         PAYROLL                * Schema of table loaded
TABlename      EMPLOYEES              * Table name
DATabasename   DBT                    * Database name
```

```
DB2instance    toronto                * DB2INSTANCE
BUFfernumber   NULL                   * Number of buffers to be used for
                                        リカバリー
SESsionnumber  NULL                   * Number of sessions to be used for
                                        リカバリー
TYPeofmedia    L                      * Type of media - L for local device
                                                       A for TSM
                                                       O for other vendors
LOCationnumber 3                      * Number of locations
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
DBP            18
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

注:

1. 重要なのは各キーワードの最初の 3 文字です。すべてのキーワードは、指定された順序になっている必要があります。ブランク行は使用できません。
2. タイム・スタンプのフォーマットは *yyyymmddhhmmss* です。
3. フィールドはすべて必須ですが、BUF と SES (NULL 可能)、および DBP (リストから省略可能) は例外です。 SES が NULL の場合には、 *dft_loadrec_ses* 構成パラメーターによって指定される値が使用されます。 BUF が NULL の場合には、デフォルト値は SES+2 です。
4. ロケーション・ファイルにある項目が 1 つでも無効な場合には、以前のロード・コピー・ロケーション・ファイルの値が使用されます。
5. メディアの種類には、ローカル装置 (テープ、ディスク、またはディスケットの場合は L)、 TSM (A)、あるいは他のベンダー (0) のいずれかを指定できます。種類が L の場合、ロケーション項目に続けてロケーションの数を指定する必要があります。種類が A の場合、それ以上入力する必要はありません。種類が 0 の場合は、共有ライブラリー名を指定する必要があります。
6. SHRlib パラメーターは、ロード・コピー・データを格納する機能を持つライブラリーを指します。
7. COPY NO または NONRECOVERABLE オプションを指定してロード操作を呼び出し、かつ操作の完了後にデータベースまたは関連する表スペースのバックアップ・コピーを取らない場合、ロード操作の後の時点でこのデータベースまたは表スペースをリストアすることはできません。つまり、ロールフォワード・リカバリーを使ってもデータベースや表スペースをロード操作後の状態に再作成することはできません。データベースや表スペースをリストアできるのは、ロード操作より前の時点の状態だけです。

特定のロード・コピーを使用する場合には、データベースのリカバリー履歴ファイルを使用して、特定のロード操作のタイム・スタンプを判別できます。複数パーティション・データベースでは、リカバリー履歴ファイルは各データベース・パーティションにローカルに存在します。

ロード・ダンプ・ファイル

dumpfile ファイル・タイプ修飾子を指定すると、リジェクトされた行を書き込む例外表の名前とロケーションをロード・ユーティリティーに対して指示できます。

パーティション・データベース環境での実行時には、パーティション化サブエージェントまたはロード・サブエージェントによって行がリジェクトされることがあります。そのため、ダンプ・ファイル名には、サブエージェントのタイプを特定する拡張子と、例外が生成されたデータベース・パーティション番号が付けられます。例えば、次のようなダンプ・ファイル値を指定したとします。

```
dumpfile = "/u/usrname/dumpit"
```

この場合、データベース・パーティション 5 でロード・サブエージェントによってリジェクトされた行は、 /u/usrname/dumpit.load.005 という名前のファイルに保管され、データベース・パーティション 2 でロード・サブエージェントによってリジェクトされた行は、 /u/usrname/dumpit.load.002 という名前のファイルに保管され、データベース・パーティション 2 でパーティション化サブエージェントによってリジェクトされた行は、 /u/usrname/dumpit.part.002 という名前のファイルに保管される、などとなります。

ロード・サブエージェントによって行がリジェクトされた場合に、その行が 32,768 バイト未満の長さであると、そのレコード全体がダンプ・ファイルにコピーされますが、それ以上の長さの場合は、行の断片 (レコードの最終バイトを含む) がファイルに書き込まれます。

パーティション化サブエージェントによって行がリジェクトされた場合、レコード・サイズに関係なく、その行全体がダンプ・ファイルにコピーされます。

ロード一時ファイル

DB2 は、ロード処理中にバイナリーの一時ファイルを作成します。このファイルは、ロード・クラッシュ・リカバリー、ロード終了操作、警告およびエラー・メッセージ、および実行時制御データに使用されます。

ロード一時ファイルは、ロード操作がエラーなしで完了した時点で削除されます。一時ファイルが書き込まれるパスは、 LOAD コマンドの *temp-pathname* パラメーターまたは db2Load API の *piTempFilesPath* パラメーターで指定できます。デフォルトのパスは、データベース・ディレクトリーのサブディレクトリーです。

一時ファイルのパスはサーバー・マシン上にあり、DB2 のインスタンスが排他的にアクセスします。そのため、*temp-pathname* パラメーターに指定するパス名の修飾はクライアントではなくサーバーのディレクトリー構造を反映したものでなければならず、 DB2 インスタンス所有者にはそのパスに対して読み取りと書き込みの両方の許可が必要です。

注: MPP システムにおいては、一時ファイルのパスは NFS マウント上ではなく、ローカル・ディスク上のパスにしてください。パスが NFS マウント上にあると、ロード操作中のパフォーマンスがかなり低下します。
重要: このパスに書き込まれる一時ファイルは、どのような状況にあっても決して手を加えないでください。一時ファイルに手を加えるとロード操作における誤動作の原因となり、データベースが危険な状態になります。

ロード・ユーティリティーのログ・レコード

ユーティリティー管理機能は、ロード・ユーティリティーなどのいくつかの DB2 ユーティリティーに関連するログ・レコードを生成します。

以下のログ・レコードには、ロード操作時の特定の活動の開始点または終了点が記録されます。

- ロード開始。このログ・レコードはロード操作の開始に関連したものです。
- ロード削除開始。このログ・レコードはロード操作の削除フェーズの開始に関連したものです。削除フェーズは、主キー値が重複している場合にのみ開始されます。削除フェーズでは、表レコードに対する各削除操作または索引キーが記録されます。
- ロード削除終了。このログ・レコードはロード操作の削除フェーズの終了に関連したものです。この削除フェーズは、正常なロード操作のロールフォワード・リカバリー中に繰り返されます。
- ロード・ペンディング・リスト。このログ・レコードは、ロード・トランザクションがコミットした時点で書き込まれ、通常のトランザクション・コミット・ログ・レコードの代わりに使用されます。

以下のリストは、ロード・ユーティリティーが、入力データのサイズに従って作成するログ・レコードを説明します。

- ユーティリティーによって、DMS 表スペースに割り振られたり削除されたりするすべての表スペース・エクステントにつき、2 つのログ・レコードが作成されます。
- 消費される ID 値の塊ごとにログ・レコードが作成されます。
- ログ・レコードは、ロード操作の削除フェーズで削除されるデータ行または索引キーごとに作成されます。
- ALLOW READ ACCESS および INDEXING MODE INCREMENTAL オプションを指定してロード操作を実行する際に、索引ツリーの整合性を保守するためのログ・レコードが作成されます。ログ記録されるレコードの数は、完全にログ記録される索引への挿入よりずっと少なくなります。

ロードの概要 - パーティション・データベース環境

複数パーティション・データベースでは、大量のデータが多数のデータベース・パーティションに散在しています。データの各部分がどのデータベース・パーティションに入るかは、分散キーによって決定されます。また、適切なデータベース・パーティションにデータをロードする前に、データを分散 しておく必要があります。

複数パーティション・データベースに表をロードする際に、ロード・ユーティリティーは以下を実行できます。

- 入力データを並列で分散します。
- データをそれぞれ対応するデータベース・パーティションに同時にロードします。
- あるシステムから別のシステムにデータを転送します。

複数パーティション・データベースへのデータのロードは、セットアップとロードの 2 つのフェーズで実行されます。セットアップ・フェーズ では表ロックなどのデータベース・パーティション・リソースが獲得され、ロード・フェーズ ではデータがデータベース・パーティションにロードされます。 LOAD コマンドの ISOLATE_PART_ERRS オプションを使用すると、これらのフェーズのいずれかで発生するエラーを処理する方法、および 1 つまたは複数のデータベース・パーティションでのエラーが、エラーのないデータベース・パーティションでのロード操作にどのように影響を与えるかを選択できます。

複数パーティション・データベースにデータをロードする際には、以下のいずれかのモードを使用できます。

PARTITION_AND_LOAD
データは (多くの場合は並列で) 分散され、それぞれ対応するデータベース・パーティションに同時にロードされます。

PARTITION_ONLY
データは (多くの場合は並列で) 分散され、それぞれのロード・データベース・パーティションの指定したファイルに出力が書き込まれます。それぞれのファイルにはパーティション・ヘッダーがあり、データがいくつかのデータベース・パーティションに分散された方法、および LOAD_ONLY モードを使用してデータベースにファイルをロードできることを示します。

LOAD_ONLY
データはすでにいくつかのデータベース・パーティションに分散されているものとします。この場合は分散プロセスが省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。

LOAD_ONLY_VERIFY_PART
データはすでにいくつかのデータベース・パーティションに分散されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。分散プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいデータベース・パーティションにあることがチェックされます。 dumpfile ファイル・タイプ修飾子が指定されている場合には、データベース・パーティション違反のある行がダンプ・ファイルに入れられます。そうでなければ、行は廃棄されます。ロードしている特定のデータベース・パーティションにデータベース・パーティション違反がある場合、そのデータベース・パーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。

ANALYZE
すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。

概念および用語

複数データベース・パーティションを持つパーティション・データベース環境でのロード・ユーティリティーの動作および操作について説明する際には、以下の用語が使用されます。

- *コーディネーター・パーティション* は、ロード操作を実行するためにユーザーが接続するデータベース・パーティションです。
  PARTITION_AND_LOAD、PARTITION_ONLY、および ANALYZE モードでは、LOAD コマンドの CLIENT オプションが指定されていない限り、データ・ファイルはこのデータベース・パーティションに存在するものとされます。 CLIENT を指定すると、ロードされるデータが、リモートで接続されるクライアントに存在することを示します。
- PARTITION_AND_LOAD、PARTITION_ONLY、および ANALYZE モードでは、*事前パーティション化エージェント* がユーザー・データを読み取り、データを分散するパーティション化エージェント にラウンドロビン方式でこれを分散します。この処理は、コーディネーター・パーティションで常に実行されます。どのロード操作の場合でも、 1 つのデータベース・パーティションにつき最大 1 つのパーティション化エージェントが許可されます。
- PARTITION_AND_LOAD、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、それぞれの出力データベース・パーティションで*ロード・エージェント* が実行し、そのデータベース・パーティションへのデータのロードを調整します。
- *ファイル・エージェントへのロード* は、 PARTITION_ONLY ロード操作時にそれぞれの出力データベース・パーティションで実行します。これらはパーティション化エージェントからデータを受信し、これをデータベース・パーティションにあるファイルに書き込みます。
- SOURCEUSEREXIT オプションを使用すると、カスタマイズしたスクリプトまたは実行ファイル (ここでは*ユーザー出口*と呼びます) をロード・ユーティリティーが実行するための機構が提供されます。

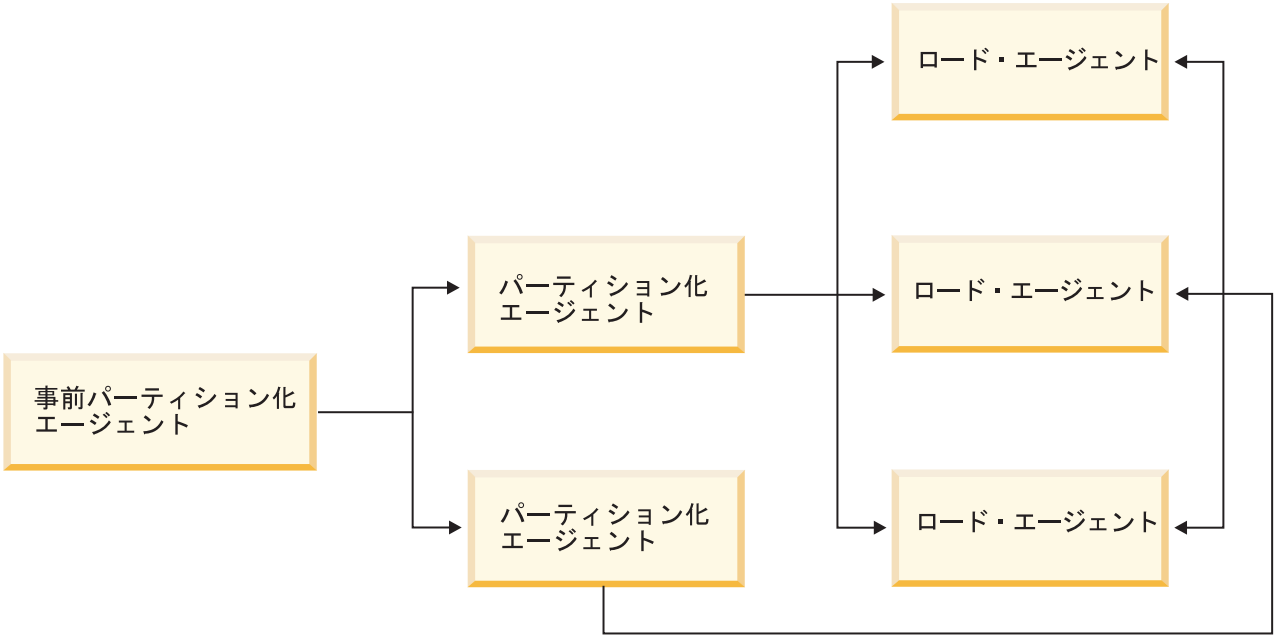


図 13. パーティション・データベース・ロードの概説： 事前パーティション化エージェントによりソース・データを読み、2 つのパーティション化エージェントそれぞれに約半分のデータが送られます。パーティション化エージェントはデータを分散し、各パーティションを 3 つのデータベース・パーティションのいずれかに送ります。各データベース・パーティションのロード・エージェントがデータをロードします。

パーティション・データベース環境でのデータのロード

ロード・ユーティリティーを使用して、パーティション・データベース環境にデータをロードします。

複数パーティションを持つデータベースに表をロードする前に確認する事項：

1. データベース・マネージャー構成パラメーター *svcename*、およびプロファイル・レジストリー変数 **DB2COMM** が正しく設定されていることを確認してください。ロード・ユーティリティーは TCP/IP を使用して事前パーティション化エージェントからパーティション化エージェントにデータを転送し、さらにパーティション化エージェントからロード・データベース・パーティションにデータを転送するため、このことは重要です。
2. ロード・ユーティリティーを起動するには、その前にデータのロード先となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。ロード・ユーティリティーは COMMIT ステートメントを発行するため、ロード・ユーティリティーの呼び出し前に COMMIT または ROLLBACK ステートメントを発行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておかなければなりません。PARTITION_AND_LOAD、 PARTITION_ONLY、または ANALYZE モードが使用されている場合には、ロードされるデータ・ファイルは、以下の状態になっていない限り、このデータベース・パーティションになければなりません。
   a. CLIENT オプションが指定されている場合。この場合には、データがクライアント・マシンになければなりません。

b. 入力ソース・データが CURSOR である場合。この場合には、入力ファイルはありません。

3. 設計アドバイザーを実行して、各表ごとに最適なデータベース・パーティションを決定します。詳しくは、データベース・パフォーマンスのチューニングの『設計アドバイザー』を参照してください。

ロード・ユーティリティーを使用して複数パーティション・データベースにデータをロードする際には、以下の制約が適用されます。

- ロード操作の入力ファイルのロケーションとして磁気テープ装置を指定することはできません。
- ANALYZE モードが使用されていない限り、ROWCOUNT オプションはサポートされません。
- ターゲット表に分散に必要な ID 列があり、identityoverride ファイル・タイプ修飾子が指定されていない場合、または複数のデータベース・パーティションを使ってデータを分散してからロードする場合、LOAD コマンドにおいて 0 より大きい SAVECOUNT 値の使用はサポートされていません。
- ID 列が分散キーの一部を構成している場合は、PARTITION_AND_LOAD モードだけがサポートされます。
- LOAD コマンドの CLIENT オプションを指定する際には、 LOAD_ONLY およびLOAD_ONLY_VERIFY_PART モードは使用できません。
- CURSOR 入力ソース・タイプを指定する際には、LOAD_ONLY_VERIFY_PART モードは使用できません。
- LOAD コマンドの ALLOW READ ACCESS および COPY YES オプションを指定する際には、分散エラー分離モード LOAD_ERRS_ONLY および SETUP_AND_LOAD_ERRS は使用できません。
- OUTPUT_DBPARTNUMS および PARTITIONING_DBPARTNUMS オプションによって指定されるデータベース・パーティションがオーバーラップしない場合には、複数のロード操作が同じ表に同時にデータをロードすることができます。例えば、表がデータベース・パーティション 0 から 3 に定義されている場合、 1 つのロード操作がデータベース・パーティション 0 および 1 にデータをロードする一方で、 2 番目のロード操作でデータベース・パーティション 2 および 3 にデータをロードすることができます。
- 区切りなし ASCII (ASC) および区切り付き ASCII (DEL) ファイルのみ、複数のデータベース・パーティションにまたがる複数の表に分散できます。 PC/IXF ファイルは分散できませんが、LOAD_ONLY_VERIFY_PART モードでロード操作を行って、複数のデータベース・パーティションに分散されている表にロードすることはできます。

以下の例では、LOAD コマンドを使用して各種のロード操作を開始する方法を説明します。以下の例で使用されるデータベースには、5 つのデータベース・パーティション、0、1、2、3、および 4 があります。データベース・パーティションにはそれぞれローカル・ディレクトリー /db2/data/ があります。データベース・パーティション 0、1、3、および 4 では、2 つの表、TABLE1 および TABLE2 が定義されます。クライアントからロードする際には、ユーザーにはデータベース・パーティションのいずれかではないリモート・クライアントへのアクセスがあります。

サーバー・パーティションからのロード

分散およびロードの例

このシナリオでは、 TABLE1 が定義されているか、または定義されていないデータベース・パーティションに接続しているものとします。データ・ファイル load.del は、このデータベース・パーティションの現行作業ディレクトリーにあります。load.del から、TABLE1 が定義されているすべてのデータベース・パーティションにデータをロードするには、次のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

注: この例では、すべてのパーティション・データベース環境の構成パラメーターでデフォルト値を使用します。MODE パラメーターのデフォルトは PARTITION_AND_LOAD で、 OUTPUT_DBPARTNUMS オプションのデフォルトは、TABLE1 が定義されているすべてのデータベース・パーティションです。また、PARTITIONING_DBPARTNUMS のデフォルトは、LOAD コマンド規則に従って選択したデータベース・パーティションのセットです。この規則は、データベース・パーティションが指定されていない場合にそれを選択するためのものです。

データがデータベース・パーティション 3 および 4 に分散されるようにロード操作を実行するには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                    PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

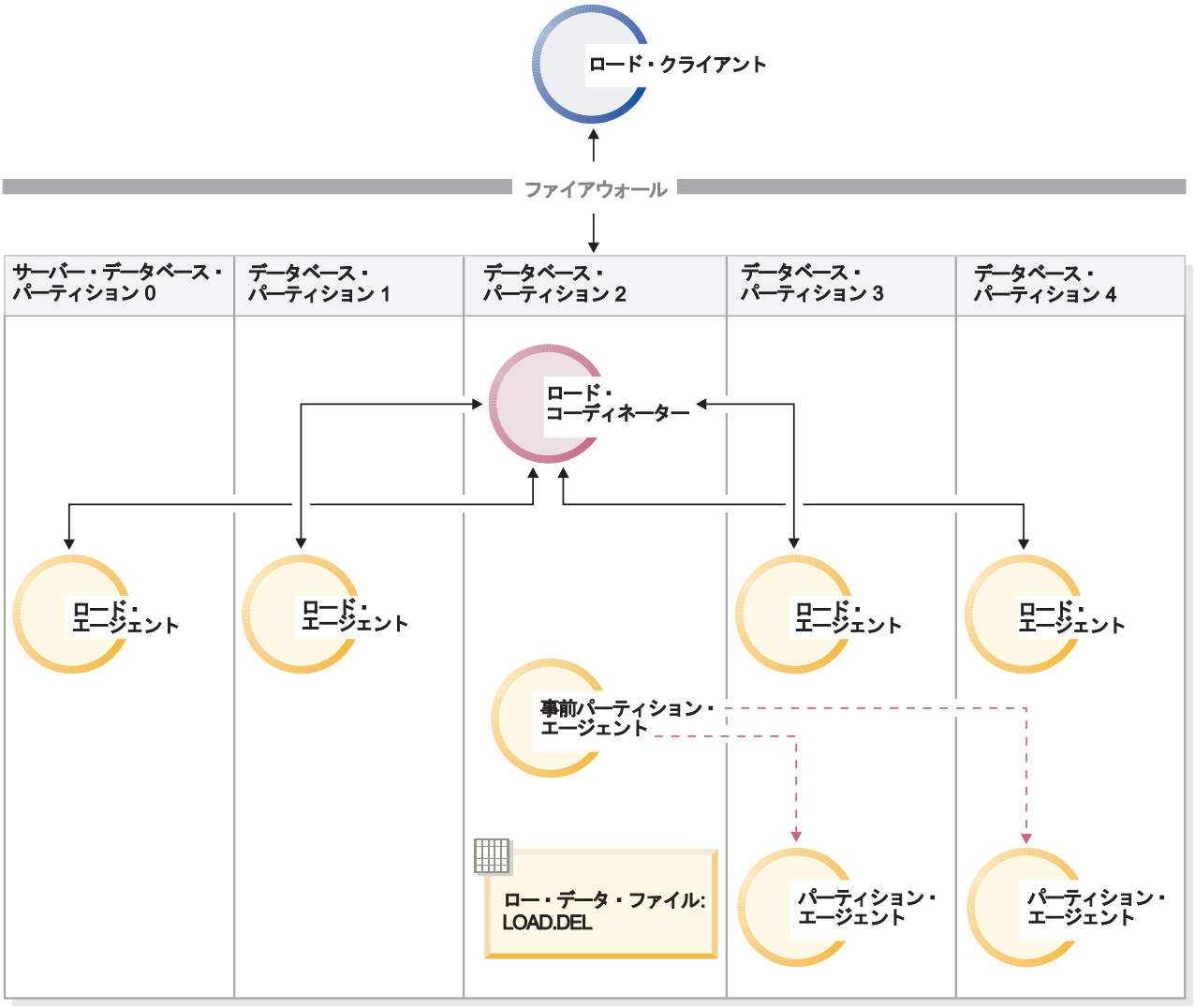


図 14. この図では、前のコマンドが発行された結果の動作を説明します。データは、データベース・パーティション 3 および 4 にロードされます。

分散のみの例

このシナリオでは、 TABLE1 が定義されているか、または定義されていないデータベース・パーティションに接続しているものとします。データ・ファイル load.del は、このデータベース・パーティションの現行作業ディレクトリーにあります。TABLE1 が定義されているすべてのデータベース・パーティションに load.del を分散する (ロードはしない) には、データベース・パーティション 3 および 4 を使用し、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                               PART_FILE_LOCATION /db2/data
                                               PARTITIONING_DBPARTNUMS (3,4)
```

これにより、ファイル load.del.xxx は、それぞれのデータベース・パーティションにある /db2/data ディレクトリーに保管されます。ここで、xxx は、3 桁表記のデータベース・パーティション番号です。

データベース・パーティション 0 (PARTITIONING_DBPARTNUMS のデフォルト) で実行しているパーティション化エージェントを 1 つだけ使用して、 load.del ファイルをデータベース・パーティション 1 および 3 に分散するには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                               PART_FILE_LOCATION /db2/data
                                               OUTPUT_DBPARTNUMS (1,3)
```

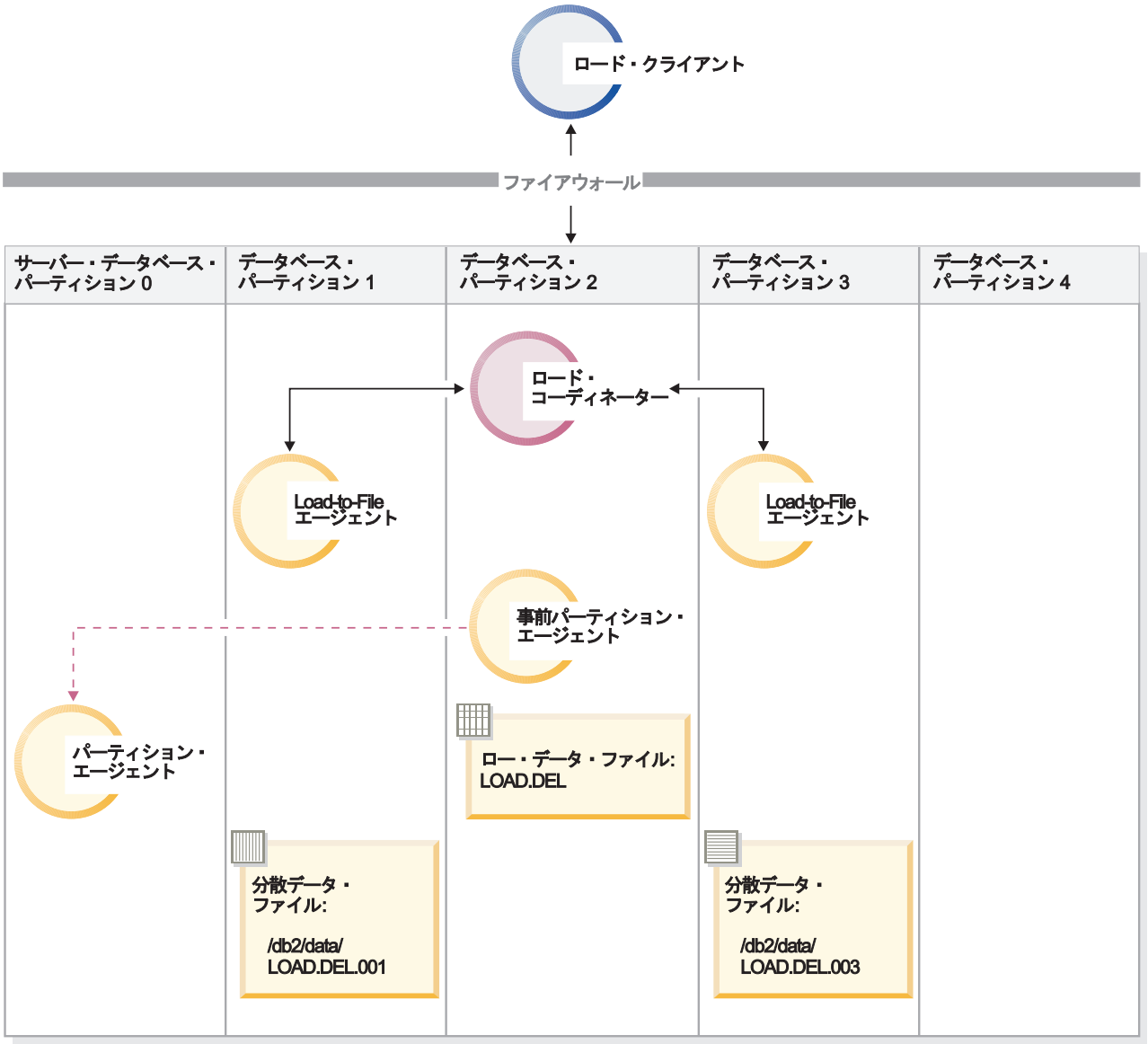


図 15. この図では、前のコマンドが発行された結果の動作を説明します。データは、データベース・パーティション 0 で実行する 1 パーティション化エージェントを使用して、データベース・パーティション 1 および 3 にロードされます。

ロードのみの例

すでに PARTITION_ONLY モードでロード操作を実行しており、 TABLE1 が定義されているすべてのデータベース・パーティションにそれぞれのロード・データベ

ース・パーティションの /db2/data ディレクトリーにあるパーティション・ファイルをロードする場合には、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                             PART_FILE_LOCATION /db2/data
```

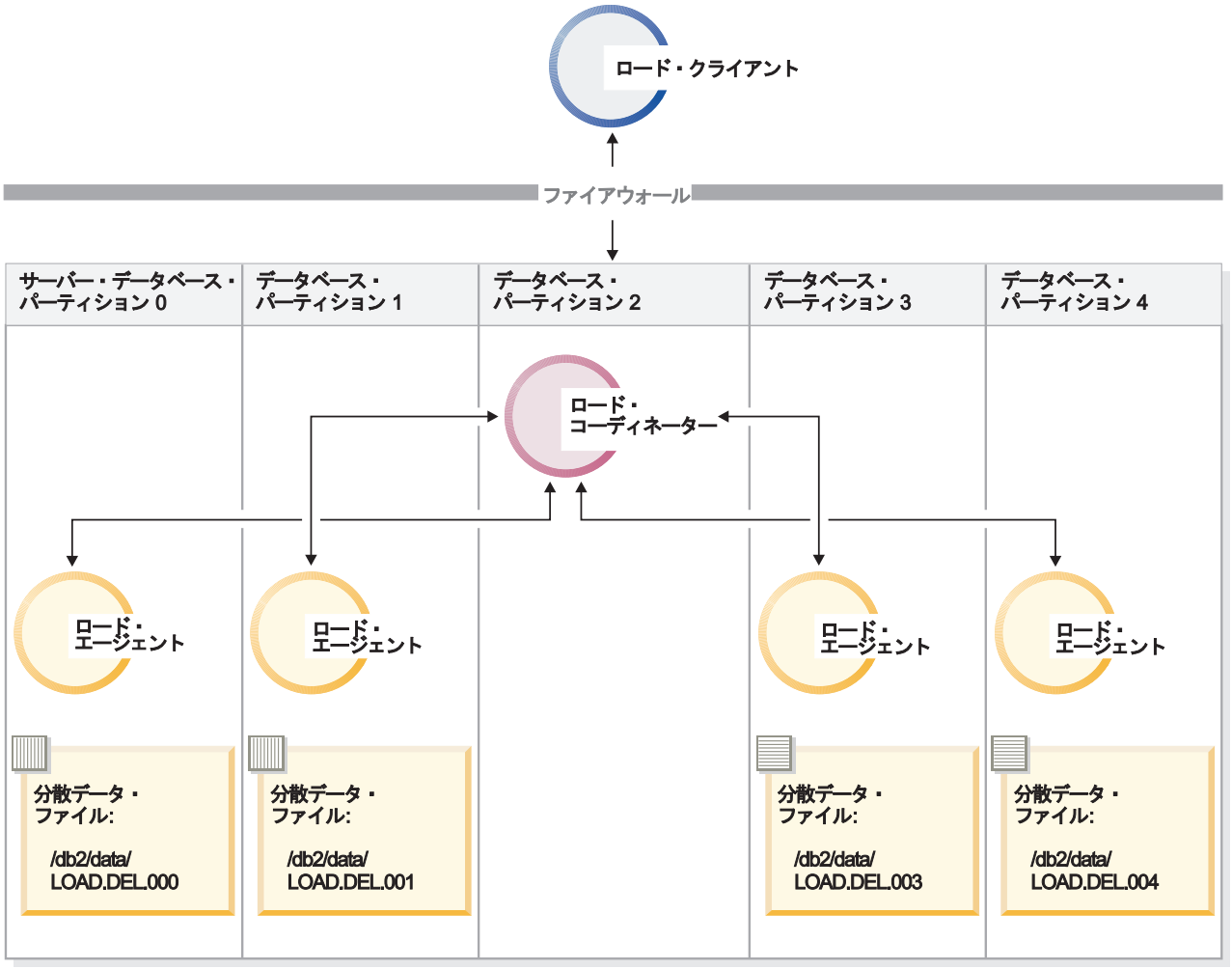


図 16. この図では、前のコマンドが発行された結果の動作を説明します。分散データは、TABLE1 が定義されているすべてのデータベース・パーティションにロードされます。

データベース・パーティション 4 にだけロードするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                             PART_FILE_LOCATION /db2/data
                                             OUTPUT_DBPARTNUMS (4)
```

分散マップ・ヘッダーのない事前分散ファイルのロード

LOAD コマンドを使用して、分散ヘッダーのないデータ・ファイルを、いくつかのデータベース・パーティションに直接ロードすることができます。 TABLE1 が定義されているそれぞれのデータベース・パーティションの /db2/data ディレクトリー

にデータ・ファイルが存在しており、この名前が load.del.xxx である場合 (xxx はデータベース・パーティション番号)、以下のコマンドを発行することによってファイルをロードできます。

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                        REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                   PART_FILE_LOCATION /db2/data
```

データベース・パーティション 1 にだけデータをロードするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                        REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                   PART_FILE_LOCATION /db2/data
                                                   OUTPUT_DBPARTNUMS (1)
```

注: ロード元のデータベース・パーティションにない行が指定された場合には、これはリジェクトされ、ダンプ・ファイルに入れられます。

リモート・クライアントから複数パーティション・データベースへのロード

リモート・クライアントにあるファイルから複数パーティション・データベースにデータをロードするには、 LOAD コマンドの CLIENT オプションを指定して、サーバー・パーティションにデータ・ファイルが存在しないことを示す必要があります。以下に例を示します。

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

注: LOAD_ONLY または LOAD_ONLY_VERIFY_PART モードを CLIENT オプションと共に使用することはできません。

カーソルからのロード

単一パーティション・データベースの場合と同様に、カーソルから複数パーティション・データベースにロードすることができます。この例では、PARTITION_ONLY および LOAD_ONLY モードの場合、 PART_FILE_LOCATION オプションは完全修飾ファイル名を指定しなければなりません。この名前は、それぞれの出力データベース・パーティションで作成またはロードされる分散ファイルの完全修飾基本ファイル名になります。ターゲット表に LOB 列がある場合には、指定されたベース名で複数のファイルを作成できます。

将来 TABLE2 にロードする目的で、 /db2/data/select.out.xxx (ここで、xxx はデータベース・パーティション番号) という名前のそれぞれのデータベース・パーティションにあるファイルに、ステートメント SELECT * FROM TABLE1 の応答セット内のすべての行を分散するには、以下のコマンドを発行します。

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                        PART_FILE_LOCATION /db2/data/select.out
```

上記の操作によって生成されるデータ・ファイルは、以下の LOAD コマンドを発行することによってロードできます。

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                        PARTITIONED CB CONFIG MODE LOAD_ONLY
                        PART_FILE_LOCATION /db2/data/select.out
```

パーティション・データベース環境でのデータのロード - ヒント

複数パーティション・データベースに表をロードする前に、以下のことを考慮してください。

- ロード構成オプションに慣れるために、まず小さなデータを使ってこのユーティリティーを操作してください。
- 入力データがあらかじめソートされている場合、または特定の順序になっている場合に、ロード・プロセス中にその順序を維持するとき、分散に使用するデータベース・パーティションは 1 つだけにしてください。並列分散では、必ずしもデータを受け取ったのと同じ順序でロードするとは限りません。 LOAD コマンドで `anyorder` 修飾子を指定しないと、ロード・ユーティリティーはデフォルトで単一パーティション化エージェントを選択します。
- 複数に分割されたファイルからラージ・オブジェクト (LOB) をロードする場合 (つまりロード・ユーティリティーの `lobsinfile` 修飾子を使っている場合)、LOB ファイルの入っているすべてのディレクトリーは、ロード先のすべてのデータベース・パーティションから読み取り可能でなければなりません。 LOB を処理する場合、ロードのパラメーター *lob-path* は完全修飾パスでなければなりません。
- ISOLATE_PART_ERRS オプションを SETUP_ERRS_ONLY または SETUP_AND_LOAD_ERRS に設定することにより、(起動時に) ロード操作でロード・データベース・パーティションや関連する表スペースまたは表がオフラインになっていることを検出した場合でも、複数パーティション・データベースで実行されているジョブを続行させることができます。
- STATUS_INTERVAL ロード構成オプションを使用して、複数パーティション・データベースで実行されているジョブの進行をモニターします。ロード操作は、指定された時間間隔でメッセージを生成し、事前パーティション化エージェントによって読み取られたデータの量をメガバイト単位で表示します。これらのメッセージは、事前パーティション化エージェント・メッセージ・ファイルにダンプされます。ロード操作中にこのファイルの内容を表示するには、コーディネーター・パーティションに接続してから、ターゲット表に対して LOAD QUERY コマンドを発行します。
- (PARTITIONING_DBPARTNUMS オプションで定義される) 分散プロセスに関係しているデータベース・パーティションが、 (OUTPUT_DBPARTNUMS オプションで定義される) ロード・データベース・パーティションと異なっている場合、CPU サイクルに対する競合が少なくなるため、パフォーマンスの改善が望めます。複数パーティション・データベースにデータをロードする際には、ロード・ユーティリティーを分散操作にもロード操作にも関係しないデータベース・パーティションで起動してください。
- LOAD コマンドに MESSAGES パラメーターを指定すると、事前パーティション化、パーティション化、およびロード・エージェントからのメッセージ・ファイルを保管して、ロード操作の終了時に参照できるようにします。ロード操作中にこれらのファイルの内容を表示するには、希望するデータベース・パーティションに接続してから、ターゲット表に対して LOAD QUERY コマンドを発行します。

- ロード・ユーティリティーは、統計情報を収集する出力データベース・パーティションを 1 つだけ選択します。そのデータベース・パーティションを指定するには、 RUN_STAT_DBPARTNUM データベース構成オプションを使用できます。
- 複数パーティション・データベースでデータをロードする場合、事前に設計アドバイザーを実行して、各表ごとに最適なパーティションを判別します。詳しくは、データベース・パフォーマンスのチューニングの『『設計アドバイザー』』を参照してください。

トラブルシューティング

ロード・ユーティリティーが停止した場合には、以下を実行できます。

- STATUS_INTERVAL パラメーターを使用して、複数パーティション・データベース・ロード操作の進行をモニターすることができます。状況インターバル情報は、コーディネーター・パーティションの事前パーティション化エージェント・メッセージ・ファイルにダンプされます。
- パーティション化エージェント・メッセージ・ファイルをチェックして、それぞれのデータベース・パーティションでのパーティション化エージェント・プロセスの状況を調べます。エラーなしでロードが進行しており、TRACE オプションが設定された場合には、これらのメッセージ・ファイル内に多くのレコードに関するトレース・メッセージがあるはずです。
- ロード・メッセージ・ファイルをチェックして、ロード・エラー・メッセージがあるかどうかを調べます。

  注: これらのファイルを出力するためには、LOAD コマンドの MESSAGES オプションを指定しなければなりません。
- ロード・プロセスのいずれかに問題が発生したことを暗示するエラーを発見した場合、現在のロード操作を中断します。

LOAD QUERY コマンドを使用したパーティション・データベース環境でのロード操作のモニター

パーティション・データベース環境でロード操作を行う際、いくつかのロード・プロセスによって、それらが実行されるデータベース・パーティションにメッセージ・ファイルが作成されます。

これらのメッセージ・ファイルには、すべての情報、ロード操作の実行時に生成される警告およびエラー・メッセージが保管されます。ユーザーが表示できるメッセージ・ファイルを生成するロード・プロセスは、ロード・エージェント、事前パーティション化エージェント、およびパーティション化エージェントです。メッセージ・ファイルの内容は、ロード操作が終了してからでなければ使用できません。

ロード操作時に個々のデータベース・パーティションに接続し、ターゲット表に対して LOAD QUERY コマンドを発行できます。このコマンドが CLP から発行されると、LOAD QUERY コマンドで指定された表の、現在このデータベース・パーティションにあるすべてのメッセージ・ファイルの内容が表示されます。

例えば、データベース・パーティション 0 から 3 で構成されるデータベース WSDB の表、TABLE1 が定義されているとします。データベース・パーティション 0 に接続され、以下の LOAD コマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

このコマンドは、データベース・パーティション 0、1、2、および 3 で実行するロード・エージェント、データベース・パーティション 1 で実行するパーティション化エージェント、データベース・パーティション 0 で実行する事前パーティション化エージェントを組み込むロード操作を開始します。

データベース・パーティション 0 には、事前パーティション化エージェントのメッセージ・ファイルが 1 つ、およびそのデータベース・パーティションでのロード・エージェントのファイルが 1 つ配備されます。これらのファイルの内容を同時に表示するには、新しいセッションを開始し、以下のコマンドを CLP から発行します。

```
set client connect_node 0
connect to wsdb
load query table table1
```

データベース・パーティション 1 には、ロード・エージェントのファイルが 1 つ、およびパーティション化エージェントのファイルが 1 つ配備されます。これらのファイルの内容を表示するには、新しいセッションを開始し、以下のコマンドを CLP から発行します。

```
set client connect_node 1
connect to wsdb
load query table table1
```

注: STATUS_INTERVAL ロード構成オプションによって生成されるメッセージは、事前パーティション化エージェント・メッセージ・ファイルに表示されます。ロード操作時にこれらのメッセージを表示するには、コーディネーター・パーティションに接続してから、LOAD QUERY コマンドを発行しなければなりません。

メッセージ・ファイルの内容の保管

db2Load API を介してロード操作が開始される場合には、メッセージ・オプション (piLocalMsgFileName) を指定しなければならず、メッセージ・ファイルはサーバーからクライアントに移動して、表示できるように保管されます。

CLP から開始される複数パーティション・データベースのロード操作では、メッセージ・ファイルがコンソールに表示されたり、保持されたりすることはありません。複数パーティション・データベースのロードの完了後にこれらのファイルの内容を保管または表示するには、LOAD コマンドの MESSAGES オプションを指定しなければなりません。このオプションが使用される場合、ロード操作が完了すると、それぞれのデータベース・パーティションのメッセージ・ファイルはクライアント・マシンに転送され、MESSAGES オプションによって示されるベース名を使用してファイルに保管されます。複数パーティション・データベースのロード操作の場合の、生成されたロード・プロセスに対応するファイルの名前を以下にリストします。

プロセス・タイプ	ファイル名
ロード・エージェント	<message-file-name>.LOAD.<dbpartition-number>

プロセス・タイプ	ファイル名
パーティション化エージェント	<message-file-name>.PART.<dbpartition-number>
事前パーティション化エージェント	<message-file-name>.PREP.<dbpartition-number>

例えば、MESSAGES オプションが `/wsdb/messages/load` を指定すると、データベース・パーティション 2 のロード・エージェント・メッセージ・ファイルは `/wsdb/messages/load.LOAD.002` です。

注: CLP から開始した複数パーティション・データベースのロード操作には、MESSAGES オプションを使用することを強くお勧めします。

パーティション・データベース環境でのロード操作の再開、再始動、または終了

パーティション・データベース環境でロード操作が失敗した場合、その後に実行すべきステップは、失敗がいつ発生したかによって異なります。

複数パーティション・データベースでのロード・プロセスは 2 つのステージで構成されます。

1. 1 つはセットアップ・ステージ です。このステージ中に、出力データベース・パーティションに対する表ロックなどのデータベース・パーティション・レベルのリソースを取得します。

   通常、セットアップ・ステージで障害が発生した場合には、操作の再始動および終了は必要ではありません。行うべき作業は、失敗したロード操作で指定されたエラー分離モードによって異なります。

   セットアップ・ステージ・エラーを分離しないことをロード操作で指定した場合、ロード操作全体がキャンセルされ、それぞれのデータベース・パーティションの表の状態は、ロード操作以前の状態にロールバックされます。

   セットアップ・ステージ・エラーを分離することをロード操作で指定した場合、ロード操作は、セットアップ・ステージが正常に実行されたデータベース・パーティションで継続しますが、失敗したそれぞれのデータベース・パーティションにある表は、ロード操作以前の状態にロールバックされます。これは、セットアップ・ステージ中に失敗するパーティションとロード・ステージ中に失敗するパーティションがある場合、単一のロード操作が複数のステージで失敗する可能性があることを意味します。
2. もう 1 つはロード・ステージ です。このステージ中にデータがフォーマットされ、データベース・パーティション上の表にロードされます。

   複数パーティション・データベースのロード操作のロード・ステージで少なくとも 1 つのデータベース・パーティションにおいてロード操作が失敗した場合には、LOAD RESTART または LOAD TERMINATE コマンドを発行しなければなりません。これが必要になるのは、複数パーティション・データベースでのデータのロードが単一のトランザクションで実行されるためです。

ロード失敗の原因となった問題を修正できる場合は LOAD RESTART を選択してください。ロードの再始動操作が開始されると、ロードが中断した時点から、すべてのデータベース・パーティションでロード操作が継続されるので、これは時間の節約になります。

表を初期ロード操作前の状態に戻す場合は LOAD TERMINATE を選択してください。

手順

ロードがいつ失敗したかの判別

パーティション環境でロード操作が失敗したらまず、どのパーティションでロード操作が失敗したか、またそれぞれどのステージで失敗したかを判別する必要があります。これは、パーティション・サマリーを調べることによって行えます。ロード・コマンドが CLP から発行された場合、パーティション・サマリーはロードの最後に表示されます (以下の例を参照)。ロード・コマンドが db2Load API から発行された場合、パーティション・サマリーは db2PartLoadOut 構造の poAgentInfoList フィールドで確認できます。

ある特定のパーティションの ″Agent Type″ の項目が ″LOAD″ となっている場合、そのパーティションがロード・ステージに達していることを示しています。それ以外の場合、失敗はセットアップ・ステージ中に発生したことを示します。負の SQL コードは失敗を示します。以下は、ロード・ステージ中にパーティション 1 で失敗したロードの例です。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________________
LOAD           001      -00000289    Error. May require RESTART.
_____________________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________________
LOAD           003      +00000000    Success.
.
.
.
```

失敗したロードの再開、再始動、または終了

セットアップ・ステージ中に失敗するのは、SETUP_ERRS_ONLY または SETUP_AND_LOAD_ERRS を指定した ISOLATE_PART_ERRS オプションを使用するロードだけです。このステージ中に少なくとも 1 つの出力データベース・パーティションで失敗したロードについては、 LOAD REPLACE または LOAD INSERT コマンドを発行することができます。 OUTPUT_DBPARTNUMS オプションを使用して、失敗が発生したデータベース・パーティションのみを指定してください。

ロード・ステージ中に少なくとも 1 つの出力データベース・パーティションで失敗するロードについては、LOAD RESTART または LOAD TERMINATE コマンドを発行してください。

セットアップ・ステージ中に少なくとも 1 つの出力データベース・パーティションで、またロード・ステージ中に少なくとも 1 つの出力データベース・パーティションで失敗するロードについては、前述したとおり、失敗したロードを再開するため

に 2 つのロード操作 (1 つはセットアップ・ステージでの失敗に対するもので、もう 1 つはロード・ステージでの失敗に対するもの) を実行する必要があります。このタイプの失敗ロード操作を効率的に取り消すには、LOAD TERMINATE コマンドを発行します。ただし、セットアップ・ステージ中に失敗したパーティション上の表に対しては変更が行われず、ロード・ステージ中に失敗したパーティションに関してはすべての変更が取り消されたため、このコマンドを発行した後はそれに伴う対応をすべてのパーティションに対して行う必要があります。

例えば、データベース・パーティション 0 から 3 で構成されるデータベース WSDB で TABLE1 が定義されているとします。以下のコマンドが発行されます。

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

セットアップ・ステージ中に出力データベース・パーティション 1 で失敗が発生します。セットアップ・ステージ・エラーは分離されるため、ロード操作は継続します。しかし、ロード・ステージ中にパーティション 3 で失敗が発生します。ロードを操作を再開するには、以下のコマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

注: ロード再始動操作では、LOAD RESTART コマンドで指定されるオプションが使用されるため、元の LOAD コマンドで指定されるものと同一であることが重要です。

マイグレーションおよびバージョン互換性

複数パーティション・データベースにおいて DB2 Universal Database バージョン 8 以前のロードの動作に戻る場合は、**DB2_PARTITIONEDLOAD_DEFAULT** レジストリー変数を使用できます。

注: バージョン 9.5 以降、DB2_PARTITIONEDLOAD_DEFAULT レジストリー変数は非推奨となり、将来のリリースでは存在しなくなる可能性があります。

複数パーティション・データベースにおいて DB2 UDBバージョン 8 以前の LOAD コマンドの動作に戻ると、パーティション・データベース構成オプションを追加で指定しなくても、有効な分散ヘッダーのあるファイルを単一データベース・パーティションにロードすることができます。これは、**DB2_PARTITIONEDLOAD_DEFAULT** の値を NO に設定することによって行えます。単一データベース・パーティションに LOAD コマンドを発行する既存のスクリプトを変更したくない場合、このオプションを使用することをお勧めします。例えば、4 つのデータベース・パーティションを持つデータベース・パーティション・グループに属する表のデータベース・パーティション 3 に配布ファイルをロードするには、以下のコマンドを発行します。

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

それから、DB2 コマンド行プロセッサーから以下のコマンドを発行します。

```
CONNECT RESET

SET CLIENT CONNECT_NODE 3
```

```
CONNECT TO DB MYDB

LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

複数パーティション・データベースでは、複数パーティション・データベース・ロード構成オプションが指定されない場合には、表が定義されているすべてのデータベース・パーティションでロード操作が実行されます。入力ファイルには分散ヘッダーは必要ではなく、 MODE オプションはデフォルトの PARTITION_AND_LOAD になります。単一データベース・パーティションをロードするには、OUTPUT_DBPARTNUMS オプションを指定しなければなりません。

リファレンス - パーティション環境でのロード

パーティション・データベース環境でのロード・セッション - CLP の例

以下の例は、複数パーティション・データベースでのデータのロードを示しています。

データベースに 0 から 3 の番号の付いた 4 つのデータベース・パーティションがあります。データベース WSDB がすべてのデータベース・パーティションで定義されており、表 TABLE1 が、すべてのデータベース・パーティションでも定義されているデフォルト・データベース・パーティション・グループにあります。

例 1

データベース・パーティション 0 にあるユーザー・データ・ファイル load.del から TABLE1 にデータをロードするには、データベース・パーティション 0 に接続してから、以下のコマンドを発行します。

```
load from load.del of del replace into table1
```

ロード操作が正常に実行された場合、出力は以下のようになります。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________
LOAD           001      +00000000    Success.
_____________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________
LOAD           003      +00000000    Success.
_____________________________________________________
PARTITION      001      +00000000    Success.
_____________________________________________________
PRE_PARTITION  000      +00000000    Success.
_____________________________________________________
RESULTS:       4 of 4 LOADs completed successfully.
_____________________________________________________

Summary of Partitioning Agents:
Rows Read                   = 100000
Rows Rejected               = 0
Rows Partitioned            = 100000

Summary of LOAD Agents:
Number of rows read         = 100000
Number of rows skipped      = 0
```

```
Number of rows loaded       = 100000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 100000
```

出力では、それぞれのデータベース・パーティションごとに 1 つのロード・エージェントがあり、それぞれが正常に実行されたことを示しています。また、コーディネーター・パーティションで実行する事前パーティション化エージェントが 1 つ、およびデータベース・パーティション 1 で実行するパーティション化エージェントが 1 つあったことも示しています。これらのプロセスは、通常の SQL 戻りコード 0 を戻して正常に完了しました。統計のサマリーでは、事前パーティション化エージェントは 100,000 行を読み取り、パーティション化エージェントは 100,000 行を分散し、ロード・エージェントによってロードされるすべての行の合計は、100,000 行であることを示します。

例 2

以下の例では、データは PARTITION_ONLY モードで TABLE1 にロードされます。分散出力ファイルは、ディレクトリー /db/data の出力データベース・パーティションのそれぞれに保管されます。

```
load from load.del of del replace into table1 partitioned db config mode
   partition_only part_file_location /db/data
```

LOAD コマンドからの出力は、以下のようになります。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________
LOAD_TO_FILE   000      +00000000    Success.
_____________________________________________________
LOAD_TO_FILE   001      +00000000    Success.
_____________________________________________________
LOAD_TO_FILE   002      +00000000    Success.
_____________________________________________________
LOAD_TO_FILE   003      +00000000    Success.
_____________________________________________________
PARTITION      001      +00000000    Success.
_____________________________________________________
PRE_PARTITION  000      +00000000    Success.
_____________________________________________________

Summary of Partitioning Agents:
Rows Read                   = 100000
Rows Rejected               = 0
Rows Partitioned            = 100000
```

出力では、それぞれの出力データベース・パーティションで実行する load-to-file エージェントがあり、これらのエージェントが正常に実行されたことを示しています。コーディネーター・パーティションには事前パーティション化エージェントがあり、データベース・パーティション 1 で実行するパーティション化エージェントがあります。統計のサマリーでは、事前パーティション化エージェントによって 100,000 行が正常に読み取られ、パーティション化エージェントによって 100,000 行が正常に分散されたことを示しています。表には行がロードされなかったため、ロードされた行の数のサマリーは表示されません。

例 3

上記の PARTITION_ONLY ロード操作時に生成されたファイルをロードするには、以下のコマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config mode
    load_only part_file_location /db/data
```

ロード・コマンドからの出力は、以下のようになります。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________
LOAD           001      +00000000    Success.
_____________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________
LOAD           003      +00000000    Success.
_____________________________________________________
RESULTS:       4 of 4 LOADs completed successfully.
_____________________________________________________

Summary of LOAD Agents:
Number of rows read         = 100000
Number of rows skipped      = 0
Number of rows loaded       = 100000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 100000
```

出力では、それぞれの出力データベース・パーティションのロード・エージェントが正常に実行し、すべてのロード・エージェントによってロードされる行の数の合計が 100,000 であることを示しています。分散は実行されなかったため、分散される行のサマリーはありません。

例 4 - ロード操作の失敗

以下の LOAD コマンドを発行したとします。

```
load from load.del of del replace into table1
```

ロード操作中に、ロード・データベース・パーティションの 1 つが表スペースでスペース不足になっているため、以下の出力が戻されます。

```
SQL0289N  Unable to allocate new pages in table space "DMS4KT".
SQLSTATE=57011

Agent Type     Node     SQL Code     Result
_____________________________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________________________
LOAD           001      -00000289    Error. May require RESTART.
_____________________________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________________________
LOAD           003      +00000000    Success.
_____________________________________________________________________
PARTITION      001      +00000000    Success.
_____________________________________________________________________
PRE_PARTITION  000      +00000000    Success.
_____________________________________________________________________
RESULTS:       3 of 4 LOADs completed successfully.
_____________________________________________________________________

Summary of Partitioning Agents:
Rows Read                   = 0
```

```
Rows Rejected               = 0
Rows Partitioned            = 0

Summary of LOAD Agents:
Number of rows read         = 0
Number of rows skipped      = 0
Number of rows loaded       = 0
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 0
```

出力では、ロード操作でエラー SQL0289 が戻されたことを示します。このデータベース・パーティションのサマリーでは、データベース・パーティション 1 でスペースが足りないことを示しています。データベース・パーティション 1 の表スペースのコンテナーにスペースが追加された場合、以下のようにしてロード操作を再始動できます。

```
load from load.del of del restart into table1
```

パーティション・データベース環境でのロード構成オプション

MODE X

複数パーティション・データベースをロードする際に実行するロード操作のモードを指定します。 PARTITION_AND_LOAD がデフォルトです。有効な値は以下のとおりです。

- PARTITION_AND_LOAD. データは (多くの場合は並列で) 分散され、それぞれ対応するデータベース・パーティションに同時にロードされます。
- PARTITION_ONLY. データは (多くの場合は並列で) 分散され、それぞれのロード・データベース・パーティションの指定したファイルに出力が書き込まれます。 CURSOR 以外のファイル・タイプの場合、各データベース・パーティションの出力ファイル名のフォーマットは `filename.xxx` です。ここで `filename` は、LOAD コマンドで指定された入力ファイル名で、 `xxx` は 3 桁のデータベース・パーティション番号です。 CURSOR ファイル・タイプの場合、各データベース・パーティションの出力ファイルの名前は PART_FILE_LOCATION オプションによって決められます。各データベース・パーティションの分散ファイルの位置を指定する方法の詳細については、 PART_FILE_LOCATION オプションを参照してください。

  注:

  1. このモードは CLI ロード操作には使用できません。
  2. 分散に必要な ID 列が表に収められている場合は、 `identityoverride` ファイル・タイプ修飾子が指定されない限り、このモードはサポートされません。
  3. ファイル・タイプ CURSOR 用に生成される分散ファイルは、 DB2 の異なるリリース間で互換性がありません。これは、前のリリースで生成されたファイル・タイプ CURSOR の分散ファイルは、LOAD_ONLY モードを使用してロードできないということを意味します。同様に、現行リリースで生成されたファイル・タイプ CURSOR の分散ファイルは、将来のリリースでは LOAD_ONLY モードを使用してロードできません。

- LOAD_ONLY。 データはすでに分散されているものとします。この場合は分散プロセスが省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。 CURSOR 以外のファイル・タイプの場合、各データベース・パーティションの入力ファイル名のフォーマットは filename.xxx となります。ここで filename は、LOAD コマンドで指定されたファイルの名前で、 xxx は 3 桁のデータベース・パーティション番号です。 CURSOR ファイル・タイプの場合、各データベース・パーティションの入力ファイルの名前は PART_FILE_LOCATION オプションによって決められます。各データベース・パーティションの分散ファイルの位置を指定する方法の詳細については、PART_FILE_LOCATION オプションを参照してください。

  注:

  1. このモードは CLI ロード操作には使用できず、 LOAD コマンドの CLIENT オプションが指定されている場合にも使用できません。
  2. 分散に必要な ID 列が表に収められている場合は、 identityoverride ファイル・タイプ修飾子が指定されない限り、このモードはサポートされません。
- LOAD_ONLY_VERIFY_PART. データはすでに分散されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。分散プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいデータベース・パーティションにあることがチェックされます。 dumpfile ファイル・タイプ修飾子が指定されている場合には、データベース・パーティション違反のある行がダンプ・ファイルに入れられます。そうでなければ、行は廃棄されます。ロードしている特定のデータベース・パーティションにデータベース・パーティション違反がある場合、そのデータベース・パーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。各データベース・パーティションの入力ファイル名のフォーマットは filename.xxx となり、ここで filename は LOAD コマンドで指定されたファイルの名前で、 xxx は 3 桁のデータベース・パーティション番号です。各データベース・パーティションの分散ファイルの位置を指定する方法の詳細については、PART_FILE_LOCATION オプションを参照してください。

  注:

  1. このモードは CLI ロード操作には使用できず、 LOAD コマンドの CLIENT オプションが指定されている場合にも使用できません。
  2. 分散に必要な ID 列が表に収められている場合は、 identityoverride ファイル・タイプ修飾子が指定されない限り、このモードはサポートされません。
- ANALYZE。 すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。

PART_FILE_LOCATION X

PARTITION_ONLY、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、このパラメーターは、分散ファイルのロケーションを指定するために使用できます。このロケーションは、OUTPUT_DBPARTNUMS オプ

ションによって指定される各データベース・パーティションに存在しなければなりません。指定されたロケーションが相対パス名の場合には、そのパスが現行ディレクトリーに追加されて、分散ファイルのロケーションが作成されます。

CURSOR ファイル・タイプの場合、このオプションを指定しなければならず、ロケーションは完全修飾されたファイル名を参照していなければなりません。この名前は、PARTITION_ONLY モードの場合は、各出力データベース・パーティションで作成された分散ファイルの完全修飾された基本ファイル名、または LOAD_ONLY モードの場合は、各データベース・パーティションから読み取ることのできるファイルのロケーションです。
PARTITION_ONLY モードの使用時に、ターゲット表中に LOB 列があると、指定した基本名のファイルが複数作成されることがあります。

CURSOR 以外のファイル・タイプの場合、このオプションが指定されないと、現行ディレクトリーが分散ファイルに使用されます。

OUTPUT_DBPARTNUMS X

X は、データベース・パーティション番号のリストを示します。データベース・パーティション番号は、ロード操作が実行されるデータベース・パーティションを示します。データベース・パーティション番号は、表が定義されているデータベース・パーティションのサブセットでなければなりません。デフォルトでは、すべてのデータベース・パーティションが選択されます。リストは括弧で囲まなければならず、リスト内のアイテムはコンマで区切らなければなりません。範囲を指定できます (例えば、(0, 2 to 10, 15))。

PARTITIONING_DBPARTNUMS X

X は、分散プロセスで使用されるデータベース・パーティション番号のリストを示します。リストは括弧で囲まなければならず、リスト内のアイテムはコンマで区切らなければなりません。範囲を指定できます (例えば、(0, 2 to 10, 15))。分散プロセスで指定されるデータベース・パーティションは、ロードされるデータベース・パーティションと異なっていてもかまいません。
PARTITIONING_DBPARTNUMS が指定されない場合には、最適なパフォーマンスを実現するために、ロード・ユーティリティーにより必要なデータベース・パーティションの数と使用するデータベース・パーティションが判別されます。

LOAD コマンドで anyorder ファイル・タイプ修飾子が指定されていない場合、ロード・セッションではパーティション化エージェントが 1 つだけ使用されます。さらに、OUTPUT_DBPARTNUMS オプションにデータベース・パーティションが 1 つだけ指定されている場合、またはロード操作のコーディネーター・パーティションが OUTPUT_DBPARTNUMS のエレメントではない場合、分散プロセスでロード操作のコーディネーター・パーティションが使用されます。その他の場合には、OUTPUT_DBPARTNUMS で最初のデータベース・パーティション (コーディネーター・パーティションではない) が分散プロセスで使用されます。

anyorder ファイル・タイプ修飾子が指定されている場合には、分散プロセスで使用されるデータベース・パーティションの数は、
(OUTPUT_DBPARTNUMS のパーティションの数)/4 + 1 で決定されます。

MAX_NUM_PART_AGENTS X

ロード・セッションで使用されるパーティション化エージェントの最大数を指定します。デフォルトは 25 です。

ISOLATE_PART_ERRS X

個々のデータベース・パーティションで発生するエラーにロード操作がどのように対応するかを指示します。 LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合 (この場合のデフォルトは NO_ISOLATION) を除き、デフォルトは LOAD_ERRS_ONLY です。有効な値は以下のとおりです。

- SETUP_ERRS_ONLY。 セットアップ時にデータベース・パーティションにエラーが発生すると (データベース・パーティションへのアクセス時の障害、またはデータベース・パーティション上の表スペースまたは表へのアクセス時の障害など)、ロード操作は障害のあるデータベース・パーティションでは停止しますが、残りのデータベース・パーティションでは実行を継続します。データのロード中にデータベース・パーティションでエラーが発生すると、全体の操作が失敗します。
- LOAD_ERRS_ONLY。 セットアップ時にデータベース・パーティションにエラーが発生すると、ロード操作全体が失敗します。データのロード中にエラーが発生する場合、ロード操作はエラーが生じたデータベース・パーティションで停止します。残りのデータベース・パーティションでは、障害が発生するか、すべてのデータがロードされるまでロードが継続されます。新しくロードされたデータは、ロード再始動操作が実行されて正常に完了するまで表示されません。

  注: LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合には、このモードは使用できません。
- SETUP_AND_LOAD_ERRS。 このモードでは、セットアップまたはデータのロード時に生じるデータベース・パーティション・レベルのエラーによって、影響を受けたデータベース・パーティション上でのみ、処理が停止します。 LOAD_ERRS_ONLY モードの場合と同様、データのロード中にパーティション・エラーが発生した場合、新しくロードされたデータはロード再始動操作が実行されて正常に完了するまで表示されません。

  注: LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合には、このモードは使用できません。
- NO_ISOLATION。 ロード操作時にエラーがあれば、ロード操作は失敗します。

STATUS_INTERVAL X

X は、読み取られたデータのボリュームを通知する頻度を示します。メジャー単位はメガバイト (MB) です。デフォルトは 100 MB です。有効な値は 1 から 4000 の整数です。

PORT_RANGE X

X は、内部通信のためのソケットの作成に使う TCP ポートの範囲を表します。デフォルトの範囲は 6000 から 6063 です。 **DB2ATLD_PORTS** レジストリー変数の値が起動時に定義される場合には、その値は PORT_RANGE ロード構成オプションの値で置き換えられます。 **DB2ATLD_PORTS** レジストリー変数の場合、範囲は以下のフォーマットで提供されます。

```
<lower-port-number:higher-port-number>
```

CLP からの場合は、以下のフォーマットです。

```
( lower-port-number, higher-port-number )
```

CHECK_TRUNCATION
プログラムが入出力時にデータ・レコードの切り捨てをチェックするように指定します。デフォルトの動作では、入出力時にはデータの切り捨てをチェックしません。

MAP_FILE_INPUT X
X は、分散マップの入力ファイル名を指定します。このパラメーターはカスタマイズされた分散マップの入ったファイルを示すため、分散マップがカスタマイズされている場合にはこのパラメーターを指定しなければなりません。カスタマイズされた分散マップを作成するには、 db2gpmap プログラムを使用してデータベース・システム・カタログ表からマップを抽出するか、または、LOAD コマンドの ANALYZE モードを使用して最適なマップを生成します。ロード操作を継続するには、その前に ANALYZE モードを使用して生成されるマップをデータベース内のそれぞれのデータベース・パーティションに移動する必要があります。

MAP_FILE_OUTPUT X
X は、分散マップの出力ファイル名を示します。出力ファイルが作成されるのは、LOAD コマンドが発行されたデータベース・パーティションです。ただし、これは、そのデータベース・パーティションが、パーティション化の実行されるデータベース・パーティション・グループに関係している場合です。パーティション化に関係していないデータベース・パーティション (PARTITIONING_DBPARTNUMS によって定義される) 上で LOAD コマンドが呼び出された場合、出力ファイルは、PARTITIONING_DBPARTNUMS パラメーターを使って最初に定義されたデータベース・パーティションに作成されます。次のパーティション・データベース環境のセットアップを考慮してください。

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

serv3 で次の LOAD コマンドを実行すると、serv1 上に分散マップが作成されます。

```
LOAD FROM file OF ASC METHOD L ( ...) INSERT INTO table CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

このパラメーターは、ANALYZE モードが指定される際に使用しなければなりません。すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。このパラメーターが指定されておらず ANALYZE モードが指定されている場合には、プログラムは終了してエラーを戻します。

TRACE X
データ変換プロセスとハッシュ値の出力のダンプを調べることが必要になった場合にトレースするレコードの数を指定します。デフォルトは 0 です。

NEWLINE
入力データ・ファイルが、それぞれのレコードが改行文字によって区切られた ASC ファイルであり、 LOAD コマンドで reclen ファイル・タイプ修飾子が指定されている場合に使用されます。このオプションが指定されると、それぞれのレコードの改行文字がチェックされます。また、reclen ファイル・タイプ修飾子で指定されたレコード長もチェックされます。

DISTFILE X
このオプションを指定すると、ロード・ユーティリティーは、指定された名前のデータベース・パーティション分散ファイルを生成します。データベース・パーティション分散ファイルには 4096 個の整数が入っており、それぞれはターゲット表の分散マップの各項目に対応しています。ファイル内の各整数は、ロードされる入力ファイルの中で、対応する分散マップ項目にハッシュされる行数を表します。この情報はデータの中のスキューを識別するのに役立ちます。さらに、ユーティリティーの ANALYZE モードを使って表の新しい分散マップを生成すべきかどうか判断するのに役立ちます。このオプションを指定しない場合、ロード・ユーティリティーのデフォルト動作として、配布ファイルを生成しません。

注: このオプションを指定すると、最大で 1 つのパーティショニング・エージェントがロード操作のために使用されます。複数のパーティショニング・エージェントを明示的に要求した場合でも、1 つのみが使用されます。

OMIT_HEADER
分散ファイルに分散マップ・ヘッダーを組み込まないように指定します。指定されていなければ、ヘッダーが生成されます。

RUN_STAT_DBPARTNUM X
LOAD コマンドに STATISTICS YES パラメーターが指定された場合には、 1 つのデータベース・パーティションでのみ統計が収集されます。このパラメーターは、統計を収集するデータベース・パーティションを指定します。値が -1 か、またはまったく指定されない場合には、出力データベース・パーティション・リストの最初のデータベース・パーティションで統計が収集されます。

リファレンス - ロード

LOAD

データを DB2 表にロードします。サーバー上に存在するデータは、ファイル、テープ、または名前付きパイプの形式にすることができます。 表の COMPRESS 属性が YES に設定されている場合、ロードされるデータは、表内にディクショナリーがすでに存在するデータおよびデータベース・パーティションごとに圧縮の対象となります。

257 ページの『ロード・ユーティリティーのファイル・タイプ修飾子』へのクイック・リンク。

制約事項

ロード・ユーティリティーでは、階層レベルのデータのロードはサポートされていません。ロード・ユーティリティーには、範囲クラスター表との互換性はありません。

有効範囲

このコマンドは、一度の要求で複数のデータベース・パーティションに対して発行できます。

許可

以下のいずれか。

- *sysadm*
- *dbadm*
- データベースに対する LOAD 権限と次のいずれか
  - ロード・ユーティリティーが INSERT モード、 TERMINATE モード (それまでのロード挿入操作を終了する)、または RESTART モード (以前のロード挿入操作を再開する) で呼び出された場合には、その表に対する INSERT 特権。
  - ロード・ユーティリティーが REPLACE モード、 TERMINATE モード (それまでのロード置換操作を終了する)、または RESTART モード (以前のロード置換操作を再開する) で呼び出された場合には、その表に対する INSERT および DELETE 特権。
  - 例外表の INSERT 特権 (例外表をロード操作の一部として使用する場合)。
- 保護列を持つ表にデータをインポートするには、表内のすべての保護列への書き込みアクセスを可能にする LBAC 信用証明情報がセッション許可 ID に必要です。そうでない場合は、ロードが失敗してエラー (SQLSTATE 5U014) が戻されます。
- 保護された行を持つ表にデータをロードするには、セッション許可 ID が、以下の基準を満たすセキュリティー・ラベルを保持していなければなりません。
  - 表を保護しているセキュリティー・ポリシーの一部である
  - 書き込みアクセスまたは全アクセスを対象としてセッション許可 ID に認可された。

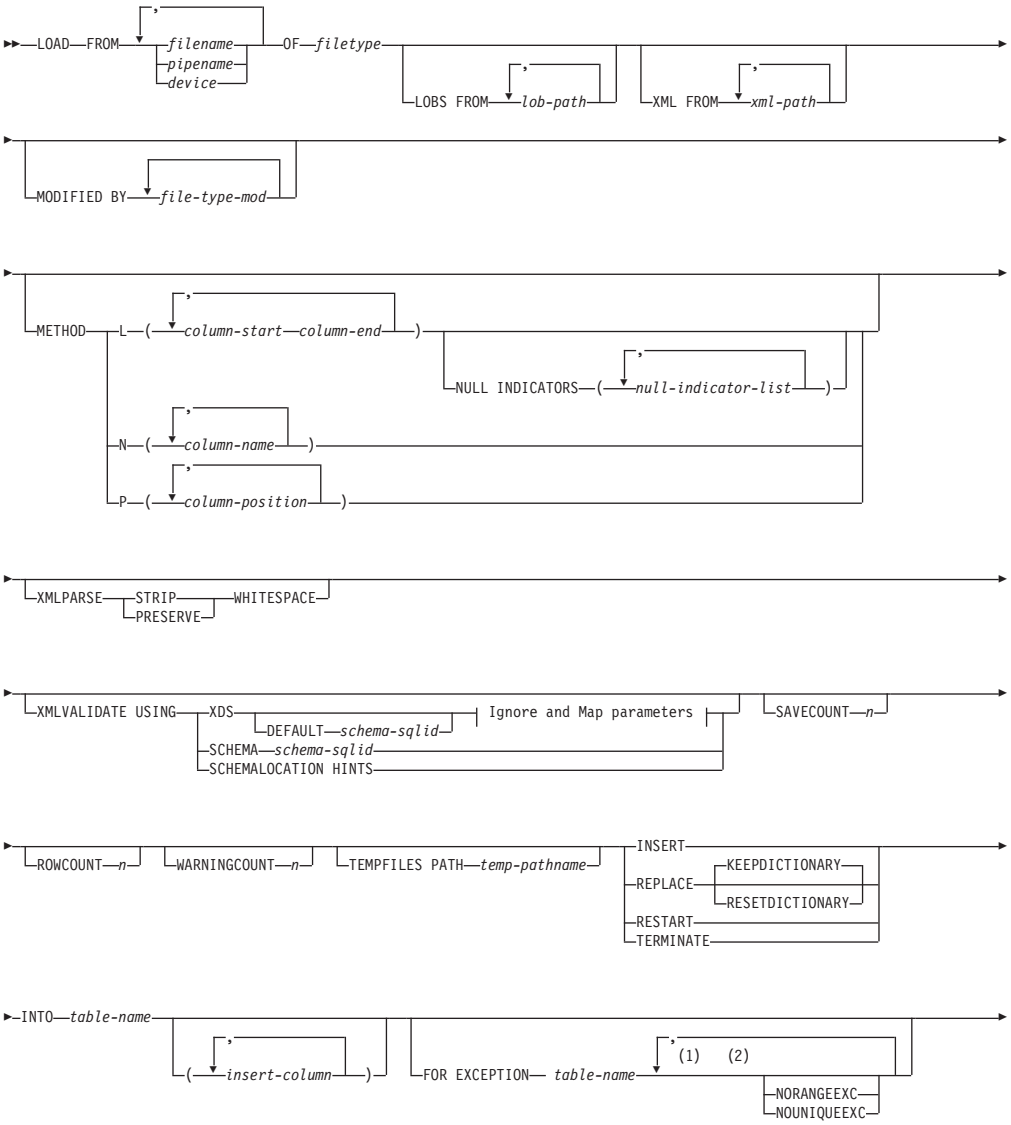
  こうしたセキュリティー・ラベルをセッション許可 ID が保持していない場合は、ロードが失敗してエラー (SQLSTATE 5U014) が戻されます。このセキュリティー・ラベルは、セッション許可 ID の LBAC 信用証明情報が、データ内のロードされる行を保護するセキュリティー・ラベルにその許可 ID が書き込むことを許可しない場合に、その行を保護するために使用されます。ただし、表を保護しているセキュリティー・ポリシーが CREATE SECURITY POLICY ステートメントの RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL オプションを使用して作成されている場合は、その状況にはなりません。その場合は、ロードが失敗してエラー (SQLSTATE 42519) が戻されます。
- REPLACE オプションを指定する場合、セッション許可 ID には表をドロップするための権限が必要です。

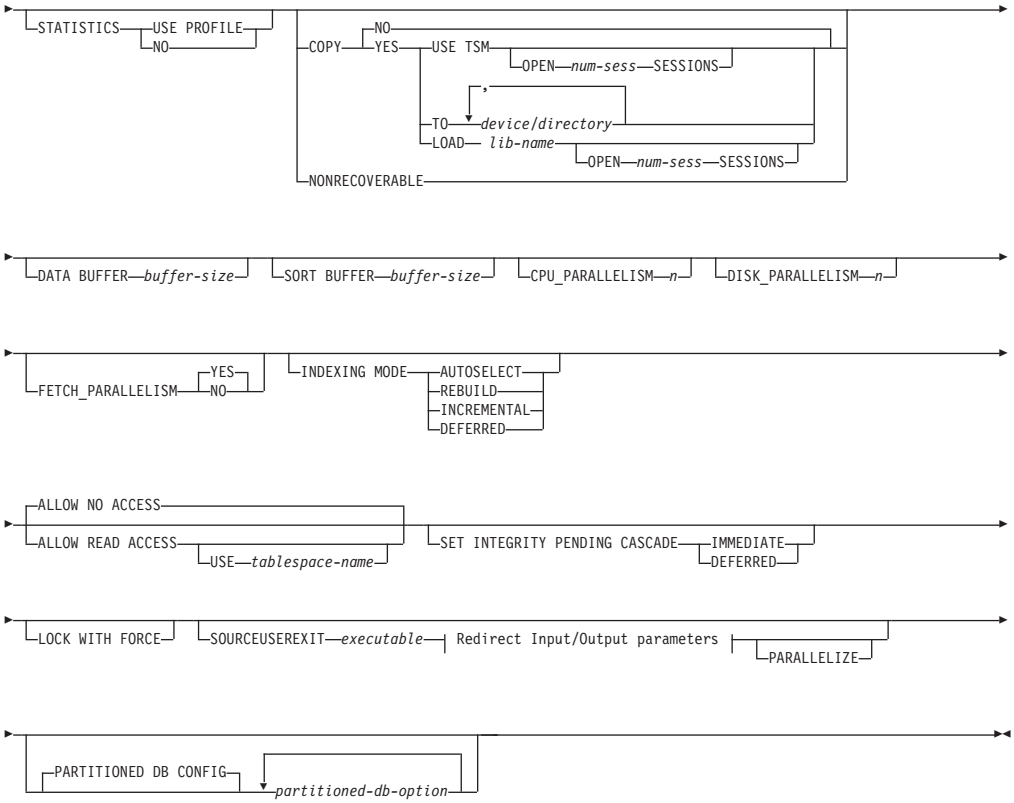
すべてのロード・プロセス (および一般にすべての DB2 サーバー・プロセス) はインスタンス所有者によって所有されており、それらのプロセスすべてにおいて、必要なファイルにアクセスするためにそのインスタンス所有者の ID を使用するため、インスタンス所有者には入力データ・ファイルに対する読み取りアクセス権が必要です。だれがコマンドを呼び出すかに関係なく、これらの入力データ・ファイルはインスタンス所有者から読み取り可能になっていなければなりません。

必要な接続

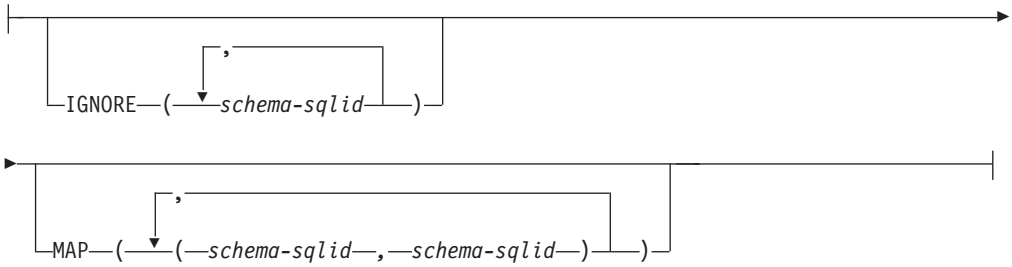
インスタンス。明示的なアタッチは必要ありません。データベースへの接続が確立されている場合には、ローカル・インスタンスへの暗黙的な接続が試みられます。

コマンド構文

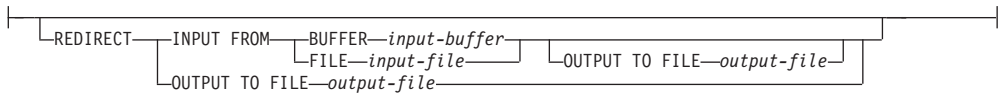




Ignore and Map parameters:



Redirect Input/Output parameters:



注:

1 これらのキーワードの出現順序は任意です。

2 これらのキーワードは、それぞれ 1 回だけ出現できます。

コマンド・パラメーター

FROM *filename* | *pipename* | *device*

注:

1. *ADMIN_CMD プロシージャーを使用した EXPORT コマンド* を使用してデータをファイルにエクスポートした場合、そのデータ・ファイルは fenced ユーザー ID によって所有されます。このファイルは通常、インスタンス所有者がアクセスすることはできません。 LOAD を CLP または ADMIN_CMD プロシージャーから実行するには、インスタンス所有者 ID はデータ・ファイルにアクセスできなければならないので、データ・ファイルに対する読み取りアクセス権限をインスタンス所有者に付与する必要があります。
2. ファイルが物理的には分割されてはいるが論理的には 1 つのファイルである場合には、複数の IXF ファイルからのデータのロードがサポートされています。ファイルが論理的にも物理的にも分割されている場合は、サポートされていません。 (複数の物理ファイルがすべて 一度の EXPORT コマンドの呼び出しで作成された場合、それらは論理的には 1 つであると見なされます。)

OF *filetype*

データのフォーマットを指定します。

- ASC (区切りなし ASCII フォーマット)。
- DEL (区切り付き ASCII フォーマット)。
- IXF (統合交換フォーマット、PC バージョン) は、DB2 専用のバイナリー・フォーマットです。
- CURSOR (SELECT または VALUES ステートメントに対して宣言されたカーソル)。

LOBS FROM *lob-path*

ロードする LOB 値が収められているデータ・ファイルへのパス。パスの最後はスラッシュでなければなりません。 LOB データ・ファイルの名前は、メイン・データ・ファイル (ASC、DEL、または IXF) の、LOB 列にロードされる列内に保管されます。指定できるパスの最大数は 999 です。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

CURSOR ファイル・タイプと併せて指定された場合、このオプションは無視されます。

MODIFIED BY *file-type-mod*

ファイル・タイプ修飾子オプションを指定します。 257 ページの『ロード・ユーティリティーのファイル・タイプ修飾子』を参照してください。

METHOD

L

データのロードを開始する列および終了する列の番号を指定します。列の番号は、データの行の先頭からのバイト単位のオフセットです。この番号は 1 から始まります。このメソッドは、ASC ファイルの場合にのみ使用することができ、そのファイル・タイプに対してのみ有効なメソッドです。

NULL INDICATORS *null-indicator-list*

このオプションは、METHOD L パラメーターを指定した場合だけ使用できます (つまり、入力ファイルが ASC ファイルの場合)。 NULL 標識リストは、コンマで区切られた正の整数のリストで、各 NULL 標識フィールドの列の番号を指

定します。列の番号は、データの行の先頭からのバイト単位の、各 NULL 標識フィールドのオフセットです。 NULL 標識リストには、METHOD L パラメーターで定義された各データ・フィールドに対する 1 つの項目がなければなりません。列の番号がゼロであることは、対応するデータ・フィールド内に必ずデータがあることを示します。

NULL 標識列中の Y の値は、その列データが NULL であることを指定します。 NULL 標識列に Y 以外 の文字を指定した場合は、列データが NULL ではなく、METHOD L オプションで指定された列データがロードされることを指定することになります。

NULL 標識文字は MODIFIED BY オプションを使用して変更できます。

N ロードするデータ・ファイルの中の列の名前を指定します。これらの列名の大文字小文字の区別は、システム・カタログ内の対応する名前の大文字小文字の区別と一致しなければなりません。 NULL 可能ではない各表の列には、METHOD N リスト内に対応する項目が必要です。例えば、データ・フィールドが F1、 F2、 F3、 F4、 F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、 `method N (F2, F1, F4, F3)` は有効な要求ですが、 `method N (F2, F1)` は無効です。この方式は、ファイル・タイプ IXF または CURSOR の場合にのみ使用することができます。

P ロードする入力データ・フィールドのフィールド番号 (1 から始まる) を指定します。 NULL 可能ではない各表の列には、METHOD P リスト内に対応する項目が必要です。たとえば、データ・フィールドが F1、 F2、 F3、 F4、 F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、 `method P (2, 1, 4, 3)` は有効な要求ですが、 `method P (2, 1)` は無効です。この方式は、ファイル・タイプ IXF、DEL、または CURSOR の場合にのみ使用でき、 DEL ファイル・タイプに対してのみ有効な方式です。

XML FROM *xml-path*

XML ファイルが含まれているパスを 1 つ以上指定します。 XDS は、メイン・データ・ファイル (ASC、DEL、または IXF) の、XML 列にロードされる列内に保管されます。

XMLPARSE

XML 文書の解析方法を指定します。このオプションが指定されていない場合、XML 文書の解析の動作は、CURRENT XMLPARSE OPTION 特殊レジスターの値によって決まります。

STRIP WHITESPACE

XML 文書の解析時に空白文字を除去することを指定します。

PRESERVE WHITESPACE

XML 文書の解析時に空白文字を除去しないことを指定します。

XMLVALIDATE

該当する場合に、XML 文書がスキーマに準拠しているかどうかの妥当性検査を実行することを指定します。

USING XDS

メイン・データ・ファイル内の XML Data Specifier (XDS) で識別される XML スキーマに照らし合わせて、XML 文書が妥当性検査されます。デフォルトでは、USING XDS 節によって XMLVALIDATE オプションが呼び出された場合、妥当性検査実行のために使用されるスキーマは、その XDS の SCH 属性によって決まります。 XDS の中で SCH 属性が指定されていない場合、DEFAULT 節によってデフォルト・スキーマが指定されているのでない限り、スキーマ妥当性検査は実行されません。

DEFAULT、IGNORE、および MAP 節を使用することにより、スキーマ決定の動作を変更することができます。これら 3 つの節はオプションであり、相互に適用されるのではなく XDS の指定に直接適用されます。例えば、DEFAULT 節で指定されているためにあるスキーマが選択された場合、それが IGNORE 節で指定されていたとしても無視されることはありません。同じように、MAP 節のペアの最初の部分で指定されているためにあるスキーマが選択された場合、それが別の MAP 節のペアの 2 番目の部分で指定されていたとしても再びマップされることはありません。

USING SCHEMA *schema-sqlid*

指定されている SQL ID の XML スキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。この場合、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

USING SCHEMALOCATION HINTS

ソース XML 文書の中で XML スキーマ・ロケーション・ヒントによって指定されているスキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。その XML 文書の中に schemaLocation 属性が指定されていない場合、妥当性検査は実行されません。 USING SCHEMALOCATION HINTS 節が指定されているなら、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

以下に示す XMLVALIDATE オプションの例を参照してください。

IGNORE *schema-sqlid*

このオプションは、USING XDS パラメーターを指定した場合にのみ使用できます。 IGNORE 節は、SCH 属性によって指定されていても無視するスキーマとして、1 つ以上のスキーマのリストを指定します。 ロードする XML 文書の XML Data Specifier の中に SCH 属性が存在し、その SCH 属性によって指定されるスキーマが IGNORE のスキーマ・リストに含まれている場合には、ロードするその XML 文書についてスキーマ妥当性検査は実行されません。

注:

あるスキーマが IGNORE 節の中で指定されている場合、MAP 節のスキーマ・ペアの左辺にそれを含めることはできません。

IGNORE 節は XDS にのみ適用されます。 あるスキーマが IGNORE 節によって指定されていても、それが MAP 節によってマップされているなら、それ以降そのスキーマが無視されることはありません。

DEFAULT *schema-sqlid*
このオプションは、USING XDS パラメーターを指定した場合にのみ使用できます。 DEFAULT 節で指定されたスキーマは、ロード対象 XML 文書の XML Data Specifier (XDS) に XML スキーマを指定する SCH 属性が含まれていない場合に、妥当性検査のために使用するスキーマとなります。

DEFAULT 節は、IGNORE 節および MAP 節よりも優先されます。 XDS が DEFAULT 節を満たすなら、IGNORE と MAP の指定は無視されます。

MAP *schema-sqlid*
このオプションは、USING XDS パラメーターを指定した場合にのみ使用できます。 MAP 節は、ロードする各 XML 文書について XML Data Specifier (XDS) の SCH 属性によって指定されるスキーマの代わりに使用する代替スキーマを指定するのに使用します。MAP 節には、それぞれがあるスキーマから別のスキーマへのマッピングを表すスキーマ・ペアを 1 つ以上列挙したリストを指定します。 ペア中の最初のスキーマは、XDS 内の SCH 属性によって参照されるスキーマを表します。ペア中の 2 番目のスキーマは、スキーマ検証の実行で使用する必要のあるスキーマを表します。

あるスキーマが MAP 節のスキーマ・ペアの左辺で指定されている場合、IGNORE 節でさらにそれを指定することはできません。

スキーマ・ペアのマッピングが適用されたなら、その結果は最終的なものです。マッピング操作は推移的ではないため、選択されたスキーマが、それ以降に別のスキーマ・ペアのマッピングに適用されることはありません。

スキーマを複数回マップすることはできません。つまり、複数のペアの左辺に指定することはできません。

SAVECOUNT *n*
ロード・ユーティリティーが *n* 行ごとに整合点を取ることを指定します。この値はページ・カウントに変換され、エクステント・サイズのインターバルに切り上げられます。メッセージは整合点において発行されるので、LOAD QUERY を使用してロード操作をモニターする場合には、このオプションを選択する必要があります。 *n* の値が十分な大きさでない場合、各整合点で実行される活動の同期化によってパフォーマンスに影響してしまいます。

デフォルト値はゼロですが、それは、必要がなければ整合点は確立されないことを意味します。

CURSOR ファイル・タイプと併せて指定された場合、このオプションは無視されます。

ROWCOUNT *n*
ロードするファイル内の物理レコードの数 *n* を指定します。ユーザーはファイル内の最初の *n* 個の行だけをロードできます。

WARNINGCOUNT *n*

n 個の警告後に、ロード操作を停止します。このパラメーターは、警告は予期されないが、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。ロード・ファイルまたはターゲット表が不適切に指定されると、ロード対象の各行ごとにロード・ユーティリティーによって警告が生成され、このためにロードが失敗する可能性があります。 *n* がゼロの場合、またはこのオプションが指定されていない場合、何度警告が出されてもロード操作は続行します。警告のしきい値に達したためにロード操作が停止された場合でも、あらためて RESTART モードでロード操作を開始できます。ロード操作は、最後の整合点から自動的に続行します。または、入力ファイルの先頭から REPLACE モードであらためてロード操作を開始できます。

TEMPFILES PATH *temp-pathname*

ロード操作時に一時ファイルを作成する場合に使用するパスの名前を指定します。これはサーバー・データベース・パーティションに従って完全に修飾しなければなりません。

一時ファイルは、ファイル・システムのスペースを使用します。場合によっては、このスペースが相当必要になります。以下に示すのは、すべての一時ファイルにどの程度のファイル・システム・スペースを割り振るべきかの見積もりです。

- ロード・ユーティリティーが生成するメッセージごとに 136 バイト
- データ・ファイルに長フィールド・データまたは LOB が入っている場合は、 15 KB のオーバーヘッド。 INSERT オプションを指定した場合で、表の中に多量の長フィールドまたは LOB データがすでにある場合には、この数値はこれよりもかなり大きくなる場合があります。

INSERT

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。既存の表データを変更することなく、ロードされたデータを表に追加します。

REPLACE

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。表から既存データをすべて削除し、ロードされたデータを挿入します。表定義および索引定義は変更されません。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

KEEPDICTIONARY

LOAD REPLACE 操作の後も、既存のコンプレッション・ディクショナリーを保持します。表の COMPRESS 属性が YES になっていると、新しく置換するデータは、ロードの呼び出し前に存在していたディクショナリーに基づく圧縮の対象になります。表にディクショナリーが存在していなかった場合は、表の COMPRESS 属性が YES になっている限り、置換によって表に挿入されるデータによって新しいディクショナリーが作成されます。この場合、コンプレッション・ディクショナリーを作成するために必要なデータの量は、ADC のポリシーによって左右されます。そのデータは、圧縮されていない状態で表に取り込まれます。表にディクショナリーが挿入されると、その後にロードされる残りのデータは、そのディクショ

ナリーによる圧縮の対象になります。これはデフォルトのパラメーターです。要約を以下の表 1 に示します。

表 27. LOAD REPLACE KEEPDICTIONARY

圧縮	ディクショナリーが存在するかどうか	結果
Y	Y	ディクショナリーを保存します。すべての入力行が既存のディクショナリーによる圧縮の対象になります。
Y	N	十分なユーザー・データが存在する場合にのみ、新しいディクショナリーを表に挿入します。残りの行は、ディクショナリーの作成後に圧縮の対象になります。
N	Y	ディクショナリーを保存します。すべての入力行が圧縮されません。
N	N	影響はありません。すべての行が圧縮されません。

RESETDICTIONARY

表の COMPRESS 属性が YES の場合にこのディレクティブを指定すると、LOAD REPLACE 処理の実行時に、表のデータ・オブジェクトに対応した新しいディクショナリーが作成されます。
COMPRESS 属性が NO で、表の中にディクショナリーがすでに存在している場合は、そのディクショナリーが除去されるだけで、新しいディクショナリーが表に挿入されることはありません。コンプレッション・ディクショナリーは、1 つのユーザー・レコードだけでも作成できます。ロードするデータ・セットのサイズがゼロの場合は、既存のディクショナリーが存在していても、そのディクショナリーは保持されません。このディレクティブを指定した場合、ディクショナリーを作成するために必要なデータの量は、ADC のポリシーに左右されません。要約を以下の表 2 に示します。

表 28. LOAD REPLACE RESETDICTIONARY

圧縮	ディクショナリーが存在するかどうか	結果
Y	Y	新しいディクショナリーを作成します。* ロードする残りの行は、ディクショナリーの作成後に圧縮の対象になります。
Y	N	新しいディクショナリーを作成します。残りの行は、ディクショナリーの作成後に圧縮の対象になります。
N	Y	ディクショナリーを除去します。すべての入力行が圧縮されません。
N	N	影響はありません。すべての行が圧縮されません。

* ディクショナリーが存在し、圧縮属性が有効になっていても、表パーティションにロードするレコードがない場合は、新しいディクショナリーを作成できません。RESETDICTIONARY 操作では、既存のディクショナリーが維持されなくなります。

TERMINATE

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。以前に割り込みを受けたロード操作を終了し、ロード操作が開始された時点まで操作をロールバックします。途中に整合点があっても通過します。その操作に関係する表スペースの状態は通常に戻され、すべての表オブジェクトの整合性が保たれます (索引オブジェクトが無効とマークされる場合がありますが、そのような場合には、次回のアクセス時に索引の再作成が自動的に行われます)。終了するロード操作が LOAD REPLACE の場合、その表は LOAD TERMINATE 操作完了後に空の表まで切り捨てられます。終了するロード操作が LOAD INSERT の場合、その表は LOAD TERMINATE 操作完了後も元のレコードをすべて保持します。ディクショナリー管理の要約を以下の表 3 に示します。

LOAD TERMINATE オプションでは、表スペースのバックアップ・ペンディング状態は解除されません。

RESTART

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。以前に割り込みを受けたロード操作を再開します。ロード操作は、ロード、作成、または削除フェーズの最後の整合点から自動的に続行されます。ディクショナリー管理の要約を以下の表 4 に示します。

INTO *table-name*

データのロード先となるデータベース表を指定します。この表として、システム表または宣言一時表は指定できません。別名、完全修飾、または非修飾の表名を指定することができます。修飾子付き表名は、`schema.tablename` の形式です。非修飾の表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

insert-column

データの挿入先となる表の列を指定します。

ロード・ユーティリティーは、1 つ以上のスペースを使った名前の列を解析できません。例えば、次のようにします。

は、`Int 4` 列があるためエラーになります。これは、次のようにして二重引用符で列名を囲むことによって解決できます。

FOR EXCEPTION *table-name*

エラーが発生した行のコピー先となる例外表を指定します。ユニーク索引または主キー索引に違反した行がすべてコピーされます。非修飾の表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

例外表に書き込まれる情報は、ダンプ・ファイルには書き込まれません 。パーティション・データベース環境では、ロードする表を定義されたデータベース・パーティションの例外表を定義する必要があります。ダンプ・ファイルには、無効であるか構文エラーであるためにロードできない行が入ります。

NORANGEEXC

範囲違反のためにリジェクトされた行は、例外表に挿入しないことを指定します。

NOUNIQUEEXC
ユニーク制約に違反しているためにリジェクトされた行は、例外表に挿入しないことを指定します。

STATISTICS USE PROFILE
この表で定義されているプロファイルに従ってロード中に統計を収集するようロード操作に指示します。そのプロファイルは、ロードの実行前に作成されていなければなりません。そのプロファイルは、RUNSTATS コマンドで作成します。プロファイルが存在しない場合に、プロファイルに従って統計を収集するようロード操作に指示すると、警告メッセージが戻されて統計は収集されません。

STATISTICS NO
統計データを収集せず、したがってカタログ内の統計データも変更しないことを指定します。これはデフォルトです。

COPY NO
順方向リカバリーが使用可能 (つまり、*logretain* または *userexit* がオン) になっていれば、表が存在している表スペースをバックアップ・ペンディング状態にするよう指定します。 COPY NO オプションを使用する場合も、表スペース状態は LOAD IN PROGRESS になります。これは、一時的な状態であり、ロードが完了するか打ち切られると解除されます。表スペースのバックアップまたはデータベースの完全バックアップを実行しない限り、表スペースのどの表のデータも更新または削除できません。ただし、SELECT ステートメントを使用すれば、どの表のデータにもアクセス可能です。

リカバリー可能データベースでの COPY NO を指定した LOAD は、表スペースをバックアップ・ペンディング状態のままにします。例えば、COPY NO を指定した LOAD および INDEXING MODE DEFERRED を実行すると、索引はリフレッシュが必要な状態になります。表での照会には、索引スキャンが必要なものがあり、索引がリフレッシュされるまで、成功しません。バックアップ・ペンディング状態にある表スペース内に常駐する場合、索引はリフレッシュできません。この場合、表へのアクセスは、バックアップが行われるまで許可されません。索引リフレッシュは、索引が照会によってアクセスされたときに、データベースによって自動的に行われます。
COPY NO、COPY YES、NONRECOVERABLE のいずれも指定しない場合に、データベースがリカバリー可能であれば (つまり、**logretain** または **logarchmeth1** が有効になっていれば)、COPY NO がデフォルトになります。

COPY YES
ロードするデータのコピーを保存することを指定します。このオプションは、フォワード・リカバリーが使用不可に設定されている場合には無効です。

USE TSM
Tivoli Storage Manager (TSM) を使ってコピーを保管することを指定します。

OPEN *num-sess* **SESSIONS**
TSM またはベンダー製品とともに使用する入出力セッションの数です。デフォルト値は 1 です。

TO *device/directory*
コピー・イメージを作成する先の装置またはディレクトリーを指定します。

LOAD *lib-name*
使用するバックアップおよびリストア I/O ベンダー関数を含む共有ライブラリー (Windows オペレーティング・システムでは DLL) の名前。絶対パスで指定することができます。絶対パスを指定しない場合、デフォルトでユーザー出口プログラムの存在するパスになります。

NONRECOVERABLE
ロード・トランザクションがリカバリー不能としてマークされており、それ以降のロールフォワード・アクションによってそれをリカバリーさせることは不可能であることを指定します。ロールフォワード・ユーティリティーは、そのトランザクションをスキップし、データのロード先の表に「invalid」(無効) としてマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。ロールフォワード操作が完了すると、そのような表は、ドロップするか、またはリカバリー不能なロード操作完了後のコミット・ポイントの後に取られたバックアップ (全バックアップまたは表スペースのバックアップ) からのみ、リストアすることができます。

このオプションを指定すると、表スペースはロード操作後にバックアップ・ペンディング状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。 COPY NO、COPY YES、NONRECOVERABLE のいずれも指定しない場合に、データベースがリカバリー不能であれば (つまり、**logretain** または **logarchmeth1** が有効になっていなければ)、NONRECOVERABLE がデフォルトになります。

WITHOUT PROMPTING
データ・ファイルのリストにロードするすべてのファイルを含め、しかもリストに入っている装置またはディレクトリーがロード操作全体で十分であるということを指定します。続きの入力ファイルが見つからなかったり、ロード操作が終了する前にコピー先がいっぱいになるとロード操作は失敗し、表はロード・ペンディング状態のままになります。

DATA BUFFER *buffer-size*
ユーティリティー内でデータを転送するためのバッファー・スペースとして使用する 4 KB ページ数を設定します (並列処理の度合いには依存しません)。指定する値がアルゴリズム上の最小値より小さい場合、最小限必要なリソースが使用され、警告は戻されません。

このメモリーは、ユーティリティー・ヒープから直接に割り当てられ、そのサイズは *util_heap_sz* データベース構成パラメーターで修正可能です。

値が指定されていない場合、実行時にユーティリティーによって適切なデフォルトが計算されます。デフォルトは、ローダーのインスタンス生成時にユーティリティー・ヒープで使用できるフリー・スペースの割合と、表の一部の特性に基づいて決まります。

SORT BUFFER *buffer-size*
このオプションは、ロード操作時に SORTHEAP データベース構成パラメー

ターをオーバーライドする値を指定します。これは、索引とともに表をロードする場合、また INDEXING MODE パラメーターが DEFERRED として指定されていない場合にのみ関係があります。指定される値は、SORTHEAP の値を超えることはできません。このパラメーターは、SORTHEAP の値を変更せずに多くの索引を持つ表をロードする際に使用されるソート・メモリーのスロットルで役に立ちます。これは、一般的な照会処理にも影響を与えます。

CPU_PARALLELISM *n*

表オブジェクトの作成時に、レコードの解析、変換、およびフォーマット設定のためにロード・ユーティリティーによって作成されるプロセスまたはスレッドの数を指定します。このパラメーターは、データベース・パーティションごとに実行するプロセス数を活用するために設計されています。これは、事前にソートされたデータをロードする際に役立ちます (ソース・データのレコード順序が保持されるため)。このパラメーターの値が 0 の場合や、このパラメーターを指定しなかった場合、ロード・ユーティリティーは、実行時に自動的に計算された適切なデフォルト値 (通常は使用可能な CPU の数に基づく) を使用します。

注:

1. LOB または LONG VARCHAR フィールドのどちらかの入った表でこのパラメーターを使用する場合、システムの CPU の数またはユーザーが指定した値には関係なく、値は 1 になります。
2. SAVECOUNT パラメーターに指定する値が小さいと、データと表のメタデータの両方をフラッシュするために、ローダーがさらに多くの入出力操作を実行することになります。 CPU_PARALLELISM が 1 より大きいなら、フラッシュ操作は非同期になり、ローダーは CPU を活用できます。 CPU_PARALLELISM が 1 に設定されている場合、ローダーは整合点において入出力を待ちます。 CPU_PARALLELISM を 2 に設定し、SAVECOUNT を 10 000 に設定したロード操作は、CPU が 1 つしかなくても、同じ操作で CPU_PARALLELISM を 1 に設定した場合より速く完了します。

DISK_PARALLELISM *n*

表スペース・コンテナーにデータを書き込むためにロード・ユーティリティーが作成するプロセスまたはスレッドの数を指定します。値を指定しない場合、ユーティリティーは表スペース・コンテナーの数と表の特性に基づいて、自動的に計算された適切なデフォルトを選択します。

FETCH_PARALLELISM YES | NO

DATABASE キーワードを使用してカーソルが宣言されていてカーソルからのロードを実行するとき、または API の `sqlu_remotefetch_entry` メディア項目を使用するとき、このオプションが YES に設定されていると、ロード・ユーティリティーは、リモート・データ・ソースからのフェッチの並列化を試みます (可能な場合)。 NO に設定されている場合、並列フェッチは行われません。デフォルト値は、YES です。詳細については、『*CURSOR ファイル・タイプを使用したデータの移動*』を参照してください。

INDEXING MODE

ロード・ユーティリティーが索引を再作成するのか、それとも索引を増分で拡張するのかを指定します。有効な値は以下のとおりです。

AUTOSELECT

REBUILD モードと INCREMENTAL モードのいずれにするかを、ロード・ユーティリティーが自動的に決定します。決定は、ロードされるデータ量と索引ツリーの深さに基づいて行われます。索引ツリーの深さに関連する情報は索引オブジェクトに保管されています。この情報を設定するために、RUNSTATS は不要です。AUTOSELECT がデフォルトの索引付けモードです。

REBUILD

すべての索引が再作成されます。古い表データの索引キー部分も、追加される新しい表データの索引キー部分もすべてソートできるようにするため、ロード・ユーティリティーには十分なリソースが必要となります。

INCREMENTAL

索引に新しいデータが取り込まれて拡張します。このアプローチでは、索引のフリー・スペースが消費されます。このアプローチでは、新たに挿入されるレコードの索引キーを追加するためのソート・スペースだけがあれば十分です。この方式がサポートされるのは、索引オブジェクトが有効で、かつロード操作の開始時にアクセス可能な場合だけです (例えば、 DEFERRED モードが指定されたロード操作の直後では、この方式は無効です)。このモードを指定したものの、索引の状態などの理由でサポートされない場合は、警告が戻され、REBUILD モードでロード操作が続行されます。同様に、ロード作成フェーズでロード再開操作を開始した場合も、INCREMENTAL モードはサポートされません。

以下の条件がすべて真の場合、増分索引の作成はサポートされません。

- LOAD COPY オプションが指定されている (USEREXIT または LOGRETAIN オプションを指定した *logarchmeth1*)。
- 表が DMS 表スペース内に存在している。
- 索引オブジェクトの存在している表スペースが、ロードしようとしている表に属する他の表オブジェクトによって共有されている。

この制限を迂回するため、索引は別々の表スペースに置くようお勧めします。

DEFERRED

このモードが指定されている場合、ロード・ユーティリティーは索引の作成を試みません。リフレッシュが必要であることを示すマークが索引に付けられます。ロード操作とは関係のないこのような索引に最初にアクセスするときは、再作成が強制的に実行されたり、データベースの再始動時に索引が再作成されたりする場合があります。 このアプローチでは、最も大きい索引のキー部分をすべて処理できるだけのソート・スペースが必要です。索引を作成するために

その後かかる合計時間は、REBUILD モードの場合よりも長くなります。したがって、この索引作成据え置きモードで複数のロード操作を実行する場合、最初の非ロード・アクセス時に索引を再作成できるようにしておくよりも、順序列内の最後のロード操作で索引の再作成を実行できるようにした方が (パフォーマンスの観点から) 賢明であるといえます。

据え置き索引作成がサポートされるのは、非ユニーク索引がある表だけです。そのため、ロード・フェーズで挿入される重複キーがロード操作後は永続的ではなくなります。

ALLOW NO ACCESS

ロードを使用すると、ロード中に、排他的アクセスのためにターゲット表がロックされます。ロード中、表の状態は LOAD IN PROGRESS に設定されます。 ALLOW NO ACCESS はデフォルトの動作です。これは、LOAD REPLACE で唯一有効なオプションです。

表に制約があると、表の状態は、LOAD IN PROGRESS の他に、SET INTEGRITY PENDING に設定されます。表の SET INTEGRITY PENDING 状態を解除するには、SET INTEGRITY ステートメントを使用する必要があります。

ALLOW READ ACCESS

ロードを使用すると、ターゲット表は共用モードでロックされます。表の状態は、LOAD IN PROGRESS および READ ACCESS の両方に設定されます。表のロード中、データの非デルタ部分にアクセスすることができます。つまり、表を読み取る側はロードの開始前に存在していたデータにはアクセスができ、ロード中のデータはロードが完了するまで利用できない、ということです。 ALLOW READ ACCESS ロードの LOAD TERMINATE または LOAD RESTART はこのオプションを使用できますが、ALLOW NO ACCESS ロードの LOAD TERMINATE または LOAD RESTART はこのオプションを使用できません。また、ターゲット表上の索引が要再作成のマークが付けられると、このオプションは無効になります。

表に制約があると、表の状態は、LOAD IN PROGRESS、および READ ACCESS の他に、SET INTEGRITY PENDING に設定されます。ロードの終了時に、表の状態 LOAD IN PROGRESS は解除されますが、表の状態 SET INTEGRITY PENDING および READ ACCESS はそのまま残ります。表の SET INTEGRITY PENDING を解除するには、SET INTEGRITY ステートメントを使用する必要があります。表が SET INTEGRITY PENDING および READ ACCESS の状態にある間、データの非デルタ部分には引き続き読み取りアクセスできますが、データの新しい (デルタ) 部分には、 SET INTEGRITY ステートメントが完了するまでアクセス不能のままになります。ユーザーは、SET INTEGRITY ステートメントを発行しないで、同じ表上で複数のロードを実行できます。ただし、元の (チェック済み) データは、 SET INTEGRITY ステートメントが発行されるまで可視のままです。

ALLOW READ ACCESS は、以下の修飾子もサポートします。

USE *tablespace-name*

索引が再作成される場合、表スペース *tablespace-name* に索引のシャドー・コピーが作成され、ロード終了時の INDEX COPY PHASE で、元の表スペース上にコピーされます。 SYSTEM TEMPORARY

表スペースのみ、このオプションを使用できます。指定されない場合、シャドー索引が、索引オブジェクトと同じ表スペース内に作成されます。シャドー・コピーが索引オブジェクトと同じ表スペース内に作成される場合、古い索引オブジェクトを介したシャドー索引オブジェクトのコピーは瞬時に終了します。シャドー・コピーが索引オブジェクトとは異なる表スペースにある場合、物理コピーが実行されます。これにはかなりの入出力および時間を要します。コピーは、表がオフラインの間、ロード終了時の INDEX COPY PHASE で行われます。

このオプションをしないと、シャドー索引は元の索引と同じ表スペースに作成されます。デフォルトでは、元の索引とシャドー索引の両方が同時に同じ表スペースに常駐するため、 1 つの表スペース内に両方の索引を保留するためのスペースが不足する場合があります。このオプションを使用すれば、索引用の十分な表スペースを確保できます。

ユーザーが INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT を指定しない場合、このオプションは無視されます。このオプションは INDEXING MODE AUTOSELECT が選択され、ロードが索引を徐々に更新することを選択した場合にも無視されます。

SET INTEGRITY PENDING CASCADE

LOAD によって表が SET INTEGRITY PENDING 状態になる場合、 SET INTEGRITY PENDING CASCADE オプションを使用することによって、ユーザーはロードされる表の SET INTEGRITY PENDING 状態を即時にすべての下層 (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を含む) にカスケードするかどうか指定することができます。

IMMEDIATE

SET INTEGRITY PENDING 状態が即時にすべての下層外部キー表、下層即時マテリアライズ照会表、および下層ステージング表に拡張されることを示します。 LOAD INSERT 操作の場合、 IMMEDIATE オプションが指定されている場合でも、 SET INTEGRITY PENDING 状態は下層外部キー表に拡張されません。

後で (SET INTEGRITY ステートメントの IMMEDIATE CHECKED オプションを使用して) ロードされる表の制約違反をチェックする際、SET INTEGRITY PENDING READ ACCESS 状態だった下層外部キー表は、SET INTEGRITY PENDING NO ACCESS 状態になります。

DEFERRED

ロードされる表だけが、SET INTEGRITY PENDING 状態になることを示します。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は、未変更のままになります。

下層外部キー表は、(SET INTEGRITY ステートメントの IMMEDIATE CHECKED オプションを使用して) その親表の制約違反がチェックされるとき、後で暗黙的に SET INTEGRITY PENDING 状態になる場合があります。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は、その基礎表のいずれかの保全性違反がチェックされる際、暗黙的に SET INTEGRITY PENDING 状態になります。従属表が SET INTEGRITY PENDING 状態になったことを示す警告が戻されます (SQLSTATE 01586)。この下層表がいつ SET INTEGRITY PENDING 状態になるかについては、 SQL リファレンスにある SET INTEGRITY ステートメントの「注」の項を参照してください。

SET INTEGRITY PENDING CASCADE オプションが指定されない場合、次のようになります。

- ロードされる表だけが、SET INTEGRITY PENDING 状態になります。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状態は、未変更のままになり、後にロードされた表の制約違反がチェックされる際に、暗黙的に SET INTEGRITY PENDING 状態になる場合があります。

LOAD によってターゲット表が SET INTEGRITY PENDING 状態にならない場合、SET INTEGRITY PENDING CASCADE オプションは無視されます。

LOCK WITH FORCE

ユーティリティーはロード・プロセス中に、表ロックなどの様々なロックを獲得します。ロックを獲得する際、このオプションを使用すると、ロードは待機することなく、またタイムアウトになることなく、ターゲット表に競合するロックを持つ他のアプリケーションを強制的にオフにします。システム・カタログ表に対する競合するロックを保留するアプリケーションは、ロード・ユーティリティーによって強制的にオフにされることはありません。強制されたアプリケーションは、ロールバックし、ロード・ユーティリティーが必要とするロックをリリースします。その後、ロード・ユーティリティーを続行できます。このオプションは、FORCE APPLICATIONS コマンドと同じ権限 (SYSADM または SYSCTRL) を必要とします。

ALLOW NO ACCESS は、ロード操作の開始時に競合するロックを持つアプリケーションを強制的にロールバックさせる場合があります。ロードの開始時に、ユーティリティーは、表の照会または変更を試みているアプリケーションを強制的にロールバックさせる場合があります。

ALLOW READ ACCESS は、ロード操作の開始時および終了時に競合するロックを持つアプリケーションを強制的にロールバックさせる場合があります。ロードの開始時に、ロード・ユーティリティーは、表の変更を試みているアプリケーションを強制的にロールバックさせる場合があります。ロード操作の終了時に、ロード・ユーティリティーは、表の照会または変更を試みているアプリケーションを強制的にロールバックさせる場合があります。

SOURCEUSEREXIT*executable*

このユーティリティーにデータを送るために呼び出される実行可能ファイル名を指定します。

REDIRECT

INPUT FROM

BUFFER *input-buffer*

input-buffer で指定されたバイトのストリームが、所

定の実行可能ファイルを実行するプロセスの
STDIN ファイル記述子に渡されます。

FILE *input-file*
このクライアント・サイドのファイルの内容が、所定の実行可能ファイルを実行するプロセスの
STDIN ファイル記述子に渡されます。

OUTPUT TO

FILE *output-file*
STDOUT および STDERR ファイル記述子が、指定した完全に修飾されたサーバー・サイドのファイルに取り込まれます。

PARALLELIZE
複数のユーザー出口プロセスを同時に呼び出すことによって、ロード・ユーティリティーへのデータ入力のスループットを高めます。このオプションは、複数パーティション・データベース環境でのみ適用でき、単一パーティション・データベース環境では無視されます。

詳細については、『*カスタマイズしたアプリケーション (ユーザー出口) を使用したデータの移動*』を参照してください。

PARTITIONED DB CONFIG *partitioned-db-option*
複数のデータベース・パーティションに分散した表へのロードの実行を可能にします。 PARTITIONED DB CONFIG パラメーターを使用すると、パーティション・データベース固有の構成オプションを指定することができます。 *partitioned-db-option* の値は、以下のいずれかになります。

```
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

これらのオプションの詳しい説明については、『*パーティション・データベース環境でのロード構成オプション*』を参照してください。

RESTARTCOUNT
予約済み。

USING *directory*
予約済み。

XML 文書からデータをロードする例

XML データのロード

例 1

ユーザーは、表に挿入する文書を記述するために、XDS フィールドを使用してデータ・ファイルを構成しました。内容は以下のとおりです。

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

第 1 行では、XML 文書が file1.xml というファイル名で指定されています。文字区切りである二重引用符が XDS 内でも使用されているので、XDS 内の二重引用符は二重になっています。第 2 行では、XML 文書が file2.xml というファイル名で指定されています。その文書の開始点のバイト・オフセットは 23、長さは 45 バイトです。

例 2

ユーザーは、XML 列の構文解析や妥当性検査のオプションを指定しないでロード・コマンドを実行し、データのロードに成功します。

```
LOAD FROM data.del of DEL INSERT INTO mytable
```

CURSOR からの XML データのロード

カーソルからデータをロードする操作は、通常のリレーショナル列タイプの場合と同じです。ユーザーには 2 つの表 T1 と T2 があり、それぞれは C1 という 1 つの XML 列だけで構成されています。T1 から T2 への LOAD を実行するために、ユーザーはまずカーソルを宣言します。

```
DECLARE X1 CURSOR FOR SELECT C1 FROM T1;
```

次に、ユーザーは、カーソル・タイプを使用して LOAD を実行します。

```
LOAD FROM X1 of CURSOR INSERT INTO T2
```

カーソル・タイプに XML 固有の LOAD オプションを適用する操作は、ファイルからロードする場合と同じです。

使用上の注意

- データは、入力ファイル内に並んでいる順序でロードされます。特定の順序を希望する場合には、ロードが試行される前にデータをソートしてください。ソース・データの順序を保持する必要がなければ、ANYORDER ファイル・タイプ修飾子を使用できます。この修飾子については、以下の『*ロード・ユーティリティーのファイル・タイプ修飾子*』セクションを参照してください。
- ロード・ユーティリティーは、既存の定義に基づいて索引を作成します。ユニーク・キーの重複を処理するのに、例外表が使用されます。ユーティリティーは、参照保全を強制したり、制約検査を実行したり、ロードする表に従属するマテリアライズ照会表を更新したりすることはありません。参照制約またはチェック制約を含む表は、SET INTEGRITY ペンディング状態になります。 REFRESH IMMEDIATE として定義されているサマリー表、およびロードする表に依存する

サマリー表もまた、SET INTEGRITY ペンディング状態になります。この表に関して、SET INTEGRITY ペンディング状態を解除するには、SET INTEGRITY ステートメントを発行してください。ロード操作は、複製されたマテリアライズ照会表に対しては実行できません。

- クラスタリング索引が表に存在する場合、ロード前にクラスタリング索引でデータをソートしてください。ただし、データはマルチディメンション・クラスタリング (MDC) 表にロードする前にソートする必要はありません。
- 保護された表へのロード時に例外表を指定すると、無効なセキュリティー・ラベルで保護されている行がその表に送られます。そのため、例外表にアクセスできるユーザーは、通常はアクセス権限のないデータにアクセスできてしまう可能性があります。セキュリティー・レベルを上げるために、誰に例外表アクセス権限を付与するかに注意し、行が修復されてロードする表にコピーされたら直ちにそれぞれの行を削除するとともに、使い終えた例外表は直ちにドロップしてください。
- 内部形式のセキュリティー・ラベルには、改行文字が含まれている可能性があります。 DEL ファイル形式を使用するファイルをロードする場合、この改行文字が区切り文字と間違われることがあります。この問題が起きた場合は、LOAD コマンドで delprioritychar ファイル・タイプ修飾子を指定することによって、区切り文字に以前のデフォルト優先順位を使用してください。
- DECLARE CURSOR コマンドの実行中に指定した DATABASE キーワードが CURSOR ファイル・タイプを使用してロードを実行する場合、現在接続されているデータベース (ロード用) の認証に使用されるユーザー ID およびパスワードが (DECLARE CURSOR コマンドの DATABASE オプションによって指定された) ソース・データベースの認証に使用されます。ユーザー ID またはパスワードがロード・データベースの接続に指定されない場合、ソース・データベースのユーザー ID とパスワードは DECLARE CURSOR コマンドの実行中に指定する必要があります。
- マルチパート PC/IXF ファイルの個々のパートを Windows システムから AIX システムにコピーするロード操作もサポートされています。すべてのファイルの名前を LOAD コマンドに指定する必要があります。例えば、LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1 のように記述します。論理分割 PC/IXF ファイルから Windows オペレーティング・システムにロードする操作は、サポートされていません。
- 失敗した LOAD を再開する場合の動作は、既存の動作と同じで、BUILD フェーズでは、索引の REBUILD モードの使用が強制されます。

LOAD TERMINATE と LOAD RESTART のディクショナリー管理のまとめ

TERMINATE ディレクティブの下で LOAD 処理を実行する場合のコンプレッション・ディクショナリー管理の動作を以下の表にまとめます。

表 29. LOAD TERMINATE のディクショナリー管理

表の COMPRESS 属性	LOAD の前にディクショナリーが存在するかどうか	TERMINATE: LOAD REPLACE KEEPDICTIONARY または LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	既存のディクショナリーを維持します。	何も維持しません。
YES	NO	何も維持しません。	何も維持しません。
NO	YES	既存のディクショナリーを維持します。	何も維持しません。
NO	NO	Do nothing.	Do nothing.

LOAD RESTART は、到達した最後の整合点まで表を切り捨てます。最後の LOAD 整合点が取られた時点で表にコンプレッション・ディクショナリーが存在していた場合は、LOAD RESTART 処理によって、コンプレッション・ディクショナリーが表に配置されます。その場合、LOAD RESTART が新しいディクショナリーを作成するわけではありません。考えられる条件を以下の表 4 にまとめます。

表 30. LOAD RESTART のディクショナリー管理

表の COMPRESS 属性	LOAD の整合点の前にディクショナリーが存在するかどうか	RESTART: LOAD REPLACE KEEPDICTIONARY または LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	既存のディクショナリーを維持します。	既存のディクショナリーを維持します。
YES	NO	ADC に基づいてディクショナリーを作成します。	ディクショナリーを作成します。
NO	YES	既存のディクショナリーを維持します。	既存のディクショナリーを除去します。
NO	NO	Do nothing.	Do nothing.

ロード・ユーティリティーのファイル・タイプ修飾子

表 31. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式

修飾子	説明
anyorder	この修飾子は、*cpu_parallelism* パラメーターと一緒に使用します。ソース・データの順序を保持する必要はない、という意味の指定なので、SMP システムのパフォーマンスがさらに向上します。*cpu_parallelism* の値が 1 になっていると、このオプションは無視されます。このオプションは、SAVECOUNT > 0 の場合はサポートされません。整合点の後のクラッシュ・リカバリーでは、順序のとおりにデータをロードする必要があるからです。
generatedignore	この修飾子を指定すると、ロード・ユーティリティーは、データ・ファイルに入っている、すべての生成済み列のデータを無視するようになります。その結果、すべての生成済み列の値がユーティリティーによって生成されます。この修飾子を `generatedmissing` 修飾子または `generatedoverride` 修飾子と一緒に使用することはできません。

表 31. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
generatedmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに生成済み列のデータが入っていない (NULL もない) という想定で動作します。その結果、すべての生成済み列の値がユーティリティーによって生成されます。この修飾子を generatedignore 修飾子または generatedoverride 修飾子と一緒に使用することはできません。

generatedoverride	この修飾子を指定すると、ロード・ユーティリティーは、表の中のすべての生成済み列でユーザー指定データを受け入れるようになります (この種の列の通常の規則とは反対の動作です)。別のデータベース・システムからデータをマイグレーションする場合や、ROLLFORWARD DATABASE コマンドの RECOVER DROPPED TABLE オプションを使用してリカバリーしたデータから表をロードする場合は、この修飾子を使用すると便利です。この修飾子を使用すると、NULL 不可の生成済み列にデータのない行や NULL データが入っている行は、リジェクトされます (SQL3116W)。この修飾子を使用すると、表は Set Integrity Pending 状態になります。ユーザー指定値を検証しないで表の Set Integrity Pending 状態を解除する場合は、ロード操作の後に以下のコマンドを実行します。
	`SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED`
	ユーザー指定値の検証を強制実行して表の Set Integrity Pending 状態を解除する場合は、ロード操作の後に以下のコマンドを実行します。
	`SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.`
	この修飾子を指定した場合、パーティション・キー、ディメンション・キー、分散キーのいずれかに生成済み列があれば、LOAD コマンドの実行時にその修飾子が自動的に generatedignore に変換され、ロードの処理が進められます。つまり、生成済み列のすべての値が再生成される結果になります。
	この修飾子を generatedmissing 修飾子または generatedignore 修飾子と一緒に使用することはできません。

identityignore	この修飾子を指定すると、ロード・ユーティリティーは、データ・ファイルに入っている、ID 列のデータを無視するようになります。その結果、すべての IDENTITY 値がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を identitymissing 修飾子または identityoverride 修飾子と一緒に使用することはできません。
identitymissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ID 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。この修飾子を identityignore 修飾子または identityoverride 修飾子と一緒に使用することはできません。

表 31. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
identityoverride	この修飾子を使用するのは、GENERATED ALWAYS として定義されている ID 列がロード対象の表に存在している場合に限られます。この修飾子を指定すると、ユーティリティーは、そのような列で明示的な非 NULL データを受け入れるようになります (この種の ID 列の通常の規則とは反対の動作です)。表を GENERATED ALWAYS として定義しなければならない状況で別のデータベース・システムからデータをマイグレーションする場合や、 ROLLFORWARD DATABASE コマンドの DROPPED TABLE RECOVERY オプションを使用してリカバリーしたデータから表をロードする場合は、この修飾子を使用すると便利です。この修飾子を使用すると、ID 列にデータのない行や NULL データが入っている行は、リジェクトされます (SQL3116W)。この修飾子を `identitymissing` 修飾子または `identityignore` 修飾子と一緒に使用することはできません。このオプションを使用すると、ロード・ユーティリティーは、表の ID 列に入っている値のユニーク性を保持したり検証したりする操作を試行しなくなります。
indexfreespace=*x*	*x* は、0 から 99 までの (両端を含む) 整数です。この値は、ロード操作で索引を再作成するときに、各索引ページに残すフリー・スペースのパーセンテージとして解釈されます。INDEXING MODE INCREMENTAL を指定したロード操作では、このオプションが無視されます。ページの最初の項目は、制限なしで追加されます。その後の項目は、フリー・スペースのパーセンテージしきい値を保持するために追加されます。デフォルト値は、CREATE INDEX の実行時に使用されていた値です。 この値は、CREATE INDEX ステートメントで指定されている PCTFREE 値よりも優先されます。 indexfreespace オプションの対象になるのは、索引のリーフ・ページだけです。
lobsinfile	*lob-path* では、LOB データが含まれているファイルのパスを指定します。ASC、DEL、IXF のロード入力ファイルには、LOB 列に LOB データが入っているファイルの名前が含まれています。 このオプションを CURSOR ファイル・タイプと併用することはできません。 lobsinfile 修飾子を使用するときには、LOB ファイルの配置場所を LOBS FROM 節で指定します。LOBS FROM 節を指定すると、LOBSINFILE の動作が暗黙的にアクティブになります。LOAD ユーティリティーは、データをロードするときに、LOB ファイルを検索するためのパスのリストを LOBS FROM 節から受け取ります。 各パスには、データ・ファイルの LOB ロケーション指定子 (LLS) によって参照されている LOB が少なくとも 1 つ入っているファイルが 1 つ以上含まれています。LLS は、LOB ファイル・パスに格納されているファイルの LOB の位置を示したストリング表記です。LLS の形式は、*filename.ext.nnn.mmm/* になります (*filename.ext* は、LOB が含まれているファイルの名前、*nnn* は、そのファイルに入っている LOB のオフセット (バイト単位)、*mmm* は、その LOB の長さ (バイト単位) です)。例えば、データ・ファイルにストリング `db2exp.001.123.456/` が格納されている場合は、ファイル db2exp.001 のオフセット 123 に LOB が配置されていて、その長さは 456 バイトということになります。 NULL LOB を指定する場合は、サイズとして -1 を入力します。サイズとして 0 を指定すると、長さ 0 の LOB として処理されます。長さ -1 の NULL LOB の場合は、オフセットとファイル名が無視されます。例えば、NULL LOB の LLS は、db2exp.001.7.-1/ のようになります。

表 31. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
noheader	ヘッダー検査コードをスキップします (該当するのは、単一パーティションのデータベース・パーティション・グループに存在する表へのロード操作だけです)。 単一パーティションのデータベース・パーティション・グループに存在する表に対してデフォルトの MPP ロード (モード PARTITION_AND_LOAD) を使用する場合は、ファイルにヘッダーが存在するとは考えられません。したがって、noheader 修飾子を指定する必要はありません。LOAD_ONLY モードを使用する場合は、ファイルにヘッダーが存在すると考えられます。noheader 修飾子が必要になるのは、ヘッダーのないファイルを使用して LOAD_ONLY 操作を実行する場合に限られます。
norowwarnings	リジェクトされた行についてのすべての警告を抑止します。
pagefreespace=*x*	*x* は、0 から 100 までの (両端を含む) 整数です。この値は、各データ・ページに残すフリー・スペースのパーセンテージとして解釈されます。最小行サイズのため、指定した値が無効である場合 (例えば、長さが少なくとも 3 000 バイトの行で、*x* の値が 50 である場合)、その行は新しいページに置かれます。値として 100 を指定すると、各行が新しいページに配置されます。表の PCTFREE 値は、ページごとに指定されたフリー・スペースの量を決定します。ロード操作の pagefreespace 値または表の PCTFREE 値が設定されていないと、ユーティリティーはそれぞれのページで可能なかぎり多くのスペースを満たします。pagefreespace に設定されている値は、表で指定されている PCTFREE 値をオーバーライドします。
rowchangetimestampignore	この修飾子を指定すると、ロード・ユーティリティーは、データ・ファイルに入っている、ROW CHANGE TIMESTAMP 列のデータを無視するようになります。その結果、すべての ROW CHANGE TIMESTAMP がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を `rowchangetimestampmissing` 修飾子または `rowchangetimestampoverride` 修飾子と一緒に使用することはできません。
rowchangetimestampmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ROW CHANGE TIMESTAMP 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。この修飾子を `rowchangetimestampignore` 修飾子または `rowchangetimestampoverride` 修飾子と一緒に使用することはできません。

表 31. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
rowchangetimestampoverride	この修飾子を使用するのは、GENERATED ALWAYS として定義されている ROW CHANGE TIMESTAMP 列がロード対象の表に存在している場合に限られます。この修飾子を指定すると、ユーティリティーは、そのような列で明示的な非 NULL データを受け入れるようになります (この種の ROW CHANGE TIMESTAMP 列の通常の規則とは反対の動作です)。表を GENERATED ALWAYS として定義しなければならない状況で別のデータベース・システムからデータをマイグレーションする場合や、 ROLLFORWARD DATABASE コマンドの DROPPED TABLE RECOVERY オプションを使用してリカバリーしたデータから表をロードする場合は、この修飾子を使用すると便利です。この修飾子を使用すると、ROW CHANGE TIMESTAMP 列にデータのない行や NULL データが入っている行は、リジェクトされます (SQL3116W)。この修飾子を `rowchangetimestampmissing` 修飾子または `rowchangetimestampignore` 修飾子と一緒に使用することはできません。このオプションを使用すると、ロード・ユーティリティーは、表の ROW CHANGE TIMESTAMP 列に入っている値のユニーク性を保持したり検証したりする操作を試行しなくなります。

seclabelchar	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、ストリング・フォーマットのセキュリティー・ラベル値であることを指定します。LOAD は、ロード時に各セキュリティー・ラベルを内部形式に変換します。ストリングが正しい形式になっていないと、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3242W) が戻されます。ストリングが、表を保護するセキュリティー・ポリシーの一部である有効なセキュリティー・ラベルに対応していなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3243W) が戻されます。
	
	seclabelname 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、ロードは失敗し、エラー (SQLCODE SQL3525N) が戻されます。
	
	1 つの `DB2SECURITYLABEL` 列だけで構成されている表の場合は、データ・ファイルを以下のように記述します。
	
	`"CONFIDENTIAL:ALPHA:G2"`
	`"CONFIDENTIAL;SIGMA:G2"`
	`"TOP SECRET:ALPHA:G2"`
	
	このデータのロードまたはインポートでは、以下のように SECLABELCHAR ファイル・タイプ修飾子を使用する必要があります。
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1`

表 31. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明

seclabelname	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、名前で示されていることを指定します。LOAD は、その名前に対応する適切なセキュリティー・ラベルがあれば、その名前をそのセキュリティー・ラベルに変換します。表を保護するセキュリティー・ポリシーに、その名前に対応するセキュリティー・ラベルが存在しなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3244W) が戻されます。
	
	seclabelchar 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、ロードは失敗し、エラー (SQLCODE SQL3525N) が戻されます。
	
	1 つの DB2SECURITYLABEL 列だけで構成されている表の場合は、データ・ファイルに以下のようなセキュリティー・ラベル名を組み込みます。
	
	`"LABEL1"`
	`"LABEL1"`
	`"LABEL2"`
	
	このデータのロードまたはインポートでは、以下のように SECLABELNAME ファイル・タイプ修飾子を使用する必要があります。
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1`
	
	注: ファイル・タイプが ASC の場合、セキュリティー・ラベル名の後のスペースは、名前の一部と解釈されます。そのような動作を避けるには、striptblanks ファイル・タイプ修飾子を使用して、スペースを除去するようにします。

totalfreespace=*x*	*x* は、0 以上の整数です。この値は、表の合計ページ数に対する、フリー・スペースとして表の末尾に追加するページ数のパーセンテージとして解釈されます。例えば、*x* が 20 で、データのロード後に表に 100 個のデータ・ページがある場合は、20 個の空ページが追加されます。表のデータ・ページの総数は、120 になります。データ・ページの総数は、表の索引ページの数に関する因子にはなりません。このオプションは、索引オブジェクトには影響しません。このオプションを指定して 2 つのロード操作を実行する場合、最初のロード操作で末尾に追加されたスペースが 2 番目のロード操作で再利用されるわけでありません。
usedefaults	ターゲット表の列のソース列が指定されていても、1 つ以上の行インスタンスでその列にデータが入っていない場合は、デフォルト値がロードされます。欠落データの例を以下に示します。 • DEL ファイル: 列の値として、2 つの隣接した列区切り (″,,″) や、任意の数のスペースで分離した 2 つの列区切り (″, ,″) が指定されている場合。 • DEL/ASC/WSF ファイル: 十分な数の列がない行や、元の指定に対応した十分な長さがない行。ASC ファイルの場合、NULL 列値は、明示的な欠落とは見なされず、NULL 列値の代わりにデフォルトが入ることもありません。数値、日付、時刻、タイム・スタンプの列では、全桁スペース文字で NULL 列値を表記します。また、どのタイプの列でも、NULL INDICATOR を使用すれば、その列が NULL であることを示せます。 このオプションを指定しない場合に、行インスタンスのソース列にデータが入っていないと、以下のいずれかの動作が発生します。 • DEL/ASC/WSF ファイル: 列が NULL 可能であれば、NULL がロードされます。列が NULL 可能でなければ、ユーティリティーによって行がリジェクトされます。

表 32. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL)

修飾子	説明
codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、入力データ・セットに含まれているデータのコード・ページとして解釈されます。ロード操作の実行時に、文字データ (および文字で指定されている数値データ) は、このコード・ページからデータベース・コード・ページに変換されます。 以下の規則が適用されます。 • DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。 • EBCDIC コード・ページで指定された DEL データの場合、区切り文字は DBCS のシフトイン文字およびシフトアウト文字と同じであってはなりません。 • nullindchar では、標準の ASCII セットのコード・ポイント x20 から x7F の範囲 (両端を含む) に含まれているシンボルを指定する必要があります。この修飾子では、ASCII のシンボルとコード・ポイントを参照します。EBCDIC データでは、コード・ポイントが違っていても、対応するシンボルを使用できます。 このオプションを CURSOR ファイル・タイプと併用することはできません。
dateformat="*x*"	*x* は、ソース・ファイルの日付の形式です。[1] 有効な日付エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (1 から 12 の範囲の 2 桁の数。 M とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月エレメントとは 相互に排他的) 指定されていないそれぞれのエレメントには、デフォルト値の 1 が割り当てられます。日付形式の例を以下に示します。 "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"

表 32. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明

dumpfile = *x*	*x* は、リジェクトされた行を書き込む例外ファイルの (サーバー・データベース・パーティションに応じた) 完全修飾名です。1 つのレコードごとに、最大 32 KB のデータが書き込まれます。以下は、ダンプ・ファイルの指定方法を示す例です。
	`db2 load from data of del`
	`modified by dumpfile = /u/user/filename`
	`insert into table_name`
	
	このファイルは、インスタンス所有者によって作成され、所有されます。デフォルトのファイル許可をオーバーライドする場合は、`dumpfileaccessall` ファイル・タイプ修飾子を使用します。
	注:
	1. パーティション・データベース環境では、ロードする側のデータベース・パーティションから見てローカルのパスを指定する必要があります。そうすれば、同時に実行するいくつかのロード操作が同じファイルに書き込もうとする、という事態を避けられます。
	2. ファイルの内容は、非同期のバッファー・モードでディスクに書き込まれます。ロード操作が失敗した場合や割り込みが発生した場合は、ディスクにコミットされたレコードの数を確実に把握する方法がありません。LOAD RESTART 後の整合性も保証できません。ファイルのロード操作が完了したと想定できるのは、ロード操作が開始と完了が 1 回のパスで完結している場合に限られます。
	3. 指定したファイルがすでに存在している場合は、再作成ではなく追加になります。

dumpfileaccessall	ダンプ・ファイルの作成時に、読み取りアクセスを 'OTHERS' に付与します。 このファイル・タイプ修飾子が有効なのは、以下の場合に限られます。 1. `dumpfile` ファイル・タイプ修飾子と併用する場合 2. ユーザーがロードのターゲット表に対する SELECT 特権を持っている場合 3. UNIX オペレーティング・システムに配置されている DB2 サーバー・データベース・パーティションで実行する場合 指定したファイルがすでに存在している場合は、そのファイルの許可が変更なしでそのまま使用されます。
fastparse	注意して使用してください。ユーザー指定の列値の構文検査が削減されるので、パフォーマンスは向上します。表の正しいアーキテクチャーは確保できますが (つまり、ユーティリティーは、セグメンテーション違反やトラップを回避するための十分なデータ・チェックを実行しますが)、データの一貫性に関する検証は実行しません。このオプションを使用するのは、データの一貫性と正確さに自信がある場合だけにしてください。例えば、ユーザー指定のデータに無効なタイム・スタンプ列値 `:1>0-00-20-07.11.12.000000` が含まれている場合でも、FASTPARSE が指定されていれば、その値は表に挿入されてしまいますが、FASTPARSE が指定されていなければ、その値はリジェクトされます。
implieddecimal	暗黙の小数点の位置が列定義によって決まるようになり、値の末尾という想定がなくなります。例えば、値 `12345` は DECIMAL(8,2) 列に `12345.00` としてではなく、`123.45` としてロードされます。 この修飾子を `packeddecimal` 修飾子と一緒に使用することはできません。

表 32. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
timeformat="x"	x は、ソース・ファイルの時刻の形式です。[1] 有効な時刻エレメントは、以下のとおりです。 H - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数) HH - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 2 桁の数; H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) 指定されていないそれぞれのエレメントには、デフォルト値の 0 が割り当てられます。時刻形式の例を以下に示します。 "HH:MM:SS" "HH.MM TT" "SSSSS"

表 32. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
timestampformat="*x*"	*x* は、ソース・ファイルのタイム・スタンプの形式です。[1] 有効なタイム・スタンプ・エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (01 から 12 の 2 桁の数。 M および MMM とは相互に排他的) MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月のエレメントとは相互に排他的) H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 1 桁または 2 桁の数。) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数。 H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M (分) とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的) UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999990 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM)

表 32. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明

timestampformat="x" (続き)	指定されていない YYYY、M、MM、D、DD、DDD のいずれかのエレメントには、デフォルト値の 1 が割り当てられます。指定されていない MMM エレメントには、デフォルト値の 'Jan' が割り当てられます。指定されていない他のすべてのエレメントには、デフォルト値の 0 が割り当てられます。タイム・スタンプ形式の例を以下に示します。
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	MMM エレメントの有効な値は、
	'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov'、'dec' です。これらの値では、大/小文字は区別されません。
	TIMESTAMPFORMAT 修飾子を指定しなかった場合、ロード・ユーティリティーは、タイム・スタンプ・フィールドで以下の 2 つの有効な形式のいずれかを使用します。
	`YYYY-MM-DD-HH.MM.SS`
	`YYYY-MM-DD HH:MM:SS`
	ロード・ユーティリティーは、DD と HH の間の区切り文字に基づいて形式を選択します。ダッシュ '-' になっていれば、ロード・ユーティリティーは、通常のダッシュとドットの形式 (YYYY-MM-DD-HH.MM.SS) を使用します。ブランク・スペースになっていれば、ロード・ユーティリティーは、HH と MM と SS の間を区切るためにコロン ':' を使用します。
	どちらの形式でも、マイクロ秒のフィールド (UUUUUU) を組み込むと、ロード・ユーティリティーは、区切り文字としてドット '.' を使用します。YYYY-MM-DD-HH.MM.SS.UUUUUU も YYYY-MM-DD HH:MM:SS.UUUUUU も有効です。
	ユーザー定義の日付と時刻の形式が含まれているデータを schedule という表にロードする例を以下に示します。
	`db2 load from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

表 32. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
usegraphiccodepage	usegraphiccodepage を指定すると、グラフィックまたは 2 バイト文字のラージ・オブジェクト (DBCLOB) データ・フィールドにロードするデータは、グラフィック・コード・ページのデータであるという想定で、処理が行われます。残りのデータは、文字コード・ページのデータであるという想定になります。グラフィック・コード・ページは、文字コード・ページに関連付けられています。LOAD は、codepage 修飾子が指定されていればその修飾子によって、codepage 修飾子が指定されていなければデータベースのコード・ページによって、文字コード・ページを判別します。 ドロップ済み表のリカバリーで生成される区切り付きデータ・ファイルとこの修飾子を併用するのは、リカバリーする表にグラフィック・データが入っている場合に限られます。 **制約事項** EXPORT ユーティリティーで作成される DEL ファイルでは、usegraphiccodepage 修飾子を指定しないでください。そのファイルには、1 つのコード・ページでエンコードされたデータだけが入っているからです。usegraphiccodepage 修飾子は、ファイルに含まれている 2 バイト文字ラージ・オブジェクト (DBCLOB) でも無視されます。
xmlchar	XML 文書が文字コード・ページでエンコードされていることを指定します。 指定の文字コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードが文字コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。文字コード・ページは、codepage ファイル・タイプ修飾子で指定されている値か、その修飾子が指定されていない場合はアプリケーション・コード・ページになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。
xmlgraphic	XML 文書が指定のグラフィック・コード・ページでエンコードされていることを指定します。 指定のグラフィック・コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードがグラフィック・コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。グラフィック・コード・ページは、codepage ファイル・タイプ修飾子で指定されている値のグラフィック・コンポーネントか、その修飾子が指定されていない場合はアプリケーション・コード・ページのグラフィック・コンポーネントになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。

表 33. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASC ファイル形式 (区切りなし ASCII)

修飾子	説明
binarynumerics	数値データ (DECIMAL ではない) は、文字表現ではなく 2 進形式でなければなりません。このようにして、コストのかかる変換を回避します。 このオプションは、reclen オプションで指定する固定長レコードを使用する定位置 ASC でのみサポートされています。 以下の規則が適用されます。 • データ・タイプ間の変換は実行されません (ただし、BIGINT、INTEGER、SMALLINT は例外です)。 • データ長は、ターゲット列の定義と一致している必要があります。 • FLOAT は、IEEE 浮動小数点形式でなければなりません。 • ロード・ソース・ファイルのバイナリー・データは、ロード操作を実行するプラットフォームにかかわりなく、ビッグ・エンディアンであると見なされます。 この修飾子によって影響を受ける列のデータとして NULL を入れることはできません。通常は NULL として解釈されるブランクは、この修飾子の使用時にはバイナリー値として解釈されます。
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のロードが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にロードできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。
nullindchar=*x*	*x* は、単一文字です。NULL 値を示す文字を *x* に変更します。*x* のデフォルト値は、Y です。[2] この修飾子は、EBCDIC データ・ファイルでは大文字と小文字の区別があります。ただし、文字が英字の場合は例外です。例えば、NULL 標識文字が N に指定されている場合は、n も NULL 標識として認識されます。

packeddecimal	パック 10 進数データを直接ロードします。DECIMAL フィールド・タイプは、binarynumerics 修飾子の対象に含まれていません。
	
	このオプションは、reclen オプションで指定する固定長レコードを使用する定位置 ASC でのみサポートされています。
	
	符号ニブルとしてサポートされている値は、以下のとおりです。
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	この修飾子によって影響を受ける列のデータとして NULL を入れることはできません。通常は NULL として解釈されるブランクは、この修飾子の使用時にはバイナリー値として解釈されます。
	
	サーバー・プラットフォームにかかわりなく、ロード・ソース・ファイルのバイナリー・データのバイト・オーダーは、ビッグ・エンディアンであると見なされます。したがって、Windows オペレーティング・システムでこの修飾子を使用する場合は、バイト・オーダーを逆にしてはなりません。
	
	この修飾子を implieddecimal 修飾子と一緒に使用することはできません。

表 33. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASC ファイル形式 (区切りなし ASCII) (続き)

修飾子	説明
reclen=*x*	*x* は最大値が 32 767 の整数です。各行では *x* 個の文字が読み取られ、行の終わりを示す改行文字は使用されません。
striptblanks	可変長フィールドにデータをロードするときに、末尾ブランク・スペースを切り捨てます。このオプションを指定しなければ、ブランク・スペースは維持されます。 このオプションを `striptnulls` と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの `t` の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。
striptnulls	可変長フィールドにデータをロードするときに、末尾 NULL (0x00 文字) を切り捨てます。このオプションを指定しなければ、NULL は維持されます。 このオプションを `striptblanks` と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの `padwithzero` の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。

zoneddecimal	ゾーン 10 進数をロードします。DECIMAL フィールド・タイプは、BINARYNUMERICS 修飾子の対象に含まれていません。このオプションは、RECLEN オプションで指定する固定長レコードを使用する定位置 ASC でのみサポートされています。
	
	ハーフバイト符号値は、以下のいずれかになります。
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	数字としてサポートされている値は、`0x0` から `0x9` です。
	
	ゾーンとしてサポートされている値は、`0x3` から `0xF` です。

表 34. ロード・ユーティリティーの有効なファイル・タイプ修飾子: DEL ファイル形式 (区切り付き ASCII)

修飾子	説明

chardel*x*	*x* は、単一文字ストリング区切りです。デフォルト値は、二重引用符 (″) です。文字ストリングを囲む二重引用符の代わりに指定の文字を使用します。[23] 文字ストリング区切りとして二重引用符 (″) を明示的に指定する場合は、以下のように指定します。
	`modified by chardel""`
	
	文字ストリング区切りとして単一引用符 (') を指定することもできます。その場合は、以下のようにします。
	`modified by chardel''`

coldel*x*	*x* は、単一文字列区切りです。デフォルト値は、コンマ (,) です。列の終わりを示すコンマの代わりに指定の文字を使用します。[23]
decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) の代わりにブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進値の接頭部として正符号を使用します。
decpt*x*	*x* は、小数点文字としてピリオドの代わりに使用する単一文字です。デフォルト値は、ピリオド (.) です。小数点文字として、ピリオドの代わりに指定の文字を使用します。[23]

表 34. ロード・ユーティリティーの有効なファイル・タイプ修飾子: DEL ファイル形式 (区切り付き ASCII) (続き)

修飾子	説明

delprioritychar	区切り文字に関する現在のデフォルトの優先順位は、レコード区切り、文字区切り、列区切り、という順序になっています。この修飾子を指定すると、区切り文字の優先順位が、文字区切り、レコード区切り、列区切り、という順序に戻されるので、古い優先順位に依存する既存のアプリケーションが保護されます。構文:
	`db2 load ... modified by delprioritychar ...`
	例えば、以下の DEL データ・ファイルがあるとします。
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	delprioritychar 修飾子を指定しているので、このデータ・ファイルは、2 行だけになります。1 番目と 3 番目の <row delimiter> は、実際のレコード区切りとして解釈されますが、2 番目の <row delimiter> は、第 2 行の最初のデータ列の一部として解釈されるからです。この修飾子を指定しなければ、それぞれの <row delimiter> が区切り文字として解釈され、このデータ・ファイルは 3 行になります。

keepblanks	タイプ CHAR、VARCHAR、LONG VARCHAR、CLOB の各フィールドで前後のブランクを保持します。このオプションを指定しないと、文字区切りの内側にない前後のすべてのブランクが除去され、表のすべてのブランク・フィールドに NULL が挿入されます。
	データ・ファイルにある前後のスペースをすべて保持して、TABLE1 という表にデータをロードする例を以下に示します。
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nochardel	ロード・ユーティリティーは、列区切りの間で検出するすべてのバイトを列のデータの一部と見なします。文字区切りも、列データの一部として解析されます。DB2 でエクスポートしたデータについては、このオプションを指定しないでください (ただし、エクスポート時に nochardel を指定していた場合は例外です)。このオプションは、文字区切りのないベンダー・データ・ファイルをサポートするために用意されています。正しくない使い方をすると、データが失われたり破損したりする可能性があります。 このオプションを chardelx、delprioritychar、nodoubledel のいずれかと一緒に指定することはできません。これらは、相互に排他的なオプションです。
nodoubledel	二重文字区切りの認識を抑止します。

表 35. ロード・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式

修飾子	説明
forcein	ユーティリティーは、コード・ページの不一致があってもデータを受け入れ、コード・ページの変換を抑止します。 固定長ターゲット・フィールドについては、データを収容するだけの大きさがあるかどうかのチェックが行われます。nochecklengths を指定すると、チェックなしで各行のロードが試行されます。

表 35. ロード・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式 (続き)

修飾子	説明
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のロードが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にロードできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。

注:

1. 日付形式ストリングを二重引用符で囲むのは、必須です。フィールド区切り文字には、a から z、A から Z、0 から 9 を組み込めません。フィールド区切り文字として、DEL ファイル形式の文字区切りまたはフィールド区切りと同じ文字を使用することはできません。エレメントの開始位置と終了位置があいまいでない場合は、フィールド区切り文字はオプションになります。修飾子によっては、項目が可変長の場合に D、H、M、S などのエレメントを使用することがあり、そのような場合は、開始位置と終了位置があいまいになることがあります。

   タイム・スタンプ形式の場合は、月の記述子と分の記述子の間であいまいさが残らないように注意する必要があります。どちらも、M という文字を使用するからです。月のフィールドは、他の日付フィールドと隣接している必要があります。分のフィールドは、他の時刻フィールドと隣接している必要があります。あいまいなタイム・スタンプ形式の例を以下に示します。

   ```
   "M" (月または分のどちらにもとれる)
   "M:M" (月と分の区別がつかない)
   "M:YYYY:M" (両方とも月と解釈される)
   "S:M:YYYY" (時刻値と日付値の両方に隣接している)
   ```

   あいまいな場合は、ユーティリティーによってエラー・メッセージが生成され、操作は失敗します。

   あいまいでないタイム・スタンプ形式の例を以下に示します。

   ```
   "M:YYYY" (M (月))
   "S:M" (M (分))
   "M:YYYY:S:M" (M (月)....M (分))
   "M:H:YYYY:M:D" (M (分)....M (月))
   ```

   二重引用符や円記号など、いくつかの文字の前ではエスケープ文字を使用する必要があります (¥ など)。

2. ファイル・タイプ修飾子 chardel、coldel、decpt に指定する文字値は、ソース・データのコード・ページに指定されている文字値でなければなりません。

   文字コード・ポイント (文字シンボルではない) を指定する場合は、xJJ または 0xJJ という構文を使用できます (JJ は、コード・ポイントの 16 進表記です)。例えば、列区切りとして # 文字を指定する場合は、以下のいずれかを使用します。

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

3. 区切り文字のオーバーライドとして使用できる文字に適用される制約事項については、『*データ移動のための区切り文字の制約事項*』を参照してください。
4. MODIFIED BY オプションでサポートされていないファイル・タイプを使用しようとしても、ロード・ユーティリティーからは警告が生成されません。その場合は、ロード操作が失敗し、エラー・コードが戻されます。
5. 暗黙的な非表示設定になっている Row Change Timestamp 列が含まれている表にインポートする場合は、その列の暗黙的な非表示のプロパティーが適用されません。したがって、インポートするデータに列のデータが含まれていない場合に、明示的な列リストも存在しなければ、インポート・コマンドで rowchangetimestampmissing ファイル・タイプ修飾子を指定することが必要です。

表 36. codepage と usegraphiccodepage を使用する場合の LOAD の動作

codepage=N	**usegraphiccodepage**	**LOAD の動作**
なし	なし	ファイル内のすべてのデータは、CLIENT オプションが指定されている場合でも、アプリケーション・コード・ページではなくデータベース・コード・ページのデータであるという想定になります。
あり	なし	ファイル内のすべてのデータは、コード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにロードすると、グラフィック・データが破損します。
なし	あり	ファイル内の文字データは、CLIENT オプションが指定されている場合でも、データベース・コード・ページのデータであるという想定になります。グラフィック・データは、CLIENT オプションが指定されている場合でも、データベース・グラフィック・データのコード・ページのデータであるという想定になります。 データベース・コード・ページが 1 バイトの場合は、すべてのデータがデータベース・コード・ページのデータであるという想定になります。 **警告:** グラフィック・データを 1 バイトのデータベースにロードすると、グラフィック・データが破損します。
あり	あり	文字データは、コード・ページ N のデータであるという想定になります。グラフィック・データは、N のグラフィック・コード・ページのデータであるという想定になります。 N が 1 バイトまたは 2 バイトのコード・ページの場合は、すべてのデータがコード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにロードすると、グラフィック・データが破損します。

LOAD コマンド (ADMIN_CMD プロシージャーを使用)

データを DB2 表にロードします。サーバー上に存在するデータは、ファイル、テープ、または名前付きパイプの形式にすることができます。 表の COMPRESS 属性が YES に設定されている場合、ロードされるデータは、表内にディクショナリーがすでに存在するデータおよびデータベース・パーティションごとに圧縮の対象となります。

295 ページの『ロード・ユーティリティーのファイル・タイプ修飾子』へのクイック・リンク。

制約事項

ロード・ユーティリティーでは、階層レベルのデータのロードはサポートされていません。ロード・ユーティリティーには、範囲クラスター表との互換性はありません。

有効範囲

このコマンドは、一度の要求で複数のデータベース・パーティションに対して発行できます。

許可

以下のいずれか。

- *sysadm*
- *dbadm*
- データベースに対する LOAD 権限と次のいずれか
  - ロード・ユーティリティーが INSERT モード、 TERMINATE モード (それまでのロード挿入操作を終了する)、または RESTART モード (以前のロード挿入操作を再開する) で呼び出された場合には、その表に対する INSERT 特権。
  - ロード・ユーティリティーが REPLACE モード、 TERMINATE モード (それまでのロード置換操作を終了する)、または RESTART モード (以前のロード置換操作を再開する) で呼び出された場合には、その表に対する INSERT および DELETE 特権。
  - 例外表の INSERT 特権 (例外表をロード操作の一部として使用する場合)。
- 保護列を持つ表にデータをインポートするには、表内のすべての保護列への書き込みアクセスを可能にする LBAC 信用証明情報がセッション許可 ID に必要です。そうでない場合は、ロードが失敗してエラー (SQLSTATE 5U014) が戻されます。
- 保護された行を持つ表にデータをロードするには、セッション許可 ID が、以下の基準を満たすセキュリティー・ラベルを保持していなければなりません。
  - 表を保護しているセキュリティー・ポリシーの一部である
  - 書き込みアクセスまたは全アクセスを対象としてセッション許可 ID に認可された。

  こうしたセキュリティー・ラベルをセッション許可 ID が保持していない場合は、ロードが失敗してエラー (SQLSTATE 5U014) が戻されます。このセキュリティー・ラベルは、セッション許可 ID の LBAC 信用証明情報が、データ内の

ロードされる行を保護するセキュリティー・ラベルにその許可 ID が書き込むことを許可しない場合に、その行を保護するために使用されます。ただし、表を保護しているセキュリティー・ポリシーが CREATE SECURITY POLICY ステートメントの RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL オプションを使用して作成されている場合は、その状況にはなりません。その場合は、ロードが失敗してエラー (SQLSTATE 42519) が戻されます。

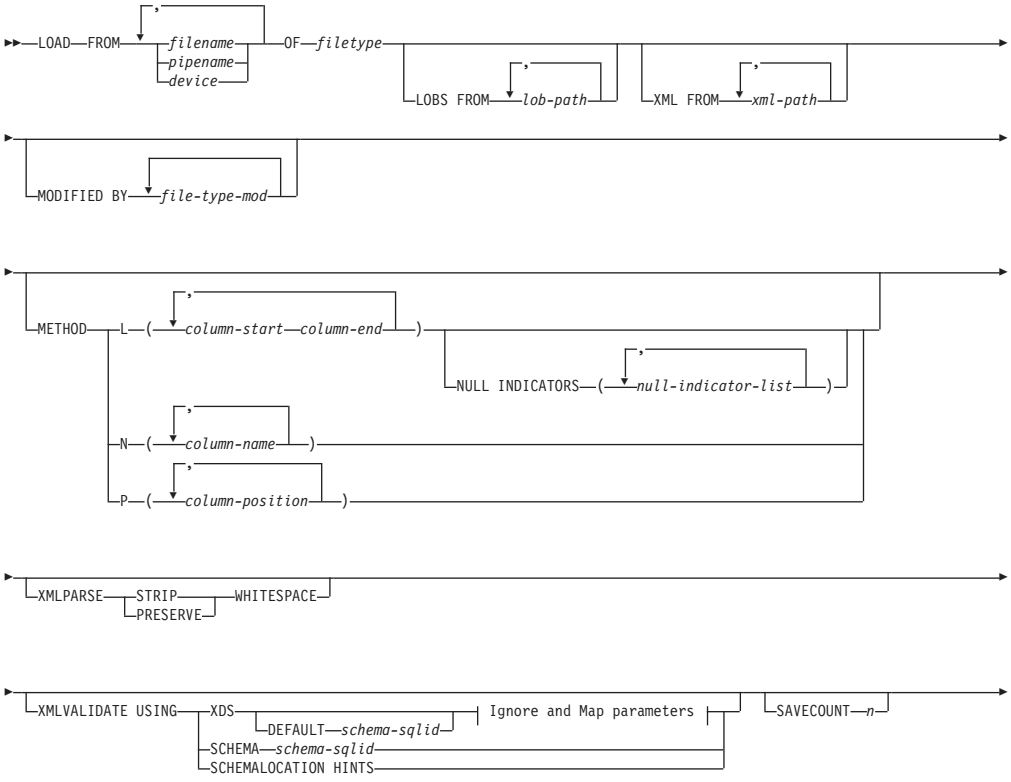
- REPLACE オプションを指定する場合、セッション許可 ID には表をドロップするための権限が必要です。

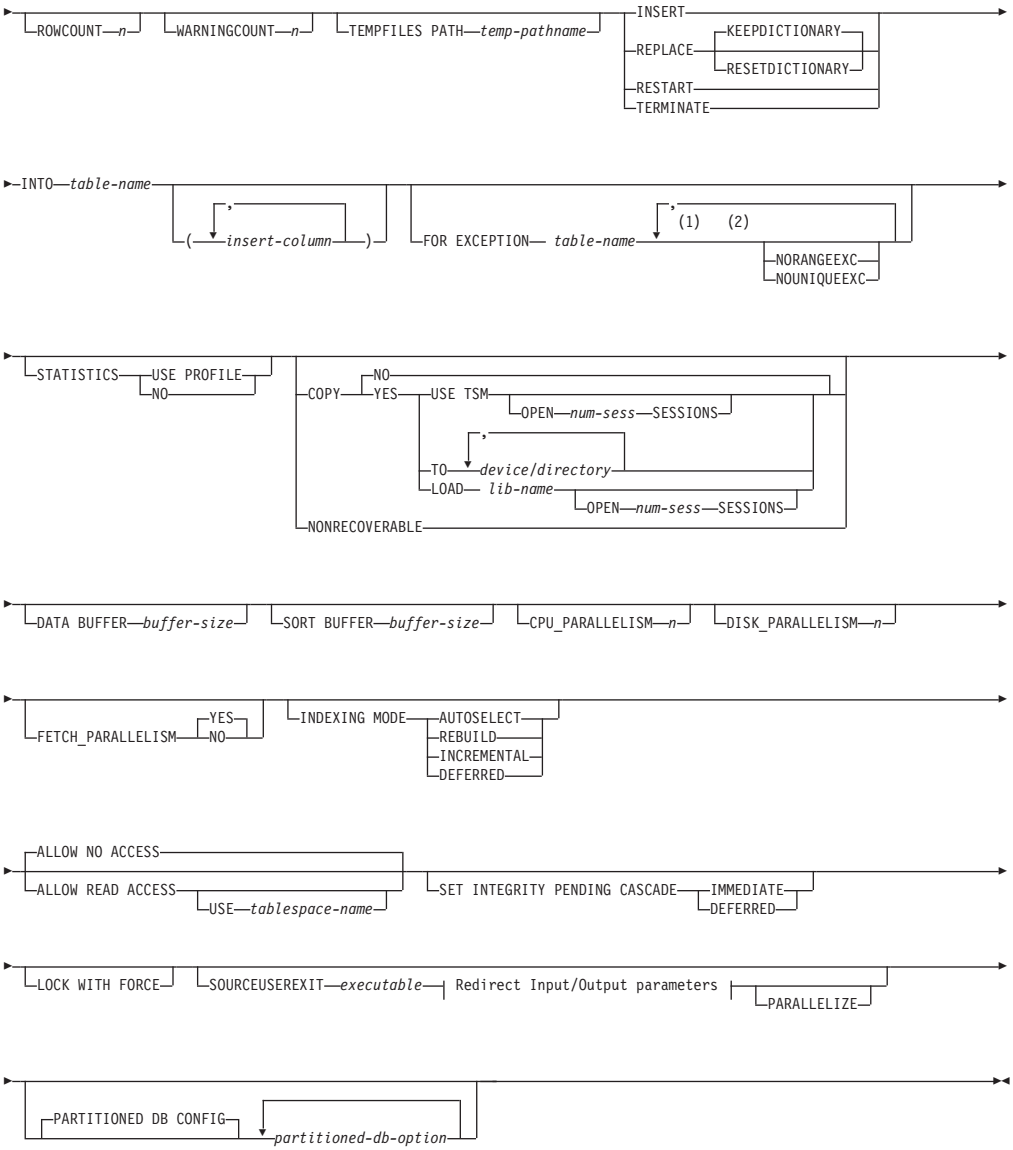
すべてのロード・プロセス (および一般にすべての DB2 サーバー・プロセス) はインスタンス所有者によって所有されており、それらのプロセスすべてにおいて、必要なファイルにアクセスするためにそのインスタンス所有者の ID を使用するため、インスタンス所有者には入力データ・ファイルに対する読み取りアクセス権が必要です。だれがコマンドを呼び出すかに関係なく、これらの入力データ・ファイルはインスタンス所有者から読み取り可能になっていなければなりません。

必要な接続

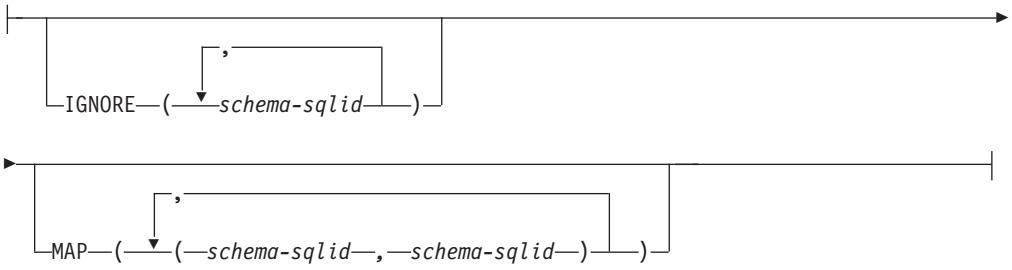
インスタンス。明示的なアタッチは必要ありません。データベースへの接続が確立されている場合には、ローカル・インスタンスへの暗黙的な接続が試みられます。

コマンド構文

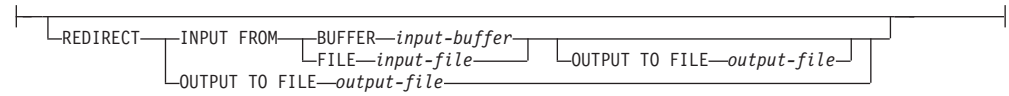




Ignore and Map parameters:



Redirect Input/Output parameters:



注:

1 これらのキーワードの出現順序は任意です。

2 これらのキーワードは、それぞれ 1 回だけ出現できます。

コマンド・パラメーター

FROM *filename* | *pipename* | *device*

注:

1. *ADMIN_CMD プロシージャーを使用した EXPORT コマンド* を使用してデータをファイルにエクスポートした場合、そのデータ・ファイルは fenced ユーザー ID によって所有されます。このファイルは通常、インスタンス所有者がアクセスすることはできません。 LOAD を CLP または ADMIN_CMD プロシージャーから実行するには、インスタンス所有者 ID はデータ・ファイルにアクセスできなければならないので、データ・ファイルに対する読み取りアクセス権限をインスタンス所有者に付与する必要があります。
2. ファイルが物理的には分割されてはいるが論理的には 1 つのファイルである場合には、複数の IXF ファイルからのデータのロードがサポートされています。ファイルが論理的にも物理的にも分割されている場合は、サポートされていません。 (複数の物理ファイルがすべて 一度の EXPORT コマンドの呼び出しで作成された場合、それらは論理的には 1 つであると見なされます。)

OF *filetype*

データのフォーマットを指定します。

- ASC (区切りなし ASCII フォーマット)。
- DEL (区切り付き ASCII フォーマット)。
- IXF (統合交換フォーマット、PC バージョン) は、DB2 専用のバイナリー・フォーマットです。
- CURSOR (SELECT または VALUES ステートメントに対して宣言されたカーソル)。

LOBS FROM *lob-path*

ロードする LOB 値が収められているデータ・ファイルへのパス。パスの最後はスラッシュでなければなりません。 LOB データ・ファイルの名前は、メイン・データ・ファイル (ASC、DEL、または IXF) の、LOB 列にロードされる列内に保管されます。指定できるパスの最大数は 999 です。これによって、LOBSINFILE 動作が暗黙的に活動化されます。

CURSOR ファイル・タイプと併せて指定された場合、このオプションは無視されます。

MODIFIED BY *file-type-mod*

ファイル・タイプ修飾子オプションを指定します。 295 ページの『ロード・ユーティリティーのファイル・タイプ修飾子』を参照してください。

METHOD

L データのロードを開始する列および終了する列の番号を指定します。列の番号は、データの行の先頭からのバイト単位のオフセットです。この番号は 1 から始まります。このメソッドは、ASC ファイルの場合にのみ使用することができ、そのファイル・タイプに対してのみ有効なメソッドです。

NULL INDICATORS *null-indicator-list*

このオプションは、METHOD L パラメーターを指定した場合だけ使用できます (つまり、入力ファイルが ASC ファイルの場合)。 NULL 標識リストは、コンマで区切られた正の整数のリストで、各 NULL 標識フィールドの列の番号を指定します。列の番号は、データの行の先頭からのバイト単位の、各 NULL 標識フィールドのオフセットです。 NULL 標識リストには、METHOD L パラメーターで定義された各データ・フィールドに対する 1 つの項目がなければなりません。列の番号がゼロであることは、対応するデータ・フィールド内に必ずデータがあることを示します。

NULL 標識列中の `Y` の値は、その列データが NULL であることを指定します。 NULL 標識列に `Y` *以外* の文字を指定した場合は、列データが NULL ではなく、METHOD L オプションで指定された列データがロードされることを指定することになります。

NULL 標識文字は MODIFIED BY オプションを使用して変更できます。

N ロードするデータ・ファイルの中の列の名前を指定します。これらの列名の大文字小文字の区別は、システム・カタログ内の対応する名前の大文字小文字の区別と一致しなければなりません。 NULL 可能ではない各表の列には、METHOD N リスト内に対応する項目が必要です。例えば、データ・フィールドが F1、 F2、 F3、 F4、 F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、 `method N (F2, F1, F4, F3)` は有効な要求ですが、 `method N (F2, F1)` は無効です。この方式は、ファイル・タイプ IXF または CURSOR の場合にのみ使用することができます。

P ロードする入力データ・フィールドのフィールド番号 (1 から始まる) を指定します。 NULL 可能ではない各表の列には、METHOD P リスト内に対応する項目が必要です。たとえば、データ・フィールドが F1、 F2、 F3、 F4、 F5、および F6 であり、表の列が C1 INT、C2 INT NOT NULL、C3 INT NOT NULL、および C4 INT の場合、 `method P (2, 1, 4, 3)` は有効な要求ですが、 `method P (2, 1)` は無効です。この方式は、ファイル・タイプ IXF、DEL、または CURSOR の場合にのみ使用でき、 DEL ファイル・タイプに対してのみ有効な方式です。

XML FROM *xml-path*

XML ファイルが含まれているパスを 1 つ以上指定します。 XDS は、メイン・データ・ファイル (ASC、DEL、または IXF) の、XML 列にロードされる列内に保管されます。

XMLPARSE

XML 文書の解析方法を指定します。このオプションが指定されていない場合、XML 文書の解析の動作は、CURRENT XMLPARSE OPTION 特殊レジスターの値によって決まります。

STRIP WHITESPACE

XML 文書の解析時に空白文字を除去することを指定します。

PRESERVE WHITESPACE

XML 文書の解析時に空白文字を除去しないことを指定します。

XMLVALIDATE

該当する場合に、XML 文書がスキーマに準拠しているかどうかの妥当性検査を実行することを指定します。

USING XDS

メイン・データ・ファイル内の XML Data Specifier (XDS) で識別される XML スキーマに照らし合わせて、XML 文書が妥当性検査されます。デフォルトでは、USING XDS 節によって XMLVALIDATE オプションが呼び出された場合、妥当性検査実行のために使用されるスキーマは、その XDS の SCH 属性によって決まります。 XDS の中で SCH 属性が指定されていない場合、DEFAULT 節によってデフォルト・スキーマが指定されているのでない限り、スキーマ妥当性検査は実行されません。

DEFAULT、IGNORE、および MAP 節を使用することにより、スキーマ決定の動作を変更することができます。これら 3 つの節はオプションであり、相互に適用されるのではなく XDS の指定に直接適用されます。例えば、DEFAULT 節で指定されているためにあるスキーマが選択された場合、それが IGNORE 節で指定されていたとしても無視されることはありません。同じように、MAP 節のペアの最初の部分で指定されているためにあるスキーマが選択された場合、それが別の MAP 節のペアの 2 番目の部分で指定されていたとしても再びマップされることはありません。

USING SCHEMA *schema-sqlid*

指定されている SQL ID の XML スキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。この場合、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

USING SCHEMALOCATION HINTS

ソース XML 文書の中で XML スキーマ・ロケーション・ヒントによって指定されているスキーマに準拠しているかどうかについて、XML 文書の妥当性検査が実行されます。その XML 文書の中に schemaLocation 属性が指定されていない場合、妥当性検査は実行さ

れません。 USING SCHEMALOCATION HINTS 節が指定されているなら、すべての XML 列について XML Data Specifier (XDS) の SCH 属性は無視されます。

以下に示す XMLVALIDATE オプションの例を参照してください。

IGNORE *schema-sqlid*

このオプションは、USING XDS パラメーターを指定した場合にのみ使用できます。 IGNORE 節は、SCH 属性によって指定されていても無視するスキーマとして、1 つ以上のスキーマのリストを指定します。 ロードする XML 文書の XML Data Specifier の中に SCH 属性が存在し、その SCH 属性によって指定されるスキーマが IGNORE のスキーマ・リストに含まれている場合には、ロードするその XML 文書についてスキーマ妥当性検査は実行されません。

注:

あるスキーマが IGNORE 節の中で指定されている場合、MAP 節のスキーマ・ペアの左辺にそれを含めることはできません。

IGNORE 節は XDS にのみ適用されます。 あるスキーマが IGNORE 節によって指定されていても、それが MAP 節によってマップされているなら、それ以降そのスキーマが無視されることはありません。

DEFAULT *schema-sqlid*

このオプションは、USING XDS パラメーターを指定した場合にのみ使用できます。 DEFAULT 節で指定されたスキーマは、ロード対象 XML 文書の XML Data Specifier (XDS) に XML スキーマを指定する SCH 属性が含まれていない場合に、妥当性検査のために使用するスキーマとなります。

DEFAULT 節は、IGNORE 節および MAP 節よりも優先されます。 XDS が DEFAULT 節を満たすなら、IGNORE と MAP の指定は無視されます。

MAP *schema-sqlid*

このオプションは、USING XDS パラメーターを指定した場合にのみ使用できます。 MAP 節は、ロードする各 XML 文書について XML Data Specifier (XDS) の SCH 属性によって指定されるスキーマの代わりに使用する代替スキーマを指定するのに使用します。MAP 節には、それぞれがあるスキーマから別のスキーマへのマッピングを表すスキーマ・ペアを 1 つ以上列挙したリストを指定します。 ペア中の最初のスキーマは、XDS 内の SCH 属性によって参照されるスキーマを表します。ペア中の 2 番目のスキーマは、スキーマ検証の実行で使用する必要のあるスキーマを表します。

あるスキーマが MAP 節のスキーマ・ペアの左辺で指定されている場合、IGNORE 節でさらにそれを指定することはできません。

スキーマ・ペアのマッピングが適用されたなら、その結果は最終的なものです。マッピング操作は推移的ではないため、選択されたスキーマが、それ以降に別のスキーマ・ペアのマッピングに適用されることはありません。

スキーマを複数回マップすることはできません。つまり、複数のペアの左辺に指定することはできません。

SAVECOUNT *n*

ロード・ユーティリティーが *n* 行ごとに整合点を取ることを指定します。この値はページ・カウントに変換され、エクステント・サイズのインターバルに切り上げられます。メッセージは整合点において発行されるので、LOAD QUERY を使用してロード操作をモニターする場合には、このオプションを選択する必要があります。 *n* の値が十分な大きさでない場合、各整合点で実行される活動の同期化によってパフォーマンスに影響してしまいます。

デフォルト値はゼロですが、それは、必要がなければ整合点は確立されないことを意味します。

CURSOR ファイル・タイプと併せて指定された場合、このオプションは無視されます。

ROWCOUNT *n*

ロードするファイル内の物理レコードの数 *n* を指定します。ユーザーはファイル内の最初の *n* 個の行だけをロードできます。

WARNINGCOUNT *n*

n 個の警告後に、ロード操作を停止します。このパラメーターは、警告は予期されないが、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。ロード・ファイルまたはターゲット表が不適切に指定されると、ロード対象の各行ごとにロード・ユーティリティーによって警告が生成され、このためにロードが失敗する可能性があります。 *n* がゼロの場合、またはこのオプションが指定されていない場合、何度警告が出されてもロード操作は続行します。警告のしきい値に達したためにロード操作が停止された場合でも、あらためて RESTART モードでロード操作を開始できます。ロード操作は、最後の整合点から自動的に続行します。または、入力ファイルの先頭から REPLACE モードであらためてロード操作を開始できます。

TEMPFILES PATH *temp-pathname*

ロード操作時に一時ファイルを作成する場合に使用するパスの名前を指定します。これはサーバー・データベース・パーティションに従って完全に修飾しなければなりません。

一時ファイルは、ファイル・システムのスペースを使用します。場合によっては、このスペースが相当必要になります。以下に示すのは、すべての一時ファイルにどの程度のファイル・システム・スペースを割り振るべきかの見積もりです。

- ロード・ユーティリティーが生成するメッセージごとに 136 バイト
- データ・ファイルに長フィールド・データまたは LOB が入っている場合は、 15 KB のオーバーヘッド。 INSERT オプションを指定した場合で、表の中に多量の長フィールドまたは LOB データがすでにある場合には、この数値はこれよりもかなり大きくなる場合があります。

INSERT

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。既存の表データを変更することなく、ロードされたデータを表に追加します。

REPLACE

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。表か

ら既存データをすべて削除し、ロードされたデータを挿入します。表定義および索引定義は変更されません。階層間でデータを移動する際にこのオプションを使用する場合は、階層全体に関係したデータだけが置き換えられます。副表は置き換えられません。

KEEPDICTIONARY

LOAD REPLACE 操作の後も、既存のコンプレッション・ディクショナリーを保持します。表の COMPRESS 属性が YES になっていると、新しく置換するデータは、ロードの呼び出し前に存在していたディクショナリーに基づく圧縮の対象になります。表にディクショナリーが存在していなかった場合は、表の COMPRESS 属性が YES になっている限り、置換によって表に挿入されるデータによって新しいディクショナリーが作成されます。この場合、コンプレッション・ディクショナリーを作成するために必要なデータの量は、ADC のポリシーによって左右されます。そのデータは、圧縮されていない状態で表に取り込まれます。表にディクショナリーが挿入されると、その後にロードされる残りのデータは、そのディクショナリーによる圧縮の対象になります。これはデフォルトのパラメーターです。要約を以下の表 1 に示します。

表 37. LOAD REPLACE KEEPDICTIONARY

圧縮	ディクショナリーが存在するかどうか	結果
Y	Y	ディクショナリーを保存します。すべての入力行が既存のディクショナリーによる圧縮の対象になります。
Y	N	十分なユーザー・データが存在する場合にのみ、新しいディクショナリーを表に挿入します。残りの行は、ディクショナリーの作成後に圧縮の対象になります。
N	Y	ディクショナリーを保存します。すべての入力行が圧縮されません。
N	N	影響はありません。すべての行が圧縮されません。

RESETDICTIONARY

表の COMPRESS 属性が YES の場合にこのディレクティブを指定すると、LOAD REPLACE 処理の実行時に、表のデータ・オブジェクトに対応した新しいディクショナリーが作成されます。

COMPRESS 属性が NO で、表の中にディクショナリーがすでに存在している場合は、そのディクショナリーが除去されるだけで、新しいディクショナリーが表に挿入されることはありません。コンプレッション・ディクショナリーは、1 つのユーザー・レコードだけでも作成できます。ロードするデータ・セットのサイズがゼロの場合は、既存のディクショナリーが存在していても、そのディクショナリーは保持されません。このディレクティブを指定した場合、ディクショナリーを作成するために必要なデータの量は、ADC のポリシーに左右されません。要約を以下の表 2 に示します。

表 38. LOAD REPLACE RESETDICTIONARY

圧縮	ディクショナリーが存在するかどうか	結果
Y	Y	新しいディクショナリーを作成します。* ロードする残りの行は、ディクショナリーの作成後に圧縮の対象になります。
Y	N	新しいディクショナリーを作成します。残りの行は、ディクショナリーの作成後に圧縮の対象になります。
N	Y	ディクショナリーを除去します。すべての入力行が圧縮されません。
N	N	影響はありません。すべての行が圧縮されません。

* ディクショナリーが存在し、圧縮属性が有効になっていても、表パーティションにロードするレコードがない場合は、新しいディクショナリーを作成できません。RESETDICTIONARY 操作では、既存のディクショナリーが維持されなくなります。

TERMINATE

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。以前に割り込みを受けたロード操作を終了し、ロード操作が開始された時点まで操作をロールバックします。途中に整合点があっても通過します。その操作に関係する表スペースの状態は通常に戻され、すべての表オブジェクトの整合性が保たれます (索引オブジェクトが無効とマークされる場合がありますが、そのような場合には、次回のアクセス時に索引の再作成が自動的に行われます)。終了するロード操作が LOAD REPLACE の場合、その表は LOAD TERMINATE 操作完了後に空の表まで切り捨てられます。終了するロード操作が LOAD INSERT の場合、その表は LOAD TERMINATE 操作完了後も元のレコードをすべて保持します。ディクショナリー管理の要約を以下の表 3 に示します。

LOAD TERMINATE オプションでは、表スペースのバックアップ・ペンディング状態は解除されません。

RESTART

ロード・ユーティリティーを実行できる 4 つのモードのうちの 1 つ。以前に割り込みを受けたロード操作を再開します。ロード操作は、ロード、作成、または削除フェーズの最後の整合点から自動的に続行されます。ディクショナリー管理の要約を以下の表 4 に示します。

INTO *table-name*

データのロード先となるデータベース表を指定します。この表として、システム表または宣言一時表は指定できません。別名、完全修飾、または非修飾の表名を指定することができます。修飾子付き表名は、`schema.tablename` の形式です。非修飾の表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

insert-column

データの挿入先となる表の列を指定します。

ロード・ユーティリティーは、1 つ以上のスペースを使った名前の列を解析できません。例えば、次のようにします。

は、Int 4 列があるためエラーになります。これは、次のようにして二重引用符で列名を囲むことによって解決できます。

FOR EXCEPTION *table-name*
エラーが発生した行のコピー先となる例外表を指定します。ユニーク索引または主キー索引に違反した行がすべてコピーされます。非修飾の表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

例外表に書き込まれる情報は、ダンプ・ファイルには書き込まれません 。パーティション・データベース環境では、ロードする表を定義されたデータベース・パーティションの例外表を定義する必要があります。ダンプ・ファイルには、無効であるか構文エラーであるためにロードできない行が入ります。

NORANGEEXC
範囲違反のためにリジェクトされた行は、例外表に挿入しないことを指定します。

NOUNIQUEEXC
ユニーク制約に違反しているためにリジェクトされた行は、例外表に挿入しないことを指定します。

STATISTICS USE PROFILE
この表で定義されているプロファイルに従ってロード中に統計を収集するようロード操作に指示します。そのプロファイルは、ロードの実行前に作成されていなければなりません。そのプロファイルは、RUNSTATS コマンドで作成します。プロファイルが存在しない場合に、プロファイルに従って統計を収集するようロード操作に指示すると、警告メッセージが戻されて統計は収集されません。

STATISTICS NO
統計データを収集せず、したがってカタログ内の統計データも変更しないことを指定します。これはデフォルトです。

COPY NO
順方向リカバリーが使用可能 (つまり、*logretain* または *userexit* がオン) になっていれば、表が存在している表スペースをバックアップ・ペンディング状態にするよう指定します。 COPY NO オプションを使用する場合も、表スペース状態は LOAD IN PROGRESS になります。これは、一時的な状態であり、ロードが完了するか打ち切られると解除されます。表スペースのバックアップまたはデータベースの完全バックアップを実行しない限り、表スペースのどの表のデータも更新または削除できません。ただし、SELECT ステートメントを使用すれば、どの表のデータにもアクセス可能です。

リカバリー可能データベースでの COPY NO を指定した LOAD は、表スペースをバックアップ・ペンディング状態のままにします。例えば、COPY NO を指定した LOAD および INDEXING MODE DEFERRED を実行すると、索引はリフレッシュが必要な状態になります。表での照会には、索引スキャンが必要なものがあり、索引がリフレッシュされるまで、成功しません。バックアップ・ペンディング状態にある表スペース内に常駐する場合、索引はリフレッシュできません。この場合、表へのアクセスは、バックアッ

プが行われるまで許可されません。索引リフレッシュは、索引が照会によってアクセスされたときに、データベースによって自動的に行われます。COPY NO、COPY YES、NONRECOVERABLE のいずれも指定しない場合に、データベースがリカバリー可能であれば (つまり、**logretain** または **logarchmeth1** が有効になっていれば)、COPY NO がデフォルトになります。

COPY YES
ロードするデータのコピーを保存することを指定します。このオプションは、フォワード・リカバリーが使用不可に設定されている場合には無効です。

USE TSM
Tivoli Storage Manager (TSM) を使ってコピーを保管することを指定します。

OPEN *num-sess* **SESSIONS**
TSM またはベンダー製品とともに使用する入出力セッションの数です。デフォルト値は 1 です。

TO *device/directory*
コピー・イメージを作成する先の装置またはディレクトリーを指定します。

LOAD *lib-name*
使用するバックアップおよびリストア I/O ベンダー関数を含む共有ライブラリー (Windows オペレーティング・システムでは DLL) の名前。絶対パスで指定することができます。絶対パスを指定しない場合、デフォルトでユーザー出口プログラムの存在するパスになります。

NONRECOVERABLE
ロード・トランザクションがリカバリー不能としてマークされており、それ以降のロールフォワード・アクションによってそれをリカバリーさせることは不可能であることを指定します。ロールフォワード・ユーティリティーは、そのトランザクションをスキップし、データのロード先の表に「invalid」(無効) としてマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。ロールフォワード操作が完了すると、そのような表は、ドロップするか、またはリカバリー不能なロード操作完了後のコミット・ポイントの後に取られたバックアップ (全バックアップまたは表スペースのバックアップ) からのみ、リストアすることができます。

このオプションを指定すると、表スペースはロード操作後にバックアップ・ペンディング状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。 COPY NO、COPY YES、NONRECOVERABLE のいずれも指定しない場合に、データベースがリカバリー不能であれば (つまり、**logretain** または **logarchmeth1** が有効になっていなければ)、NONRECOVERABLE がデフォルトになります。

WITHOUT PROMPTING
データ・ファイルのリストにロードするすべてのファイルを含め、しかもリストに入っている装置またはディレクトリーがロード操作全体で十分である

ということを指定します。続きの入力ファイルが見つからなかったり、ロード操作が終了する前にコピー先がいっぱいになるとロード操作は失敗し、表はロード・ペンディング状態のままになります。

DATA BUFFER *buffer-size*
ユーティリティー内でデータを転送するためのバッファー・スペースとして使用する 4 KB ページ数を設定します (並列処理の度合いには依存しません)。指定する値がアルゴリズム上の最小値より小さい場合、最小限必要なリソースが使用され、警告は戻されません。

このメモリーは、ユーティリティー・ヒープから直接に割り当てられ、そのサイズは *util_heap_sz* データベース構成パラメーターで修正可能です。

値が指定されていない場合、実行時にユーティリティーによって適切なデフォルトが計算されます。デフォルトは、ローダーのインスタンス生成時にユーティリティー・ヒープで使用できるフリー・スペースの割合と、表の一部の特性に基づいて決まります。

SORT BUFFER *buffer-size*
このオプションは、ロード操作時に SORTHEAP データベース構成パラメーターをオーバーライドする値を指定します。これは、索引とともに表をロードする場合、また INDEXING MODE パラメーターが DEFERRED として指定されていない場合にのみ関係があります。指定される値は、SORTHEAP の値を超えることはできません。このパラメーターは、SORTHEAP の値を変更せずに多くの索引を持つ表をロードする際に使用されるソート・メモリーのスロットルで役に立ちます。これは、一般的な照会処理にも影響を与えます。

CPU_PARALLELISM *n*
表オブジェクトの作成時に、レコードの解析、変換、およびフォーマット設定のためにロード・ユーティリティーによって作成されるプロセスまたはスレッドの数を指定します。このパラメーターは、データベース・パーティションごとに実行するプロセス数を活用するために設計されています。これは、事前にソートされたデータをロードする際に役立ちます (ソース・データのレコード順序が保持されるため)。このパラメーターの値が 0 の場合や、このパラメーターを指定しなかった場合、ロード・ユーティリティーは、実行時に自動的に計算された適切なデフォルト値 (通常は使用可能な CPU の数に基づく) を使用します。

注:

1. LOB または LONG VARCHAR フィールドのどちらかの入った表でこのパラメーターを使用する場合、システムの CPU の数またはユーザーが指定した値には関係なく、値は 1 になります。
2. SAVECOUNT パラメーターに指定する値が小さいと、データと表のメタデータの両方をフラッシュするために、ローダーがさらに多くの入出力操作を実行することになります。 CPU_PARALLELISM が 1 より大きいなら、フラッシュ操作は非同期になり、ローダーは CPU を活用できます。 CPU_PARALLELISM が 1 に設定されている場合、ローダーは整合点において入出力を待ちます。 CPU_PARALLELISM を 2 に設定

し、SAVECOUNT を 10 000 に設定したロード操作は、CPU が 1 つしかなくても、同じ操作で CPU_PARALLELISM を 1 に設定した場合より速く完了します。

DISK_PARALLELISM *n*
表スペース・コンテナーにデータを書き込むためにロード・ユーティリティーが作成するプロセスまたはスレッドの数を指定します。値を指定しない場合、ユーティリティーは表スペース・コンテナーの数と表の特性に基づいて、自動的に計算された適切なデフォルトを選択します。

FETCH_PARALLELISM YES | NO
DATABASE キーワードを使用してカーソルが宣言されていてカーソルからのロードを実行するとき、または API の `sqlu_remotefetch_entry` メディア項目を使用するとき、このオプションが YES に設定されていると、ロード・ユーティリティーは、リモート・データ・ソースからのフェッチの並列化を試みます (可能な場合)。 NO に設定されている場合、並列フェッチは行われません。デフォルト値は、YES です。詳細については、『*CURSOR ファイル・タイプを使用したデータの移動*』を参照してください。

INDEXING MODE
ロード・ユーティリティーが索引を再作成するのか、それとも索引を増分で拡張するのかを指定します。有効な値は以下のとおりです。

AUTOSELECT
REBUILD モードと INCREMENTAL モードのいずれにするかを、ロード・ユーティリティーが自動的に決定します。決定は、ロードされるデータ量と索引ツリーの深さに基づいて行われます。索引ツリーの深さに関連する情報は索引オブジェクトに保管されています。この情報を設定するために、RUNSTATS は不要です。AUTOSELECT がデフォルトの索引付けモードです。

REBUILD
すべての索引が再作成されます。古い表データの索引キー部分も、追加される新しい表データの索引キー部分もすべてソートできるようにするため、ロード・ユーティリティーには十分なリソースが必要となります。

INCREMENTAL
索引に新しいデータが取り込まれて拡張します。このアプローチでは、索引のフリー・スペースが消費されます。このアプローチでは、新たに挿入されるレコードの索引キーを追加するためのソート・スペースだけがあれば十分です。この方式がサポートされるのは、索引オブジェクトが有効で、かつロード操作の開始時にアクセス可能な場合だけです (例えば、 DEFERRED モードが指定されたロード操作の直後では、この方式は無効です)。このモードを指定したものの、索引の状態などの理由でサポートされない場合は、警告が戻され、REBUILD モードでロード操作が続行されます。同様に、ロード作成フェーズでロード再開操作を開始した場合も、INCREMENTAL モードはサポートされません。

以下の条件がすべて真の場合、増分索引の作成はサポートされません。

- LOAD COPY オプションが指定されている (USEREXIT または LOGRETAIN オプションを指定した *logarchmeth1*)。
- 表が DMS 表スペース内に存在している。
- 索引オブジェクトの存在している表スペースが、ロードしようとしている表に属する他の表オブジェクトによって共有されている。

この制限を迂回するため、索引は別々の表スペースに置くようお勧めします。

DEFERRED
このモードが指定されている場合、ロード・ユーティリティーは索引の作成を試みません。リフレッシュが必要であることを示すマークが索引に付けられます。ロード操作とは関係のないこのような索引に最初にアクセスするときは、再作成が強制的に実行されたり、データベースの再始動時に索引が再作成されたりする場合があります。 このアプローチでは、最も大きい索引のキー部分をすべて処理できるだけのソート・スペースが必要です。索引を作成するためにその後かかる合計時間は、REBUILD モードの場合よりも長くなります。したがって、この索引作成据え置きモードで複数のロード操作を実行する場合、最初の非ロード・アクセス時に索引を再作成できるようにしておくよりも、順序列内の最後のロード操作で索引の再作成を実行できるようにした方が (パフォーマンスの観点から) 賢明であるといえます。

据え置き索引作成がサポートされるのは、非ユニーク索引がある表だけです。そのため、ロード・フェーズで挿入される重複キーがロード操作後は永続的ではなくなります。

ALLOW NO ACCESS
ロードを使用すると、ロード中に、排他的アクセスのためにターゲット表がロックされます。ロード中、表の状態は LOAD IN PROGRESS に設定されます。 ALLOW NO ACCESS はデフォルトの動作です。これは、LOAD REPLACE で唯一有効なオプションです。

表に制約があると、表の状態は、LOAD IN PROGRESS の他に、SET INTEGRITY PENDING に設定されます。表の SET INTEGRITY PENDING 状態を解除するには、SET INTEGRITY ステートメントを使用する必要があります。

ALLOW READ ACCESS
ロードを使用すると、ターゲット表は共用モードでロックされます。表の状態は、LOAD IN PROGRESS および READ ACCESS の両方に設定されます。表のロード中、データの非デルタ部分にアクセスすることができます。つまり、表を読み取る側はロードの開始前に存在していたデータにはアクセスができ、ロード中のデータはロードが完了するまで利用できない、ということです。 ALLOW READ ACCESS ロードの LOAD TERMINATE または LOAD RESTART はこのオプションを使用できますが、ALLOW NO ACCESS ロードの LOAD TERMINATE または LOAD RESTART はこのオプションを使用できません。また、ターゲット表上の索引が要再作成のマークが付けられると、このオプションは無効になります。

表に制約があると、表の状態は、LOAD IN PROGRESS、および READ ACCESS の他に、SET INTEGRITY PENDING に設定されます。ロードの終了時に、表の状態 LOAD IN PROGRESS は解除されますが、表の状態 SET INTEGRITY PENDING および READ ACCESS はそのまま残ります。表の SET INTEGRITY PENDING を解除するには、SET INTEGRITY ステートメントを使用する必要があります。表が SET INTEGRITY PENDING および READ ACCESS の状態にある間、データの非デルタ部分には引き続き読み取りアクセスできますが、データの新しい (デルタ) 部分には、 SET INTEGRITY ステートメントが完了するまでアクセス不能のままになります。ユーザーは、SET INTEGRITY ステートメントを発行しないで、同じ表上で複数のロードを実行できます。ただし、元の (チェック済み) データは、 SET INTEGRITY ステートメントが発行されるまで可視のままです。

ALLOW READ ACCESS は、以下の修飾子もサポートします。

USE *tablespace-name*

索引が再作成される場合、表スペース *tablespace-name* に索引のシャドー・コピーが作成され、ロード終了時の INDEX COPY PHASE で、元の表スペース上にコピーされます。 SYSTEM TEMPORARY 表スペースのみ、このオプションを使用できます。指定されない場合、シャドー索引が、索引オブジェクトと同じ表スペース内に作成されます。シャドー・コピーが索引オブジェクトと同じ表スペース内に作成される場合、古い索引オブジェクトを介したシャドー索引オブジェクトのコピーは瞬時に終了します。シャドー・コピーが索引オブジェクトとは異なる表スペースにある場合、物理コピーが実行されます。これにはかなりの入出力および時間を要します。コピーは、表がオフラインの間、ロード終了時の INDEX COPY PHASE で行われます。

このオプションをしないと、シャドー索引は元の索引と同じ表スペースに作成されます。デフォルトでは、元の索引とシャドー索引の両方が同時に同じ表スペースに常駐するため、 1 つの表スペース内に両方の索引を保留するためのスペースが不足する場合があります。このオプションを使用すれば、索引用の十分な表スペースを確保できます。

ユーザーが INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT を指定しない場合、このオプションは無視されます。このオプションは INDEXING MODE AUTOSELECT が選択され、ロードが索引を徐々に更新することを選択した場合にも無視されます。

SET INTEGRITY PENDING CASCADE

LOAD によって表が SET INTEGRITY PENDING 状態になる場合、 SET INTEGRITY PENDING CASCADE オプションを使用することによって、ユーザーはロードされる表の SET INTEGRITY PENDING 状態を即時にすべての下層 (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を含む) にカスケードするかどうか指定することができます。

IMMEDIATE

SET INTEGRITY PENDING 状態が即時にすべての下層外部キー

表、下層即時マテリアライズ照会表、および下層ステージング表に拡張されることを示します。 LOAD INSERT 操作の場合、IMMEDIATE オプションが指定されている場合でも、 SET INTEGRITY PENDING 状態は下層外部キー表に拡張されません。

後で (SET INTEGRITY ステートメントの IMMEDIATE CHECKED オプションを使用して) ロードされる表の制約違反をチェックする際、SET INTEGRITY PENDING READ ACCESS 状態だった下層外部キー表は、SET INTEGRITY PENDING NO ACCESS 状態になります。

DEFERRED

ロードされる表だけが、SET INTEGRITY PENDING 状態になることを示します。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は、未変更のままになります。

下層外部キー表は、(SET INTEGRITY ステートメントの IMMEDIATE CHECKED オプションを使用して) その親表の制約違反がチェックされるとき、後で暗黙的に SET INTEGRITY PENDING 状態になる場合があります。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は、その基礎表のいずれかの保全性違反がチェックされる際、暗黙的に SET INTEGRITY PENDING 状態になります。従属表が SET INTEGRITY PENDING 状態になったことを示す警告が戻されます (SQLSTATE 01586)。この下層表がいつ SET INTEGRITY PENDING 状態になるかについては、 SQL リファレンスにある SET INTEGRITY ステートメントの「注」の項を参照してください。

SET INTEGRITY PENDING CASCADE オプションが指定されない場合、次のようになります。

- ロードされる表だけが、SET INTEGRITY PENDING 状態になります。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状態は、未変更のままになり、後にロードされた表の制約違反がチェックされる際に、暗黙的に SET INTEGRITY PENDING 状態になる場合があります。

LOAD によってターゲット表が SET INTEGRITY PENDING 状態にならない場合、SET INTEGRITY PENDING CASCADE オプションは無視されます。

LOCK WITH FORCE

ユーティリティーはロード・プロセス中に、表ロックなどの様々なロックを獲得します。ロックを獲得する際、このオプションを使用すると、ロードは待機することなく、またタイムアウトになることなく、ターゲット表に競合するロックを持つ他のアプリケーションを強制的にオフにします。システム・カタログ表に対する競合するロックを保留するアプリケーションは、ロード・ユーティリティーによって強制的にオフにされることはありません。強制されたアプリケーションは、ロールバックし、ロード・ユーティリティーが必要とするロックをリリースします。その後、ロード・ユーティリティーを続行できます。このオプションは、FORCE APPLICATIONS コマンドと同じ権限 (SYSADM または SYSCTRL) を必要とします。

ALLOW NO ACCESS は、ロード操作の開始時に競合するロックを持つアプリケーションを強制的にロールバックさせる場合があります。ロードの開始時に、ユーティリティーは、表の照会または変更を試みているアプリケーションを強制的にロールバックさせる場合があります。

ALLOW READ ACCESS は、ロード操作の開始時および終了時に競合するロックを持つアプリケーションを強制的にロールバックさせる場合があります。ロードの開始時に、ロード・ユーティリティーは、表の変更を試みているアプリケーションを強制的にロールバックさせる場合があります。ロード操作の終了時に、ロード・ユーティリティーは、表の照会または変更を試みているアプリケーションを強制的にロールバックさせる場合があります。

SOURCEUSEREXIT*executable*
このユーティリティーにデータを送るために呼び出される実行可能ファイル名を指定します。

REDIRECT

INPUT FROM

BUFFER *input-buffer*
input-buffer で指定されたバイトのストリームが、所定の実行可能ファイルを実行するプロセスの STDIN ファイル記述子に渡されます。

FILE *input-file*
このクライアント・サイドのファイルの内容が、所定の実行可能ファイルを実行するプロセスの STDIN ファイル記述子に渡されます。

OUTPUT TO

FILE *output-file*
STDOUT および STDERR ファイル記述子が、指定した完全に修飾されたサーバー・サイドのファイルに取り込まれます。

PARALLELIZE
複数のユーザー出口プロセスを同時に呼び出すことによって、ロード・ユーティリティーへのデータ入力のスループットを高めます。このオプションは、複数パーティション・データベース環境でのみ適用でき、単一パーティション・データベース環境では無視されます。

詳細については、『*カスタマイズしたアプリケーション (ユーザー出口) を使用したデータの移動*』を参照してください。

PARTITIONED DB CONFIG *partitioned-db-option*
複数のデータベース・パーティションに分散した表へのロードの実行を可能にします。 PARTITIONED DB CONFIG パラメーターを使用すると、パーティション・データベース固有の構成オプションを指定することができます。 *partitioned-db-option* の値は、以下のいずれかになります。

```
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
```

```
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

これらのオプションの詳しい説明については、『パーティション・データベース環境でのロード構成オプション』を参照してください。

RESTARTCOUNT
予約済み。

USING *directory*
予約済み。

XML 文書からデータをロードする例

XML データのロード

例 1

ユーザーは、表に挿入する文書を記述するために、XDS フィールドを使用してデータ・ファイルを構成しました。内容は以下のとおりです。

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

第 1 行では、XML 文書が file1.xml というファイル名で指定されています。文字区切りである二重引用符が XDS 内でも使用されているので、XDS 内の二重引用符は二重になっています。第 2 行では、XML 文書が file2.xml というファイル名で指定されています。その文書の開始点のバイト・オフセットは 23、長さは 45 バイトです。

例 2

ユーザーは、XML 列の構文解析や妥当性検査のオプションを指定しないでロード・コマンドを実行し、データのロードに成功します。

```
LOAD FROM data.del of DEL INSERT INTO mytable
```

CURSOR からの XML データのロード

カーソルからデータをロードする操作は、通常のリレーショナル列タイプの場合と同じです。ユーザーには 2 つの表 T1 と T2 があり、それぞれは C1 という 1 つの XML 列だけで構成されています。T1 から T2 への LOAD を実行するために、ユーザーはまずカーソルを宣言します。

```
DECLARE X1 CURSOR FOR SELECT C1 FROM T1;
```

次に、ユーザーは、カーソル・タイプを使用して LOAD を実行します。

```
LOAD FROM X1 of CURSOR INSERT INTO T2
```

カーソル・タイプに XML 固有の LOAD オプションを適用する操作は、ファイルからロードする場合と同じです。

使用上の注意

- データは、入力ファイル内に並んでいる順序でロードされます。特定の順序を希望する場合には、ロードが試行される前にデータをソートしてください。ソース・データの順序を保持する必要がなければ、ANYORDER ファイル・タイプ修飾子を使用できます。この修飾子については、以下の『*ロード・ユーティリティーのファイル・タイプ修飾子*』セクションを参照してください。
- ロード・ユーティリティーは、既存の定義に基づいて索引を作成します。ユニーク・キーの重複を処理するのに、例外表が使用されます。ユーティリティーは、参照保全を強制したり、制約検査を実行したり、ロードする表に従属するマテリアライズ照会表を更新したりすることはありません。参照制約またはチェック制約を含む表は、SET INTEGRITY ペンディング状態になります。 REFRESH IMMEDIATE として定義されているサマリー表、およびロードする表に依存するサマリー表もまた、SET INTEGRITY ペンディング状態になります。この表に関して、SET INTEGRITY ペンディング状態を解除するには、SET INTEGRITY ステートメントを発行してください。ロード操作は、複製されたマテリアライズ照会表に対しては実行できません。
- クラスタリング索引が表に存在する場合、ロード前にクラスタリング索引でデータをソートしてください。ただし、データはマルチディメンション・クラスタリング (MDC) 表にロードする前にソートする必要はありません。
- 保護された表へのロード時に例外表を指定すると、無効なセキュリティー・ラベルで保護されている行がその表に送られます。そのため、例外表にアクセスできるユーザーは、通常はアクセス権限のないデータにアクセスできてしまう可能性があります。セキュリティー・レベルを上げるために、誰に例外表アクセス権限を付与するかに注意し、行が修復されてロードする表にコピーされたら直ちにそれぞれの行を削除するとともに、使い終えた例外表は直ちにドロップしてください。
- 内部形式のセキュリティー・ラベルには、改行文字が含まれている可能性があります。 DEL ファイル形式を使用するファイルをロードする場合、この改行文字が区切り文字と間違われることがあります。この問題が起きた場合は、LOAD コマンドで delprioritychar ファイル・タイプ修飾子を指定することによって、区切り文字に以前のデフォルト優先順位を使用してください。
- DECLARE CURSOR コマンドの実行中に指定した DATABASE キーワードが CURSOR ファイル・タイプを使用してロードを実行する場合、現在接続されているデータベース (ロード用) の認証に使用されるユーザー ID およびパスワードが (DECLARE CURSOR コマンドの DATABASE オプションによって指定された) ソース・データベースの認証に使用されます。ユーザー ID またはパスワードがロード・データベースの接続に指定されない場合、ソース・データベースのユーザー ID とパスワードは DECLARE CURSOR コマンドの実行中に指定する必要があります。
- マルチパート PC/IXF ファイルの個々のパートを Windows システムから AIX システムにコピーするロード操作もサポートされています。すべてのファイルの名前を LOAD コマンドに指定する必要があります。例えば、LOAD FROM

DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1 のように記述します。論理分割 PC/IXF ファイルから Windows オペレーティング・システムにロードする操作は、サポートされていません。

- 失敗した LOAD を再開する場合の動作は、既存の動作と同じで、BUILD フェーズでは、索引の REBUILD モードの使用が強制されます。

LOAD TERMINATE と LOAD RESTART のディクショナリー管理のまとめ

TERMINATE ディレクティブの下で LOAD 処理を実行する場合のコンプレッション・ディクショナリー管理の動作を以下の表にまとめます。

表 39. LOAD TERMINATE のディクショナリー管理

表の COMPRESS 属性	LOAD の前にディクショナリーが存在するかどうか	TERMINATE: LOAD REPLACE KEEPDICTIONARY または LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	既存のディクショナリーを維持します。	何も維持しません。
YES	NO	何も維持しません。	何も維持しません。
NO	YES	既存のディクショナリーを維持します。	何も維持しません。
NO	NO	Do nothing.	Do nothing.

LOAD RESTART は、到達した最後の整合点まで表を切り捨てます。最後の LOAD 整合点が取られた時点で表にコンプレッション・ディクショナリーが存在していた場合は、LOAD RESTART 処理によって、コンプレッション・ディクショナリーが表に配置されます。その場合、LOAD RESTART が新しいディクショナリーを作成するわけではありません。考えられる条件を以下の表 4 にまとめます。

表 40. LOAD RESTART のディクショナリー管理

表の COMPRESS 属性	LOAD の整合点の前にディクショナリーが存在するかどうか	RESTART: LOAD REPLACE KEEPDICTIONARY または LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	既存のディクショナリーを維持します。	既存のディクショナリーを維持します。
YES	NO	ADC に基づいてディクショナリーを作成します。	ディクショナリーを作成します。
NO	YES	既存のディクショナリーを維持します。	既存のディクショナリーを除去します。
NO	NO	Do nothing.	Do nothing.

ロード・ユーティリティーのファイル・タイプ修飾子

表 41. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式

修飾子	説明
anyorder	この修飾子は、*cpu_parallelism* パラメーターと一緒に使用します。ソース・データの順序を保持する必要はない、という意味の指定なので、SMP システムのパフォーマンスがさらに向上します。*cpu_parallelism* の値が 1 になっていると、このオプションは無視されます。このオプションは、SAVECOUNT > 0 の場合はサポートされません。整合点の後のクラッシュ・リカバリーでは、順序のとおりにデータをロードする必要があるからです。
generatedignore	この修飾子を指定すると、ロード・ユーティリティーは、データ・ファイルに入っている、すべての生成済み列のデータを無視するようになります。その結果、すべての生成済み列の値がユーティリティーによって生成されます。この修飾子を generatedmissing 修飾子または generatedoverride 修飾子と一緒に使用することはできません。
generatedmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに生成済み列のデータが入っていない (NULL もない) という想定で動作します。その結果、すべての生成済み列の値がユーティリティーによって生成されます。この修飾子を generatedignore 修飾子または generatedoverride 修飾子と一緒に使用することはできません。

generatedoverride	この修飾子を指定すると、ロード・ユーティリティーは、表の中のすべての生成済み列でユーザー指定データを受け入れるようになります (この種の列の通常の規則とは反対の動作です)。別のデータベース・システムからデータをマイグレーションする場合や、ROLLFORWARD DATABASE コマンドの RECOVER DROPPED TABLE オプションを使用してリカバリーしたデータから表をロードする場合は、この修飾子を使用すると便利です。この修飾子を使用すると、NULL 不可の生成済み列にデータのない行や NULL データが入っている行は、リジェクトされます (SQL3116W)。この修飾子を使用すると、表は Set Integrity Pending 状態になります。ユーザー指定値を検証しないで表の Set Integrity Pending 状態を解除する場合は、ロード操作の後に以下のコマンドを実行します。
	`SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED`
	ユーザー指定値の検証を強制実行して表の Set Integrity Pending 状態を解除する場合は、ロード操作の後に以下のコマンドを実行します。
	`SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.`
	この修飾子を指定した場合、パーティション・キー、ディメンション・キー、分散キーのいずれかに生成済み列があれば、LOAD コマンドの実行時にその修飾子が自動的に generatedignore に変換され、ロードの処理が進められます。つまり、生成済み列のすべての値が再生成される結果になります。
	この修飾子を generatedmissing 修飾子または generatedignore 修飾子と一緒に使用することはできません。

identityignore	この修飾子を指定すると、ロード・ユーティリティーは、データ・ファイルに入っている、ID 列のデータを無視するようになります。その結果、すべての IDENTITY 値がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を identitymissing 修飾子または identityoverride 修飾子と一緒に使用することはできません。

表 41. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
identitymissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ID 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の ID 列の場合も GENERATED BY DEFAULT の ID 列の場合も同じです。この修飾子を `identityignore` 修飾子または `identityoverride` 修飾子と一緒に使用することはできません。
identityoverride	この修飾子を使用するのは、GENERATED ALWAYS として定義されている ID 列がロード対象の表に存在している場合に限られます。この修飾子を指定すると、ユーティリティーは、そのような列で明示的な非 NULL データを受け入れるようになります (この種の ID 列の通常の規則とは反対の動作です)。表を GENERATED ALWAYS として定義しなければならない状況で別のデータベース・システムからデータをマイグレーションする場合や、 ROLLFORWARD DATABASE コマンドの DROPPED TABLE RECOVERY オプションを使用してリカバリーしたデータから表をロードする場合は、この修飾子を使用すると便利です。この修飾子を使用すると、ID 列にデータのない行や NULL データが入っている行は、リジェクトされます (SQL3116W)。この修飾子を `identitymissing` 修飾子または `identityignore` 修飾子と一緒に使用することはできません。このオプションを使用すると、ロード・ユーティリティーは、表の ID 列に入っている値のユニーク性を保持したり検証したりする操作を試行しなくなります。
indexfreespace=*x*	*x* は、0 から 99 までの (両端を含む) 整数です。この値は、ロード操作で索引を再作成するときに、各索引ページに残すフリー・スペースのパーセンテージとして解釈されます。INDEXING MODE INCREMENTAL を指定したロード操作では、このオプションが無視されます。ページの最初の項目は、制限なしで追加されます。その後の項目は、フリー・スペースのパーセンテージしきい値を保持するために追加されます。デフォルト値は、CREATE INDEX の実行時に使用されていた値です。 この値は、CREATE INDEX ステートメントで指定されている PCTFREE 値よりも優先されます。 indexfreespace オプションの対象になるのは、索引のリーフ・ページだけです。

表 41. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
lobsinfile	*lob-path* では、LOB データが含まれているファイルのパスを指定します。ASC、DEL、IXF のロード入力ファイルには、LOB 列に LOB データが入っているファイルの名前が含まれています。 このオプションを CURSOR ファイル・タイプと併用することはできません。 lobsinfile 修飾子を使用するときには、LOB ファイルの配置場所を LOBS FROM 節で指定します。LOBS FROM 節を指定すると、LOBSINFILE の動作が暗黙的にアクティブになります。LOAD ユーティリティーは、データをロードするときに、LOB ファイルを検索するためのパスのリストを LOBS FROM 節から受け取ります。 各パスには、データ・ファイルの LOB ロケーション指定子 (LLS) によって参照されている LOB が少なくとも 1 つ入っているファイルが 1 つ以上含まれています。LLS は、LOB ファイル・パスに格納されているファイルの LOB の位置を示したストリング表記です。LLS の形式は、*filename.ext.nnn.mmm/* になります (*filename.ext* は、LOB が含まれているファイルの名前、*nnn* は、そのファイルに入っている LOB のオフセット (バイト単位)、*mmm* は、その LOB の長さ (バイト単位) です)。例えば、データ・ファイルにストリング `db2exp.001.123.456/` が格納されている場合は、ファイル db2exp.001 のオフセット 123 に LOB が配置されていて、その長さは 456 バイトということになります。 NULL LOB を指定する場合は、サイズとして -1 を入力します。サイズとして 0 を指定すると、長さ 0 の LOB として処理されます。長さ -1 の NULL LOB の場合は、オフセットとファイル名が無視されます。例えば、NULL LOB の LLS は、db2exp.001.7.-1/ のようになります。
noheader	ヘッダー検査コードをスキップします (該当するのは、単一パーティションのデータベース・パーティション・グループに存在する表へのロード操作だけです)。 単一パーティションのデータベース・パーティション・グループに存在する表に対してデフォルトの MPP ロード (モード PARTITION_AND_LOAD) を使用する場合は、ファイルにヘッダーが存在するとは考えられません。したがって、noheader 修飾子を指定する必要はありません。LOAD_ONLY モードを使用する場合は、ファイルにヘッダーが存在すると考えられます。noheader 修飾子が必要になるのは、ヘッダーのないファイルを使用して LOAD_ONLY 操作を実行する場合に限られます。
norowwarnings	リジェクトされた行についてのすべての警告を抑止します。
pagefreespace=*x*	*x* は、0 から 100 までの (両端を含む) 整数です。この値は、各データ・ページに残すフリー・スペースのパーセンテージとして解釈されます。最小行サイズのため、指定した値が無効である場合 (例えば、長さが少なくとも 3 000 バイトの行で、*x* の値が 50 である場合)、その行は新しいページに置かれます。値として 100 を指定すると、各行が新しいページに配置されます。表の PCTFREE 値は、ページごとに指定されたフリー・スペースの量を決定します。ロード操作の pagefreespace 値または表の PCTFREE 値が設定されていないと、ユーティリティーはそれぞれのページで可能なかぎり多くのスペースを満たします。pagefreespace に設定されている値は、表で指定されている PCTFREE 値をオーバーライドします。

表 41. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
rowchangetimestampignore	この修飾子を指定すると、ロード・ユーティリティーは、データ・ファイルに入っている、ROW CHANGE TIMESTAMP 列のデータを無視するようになります。その結果、すべての ROW CHANGE TIMESTAMP がユーティリティーによって生成されます。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。したがって、GENERATED ALWAYS 列の場合は、行がリジェクトされません。この修飾子を rowchangetimestampmissing 修飾子または rowchangetimestampoverride 修飾子と一緒に使用することはできません。
rowchangetimestampmissing	この修飾子を指定すると、ユーティリティーは、入力データ・ファイルに ROW CHANGE TIMESTAMP 列のデータが入っていない (NULL もない) という想定で動作し、各行の値を生成します。この動作は、GENERATED ALWAYS の列の場合も GENERATED BY DEFAULT の列の場合も同じです。この修飾子を rowchangetimestampignore 修飾子または rowchangetimestampoverride 修飾子と一緒に使用することはできません。
rowchangetimestampoverride	この修飾子を使用するのは、GENERATED ALWAYS として定義されている ROW CHANGE TIMESTAMP 列がロード対象の表に存在している場合に限られます。この修飾子を指定すると、ユーティリティーは、そのような列で明示的な非 NULL データを受け入れるようになります (この種の ROW CHANGE TIMESTAMP 列の通常の規則とは反対の動作です)。表を GENERATED ALWAYS として定義しなければならない状況で別のデータベース・システムからデータをマイグレーションする場合や、 ROLLFORWARD DATABASE コマンドの DROPPED TABLE RECOVERY オプションを使用してリカバリーしたデータから表をロードする場合は、この修飾子を使用すると便利です。この修飾子を使用すると、ROW CHANGE TIMESTAMP 列にデータのない行や NULL データが入っている行は、リジェクトされます (SQL3116W)。この修飾子を rowchangetimestampmissing 修飾子または rowchangetimestampignore 修飾子と一緒に使用することはできません。このオプションを使用すると、ロード・ユーティリティーは、表の ROW CHANGE TIMESTAMP 列に入っている値のユニーク性を保持したり検証したりする操作を試行しなくなります。

表 41. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明

seclabelchar	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、ストリング・フォーマットのセキュリティー・ラベル値であることを指定します。LOAD は、ロード時に各セキュリティー・ラベルを内部形式に変換します。ストリングが正しい形式になっていないと、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3242W) が戻されます。ストリングが、表を保護するセキュリティー・ポリシーの一部である有効なセキュリティー・ラベルに対応していなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3243W) が戻されます。
	
	seclabelname 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、ロードは失敗し、エラー (SQLCODE SQL3525N) が戻されます。
	
	1 つの DB2SECURITYLABEL 列だけで構成されている表の場合は、データ・ファイルを以下のように記述します。
	
	`"CONFIDENTIAL:ALPHA:G2"`
	`"CONFIDENTIAL;SIGMA:G2"`
	`"TOP SECRET:ALPHA:G2"`
	
	このデータのロードまたはインポートでは、以下のように SECLABELCHAR ファイル・タイプ修飾子を使用する必要があります。
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1`

seclabelname	入力ソース・ファイルに含まれているセキュリティー・ラベルが、デフォルトのエンコード数値形式ではなく、名前で示されていることを指定します。LOAD は、その名前に対応する適切なセキュリティー・ラベルがあれば、その名前をそのセキュリティー・ラベルに変換します。表を保護するセキュリティー・ポリシーに、その名前に対応するセキュリティー・ラベルが存在しなければ、行はロードされず、警告 (SQLSTATE 01H53、SQLCODE SQL3244W) が戻されます。
	
	seclabelchar 修飾子を指定した場合は、この修飾子を指定できません。そのようなことをすると、ロードは失敗し、エラー (SQLCODE SQL3525N) が戻されます。
	
	1 つの DB2SECURITYLABEL 列だけで構成されている表の場合は、データ・ファイルに以下のようなセキュリティー・ラベル名を組み込みます。
	
	`"LABEL1"`
	`"LABEL1"`
	`"LABEL2"`
	
	このデータのロードまたはインポートでは、以下のように SECLABELNAME ファイル・タイプ修飾子を使用する必要があります。
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1`
	
	注: ファイル・タイプが ASC の場合、セキュリティー・ラベル名の後のスペースは、名前の一部と解釈されます。そのような動作を避けるには、striptblanks ファイル・タイプ修飾子を使用して、スペースを除去するようにします。

表 41. ロード・ユーティリティーの有効なファイル・タイプ修飾子: すべてのファイル形式 (続き)

修飾子	説明
totalfreespace=*x*	*x* は、0 以上の整数です。この値は、表の合計ページ数に対する、フリー・スペースとして表の末尾に追加するページ数のパーセンテージとして解釈されます。例えば、*x* が 20 で、データのロード後に表に 100 個のデータ・ページがある場合は、20 個の空ページが追加されます。表のデータ・ページの総数は、120 になります。データ・ページの総数は、表の索引ページの数に関する因子にはなりません。このオプションは、索引オブジェクトには影響しません。このオプションを指定して 2 つのロード操作を実行する場合、最初のロード操作で末尾に追加されたスペースが 2 番目のロード操作で再利用されるわけでありません。
usedefaults	ターゲット表の列のソース列が指定されていても、1 つ以上の行インスタンスでその列にデータが入っていない場合は、デフォルト値がロードされます。欠落データの例を以下に示します。 • DEL ファイル: 列の値として、2 つの隣接した列区切り (″,,″) や、任意の数のスペースで分離した 2 つの列区切り (″, ,″) が指定されている場合。 • DEL/ASC/WSF ファイル: 十分な数の列がない行や、元の指定に対応した十分な長さがない行。ASC ファイルの場合、NULL 列値は、明示的な欠落とは見なされず、NULL 列値の代わりにデフォルトが入ることもありません。数値、日付、時刻、タイム・スタンプの列では、全桁スペース文字で NULL 列値を表記します。また、どのタイプの列でも、NULL INDICATOR を使用すれば、その列が NULL であることを示せます。 このオプションを指定しない場合に、行インスタンスのソース列にデータが入っていないと、以下のいずれかの動作が発生します。 • DEL/ASC/WSF ファイル: 列が NULL 可能であれば、NULL がロードされます。列が NULL 可能でなければ、ユーティリティーによって行がリジェクトされます。

表 42. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL)

修飾子	説明
codepage=*x*	*x* は、ASCII 文字ストリングです。この値は、入力データ・セットに含まれているデータのコード・ページとして解釈されます。ロード操作の実行時に、文字データ (および文字で指定されている数値データ) は、このコード・ページからデータベース・コード・ページに変換されます。 以下の規則が適用されます。 • DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。 • EBCDIC コード・ページで指定された DEL データの場合、区切り文字は DBCS のシフトイン文字およびシフトアウト文字と同じであってはなりません。 • nullindchar では、標準の ASCII セットのコード・ポイント x20 から x7F の範囲 (両端を含む) に含まれているシンボルを指定する必要があります。この修飾子では、ASCII のシンボルとコード・ポイントを参照します。EBCDIC データでは、コード・ポイントが違っていても、対応するシンボルを使用できます。 このオプションを CURSOR ファイル・タイプと併用することはできません。

表 42. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
dateformat="*x*"	*x* は、ソース・ファイルの日付の形式です。[1] 有効な日付エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (1 から 12 の範囲の 2 桁の数。 M とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月エレメントとは 相互に排他的) 指定されていないそれぞれのエレメントには、デフォルト値の 1 が割り当てられます。日付形式の例を以下に示します。 "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
dumpfile = *x*	*x* は、リジェクトされた行を書き込む例外ファイルの (サーバー・データベース・パーティションに応じた) 完全修飾名です。1 つのレコードごとに、最大 32 KB のデータが書き込まれます。以下は、ダンプ・ファイルの指定方法を示す例です。 db2 load from data of del modified by dumpfile = /u/user/filename insert into table_name このファイルは、インスタンス所有者によって作成され、所有されます。デフォルトのファイル許可をオーバーライドする場合は、dumpfileaccessall ファイル・タイプ修飾子を使用します。 **注:** 1. パーティション・データベース環境では、ロードする側のデータベース・パーティションから見てローカルのパスを指定する必要があります。そうすれば、同時に実行するいくつかのロード操作が同じファイルに書き込もうとする、という事態を避けられます。 2. ファイルの内容は、非同期のバッファー・モードでディスクに書き込まれます。ロード操作が失敗した場合や割り込みが発生した場合は、ディスクにコミットされたレコードの数を確実に把握する方法がありません。LOAD RESTART 後の整合性も保証できません。ファイルのロード操作が完了したと想定できるのは、ロード操作が開始と完了が 1 回のパスで完結している場合に限られます。 3. 指定したファイルがすでに存在している場合は、再作成ではなく追加になります。

表 42. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
dumpfileaccessall	ダンプ・ファイルの作成時に、読み取りアクセスを 'OTHERS' に付与します。 このファイル・タイプ修飾子が有効なのは、以下の場合に限られます。 1. dumpfile ファイル・タイプ修飾子と併用する場合 2. ユーザーがロードのターゲット表に対する SELECT 特権を持っている場合 3. UNIX オペレーティング・システムに配置されている DB2 サーバー・データベース・パーティションで実行する場合 指定したファイルがすでに存在している場合は、そのファイルの許可が変更なしでそのまま使用されます。
fastparse	注意して使用してください。ユーザー指定の列値の構文検査が削減されるので、パフォーマンスは向上します。表の正しいアーキテクチャーは確保できますが (つまり、ユーティリティーは、セグメンテーション違反やトラップを回避するための十分なデータ・チェックを実行しますが)、データの一貫性に関する検証は実行しません。このオプションを使用するのは、データの一貫性と正確さに自信がある場合だけにしてください。例えば、ユーザー指定のデータに無効なタイム・スタンプ列値 :1>0-00-20-07.11.12.000000 が含まれている場合でも、FASTPARSE が指定されていれば、その値は表に挿入されてしまいますが、FASTPARSE が指定されていなければ、その値はリジェクトされます。
implieddecimal	暗黙の小数点の位置が列定義によって決まるようになり、値の末尾という想定がなくなります。例えば、値 12345 は DECIMAL(8,2) 列に 12345.00 としてではなく、123.45 としてロードされます。 この修飾子を packeddecimal 修飾子と一緒に使用することはできません。
timeformat="*x*"	*x* は、ソース・ファイルの時刻の形式です。[1] 有効な時刻エレメントは、以下のとおりです。 H - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 1 桁または 2 桁の数) HH - 時 (12 時間制の場合は 0 から 12、 24 時間制では 0 から 24 の範囲の 2 桁の数; H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM) 指定されていないそれぞれのエレメントには、デフォルト値の 0 が割り当てられます。時刻形式の例を以下に示します。 "HH:MM:SS" "HH.MM TT" "SSSSS"

表 42. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
timestampformat="x"	x は、ソース・ファイルのタイム・スタンプの形式です。[1] 有効なタイム・スタンプ・エレメントは、以下のとおりです。 YYYY - 年 (0000 から 9999 の範囲の 4 桁の数) M - 月 (1 から 12 の範囲の 1 桁または 2 桁の数) MM - 月 (01 から 12 の 2 桁の数。 M および MMM とは相互に排他的) MMM - 月 (大文字小文字を区別しない月名の 3 文字の省略形。 M と MM とは相互に排他的) D - 日 (1 から 31 の範囲の 1 桁または 2 桁の数) DD - 日 (1 から 31 の範囲の 2 桁の数。 D とは相互に排他的) DDD - 元日から数えた日数 (001 から 366 の範囲の 3 桁の数。 他の日または月のエレメントとは相互に排他的) H - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 1 桁または 2 桁の数。) HH - 時 (12 時間制の場合は 0 から 12、24 時間制では 0 から 24 の 範囲の 2 桁の数。 H と相互に排他的) M - 分 (0 から 59 の範囲の 1 桁または 2 桁の数) MM - 分 (0 から 59 の範囲の 2 桁の数。 M (分) とは相互に排他的) S - 秒 (0 から 59 の範囲の 1 桁または 2 桁の数) SS - 秒 (0 から 59 の範囲の 2 桁の数。 S と相互に排他的) SSSSS - 夜中の 12 時から数えた秒数 (00000 から 86399 の範囲の 5 桁の数。 他の時刻エレメントとは相互に排他的) UUUUUU - マイクロ秒 (000000 から 999999 の範囲の 6 桁の数。 他のマイクロ秒エレメントとは相互に排他的) UUUUU - マイクロ秒 (00000 から 99999 の範囲の 5 桁の数。 000000 から 999990 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUUU - マイクロ秒 (0000 から 9999 の範囲の 4 桁の数。 000000 から 999900 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UUU - マイクロ秒 (000 から 999 の範囲の 3 桁の数。 000000 から 999000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) UU - マイクロ秒 (00 から 99 の範囲の 2 桁の数。 000000 から 990000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) U - マイクロ秒 (0 から 9 の範囲の 1 桁の数。 000000 から 900000 の範囲にマップされる。 他のマイクロ秒エレメントとは相互に排他的) TT - 午前/午後の指定子 (AM または PM)

表 42. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明

timestampformat="x" (続き)	指定されていない YYYY、M、MM、D、DD、DDD のいずれかのエレメントには、デフォルト値の 1 が割り当てられます。指定されていない MMM エレメントには、デフォルト値の 'Jan' が割り当てられます。指定されていない他のすべてのエレメントには、デフォルト値の 0 が割り当てられます。タイム・スタンプ形式の例を以下に示します。
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	MMM エレメントの有効な値は、
	'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov'、'dec' です。これらの値では、大/小文字は区別されません。
	TIMESTAMPFORMAT 修飾子を指定しなかった場合、ロード・ユーティリティーは、タイム・スタンプ・フィールドで以下の 2 つの有効な形式のいずれかを使用します。
	`YYYY-MM-DD-HH.MM.SS`
	`YYYY-MM-DD HH:MM:SS`
	ロード・ユーティリティーは、DD と HH の間の区切り文字に基づいて形式を選択します。ダッシュ '-' になっていれば、ロード・ユーティリティーは、通常のダッシュとドットの形式 (YYYY-MM-DD-HH.MM.SS) を使用します。ブランク・スペースになっていれば、ロード・ユーティリティーは、HH と MM と SS の間を区切るためにコロン ':' を使用します。
	どちらの形式でも、マイクロ秒のフィールド (UUUUUU) を組み込むと、ロード・ユーティリティーは、区切り文字としてドット '.' を使用します。YYYY-MM-DD-HH.MM.SS.UUUUUU も YYYY-MM-DD HH:MM:SS.UUUUUU も有効です。
	ユーザー定義の日付と時刻の形式が含まれているデータを schedule という表にロードする例を以下に示します。
	`db2 load from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

表 42. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASCII ファイル形式 (ASC/DEL) (続き)

修飾子	説明
usegraphiccodepage	usegraphiccodepage を指定すると、グラフィックまたは 2 バイト文字のラージ・オブジェクト (DBCLOB) データ・フィールドにロードするデータは、グラフィック・コード・ページのデータであるという想定で、処理が行われます。残りのデータは、文字コード・ページのデータであるという想定になります。グラフィック・コード・ページは、文字コード・ページに関連付けられています。LOAD は、codepage 修飾子が指定されていればその修飾子によって、codepage 修飾子が指定されていなければデータベースのコード・ページによって、文字コード・ページを判別します。 ドロップ済み表のリカバリーで生成される区切り付きデータ・ファイルとこの修飾子を併用するのは、リカバリーする表にグラフィック・データが入っている場合に限られます。 **制約事項** EXPORT ユーティリティーで作成される DEL ファイルでは、usegraphiccodepage 修飾子を指定しないでください。そのファイルには、1 つのコード・ページでエンコードされたデータだけが入っているからです。usegraphiccodepage 修飾子は、ファイルに含まれている 2 バイト文字ラージ・オブジェクト (DBCLOB) でも無視されます。
xmlchar	XML 文書が文字コード・ページでエンコードされていることを指定します。 指定の文字コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードが文字コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。文字コード・ページは、codepage ファイル・タイプ修飾子で指定されている値か、その修飾子が指定されていない場合はアプリケーション・コード・ページになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。
xmlgraphic	XML 文書が指定のグラフィック・コード・ページでエンコードされていることを指定します。 指定のグラフィック・コード・ページでエンコードされているものの、エンコード宣言が含まれていない XML 文書を処理するときに、このオプションは便利です。 それぞれの文書で、宣言タグが存在していて、エンコード属性が含まれている場合は、そのエンコードがグラフィック・コード・ページと一致している必要があります。そうでないと、その文書が含まれている行はリジェクトされます。グラフィック・コード・ページは、codepage ファイル・タイプ修飾子で指定されている値のグラフィック・コンポーネントか、その修飾子が指定されていない場合はアプリケーション・コード・ページのグラフィック・コンポーネントになります。デフォルトでは、Unicode で文書がエンコードされているか、エンコード属性の宣言タグが含まれている、という想定になります。

表 43. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASC ファイル形式 (区切りなし ASCII)

修飾子	説明
binarynumerics	数値データ (DECIMAL ではない) は、文字表現ではなく 2 進形式でなければなりません。このようにして、コストのかかる変換を回避します。 このオプションは、reclen オプションで指定する固定長レコードを使用する定位置 ASC でのみサポートされています。 以下の規則が適用されます。 • データ・タイプ間の変換は実行されません (ただし、BIGINT、INTEGER、SMALLINT は例外です)。 • データ長は、ターゲット列の定義と一致している必要があります。 • FLOAT は、IEEE 浮動小数点形式でなければなりません。 • ロード・ソース・ファイルのバイナリー・データは、ロード操作を実行するプラットフォームにかかわりなく、ビッグ・エンディアンであると見なされます。 この修飾子によって影響を受ける列のデータとして NULL を入れることはできません。通常は NULL として解釈されるブランクは、この修飾子の使用時にはバイナリー値として解釈されます。
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のロードが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にロードできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。
nullindchar=*x*	*x* は、単一文字です。NULL 値を示す文字を *x* に変更します。*x* のデフォルト値は、Y です。[2] この修飾子は、EBCDIC データ・ファイルでは大文字と小文字の区別があります。ただし、文字が英字の場合は例外です。例えば、NULL 標識文字が N に指定されている場合は、n も NULL 標識として認識されます。

packeddecimal	パック 10 進数データを直接ロードします。DECIMAL フィールド・タイプは、binarynumerics 修飾子の対象に含まれていません。
	
	このオプションは、reclen オプションで指定する固定長レコードを使用する定位置 ASC でのみサポートされています。
	
	符号ニブルとしてサポートされている値は、以下のとおりです。
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	この修飾子によって影響を受ける列のデータとして NULL を入れることはできません。通常は NULL として解釈されるブランクは、この修飾子の使用時にはバイナリー値として解釈されます。
	
	サーバー・プラットフォームにかかわりなく、ロード・ソース・ファイルのバイナリー・データのバイト・オーダーは、ビッグ・エンディアンであると見なされます。したがって、Windows オペレーティング・システムでこの修飾子を使用する場合は、バイト・オーダーを逆にしてはなりません。
	
	この修飾子を implieddecimal 修飾子と一緒に使用することはできません。

表 43. ロード・ユーティリティーの有効なファイル・タイプ修飾子: ASC ファイル形式 (区切りなし ASCII) (続き)

修飾子	説明
reclen=*x*	*x* は最大値が 32 767 の整数です。各行では *x* 個の文字が読み取られ、行の終わりを示す改行文字は使用されません。
striptblanks	可変長フィールドにデータをロードするときに、末尾ブランク・スペースを切り捨てます。このオプションを指定しなければ、ブランク・スペースは維持されます。 このオプションを striptnulls と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの t の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。
striptnulls	可変長フィールドにデータをロードするときに、末尾 NULL (0x00 文字) を切り捨てます。このオプションを指定しなければ、NULL は維持されます。 このオプションを striptblanks と一緒に指定することはできません。これらは、相互に排他的なオプションです。このオプションは、廃止オプションの padwithzero の代わりに用意されています。その廃止オプションは、旧バージョンとの互換性のためだけにサポートされています。

zoneddecimal	ゾーン 10 進数をロードします。DECIMAL フィールド・タイプは、BINARYNUMERICS 修飾子の対象に含まれていません。このオプションは、RECLEN オプションで指定する固定長レコードを使用する定位置 ASC でのみサポートされています。
	
	ハーフバイト符号値は、以下のいずれかになります。
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	数字としてサポートされている値は、0x0 から 0x9 です。
	
	ゾーンとしてサポートされている値は、0x3 から 0xF です。

表 44. ロード・ユーティリティーの有効なファイル・タイプ修飾子: DEL ファイル形式 (区切り付き ASCII)

修飾子	説明

chardel*x*	*x* は、単一文字ストリング区切りです。デフォルト値は、二重引用符 (") です。文字ストリングを囲む二重引用符の代わりに指定の文字を使用します。[23] 文字ストリング区切りとして二重引用符 (") を明示的に指定する場合は、以下のように指定します。
	`modified by chardel""`
	
	文字ストリング区切りとして単一引用符 (') を指定することもできます。その場合は、以下のようにします。
	`modified by chardel''`

coldel*x*	*x* は、単一文字列区切りです。デフォルト値は、コンマ (,) です。列の終わりを示すコンマの代わりに指定の文字を使用します。[23]
decplusblank	正符号文字。正の 10 進値の接頭部として、正符号 (+) の代わりにブランク・スペースを使用します。デフォルトのアクションでは、正の 10 進値の接頭部として正符号を使用します。
decpt*x*	*x* は、小数点文字としてピリオドの代わりに使用する単一文字です。デフォルト値は、ピリオド (.) です。小数点文字として、ピリオドの代わりに指定の文字を使用します。[23]

表 44. ロード・ユーティリティーの有効なファイル・タイプ修飾子: DEL ファイル形式 (区切り付き ASCII) (続き)

修飾子	説明

delprioritychar	区切り文字に関する現在のデフォルトの優先順位は、レコード区切り、文字区切り、列区切り、という順序になっています。この修飾子を指定すると、区切り文字の優先順位が、文字区切り、レコード区切り、列区切り、という順序に戻されるので、古い優先順位に依存する既存のアプリケーションが保護されます。構文:
	`db2 load ... modified by delprioritychar ...`
	例えば、以下の DEL データ・ファイルがあるとします。
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	delprioritychar 修飾子を指定しているので、このデータ・ファイルは、2 行だけになります。1 番目と 3 番目の <row delimiter> は、実際のレコード区切りとして解釈されますが、2 番目の <row delimiter> は、第 2 行の最初のデータ列の一部として解釈されるからです。この修飾子を指定しなければ、それぞれの <row delimiter> が区切り文字として解釈され、このデータ・ファイルは 3 行になります。

keepblanks	タイプ CHAR、VARCHAR、LONG VARCHAR、CLOB の各フィールドで前後のブランクを保持します。このオプションを指定しないと、文字区切りの内側にない前後のすべてのブランクが除去され、表のすべてのブランク・フィールドに NULL が挿入されます。
	データ・ファイルにある前後のスペースをすべて保持して、TABLE1 という表にデータをロードする例を以下に示します。
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nochardel	ロード・ユーティリティーは、列区切りの間で検出するすべてのバイトを列のデータの一部と見なします。文字区切りも、列データの一部として解析されます。DB2 でエクスポートしたデータについては、このオプションを指定しないでください (ただし、エクスポート時に nochardel を指定していた場合は例外です)。このオプションは、文字区切りのないベンダー・データ・ファイルをサポートするために用意されています。正しくない使い方をすると、データが失われたり破損したりする可能性があります。 このオプションを chardelx、delprioritychar、nodoubledel のいずれかと一緒に指定することはできません。これらは、相互に排他的なオプションです。
nodoubledel	二重文字区切りの認識を抑止します。

表 45. ロード・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式

修飾子	説明
forcein	ユーティリティーは、コード・ページの不一致があってもデータを受け入れ、コード・ページの変換を抑止します。 固定長ターゲット・フィールドについては、データを収容するだけの大きさがあるかどうかのチェックが行われます。nochecklengths を指定すると、チェックなしで各行のロードが試行されます。

表 45. ロード・ユーティリティーの有効なファイル・タイプ修飾子: IXF ファイル形式 (続き)

修飾子	説明
nochecklengths	nochecklengths を指定すると、ターゲット表の列のサイズを超える列定義がソース・データに含まれている場合でも、各行のロードが試行されるようになります。コード・ページ変換によってソース・データが縮小される場合は、そのような行を正常にロードできます。例えば、ソースにある 4 バイトの EUC データがターゲットで 2 バイトの DBCS データに縮小されれば、必要なスペースが半分になります。列定義の不一致があっても、すべてのソース・データがターゲットに収まることがわかっている場合は、このオプションが特に便利です。

注:

1. 日付形式ストリングを二重引用符で囲むのは、必須です。フィールド区切り文字には、a から z、A から Z、0 から 9 を組み込めません。フィールド区切り文字として、DEL ファイル形式の文字区切りまたはフィールド区切りと同じ文字を使用することはできません。エレメントの開始位置と終了位置があいまいでない場合は、フィールド区切り文字はオプションになります。修飾子によっては、項目が可変長の場合に D、H、M、S などのエレメントを使用することがあり、そのような場合は、開始位置と終了位置があいまいになることがあります。

   タイム・スタンプ形式の場合は、月の記述子と分の記述子の間であいまいさが残らないように注意する必要があります。どちらも、M という文字を使用するからです。月のフィールドは、他の日付フィールドと隣接している必要があります。分のフィールドは、他の時刻フィールドと隣接している必要があります。あいまいなタイム・スタンプ形式の例を以下に示します。

   ```
   "M" (月または分のどちらにもとれる)
   "M:M" (月と分の区別がつかない)
   "M:YYYY:M" (両方とも月と解釈される)
   "S:M:YYYY" (時刻値と日付値の両方に隣接している)
   ```

   あいまいな場合は、ユーティリティーによってエラー・メッセージが生成され、操作は失敗します。

   あいまいでないタイム・スタンプ形式の例を以下に示します。

   ```
   "M:YYYY" (M (月))
   "S:M" (M (分))
   "M:YYYY:S:M" (M (月)....M (分))
   "M:H:YYYY:M:D" (M (分)....M (月))
   ```

   二重引用符や円記号など、いくつかの文字の前ではエスケープ文字を使用する必要があります (¥ など)。

2. ファイル・タイプ修飾子 chardel、coldel、decpt に指定する文字値は、ソース・データのコード・ページに指定されている文字値でなければなりません。

   文字コード・ポイント (文字シンボルではない) を指定する場合は、xJJ または 0xJJ という構文を使用できます (JJ は、コード・ポイントの 16 進表記です)。例えば、列区切りとして # 文字を指定する場合は、以下のいずれかを使用します。

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

3. 区切り文字のオーバーライドとして使用できる文字に適用される制約事項については、『*データ移動のための区切り文字の制約事項*』を参照してください。
4. MODIFIED BY オプションでサポートされていないファイル・タイプを使用しようとしても、ロード・ユーティリティーからは警告が生成されません。その場合は、ロード操作が失敗し、エラー・コードが戻されます。
5. 暗黙的な非表示設定になっている Row Change Timestamp 列が含まれている表にインポートする場合は、その列の暗黙的な非表示のプロパティーが適用されません。したがって、インポートするデータに列のデータが含まれていない場合に、明示的な列リストも存在しなければ、インポート・コマンドで rowchangetimestampmissing ファイル・タイプ修飾子を指定することが必要です。

表 46. codepage と usegraphiccodepage を使用する場合の LOAD の動作

codepage=N	**usegraphiccodepage**	**LOAD の動作**
なし	なし	ファイル内のすべてのデータは、CLIENT オプションが指定されている場合でも、アプリケーション・コード・ページではなくデータベース・コード・ページのデータであるという想定になります。
あり	なし	ファイル内のすべてのデータは、コード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにロードすると、グラフィック・データが破損します。
なし	あり	ファイル内の文字データは、CLIENT オプションが指定されている場合でも、データベース・コード・ページのデータであるという想定になります。グラフィック・データは、CLIENT オプションが指定されている場合でも、データベース・グラフィック・データのコード・ページのデータであるという想定になります。 データベース・コード・ページが 1 バイトの場合は、すべてのデータがデータベース・コード・ページのデータであるという想定になります。 **警告:** グラフィック・データを 1 バイトのデータベースにロードすると、グラフィック・データが破損します。
あり	あり	文字データは、コード・ページ N のデータであるという想定になります。グラフィック・データは、N のグラフィック・コード・ページのデータであるという想定になります。 N が 1 バイトまたは 2 バイトのコード・ページの場合は、すべてのデータがコード・ページ N のデータであるという想定になります。 **警告:** N が 1 バイト・コード・ページの場合に、グラフィック・データをデータベースにロードすると、グラフィック・データが破損します。

db2Load - 表へのデータのロード

データを DB2 表にロードします。サーバー上にあるデータは、ファイル、カーソル、テープ、または名前付きパイプの形式とすることができます。リモートで接続しているクライアント上にあるデータは、完全修飾ファイル、カーソル、または名前付きパイプの形式とすることができます。ロード・ユーティリティーはインポート・ユーティリティーよりも高速ですが、階層レベルでのデータのロードまたはニックネームへのロードをサポートしません。

許可

以下のいずれか。

- *sysadm*
- *dbadm*
- データベースに対するロード権限と以下のもの
  - 表の INSERT 特権 (ロード・ユーティリティーが INSERT モード、TERMINATE モード、または RESTART モードで呼び出される場合)。TERMINATE モードは直前のロード挿入操作を終了するためのもので、RESTART モードは直前のロード挿入操作を再開するためのものです。
  - 表の INSERT および DELETE 特権 (ロード・ユーティリティーが REPLACE モード、 TERMINATE モード、または RESTART モードで呼び出される場合)。 TERMINATE モードは直前のロード置換操作を終了するためのもので、RESTART モードは直前のロード置換操作を再開するためのものです。
  - 例外表の INSERT 特権 (例外表をロード操作の一部として使用する場合)。

注: 一般的に、すべてのロード処理、およびすべての DB2 サーバー処理は、インスタンス所有者に所有されています。これらのすべての処理では、インスタンス所有者の ID を使用して、必要なファイルにアクセスします。そのため、インスタンス所有者は、誰がコマンドを呼び出すかに関係なく、入力ファイルへの読み取りアクセスを持っている必要があります。

必要な接続

データベース。暗黙的な接続が可能である場合には、デフォルトのデータベースへの接続が確立されます。 Linux、UNIX、または Windows クライアントから Linux、UNIX、または Windows データベース・サーバーへのユーティリティー・アクセスは、DB2 Connect ゲートウェイまたはループバック環境を経由してではなく、エンジンを使用した直接接続でなければなりません。

インスタンス。明示的なアタッチは必要ありません。データベースへの接続が確立されている場合には、ローカル・インスタンスへの暗黙的な接続が試みられます。

API 組み込みファイル

```
db2ApiDf.h
```

API およびデータ構造構文

```
SQL_API_RC SQL_API_FN
  db2Load (
   db2Uint32 versionNumber,
   void * pParmStruct,
```

```
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2LoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2PartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2LoadStruct;

typedef SQL_STRUCTURE db2LoadUserExit
{
   db2Char iSourceUserExitCmd;
   struct db2Char *piInputStream;
   struct db2Char *piInputFileName;
   struct db2Char *piOutputFileName;
   db2Uint16 *piEnableParallelism;
} db2LoadUserExit;

typedef SQL_STRUCTURE db2LoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   db2Uint16 iSetIntegrityPending;
   struct db2LoadUserExit *piSourceUserExit;
} db2LoadIn;

typedef SQL_STRUCTURE db2LoadOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsLoaded;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsDeleted;
   db2Uint64 oRowsCommitted;
} db2LoadOut;
```

```
typedef SQL_STRUCTURE db2PartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   SQL_PDB_NODE_TYPE *piRunStatDBPartNum;
} db2PartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint16 iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
   db2Uint16 iPortMin;
   db2Uint16 iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
   db2Uint64 oRowsRdPartAgents;
   db2Uint64 oRowsRejPartAgents;
   db2Uint64 oRowsPartitioned;
   struct db2LoadAgentInfo *poAgentInfoList;
   db2Uint32 iMaxAgentInfoEntries;
   db2Uint32 oNumAgentInfoEntries;
} db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
{
   db2int32 oSqlcode;
   db2Uint32 oTableState;
   SQL_PDB_NODE_TYPE oNodeNum;
   db2Uint16 oAgentType;
} db2LoadAgentInfo;

SQL_API_RC SQL_API_FN
  db2gLoad (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
```

```
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2gLoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2gPartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   db2Uint16 iFileTypeLen;
   db2Uint16 iLocalMsgFileLen;
   db2Uint16 iTempFilesPathLen;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2gLoadStruct;

typedef SQL_STRUCTURE db2gLoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 iUseTablespaceLen;
   db2Uint16 iSetIntegrityPending;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   struct db2LoadUserExit *piSourceUserExit;
} db2gLoadIn;

typedef SQL_STRUCTURE db2gPartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   void *piReserved1;
   db2Uint16 iHostnameLen;
   db2Uint16 iFileTransferLen;
   db2Uint16 iPartFileLocLen;
   db2Uint16 iMapFileInputLen;
```

```
   db2Uint16 iMapFileOutputLen;
   db2Uint16 iDistfileLen;
} db2gPartLoadIn;


/* Definitions for iUsing value of db2DMUXmlValidate structure               */
#define DB2DMU_XMLVAL_XDS              1         /* Use XDS                   */
#define DB2DMU_XMLVAL_SCHEMA           2         /* Use a specified schema    */
#define DB2DMU_XMLVAL_SCHEMALOC_HINTS  3         /* Use schemaLocation hints  */
#define DB2DMU_XMLVAL_ORIGSCHEMA       4         /* Use schema that document was
                                                    originally validated against
                                                    (load from cursor only) */
```

db2Load API パラメーター

versionNumber

入力。 2 番目のパラメーター pParmStruct として渡される、構造のバージョンとリリース・レベルを指定します。

pParmStruct

入力。 db2LoadStruct 構造を指すポインター。

pSqlca 出力。sqlca 構造を指すポインター。

db2LoadStruct データ構造パラメーター

piSourceList

入力。ソース・ファイル、装置、ベンダー、パイプ、または SQL ステートメントを提供するのに使用される、 sqlu_media_list 構造を指すポインター。

この構造に提供される情報は、media_type フィールドの値によって異なります。有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQLU_SQL_STMT

media_type フィールドがこの値に設定されている場合、呼び出し側は、ターゲット・フィールドの pStatement フィールドで SQL 照会を提供します。 pStatement フィールドは、sqlu_statement_entry のタイプです。セッション・フィールドは値を 1 に設定していなければなりません。これは、ロード・ユーティリティーはロードごとに 1 つの SQL 照会だけを受け取るからです。

SQLU_SERVER_LOCATION

media_type フィールドがこの値に設定されている場合、呼び出し側から sqlu_location_entry 構造によって情報が提供されます。
sessions フィールドは、提供される sqlu_location_entry 構造の数を示します。これは、ファイル、装置、および Named PIPE に使用されます。

SQLU_CLIENT_LOCATION

media_type フィールドがこの値に設定されている場合、呼び出し側から sqlu_location_entry 構造によって情報が提供されます。
sessions フィールドは、提供される sqlu_location_entry 構造の数を示します。これは、完全修飾ファイル、および Named PIPE に使用

されます。この media_type が有効なのは、リモートで接続されているクライアントを使用して API を呼び出している場合だけであることに注意してください。

SQLU_TSM_MEDIA
media_type フィールドがこの値に設定されている場合、sqlu_vendor 構造が使用されます。 filename には、ロードされるデータに固有の ID が入ります。 sessions の値がいくつであっても、 sqlu_vendor 項目の数は 1 つだけにする必要があります。 sessions フィールドは、開始される TSM セッションの数を示します。ロード・ユーティリティーは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの sqlu_vendor 項目にあるものと同じです。

SQLU_OTHER_MEDIA
media_type フィールドがこの値に設定されている場合、sqlu_vendor 構造が使用されます。 shr_lib には共有ライブラリー名、filename にはロードされるデータに固有の ID が入ります。 sessions の値がいくつであっても、 sqlu_vendor 項目の数は 1 つだけにする必要があります。 sessions フィールドは、開始されるその他のベンダー・セッションの数を示します。ロード・ユーティリティーは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの sqlu_vendor 項目にあるものと同じです。

SQLU_REMOTEFETCH
media_type フィールドがこの値に設定されている場合、呼び出し側から sqlu_remotefetch_entry 構造によって情報が提供されます。セッション・フィールドは値を 1 に設定していなければなりません。

piLobPathList
入力。 sqlu_media_list 構造を指すポインター。ファイル・タイプが IXF、ASC、および DEL の場合は、ロードされる個々の LOB ファイルのロケーションを識別する、完全修飾パスまたは装置のリスト。ファイル名は、IXF、ASC、または DEL ファイルで検索され、提供されたパスに追加されます。

この構造に提供される情報は、media_type フィールドの値によって異なります。有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQLU_LOCAL_MEDIA
この値に設定されている場合、呼び出し側から sqlu_media_entry 構造によって情報が提供されます。 sessions フィールドは、提供される sqlu_media_entry 構造の数を示します。

SQLU_TSM_MEDIA
この値に設定されている場合、sqlu_vendor 構造が使用されます。 filename には、ロードされるデータに固有の ID が入ります。 sessions の値がいくつであっても、 sqlu_vendor 項目の数は 1 つだけにする必要があります。 sessions フィールドは、開始される TSM セッションの数を示します。ロード・ユーティリティーは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの sqlu_vendor 項目にあるものと同じです。

SQLU_OTHER_MEDIA
この値に設定されている場合、sqlu_vendor 構造が使用されます。shr_lib には共有ライブラリー名、filename にはロードされるデータに固有の ID が入ります。 sessions の値がいくつであっても、sqlu_vendor 項目の数は 1 つだけにする必要があります。 sessions フィールドは、開始されるその他のベンダー・セッションの数を示します。ロード・ユーティリティーは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの sqlu_vendor 項目にあるものと同じです。

piDataDescriptor
入力。外部ファイルからロードするよう選択された列に関する情報を含む sqldcol 構造を指すポインター。

piFileType パラメーターが SQL_ASC に設定されている場合、この構造の dcolmeth フィールドを SQL_METH_L に設定する必要があります。ユーザーは、ロードする各列の開始位置と終了位置を指定します。

ファイル・タイプ SQL_DEL の場合、 dcolmeth は SQL_METH_P または SQL_METH_D のどちらかにすることができます。 SQL_METH_P の場合、ソース列の位置を提供する必要があります。 SQL_METH_D の場合は、ファイル内の最初の列が表の最初の列にロードされ、以下同様に続きます。

ファイル・タイプが SQL_IXF の場合、dcolmeth は SQL_METH_P、SQL_METH_D、または SQL_METH_N のいずれかにすることができます。この場合は、SQL_METH_N が sqldcol 構造でファイル列名が提供されるべきであることを示す点を除き、 DEL ファイルに関する規則が適用されます。

piActionString
推奨されません。piLongActionString に換わりました。

piLongActionString
入力。 4 バイト長のフィールドが含まれる *sqllob* 構造を指すポインターと、それに続いて表に影響するアクションを指定する文字の配列。

文字配列の形式は、以下のようになります。

```
"INSERT|REPLACE KEEPDICTIONARY|REPLACE RESETDICTIONARY|RESTART|TERMINATE
INTO tbname [(column_list)]
[FOR EXCEPTION e_tbname]"
```

INSERT
既存の表データを変更することなく、ロードされたデータを表に追加します。

REPLACE
表から既存データをすべて削除し、ロードされたデータを挿入します。表定義および索引定義は変更されません。

RESTART
以前に割り込みを受けたロード操作を再開します。ロード操作は、ロード、作成、または削除フェーズの最後の整合点から自動的に続行されます。

TERMINATE
以前に割り込みを受けたロード操作を終了し、ロード操作が開始された時点まで操作をロールバックします。途中に整合点があっても通過します。その操作に関係する表スペースの状態は通常に戻され、すべての表オブジェクトの整合性が保たれます (索引オブジェクトが無効とマークされる場合がありますが、そのような場合には、次回のアクセス時に索引の再作成が自動的に行われます)。表の存在する表スペースがロード・ペンディング状態でなければ、このオプションは表スペースの状態に影響しません。

ロード終了オプションでは、表スペースのバックアップ・ペンディング状態は解除されません。

tbname
データのロード先の表の名前。システム表または宣言された一時表を指定することはできません。別名、完全修飾、または非修飾の表名を指定することができます。修飾された表名は、schema.tablename の形式になります。非修飾の表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

(column_list)
データの挿入先の表の列名のリスト。列名は、コンマで区切らなければなりません。名前にスペースまたは小文字が含まれている場合には、それを引用符で囲まなければなりません。

FOR EXCEPTION e_tbname
エラーが発生した行のコピー先となる例外表を指定します。例外表は、ユニーク索引の規則、範囲制約、およびセキュリティー・ポリシーに違反する行のコピーを保管するために使用されます。

NORANGEEXC
範囲違反のためにリジェクトされた行は、例外表に挿入しないことを指定します。

NOUNIQUEEXC
ユニーク制約に違反しているためにリジェクトされた行は、例外表に挿入しないことを指定します。

piFileType
入力。入力データ・ソースの形式を示すストリング。サポートされている外部の形式 (sqlutil で定義) は、以下のとおりです。

SQL_ASC
区切り文字なし ASCII。

SQL_DEL
区切り文字付き ASCII。これは dBase プログラム、BASIC プログラム、 IBM パーソナル・デシジョン・シリーズ・プログラム、およびその他の多数のデータベース・マネージャー/ファイル・マネージャーとの交換のための形式です。

SQL_IXF
IXF (統合交換フォーマットの PC バージョン)。表からデータをエ

クスポートする場合の推奨方式で、同じ表または別のデータベース・マネージャー表にそれをロードすることが可能です。

SQL_CURSOR
SQL 照会。 piSourceList パラメーターによって渡された sqlu_media_list 構造のタイプは SQLU_SQL_STMT または SQLU_REMOTEFETCH のいずれかであり、SQL 照会または表名を参照します。

piFileTypeMod
入力。 sqlchar 構造を指すポインターと、それに続いて 1 つ以上の処理オプションを指定する文字の配列。このポインターが NULL であるか、このポインターが指す構造に 1 文字も入っていない場合、このアクションはデフォルトの指定が選択されたものとして解釈されます。

サポートされるすべてのファイル・タイプに、すべてのオプションを使用できるわけではありません。関連リンクの「ロード・ユーティリティー用のファイル・タイプ修飾子」を参照してください。

piLocalMsgFileName
入力。出力メッセージの書き込み先となるローカル・ファイルの名前を含むストリング。

piTempFilesPath
入力。一時ファイル用のサーバー上で使用されるパス名を含むストリング。一時ファイルは、メッセージや整合点を格納したり、フェーズ情報を削除したりするために作成されます。

piVendorSortWorkPaths
入力。ベンダー・ソート作業ディレクトリーを指定する sqlu_media_list 構造を指すポインター。

piCopyTargetList
入力。 sqlu_media_list 構造へのポインター。これは、 (コピー・イメージを作成する予定の場合) コピー・イメージの書き込み先となるターゲット・パス、装置、または共有ライブラリーのリストを提供するときに使用します。

この構造に入力する値は、media_type フィールドの値によって異なります。このパラメーターの有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQLU_LOCAL_MEDIA
コピーをローカル・メディアに書き込む予定の場合、media_type をこの値に設定し、ターゲットに関する情報を sqlu_media_entry 構造に提供してください。 sessions フィールドは、提供される sqlu_media_entry 構造の数を示します。

SQLU_TSM_MEDIA
コピーを TSM に書き込む予定の場合、この値を使用してください。それ以外の情報は特に必要ありません。

SQLU_OTHER_MEDIA
ベンダー製品を使用する予定の場合、この値を使用し、 sqlu_vendor 構造を介して追加の情報を提供してください。この構造の shr_lib フィールドをベンダー製品の共用ライブラリー名に設定してくださ

い。 sessions の値に関係なく、1 つの sqlu_vendor 項目だけを提供してください。 sessions フィールドは、提供される sqlu_media_entry 構造の数を示します。ロード・ユーティリティーは、異なるシーケンス番号を持つセッションを開始しますが、ロードされるデータは、1 つの sqlu_vendor 項目で提供されているものと同じです。

piNullIndicators
入力。 ASC ファイルの場合にのみ使用します。列データが NULL 可能であるかどうかを示す整数の配列です。この配列のエレメントと、データ・ファイルからロードされる列との間には、 1 対 1 の順序付けられた対応関係があります。要するに、エレメントの数は、piDataDescriptor パラメーターの dcolnum フィールドと同じでなければなりません。配列の各エレメントには、NULL 標識フィールドとして使用される、データ・ファイル内のロケーションを識別する数値、または表列が NULL 可能ではないことを示すゼロが含まれます。エレメントがゼロでない場合には、データ・ファイル内の識別されたロケーションに Y または N が入っていなければなりません。 Y は表列のデータが NULL であることを示し、 N は表列のデータが NULL ではないことを示します。

piLoadInfoIn
入力。 db2LoadIn 構造を指すポインター。

poLoadInfoOut
出力。db2LoadOut 構造を指すポインター。

piPartLoadInfoIn
入力。 db2PartLoadIn 構造を指すポインター。

poPartLoadInfoOut
出力。db2PartLoadOut 構造を指すポインター。

iCallerAction
入力。呼び出し側が要求するアクションを示します。有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQLU_INITIAL
最初の呼び出し。この値 (または SQLU_NOINTERRUPT) は、API への最初の呼び出しの際には必ず使用してください。

SQLU_NOINTERRUPT
最初の呼び出し。処理を中断しません。この値 (または SQLU_INITIAL) は、API への最初の呼び出しの際には必ず使用してください。

最初の呼び出しまたは後続の呼び出しのいずれかが戻され、要求されたロード操作が完了する前に呼び出し側のアプリケーションが何らかのアクションを行うことが必要な場合、呼び出し側のアクションを以下のどちらかに設定する必要があります。

SQLU_CONTINUE
処理の継続。この値を使用できるのは、最初の呼び出しが戻されたときにユーティリティーがユーザー入力 (例えば、テープの終わり

条件への応答) を要求した後で、 API への後続呼び出しを出す場合だけです。この値は、ユーティリティーが要求したユーザー・アクションが完了したら、ユーティリティーが最初の要求の処理を続行するよう指定するものです。

SQLU_TERMINATE
処理の終了。ロード中の表スペースを LOAD_PENDING 状態にしたまま、ロード・ユーティリティーを早期に終了させます。このオプションは、これ以上データの処理が行われない場合に指定します。

SQLU_ABORT
処理の終了。ロード中の表スペースを LOAD_PENDING 状態にしたまま、ロード・ユーティリティーを早期に終了させます。このオプションは、これ以上データの処理が行われない場合に指定します。

SQLU_RESTART
処理の再開。

SQLU_DEVICE_TERMINATE
単一の装置の終了。このオプションは、ユーティリティーが装置からの読み取りを停止しても、データの処理をさらに続ける場合に指定します。

piXmlPathList
入力。 media_type フィールドを SQLU_LOCAL_MEDIA に設定された sqlu_media_list、および xml ファイルが置かれているクライアント上のパスをリストするその sqlu_media_entry 構造を指すポインター。

db2LoadUserExit データ構造パラメーター

iSourceUserExitCmd
入力。データをユーティリティーに送るために使用される実行可能ファイルの完全修飾名。セキュリティー上の理由で、この実行可能ファイルはサーバー上の `sqllib/bin` ディレクトリー内に置く必要があります。
piSourceUserExit 構造が NULL ではない場合、このパラメーターは必須です。

piInputStream、piInputFileName、piOutputFileName、piEnableParallelism の各フィールドはオプションです。

piInputStream
入力。 STDIN を経由してユーザー出口アプリケーションに直接渡される汎用バイト・ストリーム。このバイト・ストリームにどんなデータを含めるか、およびどんなフォーマットにするかについては、ユーザーが完全に制御できます。ロード・ユーティリティーは、このバイト・ストリームをサーバーに送り、プロセスの STDIN にフィードして、ユーザー出口アプリケーションに渡すだけです (コード・ページの変換またはバイト・ストリームの変更は行われません)。ユーザー出口アプリケーションは STDIN から引数を読み取り、適切な方法でデータを使用します。

このフィーチャーの重要な属性の 1 つは、機密情報 (ユーザー ID、パスワードなど) を隠す機能です。

piInputFileName
入力。完全修飾されたクライアント・サイド・ファイルの名前を含みます。そのファイルの内容は、プロセスの STDIN をフィードすることによりユーザー出口アプリケーションに渡されます。

piOutputFileName
入力。サーバー・サイド・ファイルの完全修飾名。ユーザー出口アプリケーションを実行しているプロセスの STDOUT および STDERR ストリームは、このファイルにストリーム入力されます。 piEnableParallelism が TRUE のとき、複数のファイル (ユーザー出口インスタンスごとに 1 つ) が作成されて、各ファイル名には 3 桁の数字のノード番号値が付加されます (`<filename>.000` など)。

piEnableParallelism
入力。ユーザー出口アプリケーションの起動を並列化するようにユーティリティーに指示するフラグ。

db2LoadIn データ構造パラメーター

iRowcount
入力。ロードされる物理レコードの数。これを使用すると、ファイル内の最初の rowcnt 個の行だけをロードすることができます。

iRestartcount
入力。将来の使用のために予約されています。

piUseTablespace
入力。索引が再作成されている場合、索引のシャドー・コピーが表スペース iUseTablespaceName 内に作成され、ロード終了時に元の表スペースにコピーされます。 SYSTEM TEMPORARY 表スペースのみ、このオプションを使用できます。指定されない場合、シャドー索引が、索引オブジェクトと同じ表スペース内に作成されます。

シャドー・コピーが索引オブジェクトと同じ表スペース内に作成される場合、古い索引オブジェクトを介したシャドー索引オブジェクトのコピーは瞬時に終了します。シャドー・コピーが索引オブジェクトとは異なる表スペースにある場合、物理コピーが実行されます。これにはかなりの入出力および時間を要します。コピーは、表がロード終了時にオフラインであるときに行われます。

iAccessLevel が SQLU_ALLOW_NO_ACCESS である場合、このフィールドは無視されます。

ユーザーが INDEXING MODE REBUILD または INDEXING MODE AUTOSELECT を指定しない場合、このオプションは無視されます。このオプションは INDEXING MODE AUTOSELECT が選択され、ロードが索引を徐々に更新することを選択した場合にも無視されます。

iSavecount
整合点を確立する前にロードするレコードの数。この値はページ・カウントに変換され、エクステント・サイズのインターバルに切り上げられます。それぞれの整合点でメッセージが発行されるため、 db2LoadQuery - 照会のロードを用いてロード操作をモニターする場合には、このオプションを選択す

る必要があります。 savecount の値が十分な大きさでない場合、各整合点で実行される活動の同期化によってパフォーマンスに影響してしまいます。

デフォルト値は 0 ですが、それは、必要がなければ整合点は確立されないことを意味します。

iDataBufferSize
ユーティリティー内でデータ転送用のバッファー・スペースとして使用される 4KB ページの数 (並列処理の度合いとは無関係)。指定された値がアルゴリズムの最小値よりも小さい場合には、必要最低限のページが使用され、警告は戻されません。

このメモリーは、ユーティリティー・ヒープから直接に割り当てられ, そのサイズは util_heap_sz データベース構成パラメーターで修正可能です。

値が指定されていない場合、実行時にユーティリティーによって適切なデフォルトが計算されます。デフォルトは、ローダーのインスタンス生成時にユーティリティー・ヒープで使用できるフリー・スペースの割合と、表の一部の特性に基づいて決まります。

iSortBufferSize
入力。このオプションは、ロード操作時に SORTHEAP データベース構成パラメーターをオーバーライドする値を指定します。これは表を索引とともにロードする場合、および iIndexingMode パラメーターが SQLU_INX_DEFERRED として指定されない場合にのみ関係があります。指定される値は、SORTHEAP の値を超えることはできません。このパラメーターは、一般的な照会処理にも影響を与える SORTHEAP の値を変更せずに、 LOAD によって使用されるソート・メモリーをスロットルするために役立ちます。

iWarningcount
入力。 warningcnt 個の警告後に、ロード操作を停止します。このパラメーターは、警告は予期されないが、正しいファイルと表が使用されていることを確認するのが望ましい場合に設定してください。ロード・ファイルまたはターゲット表が不適切に指定されると、ロード対象の各行ごとにロード・ユーティリティーによって警告が生成され、このためにロードが失敗する可能性があります。 warningcnt が 0 であるか、またはこのオプションを指定していない場合には、ロード操作は、発行された警告の数に関係なく続行されます。

警告のしきい値を超過したためにロード操作が停止された場合には、RESTART モードでもう一度ロード操作を開始することができます。ロード操作は、最後の整合点から自動的に続行します。または、入力ファイルの先頭から REPLACE モードであらためてロード操作を開始できます。

iHoldQuiesce
入力。ユーティリティーによって、ロード後に表を排他静止状態のままにする場合は TRUE、それ以外の場合は FALSE に値が設定されるフラグ。

iCpuParallelism
入力。ロード・ユーティリティーが表オブジェクトの作成時にレコードを解析、変換、および形式化するために作成するプロセスつまりスレッドの数。このパラメーターは、パーティション内並列処理を活用するために設計されています。これは、事前にソートされたデータをロードする際に役立ちます

(ソース・データのレコード順序が保持されるため)。このパラメーターの値がゼロである場合には、ロード・ユーティリティーは実行時に適切なデフォルト値を使用します。注: このパラメーターが LOB または LONG VARCHAR フィールドを含む表について使用されると、システム CPU の数やユーザーによって指定された値に関係なく、値は 1 になります。

iDiskParallelism
入力。ロード・ユーティリティーがデータを表スペース・コンテナーに書き込むために作成するプロセスつまりスレッドの数。値を指定しない場合、ユーティリティーは表スペース・コンテナーの数と表の特性に基づいて、自動的に計算された適切なデフォルトを選択します。

iNonrecoverable
入力。ロード・トランザクションがリカバリー不能としてマークされ、後続のロールフォワード・アクションによってリカバリーできない場合には、SQLU_NON_RECOVERABLE_LOAD に設定します。ロールフォワード・ユーティリティーはそのトランザクションをスキップし、データのロード先の表を ″invalid (無効)″ としてマークします。さらに、このユーティリティーは、それ以降のその表に対するトランザクションをすべて無視します。ロールフォワードが完了したら、そのような表はドロップするしかありません。このオプションを指定すると、表スペースはロード操作後にバックアップ・ペンディング状態になりません。また、ロードしたデータのコピーをロード操作中に作成する必要はありません。ロード・トランザクションがリカバリー可能としてマークされる場合には、 SQLU_RECOVERABLE_LOAD に設定します。

iIndexingMode
入力。索引付けモードを指定します。有効な値は以下のとおりです (インクルード・ディレクトリーの sqlutil ヘッダー・ファイルで定義される)。

SQLU_INX_AUTOSELECT
LOAD は REBUILD と INCREMENTAL 索引モードの間で選択します。

SQLU_INX_REBUILD
表索引を再作成します。

SQLU_INX_INCREMENTAL
既存の索引を拡張します。

SQLU_INX_DEFERRED
表索引を更新しません。

iAccessLevel
入力。アクセス・レベルを指定します。有効な値は以下のとおりです。

SQLU_ALLOW_NO_ACCESS
ロードが表を排他ロックするように指定します。

SQLU_ALLOW_READ_ACCESS
表の元データ (非差分部分) が、ロードが進行中の間、リーダーに対して可視のままであるように指定します。このオプションは、ロードの付加 (例えば、ロードの挿入など) に対してのみ有効です。ロード置換に対しては無視されます。

iLockWithForce

入力。ブール・フラグ。 TRUE に設定された場合、ロードは必要に応じて他のアプリケーションに対し、必ず即時に表ロックを得るように強制します。このオプションは、FORCE APPLICATIONS コマンドと同じ権限 (SYSADM または SYSCTRL) を必要とします。

SQLU_ALLOW_NO_ACCESS ロードは、ロード操作の開始時に、アプリケーションの競合を強制終了させることができます。ロードの開始時に、このユーティリティーは、表の照会または変更を試みているアプリケーションを強制終了させることができます。

SQLU_ALLOW_READ_ACCESS ロードは、ロード操作の開始時または終了時に、アプリケーションの競合を強制終了させることができます。ロードの開始時に、このロード・ユーティリティーは、表の変更を試みているアプリケーションを強制終了させることができます。ロードの終了時に、このロード・ユーティリティーは、表の照会または変更を試みているアプリケーションを強制終了させることができます。

iCheckPending

バージョン 9.1 では、このパラメーターは推奨されていません。代わりに iSetIntegrityPending パラメーターを使用します。

iRestartphase

入力。予約済み。有効な値は、シングル・スペース文字 ' ' です。

iStatsOpt

入力。収集する統計の細分性。有効な値は以下のとおりです。

SQLU_STATS_NONE
統計は収集されません。

SQLU_STATS_USE_PROFILE
現在の表に定義されたプロファイルに基づいて、統計が収集されます。このプロファイルを作成するには、RUNSTATS コマンドを使用する必要があります。現在の表に関するプロファイルが存在しない場合、警告が戻され、統計は収集されません。

iSetIntegrityPending

入力。表を SET INTEGRITY ペンディング状態にするように指定します。SQLU_SI_PENDING_CASCADE_IMMEDIATE が指定されている場合、 SET INTEGRITY ペンディング状態は即時にすべての従属表および下層表にカスケードされます。値として SQLU_SI_PENDING_CASCADE_DEFERRED が指定されている場合、SET INTEGRITY ペンディング状態の従属表へのカスケードは、ターゲット表の保全性違反がチェックされるまで据え置かれます。このオプションが指定されない場合、
SQLU_SI_PENDING_CASCADE_DEFERRED がデフォルト値となります。

piSourceUserExit

入力。 db2LoadUserExit 構造を指すポインター。

piXmlParse

入力。 XML 文書に対して行われる必要のある解析のタイプ。 include ディレクトリーに入っている db2ApiDf ヘッダー・ファイル内の有効値は、次のとおりです。

DB2DMU_XMLPARSE_PRESERVE_WS
空白が保持されます。

DB2DMU_XMLPARSE_STRIP_WS
空白が取り除かれます。

piXmlValidate
入力。 db2DMUXmlValidate 構造を指すポインター。 XML 文書の XML スキーマ検証を行う必要があることを示します。

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                        iUsing;      /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2LoadOut データ構造パラメーター

oRowsRead
出力。ロード操作中に読み取られたレコードの数。

oRowsSkipped
出力。ロード操作が開始される前にスキップされたレコードの数。

oRowsLoaded
出力。ターゲット表にロードされた行の数。

oRowsRejected
出力。ロードできなかったレコードの数。

oRowsDeleted
出力。削除された重複行の数。

oRowsCommitted
出力。処理されたレコードの合計数。正常にロードされ、データベースにコミットされたレコードの数と、スキップまたはリジェクトされたレコードの数の合計。

db2PartLoadIn データ構造パラメーター

piHostname
入力。 iFileTransferCmd パラメーターのホスト名。 NULL の場合、ホスト名のデフォルトは「nohost」です。このパラメーターは、推奨されていません。

piFileTransferCmd
入力。ファイル転送コマンドのパラメーター。必要ない場合、NULL に設定する必要があります。このパラメーターは、推奨されていません。代わりに piSourceUserExit パラメーターを使用します。

piPartFileLocation
入力。 PARTITION_ONLY、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、このパラメーターは、パーティション・ファイルのロケーションを指定するために使用できます。このロケ

ーションは、piOutputNodes オプションで指定された各データベース・パーティションに存在している必要があります。

SQL_CURSOR ファイル・タイプの場合、このパラメーターは NULL にすることはできません。ロケーションはパスを参照しませんが、完全修飾されたファイル名を参照します。これは、PARTITION_ONLY モードの場合は、各出力データベース・パーティションで作成されたパーティション・ファイルの完全修飾された基本ファイル名であり、LOAD_ONLY モードの場合は、各データベース・パーティションから読み取られるファイルのロケーションです。 PARTITION_ONLY モードの場合、ターゲット表に LOB 列が存在するならば、指定された基本名のファイルが複数作成されることがあります。 SQL_CURSOR 以外のファイル・タイプでは、このパラメーターの値が NULL の場合、デフォルトで現行ディレクトリーになります。

piOutputNodes
入力。ロード出力データベース・パーティションのリスト。 NULL は、ターゲット表が定義されたすべてのノードを示します。

piPartitioningNodes
入力。パーティション・ノードのリスト。 NULL はデフォルトを示します。

piMode
入力。パーティション・データベースのロード・モードを指定します。有効な値は以下のとおりです (インクルード・ディレクトリーの db2ApiDf ヘッダー・ファイルで定義される)。

- **DB2LOAD_PARTITION_AND_LOAD**
  データは (多くの場合は並列で) 分散され、それぞれ対応するデータベース・パーティションに同時にロードされます。

- **DB2LOAD_PARTITION_ONLY**
  データは (多くの場合は並列で) 分散され、それぞれのロード・データベース・パーティションの指定したファイルに出力が書き込まれます。 SQL_CURSOR 以外のファイル・タイプに関して、各データベース・パーティション上の出力ファイルの名前の形式は filename.xxx となります。ここで、filename は、piSourceList で指定された最初の入力ファイルの名前で、 xxx はデータベース・パーティションの番号です。SQL_CURSOR ファイル・タイプの場合、各データベース・パーティション上の出力ファイルの名前は、piPartFileLocation パラメーターによって判別されます。各データベース・パーティション上のデータベース・パーティション・ファイルの位置の指定方法については、piPartFileLocation パラメーターを参照してください。

  注: このモードは CLI LOAD には使用できません。

DB2LOAD_LOAD_ONLY
データはすでに分散されているものとします。この場合は分散プロセスが省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。 SQL_CURSOR 以外のファイル・タイプの場合、各データベース・パーティションの入力ファイル名の形式は filename.xxx となり、ここで、filename は

piSourceList で指定された最初のファイルの名前で、 xxx は 13 桁のデータベース・パーティション番号です。 SQL_CURSOR ファイル・タイプの場合、各データベース・パーティション上の入力ファイルの名前は piPartFileLocation パラメーターによって判別されます。各データベース・パーティション上のデータベース・パーティション・ファイルの位置の指定方法については、piPartFileLocation パラメーターを参照してください。

注: このモードは、リモート・クライアント上にあるデータ・ファイルのロード時に使用したり、または CLI LOAD には使用できません。

DB2LOAD_LOAD_ONLY_VERIFY_PART
データが既に分散されているとみなします。しかし、データ・ファイルにはデータベース・パーティション・ヘッダーがありません。分散プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいデータベース・パーティションにあることがチェックされます。 dumpfile ファイル・タイプ修飾子が指定されている場合には、データベース・パーティション違反のある行がダンプ・ファイルに入れられます。そうでなければ、行は廃棄されます。データベース・パーティション違反が特定のロード・データベース・パーティションに存在する場合、1 つの警告が、そのデータベース・パーティションのロード・メッセージ・ファイルに書き込まれます。各データベース・パーティションの入力ファイル名の形式は filename.xxx となります。ここで、filename は piSourceList で指定された最初のファイルの名前で、 xxx は 13 桁のデータベース・パーティション番号です。

注: このモードは、リモート・クライアント上にあるデータ・ファイルのロード時に使用したり、または CLI LOAD には使用できません。

DB2LOAD_ANALYZE
すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。

piMaxNumPartAgents
入力。パーティション・エージェントの最大数。 NULL 値はデフォルトを示します。デフォルトは 25 です。

piIsolatePartErrs
入力。ロード操作が、個々のデータベース・パーティションで発生するエラーに対応する方法を示します。有効な値は以下のとおりです (インクルード・ディレクトリーの db2ApiDf ヘッダー・ファイルで定義される)。

DB2LOAD_SETUP_ERRS_ONLY
このモードでは、セットアップ時にデータベース・パーティションで生じるエラー (例えば、データベース・パーティションへのアクセスに関する問題や、データベース・パーティションの表スペースまたは表へのアクセスに関する問題) によって、失敗したデータベース・パーティションではロード操作が停止してしまいますが、残

りのデータベース・パーティションでは操作が続行されます。データのロード中にデータベース・パーティションで生じるエラーによって、全操作が失敗し、各データベース・パーティションの最後の整合点にロールバックされます。

DB2LOAD_LOAD_ERRS_ONLY
このモードでは、セットアップ時にデータベース・パーティションで生じるエラーによって、ロード操作全体が失敗します。データのロード中にエラーが生じた場合、エラーのあるデータベース・パーティションは最後の整合点にロールバックされます。ロード操作は、失敗が生じるまで、またはすべてのデータがロードされるまで、残りのデータベース・パーティションで続行します。すべてのデータがロードされたデータベース・パーティションでは、ロード操作後は、データは可視ではありません。他のデータベース・パーティションで生じたエラーのため、トランザクションは打ち切られます。すべてのデータベース・パーティション上のデータは、ロードの再開操作が実行されるまで、不可視のままです。これにより、ロード操作が完了したデータベース・パーティション上では新たにロードされたデータが可視になり、エラーが発生したデータベース・パーティションではロード操作が再開されます。

注: iAccessLevel が SQLU_ALLOW_READ_ACCESS に設定されている場合や、コピー・ターゲットが指定されている場合、このモードは使用できません。

DB2LOAD_SETUP_AND_LOAD_ERRS
このモードでは、セットアップまたはデータのロード時に生じるデータベース・パーティション・レベルのエラーによって、影響を受けたデータベース・パーティション上でのみ、処理が停止します。DB2LOAD_LOAD_ERRS_ONLY モードと同様、データ・ロード中にデータベース・パーティション・エラーが生じた場合、すべてのデータベース・パーティション上のデータは、ロードの再開操作が実行されるまで不可視のままです。

注: iAccessLevel が SQLU_ALLOW_READ_ACCESS に設定されている場合や、コピー・ターゲットが指定されている場合、このモードは使用できません。

DB2LOAD_NO_ISOLATION
ロード操作時にエラーが生じると、トランザクションは打ち切られます。パラメーターが NULL の場合、iAccessLevel が SQLU_ALLOW_READ_ACCESS に設定されない限り、またはコピー・ターゲットが指定されない限り、デフォルトは DB2LOAD_LOAD_ERRS_ONLY になります。設定または指定されている場合、デフォルトは DB2LOAD_NO_ISOLATION です。

piStatusInterval
入力。進行メッセージを生成する前に、ロードするデータの MB 数を指定します。有効な値は、1 から 4000 の範囲の整数です。 NULL が指定される場合、デフォルト値の 100 が使用されます。

piPortRange
入力。内部通信用の TCP ポート範囲。 NULL の場合、使用されるポート範囲は 6000 から 6063 です。

piCheckTruncation
入力。ロードで入出力時にレコードの切り捨てをチェックします。有効な値は TRUE および FALSE です。 NULL の場合、デフォルトは FALSE です。

piMapFileInput
入力。分散マップの入力ファイル名。モードが ANALYZE ではない場合、このパラメーターは NULL に設定する必要があります。モードが ANALYZE の場合、このパラメーターは指定する必要があります。

piMapFileOutput
入力。分散マップの出力ファイル名。 piMapFileInput に対する規則は、ここでも同じく適用されます。

piTrace
入力。すべてのデータ変換プロセスのダンプ、およびハッシュ値の出力を検討する必要がある場合、トレースするレコードの数を指定します。 NULL の場合、レコード数のデフォルトは 0 です。

piNewline
入力。 RECLEN ファイル・タイプ修飾子も指定されている場合、ロードで ASC データ・レコードの終端で改行文字をチェックするように強制します。指定可能な値は TRUE および FALSE です。 NULL の場合、値のデフォルトは FALSE です。

piDistfile
入力。データベース・パーティション分散ファイル名。 NULL が指定された場合、値はデフォルトの ″DISTFILE″ になります。

piOmitHeader
入力。 DB2LOAD_PARTITION_ONLY モードを使用する場合に、分散マップのヘッダーをデータベース・パーティション・ファイルに組み込まないことを示します。指定可能な値は TRUE および FALSE です。 NULL の場合、デフォルトは FALSE です。

piRunStatDBPartNum
統計を収集するデータベース・パーティションを指定します。デフォルト値は、出力データベース・パーティション・リスト内の最初のデータベース・パーティションです。

db2LoadNodeList データ構造パラメーター

piNodeList
入力。ノード番号の配列。

iNumNodes
入力。 piNodeList 配列内のノードの数。 0 がデフォルトで、これはターゲット表が定義されているすべてのノードです。

db2LoadPortRange データ構造パラメーター

iPortMin
入力。ポート番号の下限。

iPortMax
入力。ポート番号の上限。

db2PartLoadOut データ構造パラメーター

oRowsRdPartAgents
出力。すべてのパーティション・エージェントによって読み取られる行の総数。

oRowsRejPartAgents
出力。すべてのパーティション・エージェントによってリジェクトされる行の総数。

oRowsPartitioned
出力。すべてのパーティション・エージェントによってパーティション分割される行の総数。

poAgentInfoList
出力。パーティション・データベースへのロード操作時には、ロード・エージェント、パーティション・エージェント、事前パーティション・エージェント、ファイル転送コマンド・エージェント、およびファイルへのロード・エージェントなどのロード処理エンティティーが関係してくる可能性があります (これらについては、「Data Movement Guide」で説明されています)。poAgentInfoList 出力パラメーターの目的は、呼び出し側に、ロード操作に関係した各ロード・エージェントに関する情報を戻すことです。リスト内の各項目には、以下の情報が含まれます。

oAgentType
項目が記述するロード・エージェントの種類を示すタグ。

oNodeNum
エージェントが実行されたデータベース・パーティションの数。

oSqlcode
エージェントの処理の結果の最終 sqlcode。

oTableState
エージェントが実行されるデータベース・パーティション上の表の最終状況 (ロード・エージェントに関係するもののみ)。

API を呼び出す前に、このリストにメモリーを割り振るのは、API の呼び出し側の責任です。呼び出し側は、iMaxAgentInfoEntries パラメーターにメモリーを割り振った項目の数も示す必要があります。呼び出し側が poAgentInfoList を NULL に設定する場合、または iMaxAgentInfoEntries を 0 に設定する場合、ロード・エージェントに関する情報は戻されません。

iMaxAgentInfoEntries
入力。 poAgentInfoList 用にユーザーが割り振ったエージェント情報の項目の最大数。一般に、このパラメーターは、ロード操作に関係したデータベース・パーティション数の 3 倍の数に設定すれば十分です。

oNumAgentInfoEntries

出力。ロード操作によって生成されたエージェント情報の項目の実際の数。iMaxAgentInfoEntries が oNumAgentInfoEntries の値以上である場合に限り、この項目数は poAgentInfoList パラメーターでユーザーに戻されます。iMaxAgentInfoEntries が oNumAgentInfoEntries より小さい場合、poAgentInfoList に戻される項目数は iMaxAgentInfoEntries と等しくなります。

db2LoadAgentInfo データ構造パラメーター

oSqlcode

出力。エージェントの処理の結果の最終 sqlcode。

oTableState

出力。この出力パラメーターの目的は、ロード操作後に、表のいかなる状態も報告しないことです。その目的は、ロード処理中に表に何が起きたかについての一般情報を呼び出し側に提供するために、発生し得る表の状況の、小さなサブセットだけを報告することです。この値は、ロード・エージェントにのみ関係があります。可能な値は以下のとおりです。

DB2LOADQUERY_NORMAL

ロードがデータベース・パーティションで正常に完了し、表が LOAD IN PROGRESS (または LOAD PENDING) 状態ではなくなったことを示します。この場合、制約事項をさらに処理する必要があるために、表が引き続き SET INTEGRITY PENDING 状態であることがありますが、これは正常な状態なので報告はされません。

DB2LOADQUERY_UNCHANGED

エラーが原因でロード・ジョブが処理を打ち切ったが、db2Load を呼び出す前の状態がどのようなものであっても、データベース・パーティション上の表の状態はまだ変更されていないことを示します。ロードの再始動、またはそのようなデータベース・パーティション上での操作の終了を実行する必要はありません。

DB2LOADQUERY_LOADPENDING

処理中にロード・ジョブが打ち切られたが、データベース・パーティション上の表は LOAD PENDING 状態のままであることを示します。これは、そのデータベース・パーティションでのロード・ジョブを、終了または再始動する必要があることを意味しています。

oNodeNum

出力。エージェントが実行されたデータベース・パーティションの数。

oAgentType

出力。エージェント・タイプ。有効な値は以下のとおりです (インクルード・ディレクトリーの db2ApiDf ヘッダー・ファイルで定義される)。

- DB2LOAD_LOAD_AGENT
- DB2LOAD_PARTITIONING_AGENT
- DB2LOAD_PRE_PARTITIONING_AGENT
- DB2LOAD_FILE_TRANSFER_AGENT
- DB2LOAD_LOAD_TO_FILE_AGENT

db2gLoadStruct データ構造固有パラメーター

iFileTypeLen
入力。 iFileType パラメーターの長さ (バイト単位) を指定します。

iLocalMsgFileLen
入力。 iLocalMsgFileName パラメーターの長さ (バイト単位) を指定します。

iTempFilesPathLen
入力。 iTempFilesPath パラメーターの長さ (バイト単位) を指定します。

piXmlPathList
入力。 media_type フィールドを SQLU_LOCAL_MEDIA に設定された sqlu_media_list、および xml ファイルが置かれているクライアント上のパスをリストするその sqlu_media_entry 構造を指すポインター。

db2gLoadIn データ構造固有パラメーター

iUseTablespaceLen
入力。 piUseTablespace パラメーターの長さ (バイト単位)。

piXmlParse
入力。 XML 文書に対して行われる必要のある解析のタイプ。 include ディレクトリーに入っている db2ApiDf ヘッダー・ファイル内の有効値は、次のとおりです。

DB2DMU_XMLPARSE_PRESERVE_WS
空白が保持されます。

DB2DMU_XMLPARSE_STRIP_WS
空白が取り除かれます。

piXmlValidate
入力。 db2DMUXmlValidate 構造を指すポインター。 XML 文書の XML スキーマ検証を行う必要があることを示します。

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                       iUsing;       /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2gPartLoadIn データ構造固有パラメーター

piReserved1
将来の使用のために予約されています。

iHostnameLen
入力。 piHostname パラメーターの長さ (バイト単位)。

iFileTransferLen
入力。 piFileTransferCmd パラメーターの長さ (バイト単位)。

iPartFileLocLen
入力。 piPartFileLocation パラメーターの長さ (バイト単位)。

iMapFileInputLen
入力。 piMapFileInput パラメーターの長さ (バイト単位)。

iMapFileOutputLen
入力。 piMapFileOutput パラメーターの長さ (バイト単位)。

iDistfileLen
入力。 piDistfile パラメーターの長さ (バイト単位)。

使用上の注意

データは、入力ファイル内に並んでいる順序でロードされます。特定の順序を希望する場合には、ロードが試行される前にデータをソートしてください。

ロード・ユーティリティーは、既存の定義に基づいて索引を作成します。ユニーク・キーの重複を処理するのに、例外表が使用されます。ユーティリティーは、参照保全を強制したり、制約検査を実行したり、ロードする表に従属するサマリー表を更新したりすることはありません。参照制約またはチェック制約を含む表は、SET INTEGRITY ペンディング状態になります。 REFRESH IMMEDIATE として定義されているサマリー表、およびロードする表に依存するサマリー表もまた、SET INTEGRITY ペンディング状態になります。表の SET INTEGRITY ペンディング状態を解除するには、SET INTEGRITY ステートメントを発行してください。ロード操作は、複製されたサマリー表では実行できません。

クラスタリング索引の場合、ロードする前に、データをクラスタリング索引でソートする必要があります。多次元クラスターされた (MDC) 表にロードする場合は、データをソートする必要はありません。

ロード・セッション - CLP の例

例 1

TABLE1 には以下に示す 5 つの列があります。

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 には以下に示す 6 つのエレメントがあります。

- ELE1、位置 01 から 20
- ELE2、位置 21 から 22
- ELE3、位置 23 から 23
- ELE4、位置 24 から 27
- ELE5、位置 28 から 31
- ELE6、位置 32 から 32
- ELE7、位置 33 から 40

データ・レコード:

```
1...5...10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    XXN
Test data 4,5 and 6 WWN6789    Y
```

以下に示すコマンドで、ファイルから表をロードします。

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
   method L (1 20, 21 22, 24 27, 28 31)
   null indicators (0,0,23,32)
   insert into table1 (col1, col5, col2, col3)
```

注:

1. MODIFIED BY パラメーターで striptblanks を指定すると、 VARCHAR 列の中のブランクが切り捨てられるようになります (例えば、行 1、2、および 3 の長さがそれぞれ 11、17、および 19 バイトである COL1)。
2. MODIFIED BY パラメーターに reclen=40 を指定することにより、各入力レコードの末尾には改行文字がなく、各レコードの長さが 40 バイトであることを示しています。最後の 8 バイトは、表のロードには使用されません。
3. COL4 は入力ファイルにはないので、そのデフォルト値 (NOT NULL WITH DEFAULT と定義されている) を使用して TABLE1 に挿入されます。
4. 位置 23 および 32 は、TABLE1 の COL2 と COL3 に NULL をロードするかどうかを行ごとに示すために使用されます。特定のレコードのうち列の NULL 標識位置が Y なら、その列は NULL になります。それが N の場合は、入力レコードのその列のデータ位置 (L(.....) で定義) にあるデータ値が、その行の列データのソースとして使用されます。この例では、行 1 のどの列も NULL ではなく、行 2 の COL2 は NULL であり、行 3 の COL3 は NULL です。
5. この例では、COL1 と COL5 の NULL INDICATORS は 0 (ゼロ) として指定されますが、それはそのデータを NULL 不可能であることを示しています。
6. 特定の列に対する NULL INDICATOR は入力レコードのどの位置でも可能ですが、その位置は必ず指定しなければならず、 Y または N のいずれかの値が提供される必要があります。

例 2 (ダンプ・ファイルの使用)

表 FRIENDS は以下のように定義されています。

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

この表に対して、以下に示すデータ・レコードのロードを試みたとします。

```
23, 24, bobby
, 45, john
4,, mary
```

第 2 行は最初の INT が NULL であり、列定義では NOT NULL が指定されているためにリジェクトされます。 DEL フォーマットと互換でない開始文字の入った列は、エラーを生成し、レコードはリジェクトされます。そのようなレコードはダンプ・ファイルに書き込まれます。

区切り文字の外側の列の中にある DEL データは無視されますが、警告が生成されます。以下に例を示します。

```
22,34,"bob"
24,55,"sam" sdf
```

ユーティリティーはこの表の第 3 列にある ″sam″ をロードし、文字 ″sdf″ には警告のフラグが付けられます。このレコードはリジェクトされません。別の例として、

```
22 3, 34,"bob"
```

ユーティリティーは 22,34,"bob" をロードし、列 1 の 22 の後にある一部のデータが無視されたという警告を生成します。このレコードはリジェクトされません。

例 3 (ID 列がある表のロード)

TABLE1 には、以下の 4 つの列があります。

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 は、C2 が GENERATED ALWAYS ID 列であることを除き、 TABLE1 と同じです。

DATAFILE1 内のデータ・レコード (DEL フォーマット):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2 内のデータ・レコード (DEL フォーマット):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

注:

1. 以下のコマンドでは、行 1 および 2 の ID 値が生成されます。これは、DATAFILE1 内にこれらの行の ID 値が存在しないためです。ただし、行 3 および 4 には、ユーザー提供の ID 値である 100 および 101 がそれぞれ割り当てられます。

   ```
   db2 load from datafile1.del of del replace into table1
   ```

2. DATAFILE1 を TABLE1 にロードして、すべての行の ID 値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

   ```
   db2 load from datafile1.del of del method P(1, 3, 4)
      replace into table1 (c1, c3, c4)
   db2load from datafile1.del of del modified by identityignore
      replace into table1
   ```

3. DATAFILE2 を TABLE1 にロードして、それぞれの行ごとに ID 値が生成されるようにするには、以下のコマンドのいずれかを発行してください。

   ```
   db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
   db2 load from datafile2.del of del modified by identitymissing
      replace into table1
   ```

4. DATAFILE1 を TABLE2 にロードして、 ID 値である 100 および 101 が行 3 および 4 にそれぞれ割り当てられるようにするには、以下のコマンドを発行してください。

```
db2 load from datafile1.del of del modified by identityoverride
   replace into table2
```

この場合、行 1 および 2 はリジェクトされます。これは、ユーティリティーへの指示により、ユーザー提供の値を優先し、システムが生成した ID 値をオーバーライドするようになっているためです。ユーザー提供の値が存在しない場合でも、 ID 列が暗黙的に非 NULL であるため、この行はリジェクトする必要があります。

5. 識別に関係するファイル・タイプ修飾子を使用せずに DATAFILE1 を TABLE2 にロードすると、行 1 と 2 はロードされますが、行 3 と 4 はリジェクトされます。これは、行 3 と 4 では独自に非 NULL 値が提供されており、 ID 列が GENERATED ALWAYS であるためです。

例 4 (CURSOR からのロード)

MY.TABLE1 には次の 3 つの列があります。

- ONE INT
- TWO CHAR(10)
- THREE DATE

MY.TABLE2 には次の 3 つの列があります。

- ONE INT
- TWO CHAR(10)
- THREE DATE

カーソル MYCURSOR は以下のように定義されます。

```
declare mycursor cursor for select * from my.table1
```

次のコマンドは、すべてのデータを MY.TABLE1 から MY.TABLE2 にロードします。

```
load from mycursor of cursor method P(1,2,3) insert into
  my.table2(one,two,three)
```

注:

1. 単一の LOAD コマンドには、カーソル名を 1 つだけ指定できます。つまり、`load from mycurs1, mycurs2 of cursor...` は許可されません。
2. カーソルからのロードに有効な METHOD 値は、`P` および `N` だけです。
3. この例では、METHOD P および挿入列リスト `(one,two,three)` はデフォルト値を示すため、これらを省略してもかまいません。
4. MY.TABLE1 は、表、ビュー、別名、またはニックネームが使用できます。

SET INTEGRITY

SET INTEGRITY ステートメントは、以下の目的で使用されます。

- 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もあります。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もあります。

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。

呼び出し方法

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。つまり、以下のようになります。
      - 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。

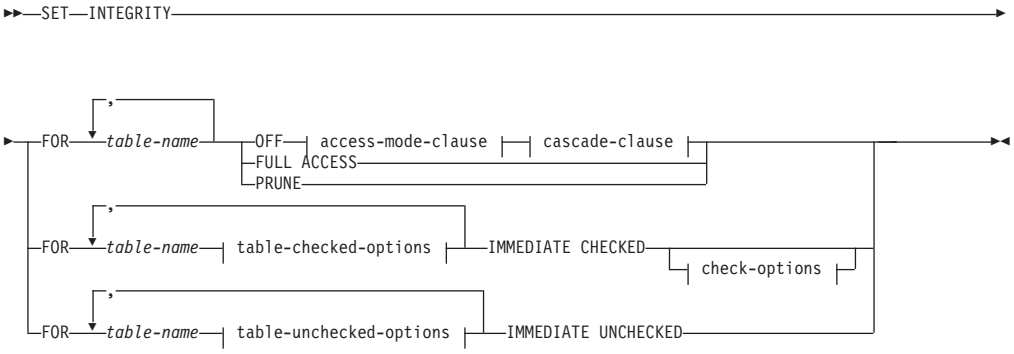
  ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかが含まれていなければなりません。
  - 以下に対する CONTROL 特権
    - 指定された表
    - ステートメントによって SET INTEGRITY ペンディング状態にされる下層外部キー表
    - ステートメントによって SET INTEGRITY ペンディング状態にされる下層即時マテリアライズ照会表
    - ステートメントによって SET INTEGRITY ペンディング状態にされる下層即時ステージング表
  - LOAD 権限
  - SYSADM または DBADM 権限
- 表をフル・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
- 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。

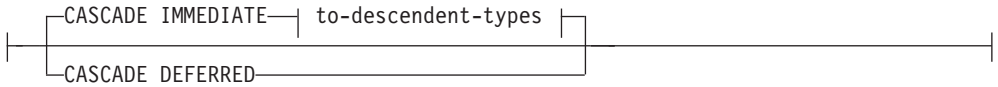
– 整理する表に対する CONTROL 特権
– SYSADM または DBADM 権限

構文



access-mode-clause:

cascade-clause:



to-descendent-types:

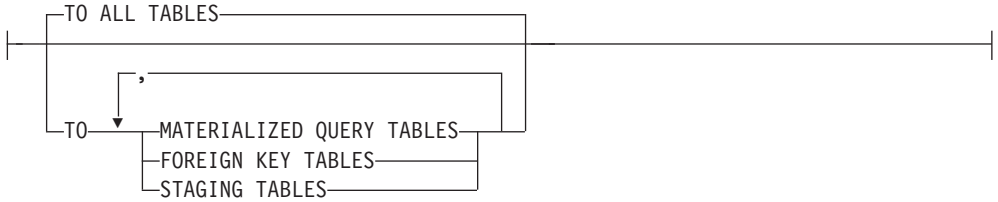
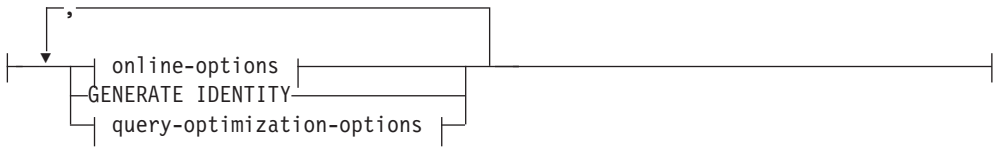


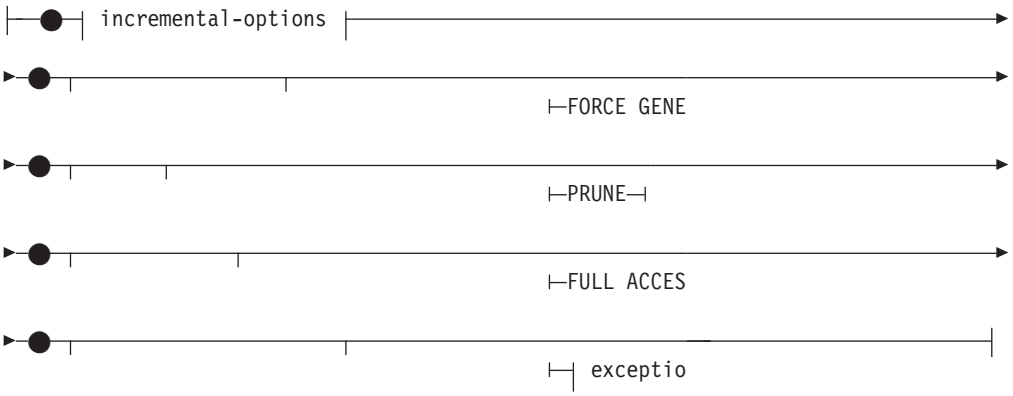
table-checked-options:



online-options:

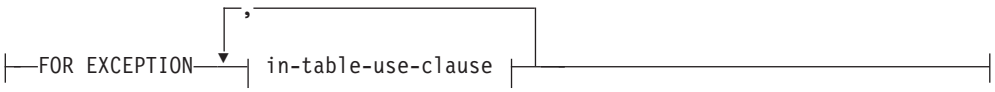
query-optimization-options:

check-options:



incremental-options:

exception-clause:



in-table-use-clause:

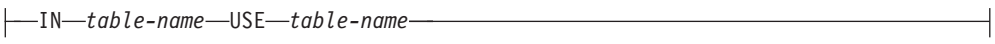
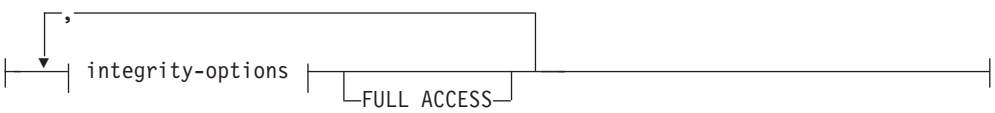
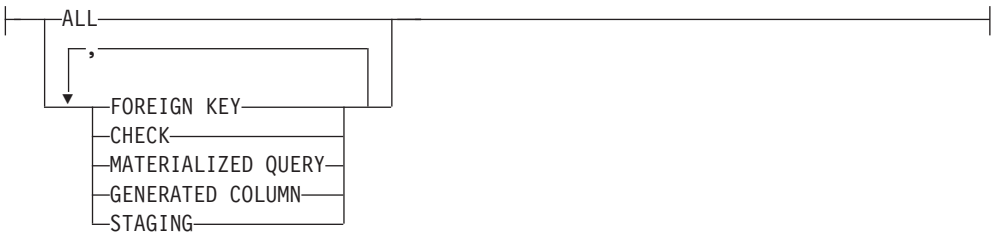


table-unchecked-options:



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保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
必要な保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。つまり、以下のようになります。

- 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す。

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、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にバイパスする、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 ユーティリティーを実行できません。
- EXPORT ユーティリティーは、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 ユーティリティーは、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ーは、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れば、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)。

- 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保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ー・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ー・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ー・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保全性、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保全性、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 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
```

LOAD QUERY

処理中にロード操作の状況を調べ、表の状態を戻します。ロードが行われていない場合は、表の状態だけが戻されます。このコマンドを正常に呼び出すためには、同じデータベースへの接続と、別の CLP セッションも必要になります。このコマンドは、ローカル・ユーザーでもリモート・ユーザーでも使用できます。

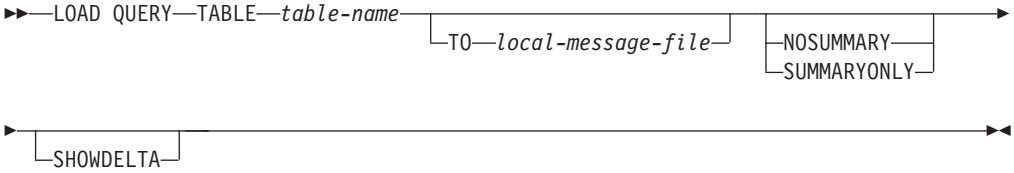
許可

なし

必要な接続

データベース

コマンド構文



コマンド・パラメーター

NOSUMMARY
ロード・サマリー情報 (読み取られた行、スキップされた行、ロードされた行、リジェクトされた行、削除された行、コミットされた行、警告の数) のレポートを生成しないよう指定します。

SHOWDELTA
新しい情報 (LOAD QUERY コマンドの最後の呼び出し以後に発生したロード・イベントに関する) だけをレポートするよう指定します。

SUMMARYONLY
ロード・サマリー情報のレポートだけを生成するよう指定します。

TABLE *table-name*
データが現在ロード中の表の名前を指定します。非修飾の表名を指定すると、その表は CURRENT SCHEMA で修飾されます。

TO *local-message-file*
ロード操作中に生じ得る警告およびエラー・メッセージの宛先を指定します。このファイルは、LOAD コマンド用に指定された *message-file* であってはなりません。ファイルが既に存在する場合、ロード・ユーティリティーが生成するメッセージはすべてそのファイルに追加されます。

例

大量のデータを BILLYBOB データベース内の STAFF 表にロードしている場合、ロード操作の状況をチェックすることが必要になるかもしれません。ユーザーは次のように指定することができます。

```
db2 connect to billybob
db2 load query table staff to /u/mydir/staff.tempmsg
```

出力ファイル /u/mydir/staff.tempmsg は、次のようになります。

```
SQL3501W  The table space(s) in which the table resides will not be placed in
backup pending state since forward recovery is disabled for the database.

SQL3109N  The utility is beginning to load data from file
"/u/mydir/data/staffbig.del"
```

```
SQL3500W  The utility is beginning the "LOAD" phase at time "03-21-2002
11:31:16.597045".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = "104416".

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = "205757".

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = "307098".

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count = "408439".

SQL3520W  Load Consistency Point was successful.

SQL3532I  The Load utility is currently in the "LOAD" phase.


Number of rows read         = 453376
Number of rows skipped      = 0
Number of rows loaded       = 453376
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 408439
Number of warnings          = 0

Tablestate:
  ロード進行中
```

使用上の注意

ロード・ユーティリティーは、ロックに加えて、表状態を使用して、表へのアクセスを制御します。 LOAD QUERY コマンドを使用して、表の状態を判別することができます。また、現在ロードされていない表に対しても、LOAD QUERY を使用することができます。パーティション表の場合、報告される状態は、対応する可視のデータ・パーティションの状態のうち、最も限定的なものです。例えば、ある 1 つのデータ・パーティションが Read Access Only 状態にあり、他のすべてのデータ・パーティションは Normal 状態にある場合、ロード照会操作からは Read Access Only 状態が戻されます。ロード操作によって、データ・パーティションのサブセットが、表の残りとは異なる状態のままになることはありません。 LOAD QUERY で記述される表の状態は次のとおりです。

正常
表が通常以外の状態 (異常な状態) にない場合はこの状態になります。通常状態は、表が作成された後に置かれる最初の状態です。

SET INTEGRITY ペンディング
表には、未確認の制約があります。表の SET INTEGRITY ペンディング状態を解除するには、SET INTEGRITY ステートメントを使用してください。ロード・ユーティリティーは、制約のある表でロード操作を開始する際に、表を整合性の設定ペンディング (Set Integrity Pending) 状態にします。

ロード進行中
これは、一時的な状態であり、ロード操作時のみ有効です。ロード操作が失敗または中断した場合に表をロード中状態から解除する方法については、『*関連リンク*』のセクションにある、ロード操作後のペンディング状態に関するセクションを参照してください。表スペースのロード中状態も参照してください。

ロード・ペンディング
この表ではロード操作がアクティブでしたが、データがコミットできるようになる前に打ち切られました。表をこの状態から解除するには、LOAD TERMINATE、LOAD RESTART、または LOAD REPLACE コマンドを発行してください。

読み取りアクセスのみ
ALLOW READ ACCESS オプションを指定した場合、表はロード操作時にこの状態になります。「読み取りアクセスのみ」は一時的な状態です。この状態になっていると、他のアプリケーションやユーティリティーは、ロード操作より前に存在していたデータに読み取りアクセスすることができます。

REORG ペンディング
REORG コマンドの推奨対象となる ALTER TABLE ステートメントが、表に対して実行されました。この表をもう一度アクセス可能にするには、まずクラシック REORG を実行する必要があります。

使用不可
表は使用できません。表のドロップまたはバックアップからのリストアのどちらかしか行えません。リカバリー不能のロード操作からロールフォワードを実行すると、表は使用できない状態になります。

ロード再始動不可
表は部分的にロードされた状態になっているので、ロードの再始動操作は行えません。さらにこの表はロード・ペンディング状態にもなります。
LOAD TERMINATE または LOAD REPLACE コマンドを使用すると、表はロード再始動不可状態から解除されます。表がロード再始動不可状態になるのは、正常に再始動も終了もしないで失敗に終わったロード操作の後でロールフォワード操作を実行した場合か、または表がロード進行中状態またはロード・ペンディング状態であった間にとられたオンライン・バックアップからリストア操作が実行された場合です。どちらの場合も、ロード再始動操作に必要な情報は信頼性に欠けるため、ロード再始動不可状態では、ロードの再始動操作はできません。

不明 LOAD QUERY コマンドは、表の状態を判別できません。

IBM DB2 データベース製品によってサポートされる表または表スペースの状態は、現在少なくとも 25 種類あります。それらの状態は、特定の環境下でのデータへのアクセスを制御したり、必要に応じて特定のユーザー・アクションを引き出して、データベースの整合性を保護するために使用されます。そのほとんどは、DB2 ユーティリティーのいずれか (例えばロード・ユーティリティーやバックアップおよびリストア・ユーティリティー) の操作に関連したイベントの結果として発生します。

ロード操作の前に従属表スペースの静止状態 (静止とは永続的なロックのこと) は解除されますが、表スペースのロード中状態のときは、ロード操作時に従属表のバックアップを行うことができません。表スペースのロード中状態は、表のロード中状態と異なります。すべてのロード操作で表のロード中状態が使用されますが、COPY NO オプションを指定した (リカバリー可能なデータベースに対する) ロード操作でも表スペースのロード中状態が使用されます。

次の表では、サポートされているそれぞれの表の状態について説明しています。また、実施例も示し、データベースの管理中に遭遇する可能性のある状態の解釈および対応の仕方を正確に示しています。これらの例は、AIX で実行されたコマンド・スクリプトから取られたものです。コピーして貼り付け、実行することができます。 UNIX 以外のシステムで DB2 製品を実行している場合は、ご使用のシステムで正しい形式のパス名となるようにしてください。例のほとんどは、DB2 データベース製品に付属する SAMPLE データベースの表に基づいています。 SAMPLE データベースの一部ではないシナリオを必要とする例も少数ながら存在しますが、開始点としては SAMPLE データベースへの接続を使用することができます。

表 47. サポートされる表の状態

状態	例

ロード・ペンディング	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。ロード操作のターゲット表 (NEWSTAFF という名前の新しい表) を含む小さな表スペースを作成します。
	`connect to sample;`
	`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256);`
	`create table newstaff like staff in ts1;`
	`load from staffdata.del of del insert into newstaff;`
	`load query table newstaff;`
	`load from staffdata.del of del terminate into newstaff;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY コマンドによって戻される情報は、NEWSTAFF 表がロード・ペンディング状態にあることを示します。ロード終了操作の後、表は通常状態になります。

ロード進行中	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。
	`connect to sample;`
	`create table newstaff like staff;`
	`load from staffdata.del of del insert into newstaff;`
	
	ロード操作の実行中に、別のセッションから次のスクリプトを実行します。
	`connect to sample;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY コマンドによって戻される情報は、NEWSTAFF 表がロード中状態にあることを示します。

正常	`connect to sample;`
	`create table newstaff like staff;`
	`load query table newstaff;`
	
	LOAD QUERY コマンドによって戻される情報は、NEWSTAFF 表が通常状態にあることを示します。

表 47. サポートされる表の状態 (続き)

状態	例

ロード再始動不可	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。
	`update db cfg for sample using logretain recovery;`
	`backup db sample;`
	`connect to sample;`
	`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256);`
	`create table newstaff like staff in ts1;`
	`connect reset;`
	`backup db sample;`
	
	このバックアップ・イメージのタイム・スタンプは 20040629205935 です。
	`connect to sample;`
	`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
	`connect reset;`
	`restore db sample taken at 20040629205935;`
	`rollforward db sample to end of logs and stop;`
	`connect to sample;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY コマンドによって戻される情報は、NEWSTAFF 表がロード再始動不可およびロード・ペンディング状態にあることを示します。
	`connect to sample;`
	`load from staffdata.del of del terminate into newstaff copy yes to /home/melnyk/backups;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY コマンドによって戻される情報は、NEWSTAFF 表が通常状態になったことを示します。

読み取りアクセスのみ	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。
	`connect to sample;`
	`export to st_data.del of del select * from staff;`
	`create table newstaff like staff;`
	`import from st_data.del of del insert into newstaff;`
	`load from staffdata.del of del insert into newstaff allow read access;`
	
	ロード操作の実行中に、別のセッションから次のスクリプトを実行します。
	`connect to sample;`
	`load query table newstaff;`
	`select * from newstaff;`
	`connect reset;`
	
	LOAD QUERY コマンドによって戻される情報は、NEWSTAFF 表が「読み取りアクセスのみ」および「ロード中」状態にあることを示します。照会は、STAFF 表のエクスポートされた内容のみを戻します。これは、ロード操作の前に NEWSTAFF 表に存在していたデータです。

表 47. サポートされる表の状態 (続き)

状態	例

SET INTEGRITY ペンディング	次の内容を持つロード入力ファイル staff_data.del があるとします。
	
	11,″Melnyk″,20,″Sales″,10,70000,15000:
	
	`connect to sample;`
	`alter table staff add constraint max_salary check (100000 - salary > 0);`
	`load from staff_data.del of del insert into staff;`
	`load query table staff;`
	
	LOAD QUERY コマンドによって戻される情報は、STAFF 表が SET INTEGRITY ペンディング状態にあることを示します。

使用不可	次の内容を持つロード入力ファイル staff_data.del があるとします。
	
	11,″Melnyk″,20,″Sales″,10,70000,15000:
	
	`update db cfg for sample using logretain recovery;`
	`backup db sample;`
	
	このバックアップ・イメージのタイム・スタンプは 20040629182012 です。
	
	`connect to sample;`
	`load from staff_data.del of del insert into staff nonrecoverable;`
	`connect reset;`
	`restore db sample taken at 20040629182012;`
	`rollforward db sample to end of logs and stop;`
	`connect to sample;`
	`load query table staff;`
	`connect reset;`
	
	LOAD QUERY コマンドによって戻される情報は、STAFF 表が使用不可状態にあることを示します。

表の状態の追加情報については、『*関連リンク*』のセクションを参照してください。

ロード操作の進行状況は、LISTUTILITIES コマンドを使ってモニターすることもできます。

LIST TABLESPACES

現行データベースの表スペースとその情報のリストを表示します。

このコマンドによって表示される情報は、表スペースのスナップショットでも使用できます。

有効範囲

このコマンドは、それが実行されたデータベース・パーティションに関する情報だけを戻します。

許可

以下のいずれか。

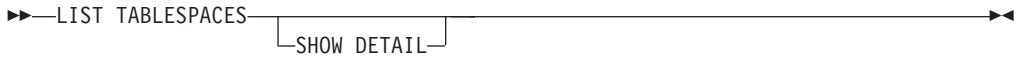
- *sysadm*

- *sysctrl*
- *sysmaint*
- *dbadm*
- LOAD 権限

必要な接続

データベース

コマンド構文



コマンド・パラメーター

SHOW DETAIL

このオプションを指定しない場合、各表スペースごとに以下の基本情報だけが表示されます。

- 表スペース ID
- 名前
- タイプ (システム管理スペースまたはデータベース管理スペース)
- 内容 (すべてのデータ、長形式または索引データ、または一時データ)
- 状態。現在の表スペースの状態を示す 16 進値。外部から見ることができる表スペースの状態は、特定の状態値が 16 進数の合計値で構成されています。 例えば、状態が ″quiesced: EXCLUSIVE″ かつ ″Load pending″ の場合、その値は 0x0004 + 0x0008、つまり 0x000c となります。 db2tbst (表スペース状態の獲得) コマンドを使うと、特定の 16 進値と関連した表スペース状態を取得できます。以下は、sqlutil.h に示されているビット定義です。

```
0x0          Normal
0x1          Quiesced: SHARE
0x2          Quiesced: UPDATE
0x4          Quiesced: EXCLUSIVE
0x8          Load pending
0x10         Delete pending
0x20         Backup pending
0x40         Roll forward in progress
0x80         Roll forward pending
0x100        Restore pending
0x100        Recovery pending (not used)
0x200        Disable pending
0x400        Reorg in progress
0x800        Backup in progress
0x1000       Storage must be defined
0x2000       Restore in progress
0x4000       Offline and not accessible
0x8000       Drop pending
0x20000      Load in progress
0x2000000    Storage may be defined
0x4000000    StorDef is in 'final' state
0x8000000    StorDef was change prior to roll forward
```

```
0x10000000   DMS rebalance in progress
0x20000000   Table space deletion in progress
0x40000000   Table space creation in progress
```

このオプションを指定した場合は、各表スペースに関して下記の付加的な情報が表示されます。

- ページの合計数
- 使用できるページの数
- 使用されたページの数
- 未使用ページの数
- 最高水準点 (ページ単位)
- ページ・サイズ (バイト単位)
- エクステント・サイズ (ページ単位)
- プリフェッチ・サイズ (ページ単位)
- コンテナーの数
- 最小リカバリー時間 (0 以外の場合のみ表示)
- 状態変更表スペース ID (表スペース状態が ″load pending″ または ″delete pending″ の場合のみ表示)
- 状態変更オブジェクト ID (表スペース状態が ″load pending″ または ″delete pending″ の場合のみ表示)
- 静止者の数 (表スペース状態が ″quiesced: SHARE″、″quiesced: UPDATE″、または ″quiesced: EXCLUSIVE″ の場合のみ表示)
- 各静止プログラムごとに表スペース ID とオブジェクト ID (静止プログラムの数が 0 より大きい場合のみ表示)

例

下記に示すのは、LIST TABLESPACES SHOW DETAIL の 2 つの出力例です。

```
          Tablespaces for Current Database
Tablespace ID                        = 0
Name                                 = SYSCATSPACE
Type                                 = Database managed space
Contents                             = Any data
State                                = 0x0000
  Detailed explanation:
    正常 Total pages                          = 895
Useable pages                        = 895
Used pages                           = 895
Free pages                           = Not applicable
High water mark (pages)              = Not applicable
Page size (bytes)                    = 4096
Extent size (pages)                  = 32
Prefetch size (pages)                = 32
Number of containers                 = 1

Tablespace ID                        = 1
Name                                 = TEMPSPACE1
Type                                 = System managed space
Contents                             = Temporary data
State                                = 0x0000
  Detailed explanation:
    正常 Total pages                          = 1
Useable pages                        = 1
```

```
 Used pages                          = 1
 Free pages                          = Not applicable
 High water mark (pages)             = Not applicable
 Page size (bytes)                   = 4096
 Extent size (pages)                 = 32
 Prefetch size (pages)               = 32
 Number of containers                = 1

 Tablespace ID                       = 2
 Name                                = USERSPACE1
 Type                                = Database managed space
 Contents                            = Any data
 State                               = 0x000c
   Detailed explanation:
     静止モードでの排他
     ロード・ペンディング
 Total pages                         = 337
 Useable pages                       = 337
 Used pages                          = 337
 Free pages                          = Not applicable
 High water mark (pages)             = Not applicable
 Page size (bytes)                   = 4096
 Extent size (pages)                 = 32
 Prefetch size (pages)               = 32
 Number of containers                = 1
 State change tablespace ID          = 2
 State change object ID              = 3
 Number of quiescers                 = 1
   Quiescer 1:
     Tablespace ID                   = 2
     Object ID                       = 3
DB21011I  In a partitioned database server environment, only the table spaces
on the current node are listed.


           Tablespaces for Current Database
 Tablespace ID                       = 0
 Name                                = SYSCATSPACE
 Type                                = System managed space
 Contents                            = Any data
 State                               = 0x0000
   Detailed explanation:
     正常
 Total pages                         = 1200
 Useable pages                       = 1200
 Used pages                          = 1200
 Free pages                          = Not applicable
 High water mark (pages)             = Not applicable
 Page size (bytes)                   = 4096
 Extent size (pages)                 = 32
 Prefetch size (pages)               = 32
 Number of containers                = 1

 Tablespace ID                       = 1
 Name                                = TEMPSPACE1
 Type                                = System managed space
 Contents                            = Temporary data
 State                               = 0x0000
   Detailed explanation:
     正常
 Total pages                         = 1
 Useable pages                       = 1
 Used pages                          = 1
 Free pages                          = Not applicable
 High water mark (pages)             = Not applicable
 Page size (bytes)                   = 4096
 Extent size (pages)                 = 32
 Prefetch size (pages)               = 32
```

```
 Number of containers                 = 1

Tablespace ID                         = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     正常
 Total pages                          = 1
 Useable pages                        = 1
 Used pages                           = 1
 Free pages                           = Not applicable
 High water mark (pages)              = Not applicable
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 1

Tablespace ID                         = 3
 Name                                 = DMS8K
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     正常
 Total pages                          = 2000
 Useable pages                        = 1952
 Used pages                           = 96
 Free pages                           = 1856
 High water mark (pages)              = 96
 Page size (bytes)                    = 8192
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 2

Tablespace ID                         = 4
 Name                                 = TEMP8K
 Type                                 = System managed space
 Contents                             = Temporary data
 State                                = 0x0000
   Detailed explanation:
     正常
 Total pages                          = 1
 Useable pages                        = 1
 Used pages                           = 1
 Free pages                           = Not applicable
 High water mark (pages)              = Not applicable
 Page size (bytes)                    = 8192
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 1
DB21011I  In a partitioned database server environment, only the table spaces
on the current node are listed.
```

使用上の注意

パーティション・データベース環境では、このコマンドがデータベースのすべての表スペースを戻すわけではありません。すべての表スペースのリストを表示するには、SYSCAT.TABLESPACES を照会します。

表スペースのバランス調整操作中には、使用できるページ数には新しく追加されたコンテナーのページ数が含まれますが、バランス調整完了までの間、それらの新し

いページは、未使用ページ数に反映されません。表スペースのバランス調整が実行されていない場合、使用されたページの数と未使用ページの数を合計すると、使用できるページ数の値に等しくなります。

IBM DB2 データベース製品によってサポートされる表または表スペースの状態は、現在少なくとも 25 種類あります。それらの状態は、特定の環境下でのデータへのアクセスを制御したり、必要に応じて特定のユーザー・アクションを引き出して、データベースの整合性を保護するために使用されます。そのほとんどは、DB2 ユーティリティーのいずれか (例えばロード・ユーティリティーやバックアップおよびリストア・ユーティリティー) の操作に関連したイベントの結果として発生します。

次の表では、サポートされているそれぞれの表スペースの状態について説明しています。また、実施例も示し、データベースの管理中に遭遇する可能性のある状態の解釈および対応の仕方を正確に示しています。これらの例は、AIX で実行されたコマンド・スクリプトから取られたものです。コピーして貼り付け、実行することができます。 UNIX 以外のシステムで DB2 製品を実行している場合は、ご使用のシステムで正しい形式のパス名となるようにしてください。例のほとんどは、DB2 データベース製品に付属する SAMPLE データベースの表に基づいています。
SAMPLE データベースの一部ではないシナリオを必要とする例も少数ながら存在しますが、開始点としては SAMPLE データベースへの接続を使用することができます。

表 48. サポートされる表スペースの状態

状態	16 進数の状態値	説明	例

バックアップ・ペンディング	0x20	ポイント・イン・タイム表スペースのロールフォワード操作の後、または COPY NO オプションを指定した (リカバリー可能データベースに対する) ロード操作の後、表スペースはこの状態になります。表スペースを使用するには、その前に表スペース (またはデータベース全体) をバックアップしておく必要があります。表スペースをバックアップしないと、その表スペース内の表を照会することはできますが、更新することができません。
		注: データベースは、ロールフォワード・リカバリーが有効になった直後にもバックアップする必要があります。データベースのリカバリーは、**logretain** データベース構成パラメーターを RECOVERY、または **userexit** データベース構成パラメーターを YES に設定した場合に行えます。そのデータベースをバックアップするまでは、活動化または接続を行えません。バックアップが取られた時点で、**backup_pending** 情報データベース構成パラメーターの値が NO に設定されます。
			1. 次の内容を持つロード入力ファイル staff_data.del があるとします。
			
			11,″Melnyk″,20,″Sales″,10,70000,15000:
			
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`load from staff_data.del of del messages load.msg`
			`insert into staff copy no;`
			`update staff set salary = 69000 where id = 11;`
			
			2.
			
			`update db cfg for sample using logretain recovery;`
			`connect to sample;`

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

バックアップ進行中	0x800	これは、一時的な状態であり、バックアップ操作時のみ有効です。	オンライン BACKUP DATABASE コマンドを発行します。
			`backup db sample online;`
			バックアップ操作の実行中に、別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			または
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			USERSPACE1 について戻される情報は、この表スペースがバックアップ進行中状態にあることを示します。

DMS 再平衡化進行中	0x10000000	これは、一時的な状態であり、データの再平衡化操作時のみ有効です。データベース管理スペース (DMS) として定義されている表スペースに新しいコンテナーが追加されるか、または既存のコンテナーが拡張されたときに、表スペースのデータの再平衡化が発生することがあります。*再平衡化*とは、データのストライピングを維持する試みにおいて、表スペースのエクステントを 1 つの場所から別の場所に移動する処理のことです。エクステント とは、コンテナー・スペースの単位 (測定基準はページ数) で、ストライプとは、表スペースの、コンテナーのセット全体にわたる エクステントの層のことです。	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。
			`connect to sample;`
			`create tablespace ts1 managed by database using`
			`(file '/home/melnyk/melnyk/NODE0000/SQL00001`
			`/ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`load from staffdata.del of del insert into newstaff`
			`nonrecoverable;`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk`
			`/NODE0000/SQL00001/ts1c2' 1024);`
			`list tablespaces;`
			`connect reset;`
			TS1 について戻される情報は、この表スペースが DMS 再平衡化進行中状態にあることを示します。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例
使用不可ペンディング	0x200	データベースのロールフォワード操作時に表スペースはこの状態になる場合があります。ロールフォワード操作の終了までには表スペースのこの状態は解除されます。この状態は、表スペースがオフラインになり、トランザクションの補正ログ・レコードが作成されなかった結果として生じる条件によってトリガーされます。この表スペース状態になり、その後解除される過程は、ユーザーから認識されません。	この表スペース状態を示す例は、本書では扱われていません。
ドロップ・ペンディング	0x8000	データベースの再始動操作時に、表スペースの 1 つ以上のコンテナーで問題が見つかった場合に表スペースはこの状態になります (データベースの再始動は、このデータベースを用いた前のセッションが (たとえば電源障害時に) 異常終了した場合に行う必要があります)。表スペースがドロップ・ペンディング状態にある場合は使用することができず、ドロップ以外に何も行えません。	この表スペース状態を示す例は、本書では扱われていません。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

ロード進行中	0x20000	これは一時的な状態で、COPY NO オプションを指定した (リカバリー可能データベースに対する) ロード操作時のみ有効です。表のロード中状態も参照してください。	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create table newstaff like staff;`
			`load from staffdata.del of del insert into newstaff copy no;`
			`connect reset;`
			
			ロード操作の実行中に、別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			`list tablespaces;`
			`connect reset;`
			
			USERSPACE1 について戻される情報は、この表スペースがロード中 (およびバックアップ・ペンディング) 状態にあることを示します。

正常	0x0	表スペースが通常以外の状態 (異常な状態) にない場合はこの状態になります。通常状態は、表スペースが作成された後に置かれる最初の状態です。	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`list tablespaces show detail;`

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

オフラインのためアクセス不可	0x4000	表スペースの 1 つ以上のコンテナーに問題があった場合に表スペースはこの状態になります。間違ってコンテナーの名前を変更したり、移動したり、損傷してしまうこともあり得ます。問題を正し、表スペースに関連付けられているコンテナーに再びアクセスできるようになった後、すべてのアプリケーションをデータベースから切断し、データベースに再接続することによって、この異常な状態から抜け出すことができます。別の方法として、SWITCH ONLINE 節を指定して ALTER TABLESPACE ステートメントを発行することによって、他のアプリケーションをデータベースから切断せずに表スペースのオフラインおよびアクセス不可状態を解除することができます。	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk /NODE0000/SQL00001/tsc2' 1024);`
			`export to st_data.del of del select * from staff;`
			`create table stafftemp like staff in ts1;`
			`import from st_data.del of del insert into stafftemp;`
			`connect reset;`
			
			表スペースのコンテナーの名前を tsc1 から tsc3 に変更し、STAFFTEMP 表の照会を試行します。
			`connect to sample;`
			`select * from stafftemp;`
			
			照会は SQL0290N (表スペース・アクセスが許されていません) を戻し、LIST TABLESPACES コマンドによって TS1 の状態値 0x4000 (オフラインおよびアクセス不可) が戻されます。表スペースのコンテナーの名前を tsc3 から tsc1 に戻します。これで照会は正常に実行されます。

静止モードでの排他	0x4	表スペースの静止機能を呼び出すアプリケーションに表スペースへの排他 (読み取りまたは書き込み) アクセスがあると、表スペースはこの状態になります。QUIESCE TABLESPACES FOR TABLE コマンドを発行することにより、表スペースを明示的に静止モードでの排他状態にすることができます。	表スペースの状態を静止モードでの排他に設定する前に通常にします。
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff exclusive;`
			`connect reset;`
			
			別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			`select * from staff where id=60;`
			`update staff set salary=50000 where id=60;`
			`list tablespaces;`
			`connect reset;`
			
			USERSPACE1 について戻される情報は、この表スペースが静止モードでの排他状態にあることを示します。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

静止モードでの共有	0x1	表スペースの静止機能を呼び出すアプリケーションと同時アプリケーションの両方に表スペースへの読み取りアクセスがあると (書き込みアクセスはない)、表スペースはこの状態になります。 QUIESCE TABLESPACES FOR TABLE コマンドを発行することにより、表スペースを明示的に静止モードでの共有状態にすることができます。	表スペースの状態を静止モードでの共有に設定する前に通常にします。
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff share;`
			`connect reset;`
			別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			`select * from staff where id=40;`
			`update staff set salary=50000 where id=40;`
			`list tablespaces;`
			`connect reset;`
			USERSPACE1 について戻される情報は、この表スペースが静止モードでの共有状態にあることを示します。

静止モードでの更新	0x2	表スペースの静止機能を呼び出すアプリケーションに表スペースへの排他書き込みアクセスがあると、表スペースはこの状態になります。 QUIESCE TABLESPACES FOR TABLE コマンドを発行することにより、表スペースを明示的に静止モードでの更新状態にすることができます。	表スペースの状態を静止モードでの更新に設定する前に通常にします。
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff intent to update;`
			`connect reset;`
			別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			`select * from staff where id=50;`
			`update staff set salary=50000 where id=50;`
			`list tablespaces;`
			`connect reset;`
			USERSPACE1 について戻される情報は、この表スペースが静止モードでの更新状態にあることを示します。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

REORG 進行中	0x400	これは、一時的な状態であり、REORG 操作時のみ有効です。	REORG TABLE コマンドを発行します。
			`connect to sample;`
			`reorg table staff;`
			`connect reset;`
			
			REORG 操作の実行中に、別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			
			1.
			`list tablespaces show detail;`
			
			または
			
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			USERSPACE1 について戻される情報は、この表スペースが REORG 進行中状態にあることを示します。
			注: SAMPLE データベースが関係する表の再編成操作は短時間で完了する可能性があり、結果として、このアプローチを用いて REORG 進行中状態を観察するのが難しくなる場合があります。

リストア・ペンディング	0x100	データベースの表スペースは、リダイレクト・リストア操作の最初の部分の後 (つまり、SET TABLESPACE CONTAINERS コマンドを発行する前)、この状態になります。表スペースを使用するには、その前に表スペース (またはデータベース全体) をリストアする必要があります。リストア操作が正常に完了するまでは、データベースへの接続を行えません。リストア操作が完了した時点で、**restore_pending** 情報データベース構成パラメーターの値が NO に設定されます。	「ストレージを定義可能」状態にあるリダイレクト・リストア操作の最初の部分が完了すると、すべての表スペースがリストア・ペンディング状態になります。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

リストア進行中	0x2000	これは、一時的な状態であり、リストア操作時のみ有効です。	`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`backup db sample tablespace (userspace1);`
			
			このバックアップ・イメージのタイム・スタンプは、次のようになります。
			
			20040611174124
			
			`restore db sample tablespace (userspace1) online taken at 20040611174124;`
			
			リストア操作の実行中に、別のセッションから次のスクリプトを実行します。
			
			`connect to sample;`
			
			1.
			
			`list tablespaces show detail;`
			
			または
			
			2.
			
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			USERSPACE1 について戻される情報は、この表スペースがリストア進行中状態にあることを示します。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例
ロールフォワード・ペンディング	0x80	リカバリー可能データベースに対するリストア操作の後、表スペースはこの状態になります。表スペースを使用するには、その前に表スペース (またはデータベース全体) をロールフォワードする必要があります。データベースのリカバリーは、**logretain** データベース構成パラメーターを RECOVERY、または **userexit** データベース構成パラメーターを YES に設定した場合に行えます。ロールフォワード操作が正常に完了するまでは、データベースの活動化または接続を行えません。リストア操作が完了した時点で、**rollfwd_pending** 情報データベース構成パラメーターの値が NO に設定されます。	リストア進行中のオンライン表スペースのリストア操作が完了すると、表スペース USERSPACE1 はロールフォワード・ペンディング状態になります。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

ロールフォワード進行中	0x40	これは、一時的な状態であり、ロールフォワード操作時のみ有効です。	大量のデータ (例えば 20000 以上のレコード) を持つロード入力ファイル staffdata.del があるとします。
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`connect reset;`
			`backup db sample tablespace (ts1) online;`
			このバックアップ・イメージのタイム・スタンプは、次のようになります。
			20040630000715
			`connect to sample;`
			`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
			`connect reset;`
			`restore db sample tablespace (ts1) online taken at 20040630000715;`
			`rollforward db sample to end of logs and stop tablespace (ts1) online;`
			ロールフォワード操作の実行中に、別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			または
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			TS1 について戻される情報は、この表スペースがロールフォワード進行中状態にあることを示します。

ストレージを定義可能	0x2000000	データベースの表スペースは、リダイレクト・リストア操作の最初の部分の後 (つまり、SET TABLESPACE CONTAINERS コマンドを発行する前)、この状態になります。これにより、必要に応じてコンテナーを再定義することができます。	`backup db sample;`
			このバックアップ・イメージのタイム・スタンプが 20040613204955 であるとします。
			`restore db sample taken at 20040613204955 redirect;`
			`list tablespaces;`
			LIST TABLESPACES コマンドによって戻される情報は、すべての表スペースが「ストレージを定義可能」および「リストア・ペンディング」状態にあることを示します。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

ストレージを定義する必要がある	0x1000	表スペース・コンテナーの設定フェーズが省略されるか、またはそのフェーズ中に指定のコンテナーを獲得できない場合、データベースの表スペースは新しいデータベースに対するリダイレクト・リストア操作時にこの状態になります。表スペース・コンテナーの設定フェーズ中に指定のコンテナーを獲得できないという状況は、例えば無効なパス名が指定されたり、十分なディスク・スペースがない場合に発生する可能性があります。	`backup db sample;`
			
			このバックアップ・イメージのタイム・スタンプが 20040613204955 であるとします。
			`restore db sample taken at 20040613204955 into mydb redirect;`
			`set tablespace containers for 2 using (path 'ts2c1');`
			`list tablespaces;`
			
			LIST TABLESPACES コマンドによって戻される情報は、表スペース SYSCATSPACE および表スペース TEMPSPACE1 が、「ストレージを定義する必要があります」、「ストレージを定義可能」、および「リストア・ペンディング」の状態にあることを示します。「ストレージを定義する必要があります」状態は、「ストレージを定義可能」状態よりも優先されます。

表スペース作成の進行中	0x40000000	これは、一時的な状態であり、表スペースの作成操作時のみ有効です。	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc3' 1024);`
			
			表スペースの作成操作の実行中に、別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			
			1.
			`list tablespaces show detail;`
			
			または
			
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			TS1、TS2、および TS3 について戻される情報は、これらの表スペースが表スペース作成の進行中状態にあることを示します。

表 48. サポートされる表スペースの状態 (続き)

状態	16 進数の状態値	説明	例

表スペース削除の進行中	0x20000000	これは、一時的な状態であり、表スペースの削除操作時のみ有効です。	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc3' 1024);`
			`drop tablespace ts1;`
			`drop tablespace ts2;`
			`drop tablespace ts3;`
			
			表スペースの削除操作の実行中に、別のセッションから次のスクリプトを実行します。
			`connect to sample;`
			
			1.
			`list tablespaces show detail;`
			
			または
			
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			TS1、TS2、および TS3 について戻される情報は、これらの表スペースが表スペース削除の進行中状態にあることを示します。

表スペースの状態の追加情報については、『*関連リンク*』のセクションを参照してください。

第 5 章 その他のデータ移動オプション

DB2 Connect によるデータの移動

ホスト・データベース・システムとワークステーションの間でデータを移動する必要のある複合環境では、DB2 Connect (ホストとワークステーションの間のデータ転送のゲートウェイ) を使用できます (図 17 を参照)。

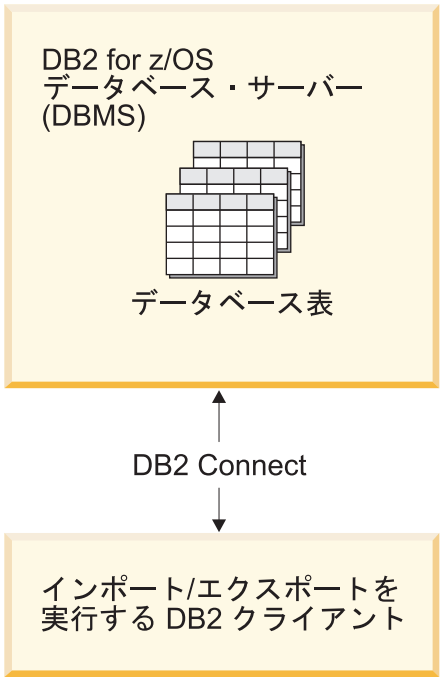


図 17. DB2 Connect によるインポート/エクスポート

DB2 のエクスポートおよびインポート・ユーティリティーを使用すると、ホストまたは System i サーバー・データベースから DB2 Connect ワークステーション上のファイルに、またはその逆にデータを移動できます。その後、このエクスポートおよびインポート・フォーマットをサポートしている他のすべてのアプリケーションやリレーショナル・データベース管理システムで、データを使用できます。例えば、ホストまたは System i サーバー・データベースから PC/IXF ファイルにデータをエクスポートして、さらにそれを DB2 for Windows データベースにインポートすることができます。

エクスポートおよびインポート操作は、データベース・クライアントから、または DB2 Connect ワークステーションから実行できます。

注:

1. エクスポートまたはインポートされるデータは、両方のデータベースに適用されるサイズとデータ・タイプの制約事項に従っていなければなりません。

2. インポートのパフォーマンスを改善するため、コンパウンド照会を使用することができます。インポート・ユーティリティーで compound ファイル・タイプ修飾子を指定することにより、指定した数の照会ステートメントをブロックにまとめてください。このようにすればネットワーク・オーバーヘッドが少なくなり、応答時間が改善されます。

DB2 Connect を使用する場合、エクスポートおよびインポートの操作は次の条件を満たしている必要があります。

- ファイル・タイプは PC/IXF でなければなりません。
- インポート開始前に、データと互換性のある属性のターゲット表がターゲット・サーバー上に作成されていなければなりません。ソース表の属性を取得するには、db2look ユーティリティーを使用できます。 DB2 Connect によるインポートでは、サポートされているオプションは INSERT だけなので、表は作成できません。

これらの条件のいずれかが満たされていない場合、操作は失敗し、エラー・メッセージが戻されます。

注: 索引定義はエクスポートにおいて保管されず、インポートにおいて使用されません。

混合データ (1 バイト・データと 2 バイト・データの両方の入った列) をエクスポートまたはインポートする場合は、以下のことを考慮してください。

- EBCDIC でデータを保管するシステム (MVS、OS/390、OS/400、VM、および VSE) では、シフトアウトおよびシフトイン文字がそれぞれ 2 バイト・データの開始と終了を表します。データベース表の列の長さを定義する場合は、これらの文字のための十分な余地を見込んでください。
- 列データのパターンが一貫しているのでない限り、文字タイプの可変長列を使用することをお勧めします。

ワークステーションからホスト・サーバーへのデータの移動

データをホストまたは System i サーバー・データベースに移動するには、以下を実行してください。

1. DB2 表から PC/IXF ファイルにデータをエクスポートします。
2. INSERT オプションを使って、PC/IXF ファイルをホスト・サーバー・データベース内の互換性のある表にインポートします。

ホスト・サーバー・データベースからワークステーションにデータを移動するには、次のようにします。

1. ホスト・サーバー・データベースの表から PC/IXF ファイルにデータをエクスポートします。
2. PC/IXF ファイルを DB2 表にインポートします。

例

以下の例では、ワークステーションからホストまたは System i サーバー・データベースにデータを移動する方法を示します。

次のコマンドを発行して、外部 IXF フォーマットにデータをエクスポートします。

```
db2 export to staff.ixf of ixf select * from userid.staff
```

次のコマンドを発行して、ターゲット DB2 データベースへの DRDA 接続を確立します。

```
db2 connect to cbc664 user admin using xxx
```

まだ存在していない場合には、次のようにしてターゲット DB2 データベース・インスタンスにターゲット表を作成します。

```
CREATE TABLE mydb.staff (ID SMALLINT NOT NULL, NAME VARCHAR(9),
       DEPT SMALLINT, JOB CHAR(5), YEARS SMALLINT, SALARY DECIMAL(7,2),
       COMM DECIMAL(7,2))
```

データをインポートするには、以下のコマンドを発行します。

```
db2 import from staff.ixf of ixf insert into mydb.staff
```

IXF フォーマットのファイルからデータの各行が読み取られ、表 mydb.staff に行を挿入するために、SQL INSERT ステートメントが発行されます。すべてのデータがターゲット表に挿入されるまで、単一行が引き続き挿入されます。

詳細については、「Moving Data Across the DB2 Family」という IBM Redbooks® を参照してください。この Redbooks は、 http://www.redbooks.ibm.com/redbooks/SG246905/ という URL にあります。

IBM レプリケーション・ツールのコンポーネント

IBM は、Q レプリケーションおよび SQL レプリケーションという 2 つの主要なレプリケーション・ソリューションを提供しています。

Q レプリケーションの主要コンポーネントは、Q キャプチャー・プログラムと Q アプライ・プログラムです。 SQL レプリケーションの主要コンポーネントは、キャプチャー・プログラムとアプライ・プログラムです。両方のタイプのレプリケーションがレプリケーション・アラート・モニター・ツールを共有します。これらのレプリケーション・コンポーネントのセットアップと管理は、レプリケーション・センターおよび ASNCLP コマンド行プログラムを使って行えます。

以下のリストは、これらのレプリケーション・コンポーネントを簡潔に要約しています。

Q キャプチャー・プログラム

DB2 リカバリー・ログを読み取って DB2 ソース表に対する変更を収集し、コミットされたソース・データを WebSphere® MQ メッセージに変換します。このメッセージは、サブスクライブ・アプリケーションへの XML 形式として発行することもでき、Qアプライ・プログラムへのコンパクト形式としてレプリケーションすることもできます。

Q アプライ・プログラム

キューから WebSphere MQ メッセージを取り、メッセージを SQL ステートメントに変換し、ターゲット表またはストアード・プロシージャーを更新します。サポー

トされるターゲットとしては、DB2 データベースまたはサブシステム、および Oracle、Sybase、Informix®、Microsoft® SQL Server のデータベース (フェデレーテッド・サーバーのニックネームを使ってアクセスする) があります。

キャプチャー・プログラム

DB2 リカバリー・ログを読み取って登録済みのソース表またはビューに対する変更を収集した後、コミットされたトランザクション・データを変更データ (CD) 表と呼ばれるリレーショナル表にステージングします。このデータはターゲット・システムでのコピー準備が整うまでこの表に保管されます。また、SQL レプリケーションは、キャプチャー・トリガーも提供します。これは、整合変更データ (CCD) 表と呼ばれるステージング表に、非 DB2 ソース表に対する変更のレコードを取り込むものです。

アプライ・プログラム

ステージング表からデータを読み取り、ターゲットに対して適切な変更を加えます。非 DB2 データ・ソースの場合、アプライ・プログラムはフェデレーテッド・データベース上の表のニックネームを使って CCD 表を読み取り、ターゲット表に対して適切な変更を加えます。

レプリケーション・アラート・モニター

Q キャプチャー、Q アプライ、キャプチャー、およびアプライ・プログラムの正常性をチェックするユーティリティー。このモニターはさまざまな状況をチェックします。例えば、プログラムの終了、警告またはエラー・メッセージの発行、指定された値のしきい値の到達、または特定のアクションの実行などの状況です。これらをチェックした後、E メール・サーバー、ページャー、または z/OS コンソールに通知を発行します。

レプリケーション・センターは、以下を行うために使用します。

- ソース表の登録、サブスクリプション、イベント発行、キュー・マップ、アラート条件、およびその他のオブジェクトの定義。
- レプリケーション・プログラムの開始、停止、中断、再開、および再初期化。
- 自動コピーの時間指定。
- データの SQL 拡張の指定。
- ソース表とターゲット表のリレーションシップの定義。

スキーマのコピー

db2move ユーティリティーおよび ADMIN_COPY_SCHEMA プロシージャーにより、データベース・スキーマのコピーを迅速に作成できます。モデル・スキーマを確立すると、新しいバージョンを作成するためのテンプレートとしてそれを使用できます。

同じデータベース内の単一のスキーマをコピーするには、ADMIN_COPY_SCHEMA プロシージャーを使用します。単一スキーマまたは複数のスキーマをソース・データベースからターゲット・データベースへコピーするには、`-co COPY` アクションを

指定して db2move ユーティリティーを使用します。ソース・スキーマからのほとんどのデータベース・オブジェクトは、新しいスキーマの下のターゲット・データベースにコピーされます。

トラブルシューティングのヒント

ADMIN_COPY_SCHEMA プロシージャーと db2move ユーティリティーはどちらも LOAD コマンドを呼び出します。ロードの処理中、データベース・ターゲット・オブジェクトのある表スペースはバックアップ・ペンディング状態になります。

ADMIN_COPY_SCHEMA プロシージャー

上記の注で説明したとおり、プロシージャーに COPYNO オプションを指定して使用すると、ターゲット・オブジェクトがある表スペースをバックアップ・ペンディング状態にします。表スペースの SET INTEGRITY ペンディング状態を解除するために、このプロシージャーは SET INTEGRITY ステートメントを発行します。ターゲット表オブジェクトに参照制約が定義されている状態では、ターゲット表も SET INTEGRITY ペンディング状態になります。表スペースはすでにバックアップ・ペンディング状態なので、ADMIN_COPY_SCHEMA プロシージャーが SET INTEGRITY ステートメントを発行する試みは失敗します。

この状態を解決するには、BACKUP DATABASE コマンドを実行して、影響を受ける表スペースのバックアップ・ペンディング状態を解除します。次いで、このプロシージャーによって生成されたエラー表の **Statement_text** 列を参照し、SET INTEGRITY ペンディング状態にある表のリストを見つけます。次に、リストされている各表に SET INTEGRITY ステートメントを発行し、各表の SET INTEGRITY ペンディング状態を解除します。

db2move ユーティリティー

このユーティリティーは、以下のタイプを除き、許容されるスキーマ・オブジェクトをすべてコピーしようとします。

- 表階層
- ステージング表 (複数パーティション・データベース環境でのロード・ユーティリティーではサポートされません)
- jars (Java™ ルーチン・アーカイブ)
- ニックネーム
- パッケージ
- ビュー階層
- オブジェクト特権 (新規オブジェクトはすべてデフォルト許可で作成されます。)
- 統計 (新規オブジェクトに統計情報は含まれません。)
- 索引拡張 (ユーザー定義構造化タイプに関連)
- ユーザー定義構造化タイプおよびそのトランスフォーム関数

サポートされないタイプに関するエラー

サポートされないタイプのいずれかのオブジェクトがソース・スキーマで検出されると、サポートされないオブジェクト・タイプが検出されたことを示すエントリーがエラー・ファイルに書き込まれます。 COPY 操作は引き続

き正常に行われます。ログに記録されたエントリーは、この操作ではコピーされないオブジェクトについて伝えるためのものです。

スキーマを伴わないオブジェクト
表スペースなどの、スキーマを伴わないオブジェクト、およびイベント・モニターは、スキーマのコピー操作時には操作されません。これらは、スキーマのコピー操作が呼び出される前に、ターゲット・データベース上で作成する必要があります。

複製された表
複製された表をコピーする場合、表の新規コピーはレプリケーションには使用できません。表は通常表として再作成されます。

異なるインスタンス
ソース・データベースは、ターゲット・データベースと同じインスタンス内にない場合は、カタログされる必要があります。

SCHEMA_MAP オプション
SCHEMA_MAP オプションを使用して、ターゲット・データベース上で異なるスキーマ名を指定する場合、元のスキーマ名を新しいスキーマ名に置き換えるために、スキーマのコピー操作はオブジェクト定義ステートメントの最小限の構文解析のみを実行します。例えば、SQL プロシージャーの内容の中に表示されるオリジナル・スキーマのインスタンスは新しいスキーマ名に置き換えられません。したがって、スキーマのコピー操作はそれらのオブジェクトを再作成できない場合があります。コピー操作の完了後に、エラー・ファイル内の DDL を使用して、それらの失敗したオブジェクトを手動で再作成することができます。

オブジェクト間の相互依存
スキーマのコピー操作は、それらのオブジェクト間の相互依存性を満たす順序でオブジェクトを再作成しようとします。例えば、表 T1 にユーザー定義関数 U1 を参照する列が含まれる場合、T1 を再作成する前に U1 を再作成します。ただし、プロシージャーの従属情報がカタログ内で前もって使用可能なわけではないため、プロシージャーを再作成する際に、スキーマのコピー操作では最初にすべてのプロシージャーを再作成し、それから失敗したものを再作成しようとします (それらのプロシージャーが前の試行時に正常に作成されたプロシージャーに依存していれば、それ以降の試行時には正常に再作成されると想定しています)。それ以降の試行時に 1 つ以上のプロシージャーを正常に再作成できる限り、この操作は失敗したプロシージャーを引き続き再作成しようとします。プロシージャーの再作成試行時に、エラー (および DDL) がエラー・ファイルに毎回記録されます。エラー・ファイルに同じプロシージャーの項目が多数表示されることがありますが、それらのプロシージャーはそれ以降の試行時に正常に再作成されている可能性がありますスキーマのコピー操作の完了時に SYSCAT.PROCEDURES 表を照会して、エラー・ファイルにリストされているそれらのプロシージャーが正常に再作成されたかどうか判断する必要があります。

詳細については、ADMIN_COPY_SCHEMA プロシージャーおよび db2move ユーティリティーを参照してください。

db2move ユーティリティーを使用したスキーマ・コピーの例

1 つまたは複数のスキーマをソース・データベースからターゲット・データベースにコピーするには、-co COPY アクションを指定して db2move ユーティリティーを使用します。モデル・スキーマを確立すると、新しいバージョンを作成するためのテンプレートとしてそれを使用できます。

例 1: -c COPY オプションの使用

-co COPY オプションを指定した以下の db2move の例は、スキーマ BAR をサンプル・データベースからターゲット・データベースにコピーして、それを FOO に名前変更します。

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u userid -p password
```

新規の (ターゲット) スキーマ・オブジェクトは、ソース・スキーマのオブジェクトと同じオブジェクト名を使用して作成されますが、修飾子はターゲット・スキーマのものが使用されます。表のコピーを作成する際には、ソース表のデータを共にコピーすることも、しないこともできます。ソース・データベースおよびターゲット・データベースは、別々のシステムに存在できます。

例 2: COPY 操作中に表スペース名のマッピングを指定する

以下の例は、db2move COPY 操作時にソース・システムからの表スペースの代わりに使用される、特定の表スペース名マッピングを指定する方法を示しています。 SYS_ANY キーワードを指定して、ターゲット表スペースを、デフォルトの表スペース選択アルゴリズムを使用して選択することを指示できます。この場合、db2move ユーティリティーは、ターゲットとして使用可能な任意の表スペースを選択します。

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u userid -p password
```

SYS_ANY キーワードは、すべての表スペースに使用できます。または、一部の表スペースに固有のマッピングを指定し、残りにはデフォルトの表スペース選択アルゴリズムを指定できます。

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
  ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
  -u userid -p password
```

これは、表スペース TS1 が TS2 にマップされ、TS3 が TS4 にマップされ、残りの表スペースがデフォルトの表スペース選択アルゴリズムを使用することを示しています。

例 3: COPY 操作後のオブジェクト所有者の変更

正常に COPY を実行した後に、ターゲット・スキーマで作成された新しい各オブジェクトの所有者を変更することができます。ターゲット・オブジェクトのデフォルト所有者は接続ユーザーです。以下のようなオプションを指定すると、新しい所有者に所有権が移転します。

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
  -u userid -p password
```

ターゲット・オブジェクトの新規所有者は jrichards です。

ソース・スキーマとターゲット・スキーマが別々のシステムにある場合、ターゲット・システム側で db2move ユーティリティーを呼び出す必要があります。あるデータベースから別のデータベースにスキーマをコピーする場合、このアクションを行うには、ソース・データベースからコピーされるスキーマ名のリスト (コンマで区切られたもの) およびターゲット・データベース名が必要です。

スキーマをコピーするには、OS のコマンド・プロンプトから db2move を次のように発行します。

```
db2move <dbname> COPY -co <COPY- options>
 -u <userid> -p <password>
```

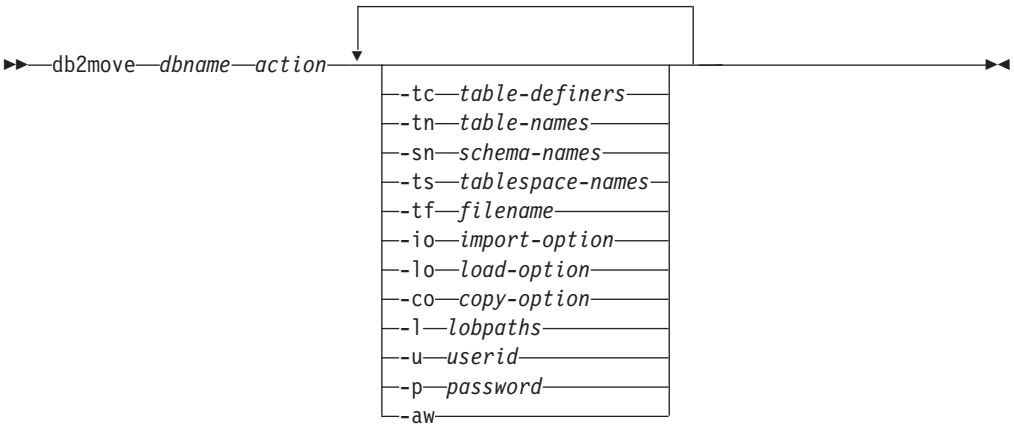
db2move - データベース移動ツール

このツールは、EXPORT/IMPORT/LOAD モードで使用された場合、ワークステーション上にある DB2 データベース間で、大量の表の移動を容易にします。また、特定のデータベースのシステム・カタログ表を照会し、すべてのユーザー表のリストをコンパイルします。そして、これらの表を PC/IXF フォーマットでエクスポートします。 PC/IXF ファイルは、同じシステム上の別のローカル DB2 データベースにインポートまたはロードするか、または別のワークステーション・プラットフォームに転送し、そのプラットフォームで DB2 データベースにインポートまたはロードすることができます。構造化タイプ列がある表は、このツールを使用しても移動しません。 COPY モードで使用すると、このツールによってスキーマの複写が容易になります。

許可

このツールは、ユーザーから要求されるアクションにしたがって、DB2 エクスポート、インポート、およびロード API を呼び出します。したがって、要求元ユーザー ID には、これらの API に求められる正しい権限がなければなりません。この権限がないと、要求は失敗します。

コマンド構文



コマンド・パラメーター

dbname
データベースの名前。

action 以下のうちの 1 つでなければなりません。

EXPORT
options のフィルター基準を満たすすべての表をエクスポートします。 options の指定がない場合には、すべての表をエクスポートします。内部ステージング情報は db2move.lst ファイルに保管されます。

IMPORT
内部ステージング・ファイル db2move.lst にリストされているすべての表をインポートします。 IMPORT の特定のアクションには、-io オプションを使用します。

LOAD 内部ステージング・ファイル db2move.lst にリストされているすべての表をロードします。 LOAD の特定のアクションには、-lo オプションを使用します。

COPY スキーマをターゲット・データベースへ複写します。 -sn オプションを使用して、1 つ以上のスキーマを指定します。 COPY 特定オプションの -co オプションを参照してください。-tn または -tf オプションを使用して、LOAD_ONLY モードの表をフィルターします。

下記は、各アクション中に生成されるファイルのリストです。

-tc *table-definers*
デフォルトはすべての定義者です。

これは EXPORT アクションのみです。指定されると、このオプションでリストされる定義者が作成する表のみがエクスポートされます。指定されない場合、デフォルトではすべての定義者を使用します。複数の定義者を指定する場合、それぞれをコンマで区切る必要があります。定義者 ID 間にブランクを入れることはできません。このオプションを -tn *table-names* オプションとともに使用すると、エクスポートする表を選択できます。

アスタリスク (*) は、ストリング中のどこにでも入れられるワイルドカード文字として使用できます。

-tn *table-names*
デフォルトはすべてのユーザー表です。

これは EXPORT または COPY アクションのみです。

EXPORT アクションで指定されると、指定されたストリング内の表と名前が一致する表だけがエクスポートされます。指定されない場合、デフォルトではすべてのユーザー表を使用します。複数の表名を指定する場合、それぞれをコンマで区切る必要があります。表名間にブランクを入れることはできません。スキーマをフィルター操作するには、修飾なしで表名をリストし、-sn オプションを使用する必要があります。

エクスポートの場合、アスタリスク (*) は、ストリング中のどこにでも入れられるワイルドカード文字として使用できます。

COPY アクションで指定する場合、これに加えて -co ″MODE″ LOAD_ONLY *copy-option* も指定する必要があります。指定された表のデータだけがターゲット・データベースで再挿入されます。表名は、スキーマ修飾子と共に ″schema″.″table″ の形式でリストします。

-sn *schema-names*

EXPORT のデフォルトは全スキーマです (COPY ではない)。

これが指定されると、一致するスキーマ名の表だけがエクスポートまたはコピーされます。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。複数のスキーマ名の間にブランクを使用することはできません。 8 文字より短いスキーマ名は、8 文字の長さになるまで埋め込まれます。

エクスポートの場合: スキーマ名の部分にアスタリスク・ワイルドカード文字 (*) が使用された場合は、それがパーセント記号 (%) に変更され、WHERE 節の LIKE 述部にパーセント付きの表名が使用されます。指定されない場合、デフォルトではすべてのスキーマを使用します。 -tn または -tc オプションと合わせて使用する場合、db2move は、スキーマが指定されたスキーマ名と一致し、定義者が指定された定義者と一致する表に対してのみ実行されます。 `fred` のようなスキーマ名の場合、アスタリスクを使用するときは、`-sn fr*d` ではなく `-sn fr*d*` のような指定が必要になります。

-ts *tablespace-names*

デフォルトはすべての表スペースです。

これは EXPORT アクションのみです。このオプションが指定されると、指定した表スペースにある表だけがエクスポートされます。表スペース名の部分にアスタリスク・ワイルドカード文字 (*) が使用された場合は、それがパーセント記号 (%) に変更され、WHERE 節の LIKE 述部にパーセント付きの表名が使用されます。 -ts オプションが指定されない場合、デフォルトではすべての表スペースを使用します。複数の表スペース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。複数の表スペース名の間にブランクを使用することはできません。 8 文字より短い表スペース名は、8 文字の長さになるまで埋め込まれます。例えば、`mytb` のような表スペース名の場合、アスタリスクを使用するときは、`-sn my*b` ではなく `-ts my*b*` のような指定が必要になります。

-tf *filename*

EXPORT アクションで指定されると、指定されたファイル内の名前と正確に一致する名前の表だけがエクスポートされます。指定されない場合、デフォルトではすべてのユーザー表を使用します。表は 1 行に 1 つずつリストする必要があり、各表は完全に修飾する必要があります。ストリング内には、ワイルドカード文字を使用できません。 以下は、ファイルの内容の例です。

```
"SCHEMA1"."TABLE NAME1"
"SCHEMA NAME77"."TABLE155"
```

COPY アクションで指定する場合、これに加えて -co ″MODE″ LOAD_ONLY *copy-option* も指定する必要があります。ファイル内に指定された表のデータだけが、ターゲット・データベースで再挿入されます。表名は、スキーマ修飾子と共に ″schema″.″table″ の形式でリストします。

-io *import-option*

デフォルトは REPLACE_CREATE です。インポート作成機能の制限については、『IMPORT コマンドの推奨されなくなったオプション CREATE および REPLACE_CREATE』を参照してください。

有効なオプションは、 INSERT、 INSERT_UPDATE、 REPLACE、CREATE、および REPLACE_CREATE です。

-lo *load-option*

デフォルトは INSERT です。

有効なオプションは、INSERT および REPLACE です。

-co db2move アクションがコピーである場合、以下の -co 追加オプションを使用できます。

「**TARGET_DB** *db name* **[USER** *userid* **USING** *password***]**」

ユーザーがターゲット・データベースの名前とユーザー/パスワードを指定できるようにします。(ソース・データベースのユーザー/パスワードは、既存の -p および -u オプションを使用して指定できます。)USER/USING 節はオプションです。 USER が userid を指定する場合は、USING 節の後にパスワードを指定します。パスワードが指定されない場合、db2move はパスワード情報を求めるプロンプトを出します。プロンプトが出されるのは、下記で説明するセキュリティーの理由によります。 TARGET_DB は COPY アクションには必須のオプションです。TARGET_DB は、ソース・データベースと同じにすることはできません。同じデータベース内のスキーマをコピーするには、ADMIN_COPY_SCHEMA プロシージャーを使用できます。COPY アクションには、少なくとも 1 つのスキーマ (-sn) または 1 つの表 (-tn または -tf) の入力が必要です。

複数の db2move コマンドを実行してスキーマを 1 つのデータベースから別のデータベースにコピーすると、デッドロックになります。1 度に 1 つのみの db2move コマンドを発行してください。コピー処理中にソース・スキーマ内の表を変更すると、ターゲット・スキーマのデータがコピー後に同一のものにならないことがあります。

「**MODE**」

DDL_AND_LOAD

ソース・スキーマの、すべてのサポートされるオブジェクトを作成し、ソース表データを表に追加します。これはデフォルト・オプションです。

DDL_ONLY

ソース・スキーマの、すべてのサポートされるオブジェクトを作成しますが、表にデータを再設定しません。

LOAD_ONLY

指定されたすべての表をソース・データベースからターゲット・データベースへロードします。表はターゲットに既に存在していなければなりません。LOAD_ONLY モードで

は、-tn または -tf オプションを使って少なくとも 1 つの表を入力する必要があります。

これは、COPY アクションでのみ使用される任意指定のオプションです。

「**SCHEMA_MAP**」

ターゲットへコピーするときにユーザーがスキーマをリネームできるようにします。ソース・ターゲット間のスキーマ・マッピングをコンマで区切り、大括弧で囲んだリストを提供します。例えば、schema_map ((s1, t1), (s2, t2)) のようになります。これは、スキーマ s1 からのオブジェクトはターゲットのスキーマ t1 にコピーされ、スキーマ s2 からのオブジェクトはターゲットのスキーマ t2 へコピーされることを意味します。ターゲット・スキーマ名がソース・スキーマ名であるのがデフォルトで、推奨されています。この理由は、db2move がオブジェクト本体内に修飾オブジェクトのあるスキーマを変更しないことにあります。したがって、異なるターゲット・スキーマ名を使用すると、オブジェクト本体内に修飾オブジェクトがある場合に問題が生じるおそれがあります。

以下に例を示します。`create view FOO.v1 as 'select c1 from FOO.t1'`

この場合、スキーマ FOO の BAR へのコピー、v1 は以下のように再生成されます。`create view BAR.v1 as 'select c1 from FOO.t1'`

これは、スキーマ FOO がターゲット・データベースに存在しないため失敗するか、または FOO が BAR と異なるために予期しない結果になります。ソースと同じスキーマ名を保つことにより、これらの問題を避けることができます。スキーマ間に相互従属関係がある場合、すべての相互に従属するスキーマがコピーされなければなりません。あるいは、相互従属関係のあるオブジェクトのコピーでエラーになります。

以下に例を示します。`create view FOO.v1 as 'select c1 from BAR.t1'`

この場合、v1 のコピーは BAR がコピーされない場合に失敗するか、または、ターゲットの BAR がソースからの BAR と異なる場合、予期しない結果になります。db2move はスキーマの相互従属関係を検出しようとはしません。

これは、COPY アクションでのみ使用される任意指定のオプションです。

「**NONRECOVERABLE**」

このオプションにより、ユーザーはロードのデフォルト動作をオーバーライドし、ロードが COPY-NO で行われるようにすることができます。デフォルトの動作では、ユーザーはロードされる各表スペースをバックアップするよう強制されます。この NONRECOVERABLE キーワードを指定すると、ユーザーは表スペースをバックアップするように即時に強制されることがありません。ただし、新しく作成された表が正しくリカバリーできるよう

に、できるだけ早くバックアップを取ることを強くお勧めします。これは、COPY アクションで使用できる任意指定のオプションです。

「**OWNER**」
正常にコピーした後、ターゲット・スキーマに作成された各新規オブジェクトの所有者をユーザーが変更できるようにします。ターゲット・オブジェクトのデフォルト所有者は接続ユーザーになりますが、このオプションが指定された場合、所有権は新規所有者に移されます。これは、COPY アクションで使用できる任意指定のオプションです。

「**TABLESPACE_MAP**」
ユーザーは、コピー中に使用する表スペース名のマッピングを、ソース・システムの表スペースの代わりに指定できます。これは、大括弧で囲まれた表スペース・マッピングが配列されたものです。例えば、`tablespace_map ((TS1, TS2),(TS3, TS4))` のようにします。これは、表スペース TS1 からのすべてのオブジェクトはターゲット・データベースの表スペース TS2 にコピーされ、表スペース TS3 からのオブジェクトはターゲットの表スペース TS4 へコピーされることを意味します。 `((T1, T2),(T2, T3))` の場合、ソース・データベースの T1 にあるすべてのオブジェクトはターゲット・データベースの T2 に再作成され、ソース・データベースの T2 にあるどのオブジェクトもターゲット・データベースの T3 に再作成されることになります。デフォルトでは、ソースの表スペース名と同じ表スペース名を使用しますが、その場合には、表スペースのマッピング入力は必要ありません。指定された表スペースが存在しない場合、その表スペースを使用したオブジェクトのコピーは失敗し、エラー・ファイルにログされます。

ユーザーには、`SYS_ANY` キーワードを使用して、ターゲット表スペースの選択にデフォルトの表スペース選択アルゴリズムの使用を指定するオプションもあります。この場合、db2move は使用できる表スペースをどれでもターゲットとしての使用に選択することができます。`SYS_ANY` キーワードはすべての表スペースに対して使用できます。例えば、`tablespace_map SYS_ANY` とします。さらに、ユーザーは特定のマッピングを表スペースのいくつかに指定し、残りにデフォルトの表スペース選択アルゴリズムを指定することもできます。例えば、`tablespace_map ((TS1, TS2),(TS3, TS4), SYS_ANY)` のようにします。これは、表スペース TS1 は TS2 に、TS3 は TS4 にマップされるが、残った表スペースはデフォルトの表スペース・ターゲットを使用することを意味します。「SYS」で始まる表スペースはあり得ないため、`SYS_ANY` キーワードが使用されます。

これは、COPY アクションで使用できる任意指定のオプションです。

-l *lobpaths*
IMPORT および EXPORT の場合、このオプションが指定されると、これは XML パスにも使用されます。デフォルトは現行ディレクトリーです。

このオプションは、LOB または XML ファイルが (EXPORT の一部として) 作成されるか、または (IMPORT または LOAD の一部として) 検索される絶対パス名を指定します。複数のパスを指定する場合、それぞれをコンマで区切る必要があります。パス間にブランクを入れることはできません。複数のパスが指定された場合、EXPORT はラウンドロビン方式でそれらを使用します。つまり、1 つの LOB 文書を最初のパスに書き込み、それから 2 番目のパスに、という順に最後まで書き込み、その後最初のパスに戻ります。 XML 文書でも同じです。最初のパスでファイルが見つからない場合 (IMPORT または LOAD 中)、 2 番目のパスが使用される、という方法でパスが使用されます。

-u *userid*

デフォルトはログオン・ユーザー ID です。

ユーザー ID とパスワードはどちらも任意指定です。しかし、一方を指定した場合、他方も必ず指定する必要があります。コマンドがリモート・サーバーに接続するクライアント上で実行される場合、ユーザー ID とパスワードを指定する必要があります。

-p *password*

デフォルトはログオン・パスワードです。ユーザー ID とパスワードはどちらも任意指定です。しかし、一方を指定した場合、他方も必ず指定する必要があります。 -p オプションが指定されてもパスワードが指定されていない場合、db2move はパスワードを求めるプロンプトを出します。これは、セキュリティーの理由によります。コマンド行にパスワードを入力するとセキュリティー問題が生じます。例えば、`ps -ef` コマンドがパスワードを表示します。しかし、db2move がスクリプトを通して呼び出される場合は、パスワードを供給する必要があります。コマンドがリモート・サーバーに接続するクライアント上で発行される場合、ユーザー ID とパスワードを指定する必要があります。

-aw 警告を許します。 -aw が指定されていない場合、エクスポート中に警告があった表は `db2move.lst` ファイルに組み込まれません (表の .ixf ファイルと .msg ファイルが生成されていても)。しかし、あるシナリオ (データ切り捨てなど) では、そのように警告があった表でも `db2move.lst` ファイルに組み込んでしまいたい場合があります。そのようなとき、このオプションを指定すると、エクスポート中に警告を受け取った表を `.lst` ファイルに組み込むことができます。

例

- SAMPLE データベースのすべての表をエクスポートするには (すべてのオプションにデフォルト値を使用)、以下を発行します。

  ```
  db2move sample export
  ```

- `userid1` または `us%rid2` のようなユーザー ID で作成され、`tbname1` という名前、または `%tbname2` のような表名を持つすべての表をエクスポートするには、以下を発行します。

  ```
  db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2
  ```

- SAMPLE データベースのすべての表をインポートするには、以下を発行します。(LOB パス `D:¥LOBPATH1` および `C:¥LOBPATH2` で、LOB ファイルが検索されます。この例は、Windows オペレーティング・システムにのみ該当します。)

```
db2move sample import -l D:¥LOBPATH1,C:¥LOBPATH2
```

- SAMPLE データベースのすべての表をロードするには、以下を発行します。(/home/userid/lobpath サブディレクトリーと tmp サブディレクトリーで、LOB ファイルが検索されます。この例は Linux および UNIX システムにのみ該当します。

```
db2move sample load -l /home/userid/lobpath,/tmp
```

- SAMPLE データベースのすべての表を、指定されたユーザー ID およびパスワードを使用して REPLACE モードでインポートするには、以下を発行します。

```
db2move sample import -io replace -u userid -p password
```

- スキーマ schema1 をソース・データベース dbsrc からターゲット・データベース dbtgt へ複写するには、以下を発行します。

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
```

- スキーマ schema1 をソース・データベース dbsrc からターゲット・データベース dbtgt へ複写し、そのターゲット上でスキーマを newschema1 に名前変更し、ソース表スペース ts1 をターゲットの ts2 へマップするには、以下を発行します。

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
     SCHEMA_MAP ((schema1,newschema1)) TABLESPACE_MAP ((ts1,ts2), SYS_ANY))
```

使用上の注意

- db2move EXPORT の後に db2move IMPORT/LOAD を指定すると、表データの移動が可能になます。表に関連した他のすべてのデータベース・オブジェクト (別名、ビュー、トリガーなど)、およびこれらの表が依存するオブジェクト (ユーザー定義タイプ、ユーザー定義関数など) を手動で移動する必要があります。
- CREATE または REPLACE_CREATE オプションを指定した IMPORT アクションを使ってターゲット・データベース上に表を作成する場合 (どちらのオプションも推奨されておらず、今後のリリースで除去される可能性があります)、『インポート済みの表の再作成』で説明されている制約が適用されます。
  REPLACE_CREATE オプションの使用時の db2move インポート・フェーズ中に想定外のエラーが生じた場合、該当する tabnnn.msg メッセージ・ファイルを調べて、表の作成に対する制限事項が原因でエラーが起きたかどうかを確かめてください。
- db2move を使用して、ID 列 GENERATED ALWAYS を含む表をインポートまたはロードすることはできません。ただし、手動でこれらの表をインポートまたはロードすることは可能です。詳しくは、『ID 列のロードに関する考慮事項』または『ID 列のインポートに関する考慮事項』を参照してください。
- エクスポート、インポート、またはロード API が db2move によって呼び出されると、 **FileTypeMod** パラメーターが lobsinfile に設定されます。つまり、LOB データが各表に対して、PC/IXF ファイルとは別のファイルに保持されます。
- LOAD コマンドは、データベースおよびデータ・ファイルが常駐するマシンでローカルに実行する必要があります。
- db2move LOAD を使用する場合、データベースの logretain が使用可能 (データベースがリカバリー可能) であれば、次のようになります。
  - NONRECOVERABLE オプションが指定されない場合、db2move はデフォルトの COPY NO オプションを使って db2Load API を呼び出します。ロードされた表が格納される表スペースは、ユーティリティー完了時にバックアップ・ペ

ンディング状態に置かれます (表スペースをバックアップ・ペンディング状態から解除するには、データベース全体または表スペース全体のバックアップが必要です)。

– NONRECOVERABLE オプションが指定されている場合、表スペースはバックアップ・ペンディング状態に置かれませんが、ロールフォワード・リカバリーが後で実行された場合、表はアクセス不能とマーク付けられるため、表をドロップする必要があります。ロード・リカバリー可能性オプションの詳細については、『ロードのパフォーマンスを改善するためのオプション』を参照してください。

- IMPORT または LOAD アクションを使用する db2move コマンドのパフォーマンスは、デフォルトのバッファー・プール IBMDEFAULTBP を変更し、構成パラメーター **sortheap**、**util_heap_sz**、**logfilsiz**、および **logprimary** を更新することによって、改善できます。

EXPORT 使用時に必要とされるファイル/生成されるファイル

- 入力: なし。
- 出力:

  EXPORT.out
  EXPORT アクションの結果の要約。

  db2move.lst
  オリジナル表名のリスト、その対応する PC/IXF ファイル名 (tabnnn.ixf)、およびメッセージ・ファイル名 (tabnnn.msg)。このリスト、エクスポートされた PC/IXF ファイル、および LOB ファイル (tabnnnc.yyy) は、 db2move IMPORT または LOAD アクションへの入力として使用されます。

  tabnnn.ixf
  特定の表の、エクスポートされる PC/IXF ファイル。

  tabnnn.msg
  対応する表のエクスポート・メッセージ・ファイル。

  tabnnnc.yyy
  特定の表の、エクスポートされる LOB ファイル。

  「nnn」は表番号です。「c」はアルファベットの文字です。「yyy」は 001 から 999 の範囲内の数値です。

  これらのファイルは、エクスポートされている表に LOB データが入っている場合のみ作成されます。作成されると、これらの LOB ファイルは「lobpath」ディレクトリーに入れられます。 LOB ファイルには、合計 26,000 の可能な名前があります。

  system.msg
  ファイルまたはディレクトリー・コマンドを作成または削除するための、システム・メッセージの入ったメッセージ・ファイル。これは、アクションが EXPORT で、LOB パスが指定される場合のみ使用されます。

IMPORT 使用時に必要とされるファイル/生成されるファイル

- 入力:

db2move.lst
EXPORT アクションからの出力ファイル。

tabnnn.ixf
EXPORT アクションからの出力ファイル。

tabnnnc.yyy
EXPORT アクションからの出力ファイル。

- 出力:

IMPORT.out
IMPORT アクションの結果の要約。

tabnnn.msg
対応する表のインポート・メッセージ・ファイル。

LOAD 使用時に必要とされるファイル/生成されるファイル

- 入力:

db2move.lst
EXPORT アクションからの出力ファイル。

tabnnn.ixf
EXPORT アクションからの出力ファイル。

tabnnnc.yyy
EXPORT アクションからの出力ファイル。

- 出力:

LOAD.out
LOAD アクションの結果の要約。

tabnnn.msg
対応する表の LOAD メッセージ・ファイル。

COPY 使用時に必要とされるファイル/生成されるファイル

- 入力: なし
- 出力:

COPYSCHEMA.msg
COPY 操作中に生成されたメッセージを含む出力ファイル。

COPYSCHEMA.err
COPY 操作中に発生した各エラーに関するエラー・メッセージが含まれる出力ファイル。これには、ターゲット・データベース上に再作成できなかった各オブジェクトに関する DDL ステートメントが含まれます。

LOADTABLE.msg
ロード・ユーティリティーのそれぞれの呼び出しによって生成されたメッセージが含まれる出力ファイル (ターゲット・データベースでのデータ再挿入に使用されます)。

LOADTABLE.err
ロード中に失敗した表の名前、またはターゲット・データベースにまだデ

ータ挿入する必要がある表の名前が含まれる出力ファイル。詳しくは『『スキーマのコピー操作が失敗した場合の再開方法』』のトピックを参照してください。

これらのファイルは、タイム・スタンプされ、1 つの実行から生成されたすべてのファイルには同一のタイム・スタンプが付きます。

自動生成スクリプトを使用したリダイレクト・リストアの実行

リダイレクト・リストア操作を実行するときは、バックアップ・イメージに保管される物理コンテナーの場所を指定し、変更される各表スペースのすべてのコンテナーを提供する必要があります。以下の手順を使用して、既存のバックアップ・イメージに基づいてリダイレクト・リストア・スクリプトを生成し、生成されたスクリプトを変更し、その後そのスクリプトを実行してリダイレクト・リストアを実行します。

リダイレクト・リストアを実行できるのは、事前に DB2 バックアップ・ユーティリティーを使って、データベースのバックアップをとってある場合に限ります。

- データベースが存在する場合、スクリプトを生成するには、データベースに接続できなければなりません。従って、データベースでマイグレーションまたはクラッシュ・リカバリーが必要な場合は、リダイレクトした復元スクリプトを生成する前に、これらの操作を行う必要があります。
- パーティション・データベース環境で作業しており、ターゲット・データベースが存在しない場合、リダイレクトされた復元スクリプトをすべてのデータベース・パーティションで同時に生成するコマンドを実行することはできません。その代わり、カタログ・パーティションからはじめて、一度に 1 つのデータベース・パーティションで、リダイレクトされた復元スクリプトを生成するコマンドを実行する必要があります。

  別の方法として、ターゲット・データベースと同じ名前を持つダミーのデータベースを最初に作成することもできます。ダミーのデータベースを作成した後、すべてのデータベース・パーティションに、リダイレクトされた復元スクリプトを同時に生成することができます。
- スクリプト生成のための RESTORE コマンドの発行時に REPLACE EXISTING オプションを指定したとしても、REPLACE EXISTING オプションはスクリプトでコメント化されます。
- セキュリティー上の理由により、パスワードは、生成されたスクリプトに現れません。パスワードは手動で入力する必要があります。
- コントロール・センターの「リストア・ウィザード」を使用してリダイレクト・リストアのスクリプトを生成することはできません。

スクリプトを使用してリダイレクト・リストアを実行するには、以下のようにします。

1. リストア・ユーティリティーを使用してリダイレクト・リストア・スクリプトを生成する。リストア・ユーティリティーは、コマンド行プロセッサー (CLP)、または db2Restore アプリケーション・プログラミング・インターフェース (API) を通して起動できます。以下は、REDIRECT オプションと GENERATE SCRIPT オプションを指定した RESTORE DATABASE コマンドの例です。

```
db2 restore db test from /home/jseifert/backups taken at 20050304090733
      redirect generate script test_node0000.clp
```

これはクライアント上に test_node0000.clp というリダイレクト・リストア・スクリプトを作成します。

2. 必要な変更を行うために、テキスト・エディターでリダイレクト・リストア・スクリプトをオープンする。変更できるのは、次のとおりです。
   - リストア・オプション
   - 自動ストレージ・パス
   - コンテナー・レイアウトおよびパス
3. 変更されたリダイレクト・リストア・スクリプトを実行します。以下に例を示します。

   ```
   db2 -tvf test_node0000.clp
   ```

RESTORE DATABASE

RESTORE DATABASE コマンドは、DB2 バックアップ・ユーティリティーを使用してバックアップされた、損傷のある、または破壊されたデータベースを再作成します。リストアされたデータベースは、バックアップ・コピーが行われた時と同じ状態になります。また、このユーティリティーを使って、別のイメージをデータベースに上書きしたり、バックアップ・コピーを新しいデータベースにリストアすることもできます。

異なるさまざまなオペレーティング・システムおよびハードウェア・プラットフォームの間で DB2 データベース・システムによってサポートされるリストア操作の詳細は、 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『異なるオペレーティング・システムおよびハードウェア・プラットフォーム間のバックアップおよびリストア操作』を参照してください。

さらに、リストア・ユーティリティーは、DB2 Universal Database バージョン 8 で作成されたバックアップ・イメージをリストアするのにも使用できます。移行が必要な場合、これはリストア操作の終了時に自動的に起動されます。

バックアップ操作の時点でデータベースのロールフォワード・リカバリーが有効になっていた場合は、リストア操作が正常に完了した後に、ロールフォワード・ユーティリティーを起動することによって、データベースを元の状態に戻すことができます。

このユーティリティーは、表スペース・レベルのバックアップをリストアすることもできます。

オペレーティング・システムまたはワード・サイズ (32 ビットか 64 ビットか) が異なる場合、増分イメージおよび「差分イメージ」(以前のキャプチャー時との差異だけをキャプチャーするイメージ) をリストアすることはできません。

ある環境から別の環境へのリストア操作を行った後は、非増分バックアップが実施されるまで、増分バックアップまたは差分バックアップを実行できません。 (同じ環境でのリストア環境の場合、この制限はありません。)

ある環境から別の環境へのリストア操作が成功した場合でも、いくつかの注意事項があります。パッケージは、使用する前に再バインドする必要があります (BIND コマンド、REBIND コマンド、または db2rbind ユーティリティーを使用)。 SQL プロシージャーは、ドロップしてから再作成する必要があります。また、外部ライブラリーは、新しいプラットフォーム上ですべて再ビルドする必要があります。 (同じ環境にリストアする場合、これらの点は該当しません。)

既存のデータベースと既存のコンテナーに対して実行されるリストア操作では、同じコンテナーと表スペース・マップが再利用されます。

新規のデータベースに対して実行されるリストア操作では、すべてのコンテナーが再取得され、最適化された表スペース・マップが再作成されます。既存のデータベースに対して実行されるリストア操作でも 1 つ以上のコンテナーがない場合は、すべてのコンテナーが再取得され、最適化された表スペース・マップが再作成されます。

有効範囲

このコマンドは、それが実行されたノードに対してだけ影響を与えます。

許可

既存のデータベースにリストアするには、次のいずれかが必要です。

- *sysadm*
- *sysctrl*
- *sysmaint*

新規のデータベースにリストアするには、次のいずれかが必要です。

- *sysadm*
- *sysctrl*

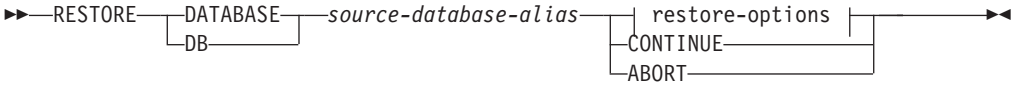
必要な接続

必要な接続は、リストア・アクションの種類によって異なります。

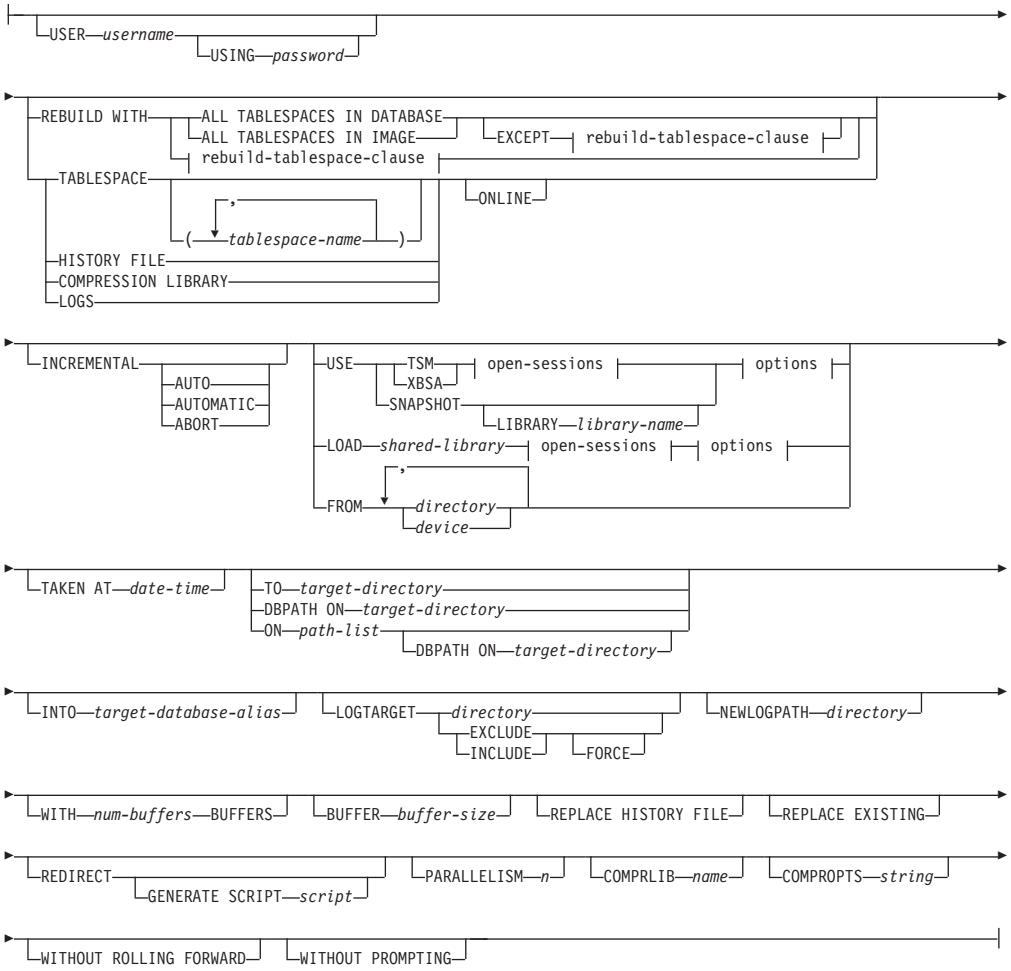
- 既存のデータベースにリストアするには、データベース接続が必要です。このコマンドは、指定されたデータベースへの排他接続を自動的に確立します。
- 新しいデータベースにリストアするには、インスタンスおよびデータベース接続が必要です。データベースを作成するには、インスタンス接続が必要です。

  現行のインスタンスとは異なるインスタンスで新規のデータベースへのリストアを行うには、まず、新規のデータベースを存在させるインスタンスにアタッチすることが必要です。新規インスタンスは、ローカルでもリモートでもかまいません。現在のインスタンスは、DB2INSTANCE 環境変数の値によって定義されます。
- スナップショット・リストアの場合、インスタンス とデータベース の接続が必要です。

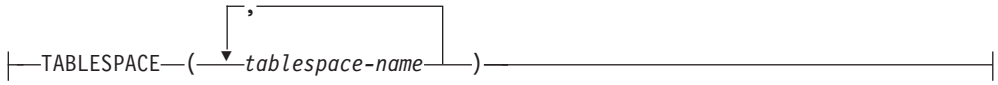
コマンド構文



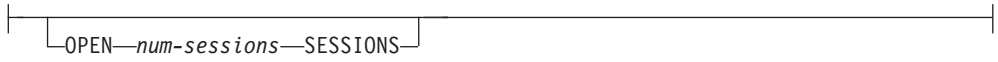
restore-options:



rebuild-tablespace-clause:



open-sessions:



オプション:

コマンド・パラメーター

DATABASE *source-database-alias*
バックアップが取得されるソース・データベースの別名です。

CONTINUE
コンテナーが再定義されていること、およびリダイレクトしたリストア操作の最終ステップを実行する必要があることを指定します。

ABORT
このパラメーターは以下を指定します。

- リダイレクトしたリストア操作を停止します。これは、1 つ以上のステップを繰り返す必要があるエラーが発生したときに便利です。 ABORT オプションを指定して RESTORE DATABASE を発行した後、 REDIRECT オプションを指定した RESTORE DATABASE を含む、リダイレクトしたリストア操作の各ステップを繰り返す必要があります。
- 完了する前に増分リストア操作を終了します。

USER *username*
データベースがリストアされる際のユーザー名を識別します。

USING *password*
ユーザー名を認証するために使用するパスワード。パスワードを省略すると、ユーザーに入力を求めるプロンプトが出ます。

REBUILD WITH ALL TABLESPACES IN DATABASE
イメージをリストアする時点でデータベースが認識しているすべての表スペースを使って、データベースをリストアします。データベースが既に存在する場合、このリストアによってそれが上書きされます。

REBUILD WITH ALL TABLESPACES IN DATABASE EXCEPT
rebuild-tablespace-clause
イメージをリストアする時点でデータベースが認識しているすべての表スペースを使って、データベースをリストアします。ただし、リストで指定されているものは除外されます。データベースが既に存在する場合、このリストアによってそれが上書きされます。

REBUILD WITH ALL TABLESPACES IN IMAGE
リストアされるイメージに含まれる表スペースだけを使ってデータベースをリストアします。データベースが既に存在する場合、このリストアによってそれが上書きされます。

REBUILD WITH ALL TABLESPACES IN IMAGE EXCEPT
rebuild-tablespace-clause
リストアされるイメージに含まれる表スペースだけを使ってデータベースを

リストアします。ただし、リストで指定されているものは除外されます。データベースが既に存在する場合、このリストアによってそれが上書きされます。

REBUILD WITH *rebuild-tablespace-clause*
指定された表スペースのリストだけを使ってデータベースをリストアします。データベースが既に存在する場合、このリストアによってそれが上書きされます。

TABLESPACE *tablespace-name*
リストアされる表スペースを指定するときに使用する名前のリストです。

ONLINE
このキーワードは、表スペース・レベルのリストア操作を行う場合のみ適用でき、これを指定するとオンラインでバックアップ・イメージがリストアできます。これは、他のエージェントが、バックアップ・イメージのリストア中にデータベースに接続できることや、指定された表スペースのリストア中に他の表スペースのデータを使用できることを意味します。

HISTORY FILE
このキーワードは、バックアップ・イメージから履歴ファイルのみをリストアするのに指定されます。

COMPRESSION LIBRARY
このキーワードは、バックアップ・イメージから圧縮ライブラリーだけをリストアする場合に指定します。バックアップ・イメージの中にオブジェクトが存在している場合、それはデータベース・ディレクトリーの中にリストアされます。バックアップ・イメージの中にオブジェクトが存在しない場合、リストア操作は失敗します。

LOGS このキーワードは、バックアップ・イメージに含まれている一連のログ・ファイルだけをリストアする場合に指定します。バックアップ・イメージの中にログ・ファイルが含まれていない場合、リストア操作は失敗します。このオプションを指定する場合は、LOGTARGET オプションも指定する必要があります。

INCREMENTAL
INCREMENTAL は、追加のパラメーターを使用しないで手動累積リストア操作を指定します。手動リストアの際、ユーザーはリストアに含まれるイメージごとに各リストア・コマンドを手動で発行する必要があります。以下の順序でこれを行ってください。最後、1 番目、2 番目、以下同様に最後のイメージまで。

INCREMENTAL AUTOMATIC/AUTO
自動累積リストア操作を指定します。

INCREMENTAL ABORT
手動累積リストア操作を指定します。

USE

TSM データベースが Tivoli Storage Manager 管理の出力からリストアされるように指定します。

XBSA XBSA インターフェースを使用するように指定します。バックアップ・サービス API (XBSA) は、バックアップやアーカイブの目的で

データ・ストレージ管理を必要とするアプリケーションまたは機能のための、オープン・アプリケーション・プログラミング・インターフェースです。

SNAPSHOT
データがスナップショット・バックアップからリストアされるように指定します。

SNAPSHOT パラメーターは、以下のパラメーターと一緒に使用することはできません。

- INCREMENTAL
- TO
- ON
- DBPATH ON
- INTO
- NEWLOGPATH
- WITH *num-buffers* BUFFERS
- BUFFER
- REDIRECT
- REPLACE HISTORY FILE
- COMPRESSION LIBRARY
- PARALLELISM
- COMPRLIB
- OPEN *num-sessions* SESSIONS
- HISTORY FILE
- LOGS

また、SNAPSHOT パラメーターは、表スペース・リストが関係するリストア操作で使用することはできません。これには、REBUILD WITH オプションが含まれます。

スナップショット・バックアップ・イメージからのデータ・リストア時のデフォルトの動作は、すべてのコンテナー、ローカル・ボリューム・ディレクトリー、データベース・パス (DBPATH)、最新のスナップショット・バックアップの 1 次ログとミラー・ログのパス (タイム・スタンプが指定されていない場合) を含む、データベースを構成するすべてのパスの FULL DATABASE OFFLINE リストアです (明示的に EXCLUDE LOGS が指定されているのでない限り、すべてのスナップショット・バックアップでのデフォルトは INCLUDE LOGS です)。タイム・スタンプが提供されている場合、そのスナップショット・バックアップ・イメージがリストアされます。

LIBRARY *library-name*
IBM Data Server には、以下のストレージ・ハードウェアのための DB2 ACS API ドライバーが組み込まれています。

- IBM TotalStorage® SAN ボリューム・コントローラー

- IBM Enterprise Storage Server® Model 800
- IBM System Storage™ DS6000™
- IBM System Storage DS8000™
- IBM System Storage N Series
- NetApp V シリーズ
- NetApp FAS

他のストレージ・ハードウェアを使用していて、そのストレージ・ハードウェア用の DB2 ACS API ドライバーがある場合、LIBRARY パラメーターを使用してその DB2 ACS API ドライバーを指定できます。

LIBRARY パラメーターの値は、完全修飾ライブラリー・ファイル名です。

OPTIONS

"options-string"
リストア操作で使用するオプションを指定します。このストリングは、二重引用符なしで、入力されたとおりに DB2 ACS API ドライバーに渡されます。 **VENDOROPT** データベース構成パラメーターを使用してスナップショット・リストア操作でのベンダー固有のオプションを指定することはできません。代わりに、リストア・ユーティリティーの OPTIONS パラメーターを使用する必要があります。

@file-name
リストア操作で使用するオプションが、 DB2 サーバー上のファイルに含まれていることを指定します。このストリングは、ベンダー・サポートのライブラリーに渡されます。ファイル名は完全修飾ファイル名でなければなりません。

OPEN *num-sessions* **SESSIONS**
TSM またはベンダー製品とともに使用する入出力セッションの数を指定します。

FROM *directory/device*
バックアップ・イメージがあるディレクトリーまたは装置の完全修飾パス名。 USE TSM、FROM、および LOAD を省略した場合のデフォルト値は、クライアント・マシンの現行作業ディレクトリーです。このターゲット・ディレクトリーまたは装置は、ターゲット・サーバー/インスタンス上に存在している必要があります。

複数の項目が指定され、項目の最後がテープ装置である場合には、他のテープが要求されます。有効な応答オプションは、次のとおりです。

c 続行。警告メッセージを生成した装置を使用し続けます (例えば、新しいテープがマウントされた場合)。

d 装置の終了。警告メッセージの原因となった装置の使用だけ を停止します (例えば、もうテープがない場合に停止する、など)。

t 終了。ユーザーが、ユーティリティーによって要求された何らかのアクションを実行しなかった場合、リストア操作を異常終了します。

LOAD *shared-library*
使用するバックアップおよびリストア I/O ベンダー関数を含む共有ライブラリー (Windows オペレーティング・システムでは DLL) の名前。名前には絶対パスを含めることができます。絶対パスを指定しない場合、ユーザー出口プログラムが置かれているパスがデフォルト値として使われます。

TAKEN AT *date-time*
データベース・バックアップ・イメージのタイム・スタンプです。タイム・スタンプはバックアップ操作が正常に終了した後に表示され、バックアップ・イメージのパス名の一部になっています。 *yyyymmddhhmmss* の形式で指定されます。タイム・スタンプを部分的に指定することもできます。例えば、2 つの異なるタイム・スタンプ 20021001010101 および 20021002010101 で指定されるバックアップ・イメージが存在する場合、`20021002` を指定することで、タイム・スタンプ `20021002010101` のイメージが使用できます。このパラメーターに値を指定しない場合は、ソース・メディア上のバックアップ・イメージは 1 つだけでなければなりません。

TO *target-directory*
このパラメーターは、ターゲット・データベース・ディレクトリーを指定します。ユーティリティーが存在するデータベースへリストアしている場合には、このパラメーターは無視されます。指定するドライブおよびディレクトリーは、ローカルのものでなければなりません。自動ストレージが有効になったデータベースがバックアップ・イメージに含まれる場合、データベース・ディレクトリーだけが変更され、そのデータベースに関連したストレージ・パスは変更されません。

DBPATH ON *target-directory*
このパラメーターは、ターゲット・データベース・ディレクトリーを指定します。ユーティリティーが存在するデータベースへリストアしている場合には、このパラメーターは無視されます。指定するドライブおよびディレクトリーは、ローカルのものでなければなりません。自動ストレージが有効になったデータベースがバックアップ・イメージに含まれ、ON パラメーターが指定されない場合、このパラメーターは TO パラメーターと同じ意味になり、データベース・ディレクトリーだけが変更されます。そのデータベースに関連したストレージ・パスは変更されません。

ON *path-list*
このパラメーターは、自動ストレージ・データベースに関連したストレージ・パスを再定義します。自動ストレージが有効になっていないデータベースに対してこのパラメーターを使用した場合、エラー (SQL20321N) が発生します。バックアップ・イメージ内に定義された既存のストレージ・パスはもはや使用されなくなり、自動ストレージ表スペースは新しいパスに自動的にリダイレクトされます。自動ストレージ・データベースに対してこのパラメーターを指定しない場合、ストレージ・パスはバックアップ・イメージ内に定義されたままの状態になります。

1 つのパス、またはコンマで区切った複数のパスを指定できます。それぞれのパスは絶対パス名でなければならず、ローカルに存在しなければなりませ

ん。データベースがディスクにまだ存在せず、DBPATH ON パラメーターが指定されていない場合には、最初のパスがターゲット・データベース・ディレクトリーとして使用されます。

複数パーティション・データベースの場合、「ON *path-list*」オプションを指定できるのはカタログ・パーティションについてだけです。 ON オプションを使用する場合、カタログ・パーティションは、他のどのパーティションがリストアされるよりも前にリストアする必要があります。新しいストレージ・パスでカタログ・パーティションをリストアすると、非カタログ・ノードのすべてが RESTORE_PENDING 状態になります。その場合、非カタログ・ノードは、リストア・コマンドに ON 節を指定することなく並列してリストアできます。

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、それらはすべて、RESTORE DATABASE コマンドの実行前に存在している必要があります。その例外の 1 つとして、ストレージ・パス内でデータベース・パーティション式を使用する場合があります。その使用によって、処理結果のパス名が各パーティションごとに異なるように、データベース・パーティション番号をストレージ・パスにおいて反映することができます。

データベース・パーティション式を指示するには、引数 " $N" ([blank]$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・パス内のどこででも使用することができ、複数のデータベース・パーティション式を指定してもかまいません。データベース・パーティション式はスペース文字で終了します。スペースの後に続く文字はすべて、データベース・パーティション式が評価された後、ストレージ・パスに付加されます。ストレージ・パス内でデータベース・パーティション式の後にスペース文字がない場合、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、以下の形式のいずれかでのみ使用できます。

表 49. : 演算子は、左から右へ評価されます。 % は、モジュラス演算子を表します。例中のデータベース・パーティション番号は 10 と想定されています。

構文	例	値
[blank]$N	″ $N″	10
[blank]$N+[number]	″ $N+100″	110
[blank]$N%[number]	″ $N%5″	0
[blank]$N+[number]%[number]	″ $N+1%5″	1
[blank]$N%[number]+[number]	″ $N%4+2″	4
[a] % はモジュラスです。		

INTO *target-database-alias*

ターゲット・データベースの別名です。ターゲット・データベースが存在しない場合には、作成されます。

データベース・バックアップを既存のデータベースにリストアするとき、リストアされたデータベースは既存のデータベースの別名およびデータベース名を継承します。データベース・バックアップを存在していないデータベースにリストアするとき、新規のデータベースが指定した別名およびデータベ

ース名を使用して作成されます。新しいデータベース名は、リストア先のシステムで固有のものでなければなりません。

LOGTARGET *directory*
スナップショット以外のリストアの場合:

バックアップ・イメージからログ・ファイルを抽出する際のターゲット・ディレクトリーとして使用する、データベース・サーバー上の既存のディレクトリーの絶対パス名。このオプションを指定する場合、バックアップ・イメージ内のログ・ファイルは、そのターゲット・ディレクトリー内に抽出されます。このオプションを指定しない場合、バックアップ・イメージ内のログ・ファイルは抽出されません。バックアップ・イメージからログ・ファイルだけを抽出する場合は、 LOGS オプションを指定してください。

スナップショット・リストアの場合:

INCLUDE
スナップショット・イメージからログ・ディレクトリー・ボリュームをリストアします。このオプションが指定されていて、バックアップ・イメージにログ・ディレクトリーが含まれている場合、それらはリストアされます。ディスク上に既存のログ・ディレクトリーとログ・ファイルは、バックアップ・イメージ中のログ・ディレクトリーと競合するのでなければ、変更なしでそのままになります。ディスク上に既存のログ・ディレクトリーがバックアップ・イメージ中のログ・ディレクトリーと競合する場合は、エラーが戻されます。

EXCLUDE
ログ・ディレクトリー・ボリュームをリストアしません。このオプションを指定すると、バックアップ・イメージからログ・ディレクトリーはリストアされません。ディスク上に既存のログ・ディレクトリーとログ・ファイルは、バックアップ・イメージ中のログ・ディレクトリーと競合するのでなければ、変更なしでそのままになります。データベースに属する 1 つのパスがリストアされ、そのために暗黙のうちに 1 つのログ・ディレクトリーがリストアされ、その結果、ログ・ディレクトリーが上書きされることになる場合、エラーが戻されます。

FORCE
スナップショット・イメージをリストアする時に現行データベースの既存のログ・ディレクトリーを上書きおよび置換することを許可します。このオプションを使用しなければ、スナップショット・イメージのログ・ディレクトリーと矛盾するディスク上の既存のログ・ディレクトリーおよびログ・ファイルが原因で、リストアは失敗します。このオプションを使用して、リストアでこれらの既存のログ・ディレクトリーを上書きおよび置換できるように指示します。

注: このオプションは注意して使用し、リカバリーに必要になる可能性があるすべてのログを常にバックアップおよびアーカイブしてください。

注: LOGTARGET がスナップショット以外のリストア用に指定されていない場合には、デフォルトの LOGTARGET ディレクトリーは LOGTARGET EXCLUDE です。

NEWLOGPATH *directory*
リストア操作後にアクティブ・ログ・ファイルに使用されるディレクトリーの絶対パス名。このパラメーターの機能は **newlogpath** データベース構成パラメーターと同じです。ただし、これが影響するのは、これを指定したリストア操作に限定されます。このパラメーターは、バックアップ・イメージのログ・パスが、リストア後の使用に適していない場合に使用することができます。例えば、パスがもはや有効でない、または別のデータベースによって使用されている、という場合などです。

WITH *num-buffers* **BUFFERS**
使用するバッファーの数です。値を明示的に指定しない場合、DB2 データベース・システムはこのパラメーターの最適値を自動的に選択します。複数のソースが読み取られる場合や、PARALLELISM の値が増やされている場合は、パフォーマンスを向上させるために複数のバッファーを使用することができます。

BUFFER *buffer-size*
リストア操作に使用するバッファーのサイズ (ページ数)。値を明示的に指定しない場合、DB2 データベース・システムはこのパラメーターの最適値を自動的に選択します。このパラメーターの最小値は 8 ページです。

リストア・バッファー・サイズは、バックアップ操作中に指定したバックアップ・バッファー・サイズに正の整数を乗算したサイズでなければなりません。誤ったバッファー・サイズを指定すると、許容可能な最小のサイズで割り振られます。

REPLACE HISTORY FILE
リストア操作において、ディスク上の履歴ファイルを、バックアップ・イメージの履歴ファイルで置換することを指定します。

REPLACE EXISTING
ターゲット・データベースの別名と同じ別名を持つデータベースが既に存在している場合、このパラメーターは、リストア・ユーティリティーが既存のデータベースをリストアしたデータベースに置換するように指定します。これはリストア・ユーティリティーを起動するスクリプトで便利です。コマンド行プロセッサーは、ユーザーに既存のデータベースの削除を検証するよう求めるプロンプトを出さないためです。 WITHOUT PROMPTING パラメーターが指定された場合、REPLACE EXISTING を指定する必要はありませんが、その場合、ユーザー介入を標準的に必要とするイベントが起こるとこの操作は失敗します。

REDIRECT
リダイレクトしたリストア操作を指定します。リダイレクトしたリストア操作を完了するには、このコマンドの後に 1 つ以上の SET TABLESPACE CONTAINERS コマンドを続け、次に CONTINUE オプションを指定して RESTORE DATABASE コマンドを続ける必要があります。同一のリダイレクトしたリストア操作に関連したコマンドはすべて、同じウィンドウまたは

CLP セッションから起動しなければなりません。自動ストレージが有効になっている表スペースに対して、リダイレクト・リストア操作を実行することはできません。

GENERATE SCRIPT *script*
指定されたファイル名を使用して、リダイレクト・リストア・スクリプトを作成します。スクリプト名は相対パスまたは絶対パスであり、そのスクリプトはクライアント・サイドで生成されます。クライアント・サイドでそのファイルを作成できない場合には、エラー・メッセージ (SQL9304N) が戻されます。ファイルが既に存在する場合は上書きされます。使用法に関する情報は、下記の例を参照してください。

WITHOUT ROLLING FORWARD
データベースを、正常にリストアされた後ロールフォワード・ペンディング状態にしないように指定します。

正常なリストアに続いて、データベースがロールフォワード・ペンディング状態にある場合には、データベースが使用できるようになる前に、ROLLFORWARD コマンドを起動する必要があります。

オンライン・バックアップ・イメージからのリストアでこのオプションを指定した場合、エラー SQL2537N が戻されます。

注: リカバリー可能データベースのバックアップ・イメージである場合、REBUILD オプションに WITHOUT ROLLING FORWARD を指定することはできません。

PARALLELISM *n*
リストア操作中に作成されるバッファー・マニピュレーターの数を指定します。値を明示的に指定しない場合、DB2 データベース・システムはこのパラメーターの最適値を自動的に選択します。

COMPRLIB *name*
解凍を実行するために使用するライブラリーの名前 (例えば、Windows では `db2compr.dll`、Linux/UNIX システムでは `libdb2compr.so` です)。この名前は、サーバー上の 1 個のファイルを参照する完全修飾パスでなければなりません。このパラメーターを指定しない場合、 DB2 はイメージ内に格納されているライブラリーの使用を試みます。バックアップが圧縮されていなかった場合、このパラメーターの値は無視されます。指定されたライブラリーをロードできない場合、リストア操作は失敗します。

COMPROPTS *string*
バイナリー・データのうち、解凍ライブラリーの初期設定ルーチンに渡すブロックを記述します。 DB2 データベース・システムはこのストリングをクライアントからサーバーに直接渡すため、バイト反転やコード・ページ変換の問題がある場合は、解凍ライブラリーで処理されます。データ・ブロックの最初の文字が「@」なら、データの残りの部分は、サーバー上に存在するファイルの名前を指定するものとして、DB2 データベース・システムは解釈します。その場合、DB2 データベース・システムは *string* の内容をこのファイルの内容で置き換え、新しい値を初期設定ルーチンに渡します。ストリングの最大長は 1 024 バイトです。

WITHOUT PROMPTING
リストア操作を無人で実行するように指定します。通常はユーザー介入を必

要とするアクションでは、エラー・メッセージが戻されます。テープやディスケットなどの取り外し可能メディア装置を使用している場合、このオプションを指定していても、その装置が終わるとプロンプトが出されます。

例

1. 以下の例で、データベース WSDB は 0 から 3 までの番号が付けられた 4 つのデータベース・パーティションすべてに定義されています。パス /dev3/backup はすべてのデータベース・パーティションからアクセスできます。以下のオフライン・バックアップ・イメージは、/dev3/backup から入手可能です。

```
wsdb.0.db2inst1.NODE0000.CATN0000.20020331234149.001
wsdb.0.db2inst1.NODE0001.CATN0000.20020331234427.001
wsdb.0.db2inst1.NODE0002.CATN0000.20020331234828.001
wsdb.0.db2inst1.NODE0003.CATN0000.20020331235235.001
```

   最初にカタログ・パーティションをリストアしてから WSDB データベースの他のすべてのデータベース・パーティションを /dev3/backup ディレクトリーからリストアするには、データベース・パーティションの 1 つから以下のコマンドを出します。

```
db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
  INTO wsdb REPLACE EXISTING'
db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
  INTO wsdb REPLACE EXISTING'
db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
  INTO wsdb REPLACE EXISTING'
db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
  INTO wsdb REPLACE EXISTING'
```

   db2_all ユーティリティーは、指定された各データベース・パーティションへのリストア・コマンドを出します。 db2_all を使用してリストアを実行する場合は、常に REPLACE EXISTING や WITHOUT PROMPTING を指定してください。これを指定しないと、プロンプトが表示された場合に操作がハングしたように見えます。それは、db2_all でユーザー・プロンプトがサポートされていないためです。

2. 以下は、別名が MYDB であるデータベースの典型的なリダイレクト・リストアのシナリオです。

   a. 次のように、REDIRECT オプションを指定して RESTORE DATABASE コマンドを発行する。

```
restore db mydb replace existing redirect
```

      ステップ 1 が正常終了した後でステップ 3 が完了する前に、次を発行してリストア操作を打ち切ることができます。

```
restore db mydb abort
```

   b. 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行する。以下に例を示します。

```
set tablespace containers for 5 using
   (file 'f:¥ts3con1' 20000, file 'f:¥ts3con2' 20000)
```

リストアしたデータベースのコンテナーが、このステップで指定したものであることを検査するために、 LIST TABLESPACE CONTAINERS コマンドを発行する。

c. ステップ 1 および 2 が正常終了した後、次を発行します。

```
restore db mydb continue
```

これはリダイレクト・リストア操作の最終ステップです。

d. ステップ 3 が失敗した場合、またはリストア操作を打ち切った場合、リダイレクト・リストアはステップ 1 から再始動できます。

3. 以下は、リカバリー可能データベース用の増分バックアップの週間予定のサンプルです。週 1 回の全データベース・バックアップ操作、1 日 1 回の非累積 (差分) バックアップ操作、および週 2 回の累積 (増分) バックアップ操作が含まれています。

```
(Sun) backup db mydb use tsm
(Mon) backup db mydb online incremental delta use tsm
(Tue) backup db mydb online incremental delta use tsm
(Wed) backup db mydb online incremental use tsm
(Thu) backup db mydb online incremental delta use tsm
(Fri) backup db mydb online incremental delta use tsm
(Sat) backup db mydb online incremental use tsm
```

金曜日の午前中に作成されたイメージを自動データベース・リストアするには、次のようにします。

```
restore db mydb incremental automatic taken at (Fri)
```

金曜日の午前中に作成されたイメージを手動データベース・リストアするには、次のようにします。

```
restore db mydb incremental taken at (Fri)
restore db mydb incremental taken at (Sun)
restore db mydb incremental taken at (Wed)
restore db mydb incremental taken at (Thu)
restore db mydb incremental taken at (Fri)
```

4. リモート・サイトに移動することを意図したバックアップ・イメージを作成し、それにログを含めるには、次のようにします。

```
backup db sample online to /dev3/backup include logs
```

このバックアップ・イメージをリストアするには、 LOGTARGET パスを指定し、ROLLFORWARD でそのパスを指定します。

```
restore db sample from /dev3/backup logtarget /dev3/logs
rollforward db sample to end of logs and stop overflow log path /dev3/logs
```

5. ログを含むバックアップ・イメージから、ログ・ファイルだけを取り出すには、

```
restore db sample logs from /dev3/backup logtarget /dev3/logs
```

6. リストア操作で使用する TSM 情報を指定するには、 USE TSM OPTIONS キーワードを使用します。 Windows プラットフォームでは、-fromowner オプションを指定しないでください。

- 区切り文字付きストリングを指定する場合、

```
restore db sample use TSM options '"-fromnode=bar -fromowner=dmcinnis"'
```

- 完全修飾ファイル名を指定する場合、

```
restore db sample use TSM options @/u/dmcinnis/myoptions.txt
```

ファイル myoptions.txt には、 -fromnode=bar -fromowner=dmcinnis というストリングが含まれています。

7. 以下に示すのは、新しいストレージ・パスによる、複数パーティション自動ストレージ対応データベースの簡単なリストアです。もともとこのデータベースは、1 つのストレージ・パス /myPath0 を使用して作成されたものです。
   - カタログ・パーティションで、次のコマンドを発行します。restore db mydb on /myPath1,/myPath2
   - カタログでないすべてのパーティションで、次のコマンドを発行します。
     restore db mydb
8. 非自動ストレージ・データベースにおいて以下のコマンドを発行すると、そのスクリプト出力は、

```
restore db sample from /home/jseifert/backups taken at 20050301100417 redirect
generate script SAMPLE_NODE0000.clp
```

下記のようなものになります。

```
-- ****************************************************************************
-- ** automatically created redirect restore script
-- ****************************************************************************
UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM  0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- ****************************************************************************
-- ** initialize redirected restore
-- ****************************************************************************
RESTORE DATABASE SAMPLE
-- USER  '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050301100417
-- DBPATH ON '<target-directory>'
INTO SAMPLE
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/SQLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- ****************************************************************************
-- ** tablespace definition
-- ****************************************************************************
-- ****************************************************************************
-- ** Tablespace name                         = SYSCATSPACE
-- **   Tablespace ID                         = 0
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 5572
-- ****************************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
```

```
);
-- ******************************************************************************
-- ** Tablespace name                         = TEMPSPACE1
-- **   Tablespace ID                         = 1
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = System Temporary data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 0
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- ******************************************************************************
-- ** Tablespace name                         = USERSPACE1
-- **   Tablespace ID                         = 2
-- **   Tablespace Type                       = System managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Total number of pages                 = 1
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0002.0'
);
-- ******************************************************************************
-- ** Tablespace name                         = DMS
-- **   Tablespace ID                         = 3
-- **   Tablespace Type                       = Database managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   /tmp/dms1                                                  1000
, FILE   /tmp/dms2                                                  1000
);
-- ******************************************************************************
-- ** Tablespace name                         = RAW
-- **   Tablespace ID                         = 4
-- **   Tablespace Type                       = Database managed space
-- **   Tablespace Content Type               = Any data
-- **   Tablespace Page size (bytes)          = 4096
-- **   Tablespace Extent size (pages)        = 32
-- **   Using automatic storage               = No
-- **   Auto-resize enabled                   = No
-- **   Total number of pages                 = 2000
-- **   Number of usable pages                = 1960
-- **   High water mark (pages)               = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                        1000
```

```
, DEVICE '/dev/hdb2'                                        1000
);
-- *********************************************************************************
-- ** start redirect restore
-- *********************************************************************************
RESTORE DATABASE SAMPLE CONTINUE;
-- *********************************************************************************
-- ** end of file
-- *********************************************************************************
```

9. 自動ストレージ・データベースにおいて以下のコマンドを発行すると、そのスクリプト出力は、

```
restore db test from /home/jseifert/backups taken at 20050304090733 redirect
generate script TEST_NODE0000.clp
```

下記のようなものになります。

```
-- *********************************************************************************
-- ** automatically created redirect restore script
-- *********************************************************************************
UPDATE COMMAND OPTIONS USING S ON Z ON TEST_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM  0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- *********************************************************************************
-- ** initialize redirected restore
-- *********************************************************************************
RESTORE DATABASE TEST
-- USER  '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050304090733
ON '/home/jseifert'
-- DBPATH ON <target-directory>
INTO TEST
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00002/SQLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- *********************************************************************************
-- ** tablespace definition
-- *********************************************************************************
-- *********************************************************************************
-- ** Tablespace name                          = SYSCATSPACE
-- **   Tablespace ID                          = 0
-- **   Tablespace Type                        = Database managed space
-- **   Tablespace Content Type                = Any data
-- **   Tablespace Page size (bytes)           = 4096
-- **   Tablespace Extent size (pages)         = 4
-- **   Using automatic storage                = Yes
-- **   Auto-resize enabled                    = Yes
-- **   Total number of pages                  = 6144
-- **   Number of usable pages                 = 6140
-- **   High water mark (pages)                = 5968
-- *********************************************************************************
-- *********************************************************************************
-- ** Tablespace name                          = TEMPSPACE1
-- **   Tablespace ID                          = 1
-- **   Tablespace Type                        = System managed space
-- **   Tablespace Content Type                = System Temporary data
-- **   Tablespace Page size (bytes)           = 4096
```

```
-- **   Tablespace Extent size (pages)           = 32
-- **   Using automatic storage                  = Yes
-- **   Total number of pages                    = 0
-- *********************************************************************************
-- *********************************************************************************
-- ** Tablespace name                           = USERSPACE1
-- **   Tablespace ID                            = 2
-- **   Tablespace Type                          = Database managed space
-- **   Tablespace Content Type                  = Any data
-- **   Tablespace Page size (bytes)             = 4096
-- **   Tablespace Extent size (pages)           = 32
-- **   Using automatic storage                  = Yes
-- **   Auto-resize enabled                      = Yes
-- **   Total number of pages                    = 256
-- **   Number of usable pages                   = 224
-- **   High water mark (pages)                  = 96
-- *********************************************************************************
-- *********************************************************************************
-- ** Tablespace name                           = DMS
-- **   Tablespace ID                            = 3
-- **   Tablespace Type                          = Database managed space
-- **   Tablespace Content Type                  = Any data
-- **   Tablespace Page size (bytes)             = 4096
-- **   Tablespace Extent size (pages)           = 32
-- **   Using automatic storage                  = No
-- **   Auto-resize enabled                      = No
-- **   Total number of pages                    = 2000
-- **   Number of usable pages                   = 1960
-- **   High water mark (pages)                  = 96
-- *********************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   '/tmp/dms1'                                        1000
, FILE   '/tmp/dms2'                                        1000
);
-- *********************************************************************************
-- ** Tablespace name                           = RAW
-- **   Tablespace ID                            = 4
-- **   Tablespace Type                          = Database managed space
-- **   Tablespace Content Type                  = Any data
-- **   Tablespace Page size (bytes)             = 4096
-- **   Tablespace Extent size (pages)           = 32
-- **   Using automatic storage                  = No
-- **   Auto-resize enabled                      = No
-- **   Total number of pages                    = 2000
-- **   Number of usable pages                   = 1960
-- **   High water mark (pages)                  = 96
-- *********************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                        1000
, DEVICE '/dev/hdb2'                                        1000
);
-- *********************************************************************************
-- ** start redirect restore
-- *********************************************************************************
RESTORE DATABASE TEST CONTINUE;
-- *********************************************************************************
-- ** end of file
-- *********************************************************************************
```

10. SNAPSHOT オプションを使用した RESTORE DB コマンドの例を以下に示します。

ログ・ディレクトリー・ボリュームをスナップショット・イメージからリストアし、プロンプトを出しません。

```
db2 restore db sample use snapshot LOGTARGET INCLUDE without prompting
```

ログ・ディレクトリー・ボリュームをリストアせず、プロンプトを出しません。

```
db2 restore db sample use snapshot LOGTARGET EXCLUDE without prompting
```

ログ・ディレクトリー・ボリュームをリストアせず、プロンプトを出しません。 LOGTARGET が指定されていない場合には、デフォルトは LOGTARGET EXCLUDE です。

```
db2 restore db sample use snapshot without prompting
```

矛盾するログ・ディレクトリーが含まれるスナップショット・イメージをリストアする時に、プロンプトを出さずに現行データベースの既存のログ・ディレクトリーを上書きおよび置換することを許可します。

```
db2 restore db sample use snapshot LOGTARGET EXCLUDE FORCE without prompting
```

矛盾するログ・ディレクトリーが含まれるスナップショット・イメージをリストアする時に、プロンプトを出さずに現行データベースの既存のログ・ディレクトリーを上書きおよび置換することを許可します。

```
db2 restore db sample use snapshot LOGTARGET INCLUDE FORCE without prompting
```

使用上の注意

- db2 restore db <name> という形式の RESTORE DATABASE コマンドは、データベース・イメージを使ったフル・データベース・リストアの実行と、表スペース・イメージ内に検出される表スペースの表スペース・リストア操作を実行します。 db2 restore db <name> tablespace という形式の RESTORE DATABASE コマンドは、イメージ内に検出される表スペースの表スペース・リストアを実行します。さらに、そのようなコマンドで表スペースのリストが指定された場合、明示的にリストされるすべての表スペースがリストアされます。
- オンライン・バックアップのリストア操作を実行した後、ロールフォワード・リカバリーを実行する必要があります。
- バックアップ・イメージが圧縮されているなら、DB2 データベース・システムはそのことを検出し、データはリストア前に自動的に解凍されます。 db2Restore API でライブラリーが指定されている場合、データの解凍にはそれが使用されます。そうでない場合、ライブラリーがバックアップ・イメージに保管されているかどうか検査され、ライブラリーが存在する場合にはそれが使用されます。最後に、ライブラリーがバックアップ・イメージに入っていない場合には、データを解凍できず、リストア操作が失敗します。
- バックアップ・イメージから圧縮ライブラリーをリストアする場合 (COMPRESSION LIBRARY オプションを指定して明示的に、または圧縮バックアップの通常のリストアを実行することにより暗黙的に)、そのリストア操作は、バックアップが作成されたのと同じプラットフォームおよびオペレーティング・システム上で実行する必要があります。バックアップ作成時のプラットフォームとリストア操作実行時のプラットフォームが違っていると、それらの 2 つのシステムの間のクロスプラットフォーム・リストアが DB2 で通常にサポートされている場合でも、リストア操作は失敗します。

- SMS 表スペースのバックアップは SMS 表スペースにのみリストアすることができます。 SMS 表スペースのバックアップを DMS 表スペースにリストアすることはできません。その逆も同様です。
- ログ・ファイルを含むバックアップ・イメージからログ・ファイルをリストアする場合には、LOGTARGET オプションを指定する必要があります。その際、DB2 サーバー上に存在する有効な完全修飾パス名を指定する必要があります。それらの条件が満たされている場合、リストア・ユーティリティーは、イメージ内のログ・ファイルをターゲット・パスに書き込みます。ログを含まないバックアップ・イメージのリストア操作で LOGTARGET を指定した場合、表スペース・データのリストアが試行される前にエラーが戻されます。また、LOGTARGET に無効なパスや読み取り専用パスが指定された場合も、リストア操作が失敗してエラーが戻されます。
- RESTORE DATABASE コマンド発行の時点で LOGTARGET パス内にログ・ファイルが存在している場合、警告プロンプトがユーザーに対して戻されます。WITHOUT PROMPTING が指定されている場合、この警告は戻されません。
- LOGTARGET を指定したリストア操作において、いずれかのログ・ファイルを抽出できない場合には、リストア操作が失敗してエラーが戻されます。バックアップ・イメージから抽出されるいずれかのログ・ファイルの名前が、LOGTARGET パス内に存在するファイル名と同じである場合には、リストア操作が失敗してエラーが戻されます。データベース・リストア・ユーティリティーは、LOGTARGET ディレクトリー内に既存のログ・ファイルを上書きしません。
- 保管されているログ・セットだけをバックアップ・イメージからリストアすることも可能です。ログ・ファイルだけをリストアすることを指定するには、LOGTARGET パスに加えて LOGS オプションを指定します。 LOGTARGET パスを指定しないで LOGS オプションを指定すると、エラーになります。この操作モードでログ・ファイルをリストアしようとして問題が発生した場合、そのリストア操作は即座に終了し、エラーが戻されます。
- 自動増分リストア操作においては、リストア操作のターゲット・イメージに含まれるログ・ファイルだけがバックアップ・イメージから取り出されます。増分リストア処理中に参照される中間イメージに含まれるログ・ファイルは、それらの中間バックアップ・イメージから抽出されません。手動増分リストア操作の場合、 LOGTARGET パスは、最終リストア・コマンドを発行する場合にのみ指定してください。
- オフラインの全データベース・バックアップおよびオフラインの増分データベース・バックアップは、より新しいデータベース・バージョンにリストアできますが、オンライン・バックアップはそれができません。複数パーティション・データベースでは、まずカタログ・パーティションを別個にリストアしてから、その後に残りのデータベース・パーティションを (並列または逐次に) リストアする必要があります。ただし、リストア操作によって実行された暗黙的なデータベースのマイグレーションは、失敗する可能性があります。複数パーティション・データベースでは、1 つ以上のデータベース・パーティションでそれが失敗することがあります。この場合、RESTORE DATABASE コマンドの後にカタログ・パーティションから発行する単一の MIGRATE DATABASE コマンドを続けて、データベースを正常にマイグレーションすることができます。

スナップショット・リストア

従来の (スナップショット以外の) リストアのように、スナップショット・バックアップ・イメージをリストアする時のデフォルトの動作は、ログ・ディレクトリーをリストアしない、LOGTARGET EXCLUDE です。

いずれかのログ・ディレクトリーのグループ ID がリストアする他のパスのいずれかと共有されていることが DB2 マネージャーにより検出された場合、エラーが戻されます。この場合、ログ・ディレクトリーがリストアに含まれる必要があるため、LOGTARGET INCLUDE または LOGTARGET INCLUDE FORCE を指定する必要があります。

DB2 マネージャーは、バックアップ・イメージからのパスのリストアが行われる前に既存のログ・ディレクトリー (1 次、ミラー、およびオーバーフロー) を保存するために、すべての方法を試みます。

ログ・ディレクトリーをリストアする場合、ディスク上に事前に存在するログ・ディレクトリーがバックアップ・イメージ中のログ・ディレクトリーと競合することが DB2 マネージャーによって検出されたなら、DB2 マネージャーによってエラーが報告されます。その場合、LOGTARGET INCLUDE FORCE を指定した場合にはこのエラーは抑止され、イメージのログ・ディレクトリーがリストアされて、以前に存在していたログ・ディレクトリーはすべて削除されます。

LOGTARGET EXCLUDE オプションが指定され、ログ・ディレクトリー・パスがデータベース・ディレクトリーの下位 (つまり /NODExxxx/SQLxxxxx/SQLOGDIR/) にあるような特殊な事例もあります。この場合は、リストアによりログ・ディレクトリーはデータベース・パスとして上書きされ、その下位にあるすべての内容はリストアされます。このシナリオに該当することが DB2 マネージャーによって検出された場合、そのログ・ディレクトリー中にログ・ファイルが存在しているなら、エラーが報告されます。 LOGTARGET EXCLUDE FORCE を指定した場合には、このエラーは抑止され、ディスク上の矛盾するログ・ディレクトリーは、バックアップ・イメージのそれらのログ・ディレクトリーで上書きされます。

サスペンド入出力とオンライン・スプリット・ミラー・サポートによる高可用性

IBM Data Server のサスペンド入出力サポートにより、データベースをオフラインにしないで 1 次データベースのスプリット・ミラー・コピーを行うことができます。これを使用すると、1 次データベースで障害が発生した場合にテークオーバーするスタンバイ・データベースを非常に短い時間で作成することができます。

ディスク・ミラーリングは、データを 2 つの異なるハード・ディスクに同時に書き込む処理です。データの一方のコピーは、他方のミラーと呼ばれます。ミラーの分割とは、2 つのコピーを分離する処理のことです。

ディスク・ミラーリングを使用することにより、1 次データベースの 2 次コピーを保持することができます。 IBM Data Server のサスペンド入出力機能を使用すると、データベースをオフラインにしないでデータベースの 1 次ミラー・コピーと 2 次ミラー・コピーを分割することができます。 1 次データベース・コピーと 2 次データベース・コピーが分割されると、2 次データベースは 1 次データベースで障害が起きた場合に操作をテークオーバーできるようになります。

大きなデータベースは IBM Data Server バックアップ・ユーティリティーを使用してバックアップしないという場合、サスペンド入出力およびスプリット・ミラー機能を使用してミラー・イメージからコピーを作成することができます。この方法では以下の利点もあります。

- 稼働マシンからバックアップ操作のオーバーヘッドを除去します。
- 高速にシステムを複製します。
- アイドル・スタンバイ・フェイルオーバーを高速にインプリメントできます。初期のリストア操作は不要です。また、ロールフォワードが遅すぎたりエラーが発生する場合に、再初期化を高速に実行してこれらに対応することが可能です。

db2inidb コマンドは、スプリット・ミラーを初期化して以下のように使用できるようにします。

- クローン・データベースとして
- スタンバイ・データベースとして
- バックアップ・イメージとして

このコマンドは、スプリット・ミラーに対してのみ発行することができます。そのコマンドは、スプリット・ミラーを使用する前に実行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、入出力を中断してすべてのデータベース・パーティションに同時に書き込む必要はありません。 1 つ以上のデータベース・パーティションのサブセットを中断して、オフライン・バックアップを実行するためにスプリット・ミラーを作成できます。カタログ・パーティションがサブセットに含まれる場合は、そのパーティションを最後に中断するデータベース・パーティションに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、db2inidb コマンドは各データベース・パーティション上でそれぞれ実行する必要があります。それから、それらのデータベース・パーティションのスプリット・イメージを使用することができます。db2inidb コマンドは、db2_all コマンドを使用してすべてのデータベース・パーティションで同時に実行することができます。しかし、RELOCATE USING オプションを使用する場合は、db2_all コマンドを使用して全データベース・パーティションに対して同時に db2inidb を実行することはできません。データベース・パーティションごとにそれぞれ別個の構成ファイル (変更するデータベース・パーティションの NODENUM 値が含まれる) を用意する必要があります。例えば、データベースの名前を変更する場合は、すべてのデータベース・パーティションが影響を受けることになり、各データベース・パーティションごとに別個の構成ファイルを用意して db2relocatedb コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナーを移動する場合は、そのデータベース・パーティションに対して一度だけ db2relocatedb コマンドを実行することが必要です。

注: スプリット・ミラーが、データベースを構成するすべてのコンテナーおよびディレクトリー (ボリューム・ディレクトリーを含む) を含んでいることを確認してください。この情報を収集するには、DBPATHS 管理ビューを参照してください。このビューは、分割する必要のあるデータベースのすべてのファイルとディレクトリーを表示します。

db2inidb - ミラーリングされたデータベースの初期化

スプリット・ミラー環境のミラーリングされたデータベースを初期化します。ミラーリングされたデータベースは、ロールフォワード・ペンディング状態にある 1 次データベースの複製として初期化したり、1 次データベースをリストアするためのバックアップ・イメージとして使用できます。このコマンドはスプリット・ミラー・データベースに対してのみ実行可能であり、スプリット・ミラーを使用するには、その前にこのコマンドを実行しておく必要があります。

許可

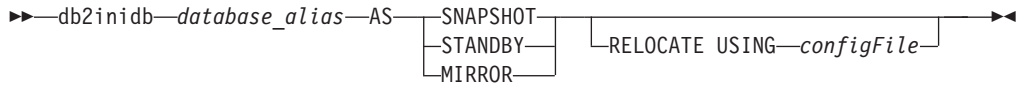
以下のいずれか。

- *sysadm*
- *sysctrl*
- *sysmaint*

必要な接続

なし

コマンド構文



コマンド・パラメーター

database_alias
初期設定するデータベースの別名を指定します。

SNAPSHOT
ミラーリングされたデータベースは、1 次データベースの複製として初期化されることを指定します。

STANDBY
データベースをロールフォワード・ペンディング状態にすることを指定します。 1 次データベースから新しいログ・ファイルをフェッチ、スタンバイ・データベースに適用することが可能です。スタンバイ・データベースは、1 次データベースがダウンした場合に、その代わりに使用できます。

MIRROR
ミラーリングされたデータベースを、1 次データベースをリストアするために使用できるバックアップ・イメージとして使用することを指定します。

RELOCATE USING *configFile*
データベースをスナップショット、スタンバイ、またはミラーとして初期化する前に、指定された *configFile* の中でリストされている情報に基づいて、データベース・ファイルを再配置することを指定します。 *configFile* の形式については、423 ページの『db2relocatedb - データベースの再配置』を参照してください。

使用上の注意

db2inidb *database_alias* as mirror コマンドを発行する前に `db2 connect to` *database-alias* 操作を発行しないでください。初期化する前にスプリット・ミラー・データベースに接続すると、ロールフォワード・リカバリーで必要なログ・ファイルが消去されてしまいます。その接続によって、データベースは、以前にそのデータベースを中断した時点の状態に戻ります。中断の時点でデータベースが整合とマーク付けされると、DB2 データベース・システムはクラッシュ・リカバリーの必要はないと判断して、将来の利用のためにログを空にします。ログが空になった場合、ロールフォワードしようとすると、SQL4970N エラー・メッセージが戻されます。

パーティション・データベース環境では、すべてのデータベース・パーティションに対して db2inidb を実行する必要があります。その後、任意のデータベース・パーティションのスプリット・ミラーを使用できるようになります。 db2_all コマンドを使用すれば、すべてのデータベース・パーティションに対して同時に db2inidb を実行することができます。

しかし、RELOCATE USING オプションを使用する場合は、 db2_all コマンドを使用して全パーティションに対して同時に db2inidb を実行することはできません。パーティションごとにそれぞれ別個の構成ファイル (変更対象のデータベース・パーティションの NODENUM 値が含まれる) を用意する必要があります。例えば、データベースの名前を変更する場合は、すべてのデータベース・パーティションが影響を受けることになり、各データベース・パーティションごとに別個の構成ファイルを用意して db2relocatedb コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナーを移動する場合は、そのデータベース・パーティションに対して一度だけ db2relocatedb コマンドを実行することが必要です。

RELOCATE USING *configFile* パラメーターが指定されており、データベースの再配置が正常に実行されたなら、指定された *configFile* はデータベース・ディレクトリーにコピーされ、その名前が `db2path.cfg` に変更されます。それ以降のクラッシュ・リカバリーまたはロールフォワード・リカバリーにおいて、このファイルは、ログ・ファイルの処理時にコンテナー・パスの名前を変更するために使用されます。

クローン・データベースを初期化している場合、指定された *configFile* は、クラッシュ・リカバリー完了後にデータベース・ディレクトリーから自動的に除去されます。

スタンバイ・データベースまたはミラーリングされたデータベースを初期化している場合、指定された *configFile* は、ロールフォワード・リカバリーの完了後またはキャンセル後に、データベース・ディレクトリーから自動的に除去されます。db2inidb 実行後には、新しいコンテナー・パスを `db2path.cfg` ファイルに追加できます。元のデータベースに対して CREATE 操作または ALTER TABLESPACE 操作を実行し、スタンバイ・データベース上で異なるパスを使用しなければならない場合には、このことが必要になります。

db2relocatedb - データベースの再配置

このコマンドは、ユーザー提供の構成ファイルで指定されたとおりに、データベースを名前変更したり、データベースやデータベースの一部 (コンテナー、ログ・ディレクトリーなど) を再配置します。このツールは、DB2 インスタンスおよびデータベース・サポート・ファイルに、必要な変更を行います。

許可

なし

コマンド構文

コマンド・パラメーター

-f *configFilename*

データベースの再配置に必要な構成情報の入ったファイルの名前を指定します。これは、相対ファイル名でも絶対ファイル名でも構いません。構成ファイルのフォーマットは以下のとおりです。

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
...
STORAGE_PATH=oldStoragePath1,newStoragePath1
STORAGE_PATH=oldStoragePath2,newStoragePath2
...
```

ここで、

DB_NAME

再配置されるデータベースの名前を指定します。データベース名を変更する場合は、古い名前と新規の名前の両方を指定する必要があります。このフィールドは必須です。

DB_PATH

再配置されるデータベースの元のパスを指定します。データベース・パスが変更される場合、古いパスと新規のパスの両方を指定する必要があります。このフィールドは必須です。

INSTANCE

データベースが存在する場所のインスタンスを指定します。データベースが新規のインスタンスに移動される場合、古いインスタンスと新規のインスタンスの両方を指定する必要があります。このフィールドは必須です。

NODENUM

変更されるデータベース・ノードのノード番号を指定します。デフォルトは 0 です。

LOG_DIR
ログ・パスのロケーション内の変更を指定します。ログ・パスが変更される場合、古いパスと新しいパスの両方を指定する必要があります。ログ・パスがデータベース・パスの下にある場合、パスは自動的に更新されるので、この指定はオプションです。

CONT_PATH
表スペース・コンテナーのロケーション内の変更を指定します。古いコンテナー・パスと新規のコンテナー・パスの両方を指定する必要があります。複数のコンテナー・パスを変更する場合、複数の CONT_PATH 行を指定できます。コンテナー・パスがデータベース・パスの下にある場合、パスは自動的に更新されるので、この指定はオプションです。同じ古いパスが共通の新規パスで置換される場所で、複数のコンテナーに変更を行う場合、単一の CONT_PATH 項目が使用されます。このような場合、古いパス、新規パスの両方にアスタリスク (*) をワイルドカードとして使用できます。

STORAGE_PATH
これは、自動ストレージが有効になっているデータベースにのみ該当します。データベースのいずれかのストレージ・パスの場所を変更することを指定します。古いストレージ・パスと新しいストレージ・パスの両方を指定する必要があります。複数のストレージ・パスを変更する場合、複数の STORAGE_PATH 行を指定できます。

ブランク行またはコメント文字 (#) で始まる行は無視されます。

例

例 1

データベース TESTDB の名前を PRODDB に、パス /home/db2inst1 にあるインスタンス db2inst1 で変更するには、以下の構成ファイルを作成します。

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

構成ファイルを relocate.cfg として保管し、以下のコマンドを使用して、データベース・ファイルへの変更を行います。

```
db2relocatedb -f relocate.cfg
```

例 2

データベース DATAB1 をパス /dbpath のインスタンス jsmith からインスタンス prodinst に移動するには、以下のようにします。

1. ディレクトリー /dbpath/jsmith 内のファイルを /dbpath/prodinst に移動します。
2. 以下の構成ファイルと db2relocatedb コマンドを使用して、データベース・ファイルに変更を行います。

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

例 3

パス /databases/PRODDB のインスタンス inst1 内に存在するデータベース PRODDB です。 2 つの表スペース・コンテナーのロケーションを、以下のように変更する必要があります。

- SMS コンテナー /data/SMS1 を /DATA/NewSMS1 に移動する必要があります。
- DMS コンテナー /data/DMS1 を /DATA/DMS1 に移動する必要があります。

物理ディレクトリーおよびファイルが、新規のロケーションに移動された後で、新規のロケーションを認識するように、以下の構成ファイルと db2relocatedb コマンドを使用して、データベース・ファイルに変更を行います。

```
DB_NAME=PRODDB
DB_PATH=/databases/PRODDB
INSTANCE=inst1
NODENUM=0
CONT_PATH=/data/SMS1,/DATA/NewSMS1
CONT_PATH=/data/DMS1,/DATA/DMS1
```

例 4

インスタンス db2inst1 に存在するデータベース TESTDB は、パス /databases/TESTDB に作成されました。表スペースは、以下のコンテナーと共に作成されました。

```
TS1
TS2_Cont0
TS2_Cont1
/databases/TESTDB/TS3_Cont0
/databases/TESTDB/TS4/Cont0
/Data/TS5_Cont0
/dev/rTS5_Cont1
```

TESTDB は新規システムに移動されます。新規システムのインスタンスは newinst になり、データベースのロケーションは /DB2 になります。

データベースを移動する場合、/databases/TESTDB/db2inst1 ディレクトリーに存在するすべてのファイルは、/DB2/newinst ディレクトリーに移動する必要があります。これは、最初の 5 つのコンテナーが、この移動の一部として再配置されることを意味します。(最初の 3 つはデータベース・ディレクトリーに相対で、次の 2 つはデータベース・パスに相対です。) これらのコンテナーがデータベース・ディレクトリーまたはデータベース・パス内にあるため、構成ファイルにリストする必要はありません。 2 つの残りのコンテナーが新規システムで異なるロケーションに移動された場合は、構成ファイルにリストする必要があります。

物理ディレクトリーおよびファイルが新規のロケーションに移動された後で、新規のロケーションを認識するように、以下の構成ファイルと db2relocatedb を使用して、データベース・ファイルに変更を行います。

```
DB_NAME=TESTDB
DB_PATH=/databases/TESTDB,/DB2
INSTANCE=db2inst1,newinst
```

```
NODENUM=0
CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

例 5

データベース TESTDB には、データベース・パーティション・サーバー 10 および 20 に 2 つのデータベース・パーティションがあります。このインスタンスは servinst で、データベース・パスは両方のデータベース・パーティション・サーバーで /home/servinst です。データベースの名前は SERVDB に変更され、データベース・パスは両方のデータベース・パーティション・サーバーで /databases に変更されます。さらに、ログ・ディレクトリーはデータベース・パーティション・サーバー 20 で、 /testdb_logdir から /servdb_logdir に変更されます。

両方のデータベース・パーティションに変更が行われているため、構成ファイルは各データベース・パーティションに作成され、db2relocatedb は対応する構成ファイルを使用する各データベース・パーティション・サーバーで実行される必要があります。

データベース・パーティション・サーバー 10 では、以下の構成ファイルが使用されます。

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=10
```

データベース・パーティション・サーバー 20 では、以下の構成ファイルが使用されます。

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=20
LOG_DIR=/testdb_logdir,/servdb_logdir
```

例 6

パス /home/maininst のインスタンス maininst 内に存在するデータベース MAINDB です。 4 つの表スペース・コンテナーのロケーションを、以下のように変更する必要があります。

```
/maininst_files/allconts/C0 needs to be moved to /MAINDB/C0
/maininst_files/allconts/C1 needs to be moved to /MAINDB/C1
/maininst_files/allconts/C2 needs to be moved to /MAINDB/C2
/maininst_files/allconts/C3 needs to be moved to /MAINDB/C3
```

物理ディレクトリーおよびファイルが、新規のロケーションに移動された後で、新規のロケーションを認識するように、以下の構成ファイルと db2relocatedb コマンドを使用して、データベース・ファイルに変更を行います。

同様の変更が、すべてのコンテナーに対して行われました。すなわち、/maininst_files/allconts/ が /MAINDB/ で置換され、ワイルドカード文字のある単一項目が使用できるようになります。

```
DB_NAME=MAINDB
DB_PATH=/home/maininst
INSTANCE=maininst
NODE_NUM=0
CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

使用上の注意

データベースが属するインスタンスを変更する場合、インスタンスおよびデータベース・サポート・ファイルに確実に変更が加えられるようにするため、このコマンドを実行する前に以下の事柄を行う必要があります。

- データベースが他のインスタンスに移動されている場合は、新規のインスタンスを作成します。
- 新規インスタンスが常駐するシステムにコピーされるデータベースに属するファイルとデバイスをコピーします。パス名は必要に応じて変更する必要があります。ただし、データベース・ファイルの移動先のディレクトリー内にデータベースが既にある場合、不用意に既存の sqldbdir ファイルを上書きしてしまって、既存のデータベースへの参照を除去する可能性があります。そのような事態になった場合、 db2relocatedb ユーティリティーを使用することはできません。その場合、db2relocatedb の代わりにリダイレクト・リストア操作を使用できます。
- インスタンス所有者に所有されるように、コピーされたファイル/デバイスのアクセス権を変更します。

インスタンスが変更されている場合、ツールは新規のインスタンス所有者によって実行される必要があります。

パーティション・データベース環境では、変更が必要なすべてのデータベース・パーティションに対してこのツールを実行する必要があります。データベース・パーティションごとにそれぞれ別個の構成ファイル (変更対象のデータベース・パーティションの NODENUM 値が含まれる) を用意する必要があります。例えば、データベースの名前を変更する場合は、すべてのデータベース・パーティションが影響を受けることになり、各データベース・パーティションごとに別個の構成ファイルを用意して db2relocatedb コマンドを実行する必要があります。単一データベース・パーティションに属するコンテナーを移動する場合は、そのデータベース・パーティションに対して一度だけ db2relocatedb コマンドを実行することが必要です。

db2relocatedb コマンドを使用して、ロードが進行中のデータベースや、LOAD RESTART または LOAD TERMINATE コマンドの完了を待機しているデータベースを再配置することはできません。

制約事項: パーティション・データベース環境では、同じ装置に常駐する複数のロジカル・パーティションの 1 つであるノードの全体を再配置することはできません。

db2look - DB2 統計および DDL 抽出ツール

必要とされるデータ定義言語 (DDL) ステートメントを抽出して、テスト・データベース上に実働データベースのデータベース・オブジェクトを再生成します。 db2look コマンドは、オブジェクト・タイプごとに DDL ステートメントを生成します。

このツールは、テスト・データベース内のオブジェクトに関する統計を複製するために必要な UPDATE ステートメントを生成できます。また、このツールを使用して、UPDATE DATABASE CONFIGURATION および UPDATE DATABASE MANAGER CONFIGURATION コマンドおよび db2set コマンドを生成し、テスト・データベース上の照会オプティマイザー関連の構成パラメーターとレジストリー変数を、実働データベースでの値に合わせて設定できます。

テスト・システムに実動システムのデータのサブセットを含めておくと、便利なことが多くあります。しかし、そのようなテスト・システム用に選択したアクセス・プランが、必ずしも実動システム用に選択したアクセス・プランと同じであるとは限りません。テスト・システム用のカタログ統計と構成パラメーターの両方が、実動システムのものと一致するように更新されていなければなりません。このツールを使用すると、アクセス・プランが、実動システムで使用するものと類似しているテスト・データベースを作成することが可能になります。

db2look コマンドによって生成される DDL は元の SQL オブジェクトのすべての特性を複製するとは限らないため、生成された DDL ステートメントを必ず確認するようにしてください。パーティション・データベース環境の表スペースでは、アクティブでないデータベース・パーティションが存在する場合、DDL が完全でない可能性があります。 ACTIVATE コマンドを使用することによって、すべてのデータベース・パーティションがアクティブであることを確認してください。

許可

システム・カタログ表に対する SELECT 特権。

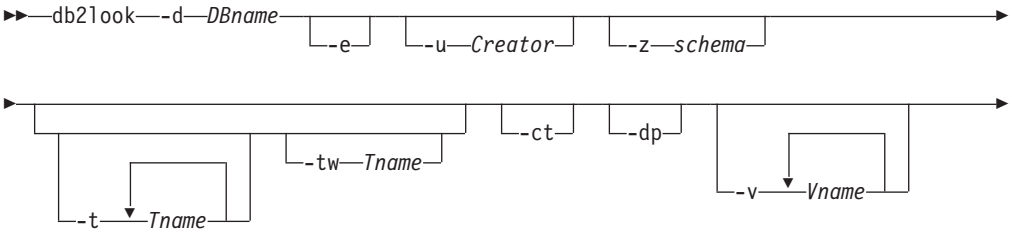
表スペース・コンテナー DDL を生成する場合 (つまり、sqlbotcq、sqlbftcq、および sqlbctcq の API を呼び出す場合) などでは、以下のいずれか 1 つが必要です。

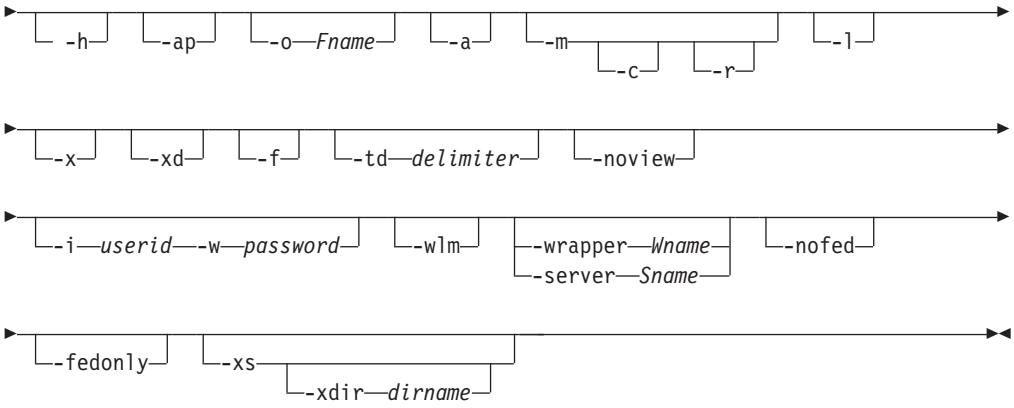
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

必要な接続

なし

コマンド構文





コマンド・パラメーター

-d *DBname*

照会する実動データベースの別名。 *DBname* としては、 DB2 Database for Linux, UNIX, and Windows または DB2 Version 9.1 for z/OS (DB2 for z/OS) データベースの名前を指定できます。*DBname* が DB2 for z/OS データベースの場合には、db2look ユーティリティーは、OS/390 オブジェクトおよび z/OS オブジェクト用の DDL および UPDATE 統計ステートメントを抽出します。これらの DDL および UPDATE 統計ステートメントは、DB2 Database for Linux, UNIX, and Windows データベースには適用できますが、 DB2 for z/OS データベースには適用できません。これは、OS/390 オブジェクトおよび z/OS オブジェクトを抽出して、それらを DB2 Database for Linux, UNIX, and Windows データベースで再作成しようとするユーザーに役立ちます。

DBname が DB2 for z/OS データベースの場合、db2look コマンドの出力は以下のものに制限されます。

- 表、索引、ビュー、およびユーザー定義特殊タイプ用の DDL の生成
- 表、列、列分散および索引用の UPDATE 統計ステートメントの生成

-e データベース・オブジェクト用の DDL ステートメントを抽出します。 -e オプションを使用する場合には、以下のデータベース・オブジェクト用の DDL を抽出します。

- 監査ポリシー
- スキーマ
- 表
- ビュー
- マテリアライズ照会表 (MQT)
- 別名
- 索引
- トリガー
- シーケンス
- ユーザー定義特殊タイプ
- 主キー、参照整合性、およびチェック制約

- ユーザー定義の構造化タイプ
- ユーザー定義関数
- ユーザー定義メソッド
- ユーザー定義トランスフォーム
- ラッパー
- サーバー
- ユーザー・マッピング
- ニックネーム
- タイプ・マッピング
- 関数テンプレート
- 関数マッピング
- SPECIFICATION ONLY 指定の索引
- ストアード・プロシージャー
- ロール
- トラステッド・コンテキスト
- グローバル変数
- 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
- セキュリティー・ラベル

db2look コマンドによって生成される DDL を使用して、ユーザー定義関数を正常に再作成することができます。ただし、ユーザー定義関数を使用できる状態にするには、特定のユーザー定義関数 (EXTERNAL NAME 節など) が参照するユーザー・ソース・コードが使用できる状態でなければなりません。

-u *Creator*

作成者 ID。出力をこの作成者 ID があるオブジェクトだけに制限します。オプション -a を指定した場合、このパラメーターは無視されます。出力に作動不能オブジェクトは含まれません。作動不能オブジェクトを表示するには、-a オプションを使用します。

-z *schema*

スキーマ名。出力をこのスキーマ名のオブジェクトに制限します。出力に作動不能オブジェクトは含まれません。作動不能オブジェクトを表示するには、-a オプションを使用します。このパラメーターが指定されない場合は、すべてのスキーマ名のオブジェクトが抽出されます。 -a オプションを指定した場合、このパラメーターは無視されます。このオプションは、フェデレーテッド DDL では無視されます。

-t *Tname1 Tname2 ... TnameN*

表名のリストです。表のリストにある特定の表への出力を制限します。表の最大数は 30 です。表名はブランク・スペースで区切られます。大文字と小文字を区別する名前および 2 バイト文字セット (DBCS) 名は、 ¥″ MyTabLe ¥″ のように、円記号と二重引用符の区切り文字で囲む必要があります。複数語表名の場合、区切り文字は (″¥″My Table¥″″ のように) 引用符の中に置く必要があり、そのようにしてその対が一語ごとにコマンド行プロ

セッサーで評価されないようにします。複数語表名が (″My Table″ のように) 円記号と二重引用符で囲まれていない場合、すべての語は大文字に変換され、db2look コマンドは (″MY TABLE″ のように) 大文字の表を探します。 -t が -l と共に使用された場合、その組み合わせでは DB2 バージョン 9.5 のパーティション表がサポートされます。

-tw *Tname*
Tname に指定したパターン基準に一致する表名の DDL を生成します。また、戻されたすべての表のすべての従属オブジェクトの DDL も生成します。 *Tname* は、1 つの値だけでもかまいません。 *Tname* 内の下線文字 (_) は、任意の 1 文字を表します。パーセント記号 (%) は、ゼロ個以上の文字のストリングを表します。*Tname* 内の他のすべての文字は、その文字そのものを表します。 -tw を指定した場合は、-t オプションが無視されます。

-ct オブジェクト作成時刻に基づいて DDL を生成します。オブジェクト作成時刻に基づいて DDL を生成した場合、オブジェクト DDL のすべてが従属関係の点で正しい順序で表示されることが保証されません。 db2look コマンドで、-ct オプションと共に指定することがサポートされているオプションは、-e、-a、-u、-z、-t、-tw、-v、-l、-noview、-wlm だけです。

-dp CREATE ステートメントの前に DROP ステートメントを生成します。ドロップされるオブジェクトに依存するオブジェクトが存在する場合、その DROP ステートメントは機能しない可能性があります。例えば、スキーマをドロップする場合、そのスキーマに依存する表が存在するなら、そのドロップは失敗します。あるいは、ユーザー定義のタイプ/関数をドロップする場合、それに依存する他のタイプ、関数、トリガー、または表が存在するなら、そのドロップは失敗します。型付き表の場合、DROP TABLE HIERARCHY ステートメントはルート表についてのみ生成されます。表がドロップされると索引、主キーと外部キー、および制約も常にドロップされるため、それらについての DROP ステートメントは生成されません。表に RESTRICT ON DROP 属性が設定されている場合、それはドロップできません。

-v *Vname1 Vname2 ... VnameN*
指定したビューの DDL を生成します。ビューの最大数は 30 です。 -t オプションを指定した場合は、-v オプションが無視されます。大/小文字の区別、DBCS、および複数語表名を制御する規則は、ビュー名にも適用されます。

-h ヘルプ情報を表示します。 このオプションを指定すると、他のすべてのオプションは無視され、ヘルプ情報だけが表示されます。

-ap 必須の AUDIT USING ステートメントを生成し、監査ポリシーを他のデータベース・オブジェクトと関連付けます。

-o *Fname*
出力を *filename*.sql に書き込みます。このオプションを指定しない場合、出力は標準出力に書き込まれます。ファイル名が拡張子付きで指定されている場合、出力はそのファイルに書き込まれます。

-a このオプションが指定されている場合には、特定の作成者 ID で作成されたオブジェクトだけに出力が制限されることはありません。すべてのユーザーによって作成されたすべてのオブジェクト (作動不能オブジェクトを含む)

が対象になります。例えば、このオプションと -e オプションが共に指定される場合、データベース内のすべてのオブジェクト用の DDL ステートメントが抽出されます。このオプションと -m オプションが共に指定される場合、データベース内のすべてのユーザー作成表および索引用の UPDATE 統計ステートメントが抽出されます。 -u と -a のどちらも指定しない場合には、環境変数 USER が使用されます。 UNIX オペレーティング・システムでは、この変数を明示的に設定する必要はありません。しかし Windows システムの場合、USER 環境変数にデフォルト値がありません。SYSTEM 変数の中のユーザー変数を設定するか、または set USER=*username* をセッションに発行する必要があります。

-m 必要な UPDATE ステートメントを生成して、表、統計ビュー、列、および索引についての統計を複製します。

-c このオプションを -m オプションと共に指定する場合、db2look コマンドは COMMIT、CONNECT、および CONNECT RESET ステートメントを生成しません。デフォルト・アクションでは、これらのステートメントを生成します。

-r このオプションと -m オプションを共に指定する場合、db2look コマンドは RUNSTATS コマンドを生成しません。デフォルト・アクションでは、RUNSTATS コマンドを生成します。

注: 模倣モード (-m オプション) で db2look を使用して作成されたコマンド・プロセッサー・スクリプトを他のデータベースに対して実行しようとする場合 (例えば、テスト・データベースのカタログ統計を実動データベースと一致させる場合)、データベースは両方とも同じコード・セットおよびテリトリーを使用していなければなりません。

-l このオプションを指定すると、db2look コマンドは、ユーザー定義の表スペース、データベース・パーティション・グループ、およびバッファー・プール用の DDL を生成します。以下のデータベース・オブジェクト用の DDL は、 -l オプションを使用すると抽出されます。

- ユーザー定義表スペース
- ユーザー定義データベース・パーティション・グループ
- ユーザー定義バッファー・プール

-x このオプションを指定すると、db2look コマンドは、許可 DDL (GRANT ステートメントなど) を生成します。

サポートされている許可には、以下のものが含まれます。

- 表: ALTER、SELECT、INSERT、DELETE、UPDATE、INDEX、REFERENCE、CONTROL
- ビュー: SELECT、INSERT、DELETE、UPDATE、CONTROL
- 索引: CONTROL
- スキーマ: CREATEIN、DROPIN、ALTERIN
- データベース: CREATEDB、BINDADD、CONNECT、CREWATE_NOT_FENCED、IMPLICIT_SCHEMA
- ユーザー定義関数 (UDF): EXECUTE
- ユーザー定義メソッド: EXECUTE

- ストアード・プロシージャー: EXECUTE
- パッケージ: CONTROL、BIND、EXECUTE
- 列: UPDATE、REFERENCES
- 表スペース: USE
- シーケンス: USAGE、ALTER
- ワークロード: USAGE
- グローバル変数
- ロール
- セキュリティー・ラベル
- 免除

-xd このオプションを指定すると、オブジェクトの作成時に SYSIBM によって権限を付与されたオブジェクトの権限 DDL を含むすべての権限 DDL が、db2look コマンドによって生成されます。

-f このオプションを使用して、照会オプティマイザーに影響を与える構成パラメーターおよびレジストリー変数を抽出します。

-td *delimiter*
db2look コマンドによって生成される SQL ステートメントのステートメント区切り文字を指定します。このオプションが指定されていない場合のデフォルトはセミコロン (;) です。このオプションは、-e オプションを指定した場合に使用することをお勧めします。 この場合、抽出されたオブジェクトにはトリガーまたは SQL ルーチンが含まれる可能性があります。

-noview
このオプションを指定すると、CREATE VIEW DDL ステートメントが抽出されません。

-i *userid*
リモート・データベースで作業する場合には、このオプションを使用してください。

-w *password*
-i オプションと共にこのパラメーターを使用すると、ユーザーは、リモート・システムにあるデータベースに対して db2look コマンドを実行できるようになります。 db2look コマンドでは、リモート・システムにログオンするために、ユーザー ID およびパスワードが使用されます。リモート・データベースで作業を行っている場合は、リモート・データベースは、ローカル・データベースと同じバージョンである必要があります。 db2look コマンドには、下位レベルまたは上位レベル・サポートはありません。

-wlm このオプションは、WLM 固有の DDL 出力を生成します。この出力は、以下のものに関する CREATE および ALTER ステートメントの生成に使用できます。

- ヒストグラム
- WLM イベント・モニター
- サービス・クラス
- ワークロード
- しきい値

- 作業クラス・セット
- 作業アクション・セット

-wrapper *Wname*
このラッパーに適用するフェデレーテッド・オブジェクト用の DDL ステートメントを生成します。生成される可能性のあるフェデレーテッド DDL ステートメントには、以下のものが含まれます: CREATE WRAPPER、CREATE SERVER、 CREATE USER MAPPING、 CREATE NICKNAME、CREATE TYPE MAPPING、 CREATE FUNCTION ... AS TEMPLATE、CREATE FUNCTION MAPPING、 CREATE INDEX SPECIFICATION、および GRANT (ニックネーム、サーバー、索引への特権)。 1 つのラッパー名のみがサポートされています。1 つも指定されない場合、または複数が指定された場合は、エラーが返されます。このオプションでは、非リレーショナル・データ・ソースはサポートされません。

-server *Sname*
このサーバーに適用するフェデレーテッド・オブジェクト用の DDL ステートメントを生成します。生成される可能性のあるフェデレーテッド DDL ステートメントには、以下のものが含まれます: CREATE WRAPPER、CREATE SERVER、 CREATE USER MAPPING、 CREATE NICKNAME、CREATE TYPE MAPPING、 CREATE FUNCTION ... AS TEMPLATE、CREATE FUNCTION MAPPING、 CREATE INDEX SPECIFICATION、および GRANT (ニックネーム、サーバー、索引への特権)。 1 つのサーバー名のみがサポートされています。1 つも指定されない場合、または複数が指定された場合は、エラーが返されます。このオプションでは、非リレーショナル・データ・ソースはサポートされません。

-nofed フェデレーテッド DDL ステートメントが生成されないことを指定します。このオプションが指定された場合、-wrapper および -server オプションは無視されます。

-fedonly
フェデレーテッド DDL ステートメントのみ生成されることを指定します。

-xs XML スキーマと DTD をターゲット・データベースに登録するために必要なすべてのファイルをエクスポートし、それらを登録するための該当するコマンドを生成します。エクスポートされる XSR オブジェクトのセットは、-u、-z、-a の各オプションによって制御されます。

-xdir *dirname*
エクスポートされた XML 関連ファイルを、指定されたパスに配置します。このオプションが指定されていない場合、XML 関連ファイルはすべて現行ディレクトリーにエクスポートされます。

例

- データベース DEPARTMENT でユーザー `walid` によって作成されたオブジェクト用の DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル `db2look.sql` に書き込みます。

```
db2look -d department -u walid -e -o db2look.sql
```

- ianhe というスキーマ名を持ち、データベース DEPARTMENT でユーザー walid によって作成されたオブジェクト用の DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -u walid -z ianhe -e -o db2look.sql
  ```

- UPDATE ステートメントを生成して、データベース DEPARTMENT でユーザー walid によって作成されたデータベース・オブジェクトの統計を複製します。出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -u walid -m -o db2look.sql
  ```

- ユーザー walid によって作成されたオブジェクト用の DDL ステートメントおよび UPDATE ステートメントの両方を生成して、同じユーザーによって作成されたデータベース・オブジェクトについての統計を複製します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -u walid -e -m -o db2look.sql
  ```

- データベース DEPARTMENT ですべてのユーザーによって作成されたオブジェクトの DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -a -e -o db2look.sql
  ```

- すべてのユーザー定義のデータベース・パーティション・グループ、バッファー・プール、および表スペース用の DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -l -o db2look.sql
  ```

- オプティマイザー関連のデータベースおよびデータベース・マネージャーの構成パラメーター用の UPDATE ステートメント、およびデータベース DEPARTMENT にあるオプティマイザー関連のレジストリー変数用の db2set ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -f -o db2look.sql
  ```

- データベース DEPARTMENT にあるすべてのオブジェクト用の DDL、データベース DEPARTMENT にあるすべての表および索引についての統計を複製するための UPDATE ステートメント、GRANT 許可ステートメント、オプティマイザー関連データベースおよびデータベース・マネージャー構成パラメーター用の UPDATE ステートメント、オプティマイザー関連レジストリー変数用の db2set ステートメント、およびデータベース DEPARTMENT にあるすべてのユーザー定義のデータベース・パーティション・グループ、バッファー・プール、および表スペース用の DDL を生成します。出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -a -e -m -l -x -f -o db2look.sql
  ```

- オリジナルの作成者によって作成されたオブジェクトも含む、データベース DEPARTMENT 内のすべてのオブジェクトのすべての許可 DDL ステートメントを生成します。 (この場合には、オブジェクトの作成時に SYSIBM によって権限が付与されました。) db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -xd -o db2look.sql
  ```

- データベース DEPARTMENT ですべてのユーザーによって作成されたオブジェクトの DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -a -e -td % -o db2look.sql
  ```

  出力は CLP によって読み取ることができます。

  ```
  db2 -td% -f db2look.sql
  ```

- データベース DEPARTMENT 内のオブジェクト用に、 CREATE VIEW ステートメントを除く DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -e -noview -o db2look.sql
  ```

- 指定した表に関連するデータベース DEPARTMENT 内のオブジェクト用に、 DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -e -t tab1 ¥"My TaBlE2¥" -o db2look.sql
  ```

- フェデレーテッド・データベース FEDDEPART にすべてのオブジェクト (フェデレーテッドおよび非フェデレーテッド) 用の DDL ステートメントを生成します。フェデレーテッド DDL ステートメントでは、指定されたラッパー FEDWRAP に適用されるもののみが生成されます。 db2look 出力が標準出力に書き込まれます。

  ```
  db2look -d feddepart -e -wrapper fedwrap
  ```

- 非フェデレーテッド DDL ステートメントのみを含むスクリプト・ファイルを生成します。以下のシステム・コマンドは、フェデレーテッド・データベース (FEDDEPART) に対して実行でき、フェデレーテッドではないデータベースの実行時に検出されたような出力を生成するだけです。 db2look 出力がファイル out.sql に書き込まれます。

  ```
  db2look -d feddepart -e -nofed -o out
  ```

- データベース DEPARTMENT の中でスキーマ名が walid であるオブジェクト用の DDL ステートメントを生成します。組み込み XML スキーマおよび DTD を登録するために必要なファイルは、現行ディレクトリーにエクスポートされます。 db2look の出力は、以下のようにしてファイル db2look.sql に書き込みます。

  ```
  db2look -d department -z walid -e -xs -o db2look.sql
  ```

- データベース DEPARTMENT ですべてのユーザーによって作成されたオブジェクトの DDL ステートメントを生成します。組み込み XML スキーマおよび DTD を登録するために必要なファイルは、ディレクトリー /home/ofer/ofer/ にエクスポートされます。 db2look 出力が標準出力に書き込まれます。

  ```
  db2look -d department -a -e -xs -xdir /home/ofer/ofer/
  ```

- データベース DEPARTMENT 中の WLM 固有の DDL を排他的に生成します。

  ```
  db2look -d department -wlm
  ```

  データベース DEPARTMENT 中のすべてのオブジェクトに関する DDL を生成します。

  ```
  db2look -d department -wlm -e -l
  ```

使用上の注意

Windows オペレーティング・システムにおいて、db2look コマンドは DB2 コマンド・ウィンドウから実行される必要があります。

既存のオプションのいくつかは、フェデレーテッド環境をサポートします。以下の db2look コマンド行オプションがフェデレーテッド環境で使用されます。

- -ap

  使用されると、AUDIT USING ステートメントが生成されます。

- -e

  使用されると、フェデレーテッド DDL ステートメントが生成されます。

- -x

  使用されると、フェデレーテッド・オブジェクトへの特権を付与するために、GRANT ステートメントが生成されます。

- -xd

  使用されると、システム付与の特権をフェデレーテッド・オブジェクトに追加するために、フェデレーテッド DDL ステートメントが生成されます。

- -f

  使用されると、フェデレーテッド関連情報がデータベース・マネージャー構成から抽出されます。

- -m

  使用されると、ニックネームの統計が抽出されます。

- -wlm

  使用されると、WLM 固有の DDL が出力されます。

フェデレーテッド・システムを使用する能力は、フェデレーテッド DDL ステートメントを作成するために、データベース・マネージャー構成で有効にされる必要があります。 db2look コマンドがスクリプト・ファイルを生成した後、そのスクリプトを実行する前に、**federated** 構成パラメーターを YES に設定する必要があります。

出力スクリプトを変更して、CREATE USER MAPPING ステートメントのリモート・パスワードを追加する必要があります。

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のに使用される、これらの CREATE SERVER ステートメントに、AUTHORIZATION および PASSWORD を追加して、db2look コマンド出力スクリプトを変更する必要があります。

-tw オプションの使用法は、次のとおりです。

- abc で始まる名前を持つ表に関連した、 DEPARTMENT データベース内のオブジェクトの DDL ステートメントを生成し、その出力を db2look.sql ファイルに送信するには、次のようにします。

```
db2look -d department -e -tw abc% -o db2look.sql
```

- 名前の 2 番目の文字が d である表に関連した、 DEPARTMENT データベース内のオブジェクトの DDL ステートメントを生成し、その出力を db2look.sql ファイルに送信するには、次のようにします。

  ```
  db2look -d department -e -tw _d% -o db2look.sql
  ```

- db2look コマンドは、LIKE 述部を使用して、どの表名が *Tname* 引数に指定されたパターンに一致するかを評価します。 LIKE 述部を使用する以上、_ 文字または % 文字のいずれかが表名の一部である場合には、_ または % のすぐ前に円記号 (¥) エスケープ文字を置かなければなりません。この場合、_ も % も、*Tname* 内でワイルドカード文字として使用することはできません。例えば、名前の最初でも最後でもない場所にパーセント (%) 記号を持つ表に関連した、DEPARTMENT データベース内のオブジェクトの DDL ステートメントを生成するには、次のようにします。

  ```
  db2look -d department -e -tw string¥%string
  ```

- 大/小文字の区別がある名前、DBCS 名、複数語表名および複数語ビュー名は、円記号および二重引用符の両方で囲む必要があります。以下に例を示します。

  ```
  ¥"My TabLe¥"
  ```

  マルチバイト文字セット (MBCS) 名または 2 バイト文字セット (DBCS) 名が円記号と二重引用符の区切り文字で囲まれていない場合、小文字と同じバイトの名前が含まれていると大文字に変換され、db2look は変換された名前を使用してデータベース・オブジェクトを探します。その結果、DDL ステートメントは抽出されません。

- -tw オプションは、-x オプション (GRANT 特権を生成する場合)、-m オプション (表統計および列統計を戻す場合)、および -l オプション (ユーザー定義表スペース、データベース・パーティション・グループ、およびバッファー・プールの DDL を生成する場合) とともに使用できます。 -t オプションを -tw オプションとともに指定すると、-t オプション (およびそれに関連した *Tname* 引数) が無視されます。
- フェデレーテッド・データ・ソース上、または DB2 Universal Database for z/OS and OS/390、 DB2 for i5/OS®、または DB2 Server for VSE & VM 上にある表 (およびそれらに関連したオブジェクト) の DDL を生成するために -tw オプションを使用することはできません。
- -tw オプションは、CLP でのみサポートされています。

データベース・パーティション・フィーチャーを使用するシステム上で DDL を要求する場合、非アクティブ・データベース・パーティション上に存在する表スペースに関しては、DDL の代わりに警告メッセージが表示されます。すべての表スペースについて確実に正しい DDL が生成されるようにするためには、すべてのデータベース・パーティションをアクティブにする必要があります。

タイプ配列のセキュリティー・ラベル・コンポーネントに関する DDL を抽出する際に、抽出された DDL により生成されるセキュリティー・ラベル・コンポーネントの内部表記 (つまり、配列中のエレメントのエンコード) が、db2look の抽出元のデータベース中のセキュリティー・ラベル・コンポーネントの内部表記と正確に一致しないことがあります。この種の不一致は、タイプ配列のセキュリティー・ラベル・コンポーネントに変更が加えられ、1 つ以上のエレメントが追加された場合に

発生することがあります。この場合、ある表から抽出され、db2look 出力から作成された別の表に移動するデータには対応するセキュリティー・ラベル値がないので、新しい表の保護が危険にさらされる可能性があります。

関連情報

ニックネーム列および索引の名前

移行のためのアプリケーションの変更

第 6 章 ファイル形式とデータ・タイプ

エクスポート/インポート/ロード・ユーティリティーのファイル形式

ここでは、DB2 のエクスポート、インポート、およびロード・ユーティリティーによってサポートされている 5 種類のオペレーティング・システム・ファイル・フォーマットについて説明します。

DEL 区切り付き ASCII。さまざまなデータベース・マネージャーおよびファイル・マネージャーの間でのデータ交換に使用されます。このデータ保管の一般的なアプローチでは、列値を分離するために特殊な区切り文字が使用されます。

ASC 区切りなし ASCII。位置合わせされた列データをもつフラット・テキスト・ファイルを作成する他のアプリケーションからのデータのインポートまたはロードに使用されます。

PC/IXF
PC バージョンの IXF (統合交換フォーマット)。データベース・マネージャー内でのデータ交換のために望ましい方式です。 PC/IXF は、内部表の外部表記の入ったデータベース表の構造化された記述です。

WSF ワークシート・フォーマット。Lotus 1-2-3 や Symphony などの製品とのデータ交換に使用されます。ロード・ユーティリティーでは、このファイル・フォーマットはサポートされません。

CURSOR
SQL 照会に対して宣言されるカーソル。このファイル・タイプは、ロード・ユーティリティーによってのみサポートされます。

DEL、WSF、または ASC データ・ファイル・フォーマットを使用する場合は、ファイルをインポートする前に、表 (その列名とデータ・タイプを含む) を定義してください。オペレーティング・システム・ファイルのフィールドのデータ・タイプは、データベース表内の対応するデータ・タイプに変換されます。インポート・ユーティリティーは、多少の非互換性問題があるデータを受け入れます。これには、埋め込みまたは切り捨ての可能性を伴って文字データをインポートする場合、および数値データをそれとは異なるタイプの数値フィールドにインポートする場合などがあります。

PC/IXF データ・ファイル・フォーマットを使用する場合、インポート操作の前に表が存在している必要はありません。ただし、ユーザー定義特殊タイプ (UDT) は、定義する必要があります。定義しなければ、未定義名エラー (SQL0204N) を受け取ることになります。同じように、PC/IXF データ・ファイル・フォーマットにエクスポートする場合は、UDT が出力ファイルに格納されます。

CURSOR ファイル・タイプを使用する際には、ロード操作を開始する前に、列名およびデータ・タイプなどで表を定義する必要があります。 SQL 照会の列タイプは、ターゲット表の対応する列タイプと互換性がなければなりません。ロード操作を開始する前に、指定されたカーソルをオープンする必要はありません。ロード・

ユーティリティーは、カーソルが行のフェッチに使用されていたかどうかに関係なく、指定されたカーソルに関連する照会の結果全体を処理します。

プラットフォーム間のデータの移動 - ファイル形式に関する考慮事項

プラットフォームを超えてデータをエクスポート、インポート、またはロードする場合、互換性は重要です。以下の部分では、異なるオペレーティング・システム間でデータを移動する際の PC/IXF、区切り付き ASCII (DEL)、および WSF ファイル・フォーマットの考慮事項について説明します。

PC/IXF ファイル・フォーマット

PC/IXF は、プラットフォーム間でデータを転送する際に望ましいファイル・フォーマットです。 PC/IXF を使用すると、ロード・ユーティリティーやインポート・ユーティリティーは、通常はマシンによって異なる数値データをマシンから独立した形で処理できます。例えば、Intel® とその他のハードウェア体系とでは、数値データの保管および処理の方法が異なります。

DB2 ファミリー全製品で PC/IXF ファイルの互換性を保つために、エクスポート・ユーティリティーは Intel フォーマットの数値データによるファイルを作成し、インポート・ユーティリティーはこのフォーマットでデータを受け入れることを想定します。

ハードウェア・プラットフォームによっては、エクスポートおよびインポート操作中に、DB2 製品はバイト反転を使用して数値を Intel フォーマットから非 Intel フォーマットに変換します。

DB2 データベースの UNIX ベースの実装は、エクスポート時に複数パーツの PC/IXF ファイルを作成しません。しかし、DB2 の作成した複数パーツの PC/IXF ファイルをインポートすることはできます。このタイプのファイルをインポートする場合は、すべての部分を同じディレクトリーに入れる必要があります。そうしない場合エラーが戻されます。

DB2 エクスポート・ユーティリティーの UNIX ベースの実装によって作成された単一パーツの PC/IXF ファイルは、Windows 用の DB2 データベースによってインポートできます。

区切り付き ASCII (DEL) ファイル・フォーマット

DEL ファイルは、それらが作成されたオペレーティング・システムによって異なっています。相違点は次のとおりです。

- 行区切り文字
  - UNIX ベースのテキスト・ファイルでは、改行 (LF) 文字を使用します。
  - 非 UNIX ベースのテキスト・ファイルでは、復帰/改行 (CRLF) シーケンスを使用します。
- ファイル終了文字
  - UNIX ベースのテキスト・ファイルでは、ファイル終了文字を使用しません。

– 非 UNIX ベースのテキスト・ファイルでは、ファイル終了文字 (X'1A') を使用します。

DEL エクスポート・ファイルはテキスト・ファイルなので、あるオペレーティング・システムから別のオペレーティング・システムに転送できます。テキスト・モードでファイルを転送する場合、ファイル転送プログラムを使用すると、オペレーティング・システムによる違いを処理できます。バイナリー・モードの場合、行区切り文字やファイル終了文字の変換は実行されません。

注: 文字データ・フィールドに行区切り文字が入っていれば、それらもまたファイル転送中に変換されます。そのような変換によって想定外のデータの変更が発生するため、プラットフォームを超えてデータを移動する場合は DEL エクスポート・ファイルを使わないことをお勧めします。その代わりに、PC/IXF ファイル・フォーマットを使用してください。

WSF ファイル・フォーマット

WSF フォーマット・ファイルの数値データは、Intel マシン・フォーマットを使って保管されます。このフォーマットでは、異なる Lotus オペレーティング環境 (例えば Intel ベースと UNIX ベースのシステム) で Lotus WSF ファイルを転送および使用できます。

このように内部フォーマットに整合性があるため、 DB2 製品からエクスポートした WSF ファイルを、別のプラットフォームで実行している Lotus 1-2-3 または Lotus Symphony で使用することができます。また、DB2 製品は別のプラットフォームで作成された WSF をインポートできます。

オペレーティング・システム間の WSF ファイルの転送は、テキスト・モードではなくバイナリー・モードで実行してください。

注: 異なるプラットフォームの DB2 データベース間のデータ転送には WSF ファイル・フォーマットを使わないでください。そのようにすると、データが失われる可能性があります。その代わりに、PC/IXF ファイル・フォーマットを使用してください。

区切り付き ASCII (DEL) ファイル・フォーマット

区切り付き ASCII (DEL) ファイルは、行および列の区切り文字を使った順次 ASCII ファイルです。各 DEL ファイルは、行と列によって配列されたセル値から構成される ASCII 文字のストリームです。データ・ストリーム内の行は行区切り文字によって区切られ、それぞれの行の中の個々のセル値は列区切り文字によって区切られます。

以下の表は、インポートしたり、またはエクスポート・アクションの結果として生成したりできる DEL ファイルのフォーマットを示します。

```
DEL ファイル ::= 行 1 のデータ || 行区切り文字 ||
                 行 2 のデータ || 行区切り文字 ||
                 .
                 .
                 .
                 行 n のデータ || オプションの行区切り文字
```

```
行 i のデータ ::= セル値 (i,1) || 列区切り文字 ||
                  セル値 (i,2) || 列区切り文字 ||
                  .
                  .
                  .
                  セル値 (i,m)

行区切り文字 ::= ASCII 改行シーケンス[a]

列区切り文字 ::= デフォルト値 ASCII コンマ (,)[b]

セル値 (i,j) ::= 先行スペース
                 || 数値の ASCII 表記
                    (整数、10 進数、または浮動小数点数)
                 || 区切り文字ストリング
                 || 非区切り文字ストリング
                 || 後続スペース

非区切り文字ストリング ::= 行区切りまたは列区切り文字
                                以外の文字の集合

区切り文字ストリング ::= 文字ストリング区切り文字          ||
                          外字ストリング                   ||
                          文字ストリング区切り文字 ||
                          後続ガーベッジ

後続ガーベッジ ::= 行区切り文字または列区切り文字以外の
                     文字の集合

文字ストリング区切り文字 ::= デフォルト値 ASCII 二重引用符記号
                              (")[c]

外字ストリング ::= || NODOUBLEDEL 修飾子が指定されている場合、
                      行区切り文字または文字ストリング区切り文字以外の
                      文字の集合
                   || 文字ストリングが 2 つの連続した
                      文字ストリング区切り文字の一部ではない場合、
                      行区切り文字または文字ストリング区切り文字
                      以外の文字の集合

                   || 文字ストリング区切り文字が 2 つの連続した
                      文字ストリング区切り文字の一部ではなく、
                      かつ DELPRIORITYCHAR 修飾子が指定されて
                      いる場合、文字ストリング区切り文字以外の
                      文字の集合


ファイル終了 (EOF) 文字 ::= Hex '1A' (Windows オペレーティング・システムのみ)

数値の ASCII 表記 [d] ::= オプションの符号 '+' または '-'
       || 1 から 31 桁の 10 進数字に、オプションとして小数点をその前、その後、
          または数字と数字の間に入れたもの
       || オプションの指数

指数 ::= 文字 'E' または 'e'
       || オプションの符号 '+' または '-'
       || 1 から 3 桁の数字 (小数点なし)

10 進数字 ::= 文字 '0'、'1'、... '9' のいずれか

小数点 ::= デフォルト値 ASCII ピリオド (.)[e]
```

- [a] レコード区切り文字は、改行文字 (ASCII x0A) であると見なされます。 Windows オペレーティング・システムで生成されるデータでは、復帰/改行の 2 バイト標準 (0x0D0A) が使用できます。 EBCDIC コード・ページのデータで

は、レコード区切り文字として EBCDIC LF 文字 (0x25) を使用します (EBCDIC データは、LOAD コマンドの codepage ファイル・タイプ修飾子を使用してロードできます)。

- [b] 列区切り文字は、coldel ファイル・タイプ修飾子で指定できます。
- [c] 文字ストリング区切り文字は、chardel ファイル・タイプ修飾子で指定できます。

  注: 区切り文字のデフォルトの優先順位は、次のとおりです。

  1. レコード区切り文字
  2. 区切り文字
  3. 列区切り文字

- [d] 数値の ASCII 表記に指数が使用されている場合、それは FLOAT 定数です。小数点を使っているが指数は使っていない場合、それは DECIMAL 定数です。小数点も指数も使用されていない場合、それは INTEGER 定数です。
- [e] 小数点文字は、decpt ファイル・タイプ修飾子で指定できます。

エクスポート・ユーティリティーは、列データの中に埋め込まれたすべての文字ストリング区切り文字バイト (デフォルトは二重引用符または x22) を 2 つの文字ストリング区切り文字バイトに置き換えます (つまり、2 倍にします)。これは、インポート構文解析ルーチンが、列の開始または終了を定義する文字ストリング区切り文字バイトと、列データの中に埋め込まれた文字ストリング区切り文字バイトを区別できるようにするために行われます。エクスポート・ユーティリティー以外の何らかのアプリケーション用にエクスポートされた DEL ファイルを使用するときは注意を払い、この同じ文字ストリング区切り文字の倍増が、'FOR BIT' バイナリー列データの中で発生することに注意してください。

DEL のデータ・タイプの説明

表 50. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
BIGINT	-9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 の範囲の INTEGER 定数。	-9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 の範囲の数値の ASCII 表記。 10 進数および浮動小数点数は整数値に切り捨てられます。
BLOB、CLOB	区切り文字 (例えば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
BLOB_FILE、 CLOB_FILE	各 BLOB/CLOB 列ごとに文字データが個別のファイルに保管され、そのファイル名は区切り文字によって囲まれます。	データが入っているファイルの区切り付き名前または区切りなし名前。

表 50. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
CHAR	区切り文字 (例えば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、必要な場合データベース列の幅に合わせて切り捨てられるか、またはスペース (X'20') が埋められます。
DATE	区切り文字なしの *yyyymmdd* (年、月、日)。例えば、19931029。 あるいは、DATESISO オプションを使用して、すべての日付値が ISO フォーマットでエクスポートされるように指定することもできます。	ターゲット・データベースの地域別コードに一致する ISO フォーマットの日付値の入った区切り付きまたは区切りなし文字ストリング、あるいは *yyyymmdd* の形式の区切りなし文字ストリング。
DBCLOB (DBCS のみ)	GRAPHIC データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
DBCLOB_FILE (DBCS のみ)	各 DBCLOB 列ごとに文字データが個別のファイルに保管され、そのファイル名は区切り文字によって囲まれます。	データが入っているファイルの区切り付き名前または区切りなし名前。
DB2SECURITYLABEL	列のデータは、引用符 (") で囲まれて「生の」データとしてエクスポートされます。この値をセキュリティー・ラベル・ストリング・フォーマットに変換するには、 SELECT ステートメントの中で SECLABEL_TO_CHAR スカラー関数を使用します。	デフォルトでは、データ・ファイル内の値は、そのセキュリティー・ラベルの内部表記を構成する実際のバイト数と見なされます。値は引用符 (″″) で囲まれているものと想定されます。
DECIMAL	エクスポートされるフィールドの精度および位取りによる DECIMAL 定数。decplusblank ファイル・タイプ修飾子を使用して、正の 10 進値の前に正符号 (+) ではなくブランク・スペースを付けるように指定することも可能です。	フィールドのインポート先のデータベース列の範囲からあふれない数値の ASCII 表記。入力値の小数部の列数が、データベース列で受け入れ可能なものより多い場合、余分な列は切り捨てられます。
FLOAT(long)	-10E307 から 10E307 の範囲の FLOAT 定数。	-10E307 から 10E307 の範囲の数値の ASCII 表記。

表 50. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
GRAPHIC (DBCS のみ)	GRAPHIC データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、必要な場合データベース列の幅に合わせて切り捨てられるか、または 2 バイト・スペース (例えば X'8140') が埋められます。
INTEGER	-2,147,483,648 から 2,147,483,647 の範囲の INTEGER 定数。	-2,147,483,648 から 2,147,483,647 の範囲の数値の ASCII 表記。 10 進数および浮動小数点数は整数値に切り捨てられます。
LONG VARCHAR	区切り文字 (例えば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
LONG VARGRAPHIC (DBCS のみ)	GRAPHIC データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、データベースの列値として使用されます。
SMALLINT	-32,768 から 32,767 の範囲の INTEGER 定数。	-32,768 から 32,767 の範囲の数値の ASCII 表記。 10 進数および浮動小数点数は整数値に切り捨てられます。
TIME	*hh.mm.ss* (時、分、秒)。区切り文字によって囲まれた ISO フォーマットの時刻値。例えば「09.39.43」。	ターゲット・データベースの地域別コードに合致するフォーマットの時刻値を使った区切り付きまたは区切りなし文字ストリング。
TIMESTAMP	*yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒)。区切り文字によって囲まれた日時を表す文字ストリング。	データベースでの保管用に受け入れ可能なタイム・スタンプ値を使った区切り付きまたは区切りなし文字ストリング。
VARCHAR	区切り文字 (例えば二重引用符) によって囲まれた文字データ。	区切り文字ストリングまたは区切りなし文字ストリング。文字ストリングは、必要な場合データベース列の最大幅に合わせて切り捨てられます。

表 50. DEL ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
VARGRAPHIC (DBCS のみ)	GRAPHIC データは、区切り付き文字ストリングとしてエクスポートされます。	偶数バイトの長さの区切り付きまたは区切りなし文字ストリング。文字ストリングは、必要な場合データベース列の最大幅に合わせて切り捨てられます。

DEL ファイル例

DEL ファイルの例を下記に示します。各行は改行文字シーケンスで終わります (Windows オペレーティング・システムでは、各行は復帰/改行文字シーケンスで終わります)。

```
"Smith, Bob",4973,15.46
"Jones, Bill",12345,16.34
"Williams, Sam",452,193.78
```

次の例は、区切り文字なし文字ストリングの使用方法を示しています。文字データ中にコンマが使われているため、列区切り文字はセミコロンに変更されています。

```
Smith, Bob;4973;15.46
Jones, Bill;12345;16.34
Williams, Sam;452;193.78
```

注:

1. スペース (X'20') は有効な区切り文字ではありません。
2. セル値の最初の文字の前または最後の文字の後にあるスペースは、インポート時に破棄されます。セル値の途中にあるスペースは破棄されません。
3. ピリオド (.) は、タイム・スタンプ値のピリオドと競合するため、有効な文字ストリング区切り文字ではありません。
4. DBCS のみ (GRAPHIC)、混合 DBCS、および EUC の場合、区切り文字の範囲は x00 から x3F に制限されます。
5. EBCDIC コード・ページで指定された DEL データの場合、区切り文字は DBCS のシフトイン文字およびシフトアウト文字と同じであってはなりません。
6. Windows オペレーティング・システムの場合、区切り文字の中にない最初のファイル終了文字 (X'1A') がファイル終わりを示します。それ以降のデータはインポートされません。
7. NULL 値は、通常はセル値があるはずの場所にセル値がないことによって、あるいはスペースのストリングによって示されます。
8. 一部の製品では文字フィールドが 254 または 255 バイトに制限されるため、エクスポート・ユーティリティーは、最大長が 254 バイトより長い文字タイプの列がエクスポート用に選択されるたびに警告メッセージを生成します。インポート・ユーティリティーは、最も長い LONG VARCHAR および LONG VARGRAPHIC 列と同じ長さのフィールドを受け入れます。

データ移動時の区切り文字に関する考慮事項

区切り文字が使用されている ASCII (DEL) ファイルを移動する場合は、区切り文字の認識方法にかかわる問題のために、移動するデータをうっかり変更してしまわないようにする必要があります。このようなエラーを回避するために、DB2 には、いくつかの制約が設定されており、そのためのファイル・タイプ修飾子が用意されています。

区切り文字に関する制約事項

特定の区切り文字が、移動するデータの一部として処理される事態を回避するために、いくつかの制約が設定されています。第 1 に、区切り文字は、相互に排他的です。第 2 に、区切り文字として、2 進ゼロ、改行文字、復帰文字、ブランク・スペースを使用することはできません。さらに、デフォルトの小数点 (.) をストリング区切り文字として使用することはできません。最後に、DBCS 環境では、ストリング区切り文字としてパイプ (|) がサポートされていません。

ASCII 系のコード・ページと EBCDIC 系のコード・ページでは、以下の文字の指定方法が異なります。

- EBCDIC MBCS データ・ファイルでは、シフトイン (0x0F) 文字とシフトアウト (0x0E) 文字を区切り文字として使用できません。
- MBCS コード・ページ、EUC コード・ページ、DBCS コード・ページでは、0x40 より大きな文字を区切り文字として使用できません。ただし、EBCDIC MBCS データのデフォルトの小数点 (0x4b) は例外です。
- ASCII コード・ページまたは EBCDIC MBCS コード・ページのデータ・ファイルのデフォルトの区切り文字は、以下のとおりです。
  - ストリング区切り文字: "(0x22、二重引用符)
  - 列区切り文字: ,(0x2c、コンマ)
- EBCDIC SBCS コード・ページのデータ・ファイルのデフォルトの区切り文字は、以下のとおりです。
  - ストリング区切り文字: "(0x7F、二重引用符)
  - 列区切り文字: ,(0x6B、コンマ)
- ASCII データ・ファイルのデフォルトの小数点は、0x2e (ピリオド) です。
- EBCDIC データ・ファイルのデフォルトの小数点は、0x4B (ピリオド) です。
- サーバーのコード・ページとクライアントのコード・ページが異なる場合は、デフォルト以外の区切り文字の 16 進表記を指定することをお勧めします。例えば、次のようにします。

```
db2 load from ... modified by chardel0x0C coldelX1e ...
```

データ移動時の区切り文字に関する問題

二重のストリング区切り文字

DEL ファイルの文字ベース・フィールドの場合は、フィールド内にストリング区切り文字が見つかると、そのすべての出現箇所が二重のストリング区切り文字で表記される、というのがデフォルトの動作です。例えば、ストリング区切り文字が二重引用符の場合に、`I am 6" tall.` というテキストをエクスポートすると、DEL ファ

イルの出力テキストは、″I am 6″″ tall.″ になります。その逆に、DEL ファイルの入力テキストが ″What a ″″nice″″ day!″ になっていると、そのテキストは、What a ″nice″ day!

という形でインポートされます。**nodoubledel**
インポート/エクスポート/ロード・ユーティリティーで二重のストリング区切り文字の動作を無効にするには、nodoubledel ファイル・タイプ修飾子を指定します。ただし、二重のストリング区切り文字の動作が存在しているのは構文解析エラーを回避するためである、という点は忘れないようにする必要があります。エクスポートで nodoubledel を使用するときは、ストリング区切り文字が文字フィールドにあっても、そのストリング区切り文字は二重になりません。インポートとロードで nodoubledel を使用すると、二重のストリング区切り文字は、ストリング区切り文字のリテラル・インスタンスとして解釈されなくなります。

nochardel
エクスポートで nochardel ファイル・タイプ修飾子を使用すると、文字フィールドがストリング区切り文字で囲まれなくなります。インポートとロードで nochardel を使用すると、ストリング区切り文字は、特殊文字としてではなく実際のデータとして解釈されます。

chardel
その他のファイル・タイプ修飾子を使用して、デフォルトの区切り文字とデータの混同を手動で回避することも可能です。chardel ファイル・タイプ修飾子では、二重引用符 (デフォルト) の代わりに使用する文字ストリング区切り文字として、x という 1 文字を指定します。

coldel
同じように、列区切り文字としてデフォルトのコンマを使用したくない場合は、coldel を使用できます。この修飾子では、列データ区切り文字として、x という 1 文字を指定します。

delprioritychar
DEL ファイルの移動に伴うもう 1 つの問題は、区切り文字の正しい優先順位を維持することです。区切り文字のデフォルトの優先順位は、行、文字 (桁)、列です。ただし、中には、文字 (桁)、行、列という優先順位に依存するアプリケーションもあります。例えば、デフォルトの優先順位を使用する次のような DEL データ・ファイルがあるとします。

```
"Vincent <row delimiter> is a manager",<row delimiter>
```

このデータは、Vincent と is a manager という 2 つの行として解釈されます。行区切り文字 (<row delimiter>) の方がストリング区切り文字 (″) よりも優先順位が高いからです。一方、delprioritychar を使用すると、ストリング区切り文字 (″) の方が行区切り文字 (<row delimiter>) よりも優先順位が高くなり、同じ DEL ファイルでも、この場合は Vincent is a manager という 1 行として (正しく) 解釈されます。

区切りなし ASCII (ASC) ファイル形式

区切りなし ASCII フォーマット (インポートおよびロード・ユーティリティーでは ASC と言う) には、固定長 ASC と可変長 ASC の 2 つの種類があります。固定長

の ASC の場合、すべてのレコードが固定長になります。可変長の ASC の場合、レコードは行区切り文字 (常に改行) で区切られます。区切りなし ASCII における *区切りなし* という言葉は、列の値が区切り文字によって区切られないということを意味します。

ASC データのインポートまたはロード時に reclen ファイル・タイプ修飾子を指定すると、データ・ファイルは固定長の ASC になります。これを指定しないと、データ・ファイルは可変長の ASC になります。

区切りなし ASCII フォーマットは、ワード・プロセッサーを含め、縦欄フォーマットのデータを使用する ASCII 製品とのデータ交換に使用することができます。各 ASC ファイルは、行と列によって配列されたデータ値から構成される ASCII 文字のストリームです。データ・ストリーム内の行は行区切り文字によって区切られます。行内の各列は、開始/終了ロケーションの対 (IMPORT パラメーターで指定される) によって定義されます。それぞれの対は、バイト・ロケーションとして指定された行内のロケーションを表します。行内の最初の位置はバイト位置 1 です。それぞれの対の最初のエレメントは列の開始バイト、2 番目のエレメントは列の終了バイトです。列が互いに重なり合うことも可能です。 1 つの ASC ファイル内の各行の列定義はすべて同じです。

ASC ファイルの定義は、次のとおりです。

```
ASC ファイル ::= 行 1 のデータ || 行区切り文字 ||
                 行 2 のデータ || 行区切り文字 ||
                 .
                 .
                 .
                 行 n のデータ

行 i のデータ ::= ASCII 文字 || 行区切り文字

行区切り文字 ::= ASCII 改行シーケンス[a]
```

- [a] レコード区切り文字は、改行文字 (ASCII x0A) であると見なされます。Windows オペレーティング・システムで生成されるデータでは、復帰/改行の 2 バイト標準 (0x0D0A) が使用できます。 EBCDIC コード・ページのデータでは、レコード区切り文字として EBCDIC LF 文字 (0x25) を使用します (EBCDIC データは、LOAD コマンドの codepage ファイル・タイプ修飾子を使用してロードできます)。レコード区切り文字がデータのフィールドの一部として解釈されることはありません。

ASC のデータ・タイプの説明

表 51. ASC ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
BIGINT	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。 -9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 の範囲外の値があれば、その特定の値に関してはリジェクトされます。 10 進数は整数値に切り捨てられます。コンマ、ピリオド、またはコロンは小数点であると見なされます。 3 桁ごとの区切り文字は使用できません。 開始ロケーションと終了ロケーションは、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
BLOB/CLOB	文字のストリング。文字ストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられます。 ASC のブランク *切り捨て* オプションが有効な場合は、元のストリングまたは切り捨てられたストリングから後書きブランクが取り除かれます。
BLOB_FILE、 CLOB_FILE、 DBCLOB_FILE (DBCS のみ)	データが入っているファイルの区切り付き名前または区切りなし名前。
CHAR	文字のストリング。文字ストリングは、必要な場合ターゲット列の幅に合わせて切り捨てられるか、またはスペースが埋められます。
DATE	ターゲット・データベースの地域別コードに合致するフォーマットの日付値を表す文字ストリング。 開始ロケーションと終了ロケーションは、日付の外部表記の範囲内のフィールド幅になるように指定してください。
DBCLOB (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられます。
DECIMAL	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。インポート先のデータベース列の範囲外の値があれば、その特定の値に関してはリジェクトされます。入力値の小数部の桁が、データベース列の位取りより多い場合、余分な桁は切り捨てられます。コンマ、ピリオド、またはコロンは小数点であると見なされます。 3 桁ごとの区切り文字は使用できません。 開始ロケーションと終了ロケーションは、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。

表 51. ASC ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
FLOAT(long)	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。すべての値が有効です。コンマ、ピリオド、またはコロンは小数点であると見なされます。大文字または小文字の E は、FLOAT 定数の指数の始まりとして受け入れられます。 開始ロケーションと終了ロケーションは、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
GRAPHIC (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられるか、または 2 バイト・スペース (0x8140) が埋められます。
INTEGER	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。 -2,147,483,648 から 2,147,483,647 の範囲外の整数があれば、それに関してはリジェクトされます。 10 進数は整数値に切り捨てられます。コンマ、ピリオド、またはコロンは小数点であると見なされます。 3 桁ごとの区切り文字は使用できません。 開始ロケーションと終了ロケーションは、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
LONG VARCHAR	文字のストリング。文字ストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられます。 ASC のブランク *切り捨て* オプションが有効な場合は、元のストリングまたは切り捨てられたストリングから後書きブランクが取り除かれます。
LONG VARGRAPHIC (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられます。
SMALLINT	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の定数が受け入れられます。 -32,768 から 32,767 の範囲外の値があれば、その特定の値に関してはリジェクトされます。 10 進数は整数値に切り捨てられます。コンマ、ピリオド、またはコロンは小数点であると見なされます。 3 桁ごとの区切り文字は使用できません。 開始ロケーションと終了ロケーションは、幅が 50 バイト以下のフィールドになるように指定してください。整数、10 進数、および浮動小数点数の小数部は 31 桁以下です。浮動小数点数の指数は 3 桁以下です。
TIME	ターゲット・データベースの地域別コードに合致するフォーマットの時間値を表す文字ストリング。 開始ロケーションと終了ロケーションは、時間の外部表記の範囲内のフィールド幅になるように指定してください。

表 51. ASC ファイル・フォーマットで受け入れられるデータ・タイプのフォーマット (続き)

データ・タイプ	インポート・ユーティリティーにとって受け入れ可能な形式
TIMESTAMP	データベースでの保管用に受け入れ可能なタイム・スタンプ値を表す文字ストリング。 開始ロケーションと終了ロケーションは、タイム・スタンプの外部表記の範囲内のフィールド幅になるように指定してください。
VARCHAR	文字のストリング。文字ストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられます。 ASC のブランク切り捨て オプションが有効な場合は、元のストリングまたは切り捨てられたストリングから後書きブランクが取り除かれます。
VARGRAPHIC (DBCS のみ)	偶数バイトのストリング。奇数バイトのストリングは無効であり受け入れられません。有効なストリングは、必要な場合ターゲット列の最大長に合わせて右側が切り捨てられます。

ASC ファイル例

ASC ファイルの例を下記に示します。各行は改行文字シーケンスで終わります (Windows オペレーティング・システムでは、各行は復帰/改行文字シーケンスで終わります)。

```
Smith, Bob       4973     15.46
Jones, Suzanne   12345    16.34
Williams, Sam    452123  193.78
```

注:

1. ASC ファイルには、列名が入っていないものと見なされます。
2. 文字ストリングは、区切り文字によって囲まれません。 ASC ファイルの列のデータ・タイプは、データベース表のターゲット列のデータ・タイプによって決まります。
3. 次の場合、NULL 可能データベース列に NULL がインポートされます。
   - ブランクのフィールドのターゲットが数値、DATE、TIME、または TIMESTAMP タイプのデータベース列である場合
   - 開始/終了ロケーションの対のないフィールドが指定されている場合
   - 長さが 0 になる開始/終了ロケーションの対が指定されている場合
   - ある行のデータが短すぎて、ターゲット列にとって有効な値が入っていない場合
   - NULL INDICATORS ロード・オプションが使用されていて、NULL 標識列に N (またはユーザーによって指定されたその他の値) が検出された場合
4. NULL 可能でないターゲット列に数値、DATE、TIME、または TIMESTAMP 列にブランクのフィールドをインポートしようとすると、その行はリジェクトされます。
5. 入力データにターゲット列との互換性がなく、その列が NULL 可能でない場合は、エラーが検出された場所に応じて NULL がインポートされるか、または行がリジェクトされます。列が NULL 可能でない場合は、行がリジェクトされます。互換性がないことを示すメッセージがメッセージ・ファイルに書き込まれます。

PC バージョンの IXF ファイル形式

PC バージョンの IXF (PC/IXF) ファイル形式は、データベース・マネージャーに適応させた統合交換フォーマット (IXF) データ交換アーキテクチャーです。 IXF アーキテクチャーは、リレーショナル・データベースの構造とデータの交換を可能にするために特に設計されたものです。 PC/IXF アーキテクチャーを使用すれば、データベース・マネージャーは、データベースのエクスポート時に受信側の製品の要件や特性を予測する必要がなくなります。同じように、PC/IXF ファイルをインポートする側の製品で必要なことも、PC/IXF アーキテクチャーを理解するだけになります。ファイルをエクスポートした製品の特性は影響しなくなります。 PC/IXF ファイル・アーキテクチャーは、エクスポート側とインポート側の両方のデータベース・システムからの独立性を保ちます。

IXF アーキテクチャーは、特定のリレーショナル・データベース製品によってサポートされていない一部のタイプを含め、リレーショナル・データ・タイプの豊富なセットをサポートする汎用リレーショナル・データベース交換フォーマットです。PC/IXF ファイル・フォーマットではこの柔軟性が保たれます。例えば、PC/IXF アーキテクチャーでは、1 バイト文字セット (SBCS) と 2 バイト文字セット (DBCS) の両方のデータ・タイプがサポートされます。すべてのインプリメンテーションですべての PC/IXF データ・タイプがサポートされるわけではありませんが、制限付きのインプリメンテーションでも、インポート操作において、サポートされないデータ・タイプの検出と後処理が実行されます。

一般に PC/IXF ファイルは、中断なしの一連の可変長レコードから構成されます。ファイルには、次のレコードが入れられます。順序はここに示す順序です。

- レコード・タイプ H の 1 つのヘッダー・レコード
- レコード・タイプ T の 1 つの表レコード
- レコード・タイプ C の複数の列記述子レコード (表の列ごとに 1 つのレコード)
- レコード・タイプ D の複数のデータ・レコード (表の各行が 1 つまたは複数の D レコードによって表現されます)。

PC/IXF ファイルにおいて、H レコードの後であればどこにでもアプリケーション (A) レコードを入れることができます。 PC/IXF ファイルにおいてそれらのレコードは、PC/IXF フォーマットで定義されていない追加のデータをアプリケーションが PC/IXF ファイルに組み込むことができるようになっています。 PC/IXF ファイルを読み取るプログラムに、A レコード内のアプリケーション ID によって暗黙指定されるデータ・フォーマットと内容に関する特別の知識がない場合、そのレコードは無視されます。

PC/IXF ファイル内のレコードは、いずれもレコード長標識で始まります。これは、PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の、6 バイトの右寄せ文字表記です (合計レコード・サイズ - 6 バイト)。 PC/IXF ファイルを読むプログラムは、これらのレコード長を使用することにより、現行レコードの終わりと次のレコードの始まりを検出します。 H、T、および C レコードは、それらに定義されているすべてのフィールドを入れるのに十分な大きさでなければならず、当然のこととしてそれらのレコード長フィールドは、それらの実際の長さと一致していなければなりません。しかし、これらのいずれかのレコードの終わりに余分なデータ (例えば新しい フィールド) が追加された場合、

PC/IXF ファイルを読む従来のプログラムは余分のデータを無視し、警告メッセージしか生成しません。ただし、PC/IXF ファイルに書き込むプログラムの場合は、すべての定義済みフィールドを入れるのにちょうど必要な長さの H、T、および C レコードを書き込まなければなりません。

PC/IXF ファイルに LOB ロケーション指定子 (LLS) 列が入っている場合には、それぞれの LLS 列ごとに D レコードがなければなりません。 D レコードはエクスポート・ユーティリティーによって自動的に作成されますが、 PC/IXF ファイルを生成するためにサード・パーティーのツールを使用している場合には、これらを手動で作成する必要があります。さらに、NULL 値のある列を含め、表内の各 LOB 列ごとに LLS が必要です。 LOB 列が NULL である場合には、 NULL LOB を表す LLS を作成する必要があります。

各 XML 列の D レコード項目には、2 バイトのリトル・エンディアン (XML データ指定子 (XDS) の長さを示す) と、それに続く XDS そのものが入ります。

例えば、以下のような XDS があるとします。

```
XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

これは、次の D レコードのバイトによって表されます。

```
0x3D 0x00 XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

PC/IXF ファイルのレコードは、文字データの入ったフィールドで構成されます。インポートおよびエクスポート・ユーティリティーは、ターゲット・データベースの CPGID を使用してこの文字データを解釈します。ただし、2 つの例外があります。

- A レコードの IXFADATA フィールド。

  IXFADATA フィールド内に入れられる文字データのコード・ページ環境は、特定の A レコードを作成および処理するアプリケーションによって設定されます。つまり、環境は実装ごとに違います。
- D レコードの IXFDCOLS フィールド。

  IXFDCOLS フィールド内に入れられる文字データのコード・ページ環境は、特定の列とそのデータを定義する C レコードに入っている情報によって決まります。

H、T、および C レコード内の数値フィールド、そして D および A レコードの接頭部内の数値フィールドは、整数値の 1 バイト文字表記を右寄せして先行ゼロまたはブランクを埋め込んだものでなければなりません。値 0 は、ブランクではなく、少なくとも 1 つの (右寄せされた) 0 によって示されなければなりません。長さがデータ・タイプによって暗黙指定される数値フィールド (例えば IXFCLENG) が使用されない場合、それにはブランクを埋め込む必要があります。そのような数値フィールドは、次のとおりです。

```
IXFHRECL, IXFTRECL, IXFCRECL, IXFDRECL, IXFARECL,
IXFHHCNT, IXFHSBCP, IXFHDBCP, IXFTCCNT, IXFTNAML,
IXFCLENG, IXFCDRID, IXFCPOSN, IXFCNAML, IXFCTYPE,
IXFCSBCP, IXFCDBCP, IXFCNDIM, IXFCDSIZ, IXFDRID
```

注: データベース・マネージャーの PC/IXF ファイル形式は、System/370™ と同じではありません。

PC/IXF レコード・タイプ

PC/IXF の基本レコード・タイプには、次の 5 種類があります。

- ヘッダー
- 表
- 列記述子
- データ
- アプリケーション

これに、DB2 が使用する、以下の 6 つのアプリケーション・サブタイプが加わります。

- 索引
- 階層
- 副表
- 継続
- 終了
- ID

PC/IXF の各レコード・タイプは一連のフィールドとして定義されます。それらのフィールドは必須であり、示されている順序になっていなければなりません。

ヘッダー・レコード

フィールド名	長さ	タイプ	備考
IXFHRECL	06-BYTE	CHARACTER	レコード長
IXFHRECT	01-BYTE	CHARACTER	レコード・タイプ = 'H'
IXFHID	03-BYTE	CHARACTER	IXF ID
IXFHVERS	04-BYTE	CHARACTER	IXF のバージョン
IXFHPROD	12-BYTE	CHARACTER	製品
IXFHDATE	08-BYTE	CHARACTER	作成日付
IXFHTIME	06-BYTE	CHARACTER	作成時刻
IXFHHCNT	05-BYTE	CHARACTER	ヘッダー・レコードのカウント
IXFHSBCP	05-BYTE	CHARACTER	単一バイト・コード・ページ
IXFHDBCP	05-BYTE	CHARACTER	2 バイト・コード・ページ
IXFHFIL1	02-BYTE	CHARACTER	予約済み

ヘッダー・レコードには、以下のフィールドが入っています。

IXFHRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。 H レコードは、そのすべての定義済みフィールドを入れるのに十分な長さでなければなりません。

IXFHRECT
IXF レコード・タイプ (このレコードの場合、H にセットされる)。

IXFHID
ファイル・フォーマット ID (このファイルの場合、IXF にセットされる)。

IXFHVERS
ファイルの作成時に使用された PC/IXF フォーマット・レベル (0002 にセットされる)。

IXFHPROD
ファイルを作成しているプログラムが、それ自体を識別するために使用できるフィールド。このフィールドにデータが入っている場合、その最初の 6 バイトはファイルを作成した製品を識別するために使用され、最後の 6 バイトはその製品のバージョンまたはリリースを示すために使用されます。データベース・マネージャーは、データベース・マネージャー固有データの存在を通知するためにこのフィールドを使用します。

IXFHDATE
ファイルの作成日付 (*yyyymmdd* の形式)。

IXFHTIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドはオプションであり、ブランクのままにしておくことができます。

IXFHHCNT
このファイルのうち最初のデータ・レコードより前にある H、T、および C レコードの数。 A レコードは、このカウントに入りません。

IXFHSBCP
SBCS CPGID または '00000' の 1 バイト文字表記の入った 1 バイト・コード・ページ・フィールド。

エクスポート・ユーティリティーは、このフィールドを、エクスポートされるデータベース表の SBCS CPGID と等しい値にセットします。例えば、表の SBCS CPGID が 850 の場合、このフィールドの内容は '00850' です。

IXFHDBCP
DBCS CPGID または '00000' の 1 バイト文字表記の入った 2 バイト・コード・ページ・フィールド。

エクスポート・ユーティリティーは、このフィールドを、エクスポートされるデータベース表の DBCS CPGID と等しい値にセットします。例えば、表の DBCS CPGID が 301 の場合、このフィールドの内容は '00301' です。

IXFHFIL1
ホスト IXF ファイルの予約フィールドに一致させるために、2 つのブランクにセットされるスペア・フィールド。

表レコード

フィールド名	長さ	タイプ	備考
IXFTRECL	006-BYTE	CHARACTER	レコード長
IXFTRECT	001-BYTE	CHARACTER	レコード・タイプ = 'T'
IXFTNAML	003-BYTE	CHARACTER	名前の長さ
IXFTNAME	256-BYTE	CHARACTER	データの名前
IXFTQULL	003-BYTE	CHARACTER	修飾子の長さ
IXFTQUAL	256-BYTE	CHARACTER	修飾子
IXFTSRC	012-BYTE	CHARACTER	データ・ソース
IXFTDATA	001-BYTE	CHARACTER	データ規則 = 'C'
IXFTFORM	001-BYTE	CHARACTER	データ・フォーマット = 'M'
IXFTMFRM	005-BYTE	CHARACTER	マシン形式 = 'PC'
IXFTLOC	001-BYTE	CHARACTER	データ・ロケーション = 'I'
IXFTCCNT	005-BYTE	CHARACTER	'C' レコード・カウント
IXFTFIL1	002-BYTE	CHARACTER	予約済み
IXFTDESC	030-BYTE	CHARACTER	データの記述
IXFTPKNM	257-BYTE	CHARACTER	主キー名

```
IXFTDSPC        257-BYTE   CHARACTER   予約済み
IXFTISPC        257-BYTE   CHARACTER   予約済み
IXFTLSPC        257-BYTE   CHARACTER   予約済み
```

表レコードには、以下のフィールドが入っています。

IXFTRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。 T レコードは、そのすべての定義済みフィールドを入れるのに十分な長さでなければなりません。

IXFTRECT
IXF レコード・タイプ (このレコードの場合、T にセットされる)。

IXFTNAML
IXFTNAME フィールド内の表名の長さ (バイト単位)。

IXFTNAME
表の名前。各ファイルごとに 1 つの表だけがある場合、これは単なる情報フィールドです。データベース・マネージャーは、データのインポート時にこのフィールドを使用しません。 PC/IXF ファイルへの書き込み時にデータベース・マネージャーは、DOS ファイル名 (場合によってはパス情報) をこのフィールドに書き込みます。

IXFTQULL
IXFTQUAL フィールド内の表名修飾子の長さ (バイト単位)。

IXFTQUAL
リレーショナル・システム内で表の作成者を識別する表名修飾子。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。

IXFTSRC
データのオリジナル・ソースを指示するために使用されます。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。

IXFTDATA
データの記述に使用される規則。インポートまたはエクスポートの場合、このフィールドは C にセットしなければなりません。これは、個々の列属性が次の列記述子 (C) レコードで記述されており、データが PC/IXF 規則に従っていることを示します。

IXFTFORM
数値データの保管に使用される規則。このフィールドは、M にセットしなければなりません。これは、データ (D) レコード内の数値データが IXFTMFRM フィールドによって指定されるマシン (内部) フォーマットで保管されていることを示します。

IXFTMFRM

PC/IXF ファイル内のマシン・データのフォーマット。データベース・マネージャーは、このフィールドが PC*bbb* にセットされている場合にのみ、ファイルの読み取りまたは書き込みを実行します。 *b* はブランクを表し、PC は、PC/IXF ファイルのデータが IBM PC マシン・フォーマットになっていることを指定します。

IXFTLOC

データのロケーション。データベース・マネージャーは、値として I のみサポートします。これは、データがこのファイルの内部にあることを意味します。

IXFTCCNT

この表内の C レコードの数。これは、整数値の右寄せ文字表記です。

IXFTFIL1

ホスト IXF ファイルの予約フィールドに一致させるために、2 つのブランクにセットされるスペア・フィールド。

IXFTDESC

表に関する記述データ。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。列がデフォルトによって NULL になっておらず、表名がワークステーションのデータベースからのものである場合、このフィールドには NOT NULL WITH DEFAULT が入ります。

IXFTPKNM

表で定義されている主キーの名前 (ある場合)。この名前は NULL 文字で終了するストリングとして格納されます。

IXFTDSPC

このフィールドは将来の利用のために予約済み。

IXFTISPC

このフィールドは将来の利用のために予約済み。

IXFTLSPC

このフィールドは将来の利用のために予約済み。

列記述子レコード

フィールド名	長さ	タイプ	備考
IXFCRECL	006-BYTE	CHARACTER	レコード長
IXFCRECT	001-BYTE	CHARACTER	レコード・タイプ = 'C'
IXFCNAML	003-BYTE	CHARACTER	列名長
IXFCNAME	256-BYTE	CHARACTER	列名
IXFCNULL	001-BYTE	CHARACTER	列が NULL 可能
IXFCDEF	001-BYTE	CHARACTER	列にデフォルトがある
IXFCSLCT	001-BYTE	CHARACTER	列選択フラグ
IXFCKPOS	002-BYTE	CHARACTER	主キーでの位置
IXFCCLAS	001-BYTE	CHARACTER	データ・クラス
IXFCTYPE	003-BYTE	CHARACTER	データ・タイプ
IXFCSBCP	005-BYTE	CHARACTER	単一バイト・コード・ページ
IXFCDBCP	005-BYTE	CHARACTER	2 バイト・コード・ページ
IXFCLENG	005-BYTE	CHARACTER	列データ長

```
IXFCDRID       003-BYTE   CHARACTER   'D' レコード ID
IXFCPOSN       006-BYTE   CHARACTER   列位置
IXFCDESC       030-BYTE   CHARACTER   列記述
IXFCLOBL       020-BYTE   CHARACTER   LOB 列の長さ
IXFCUDTL       003-BYTE   CHARACTER   UDT 名の長さ
IXFCUDTN       256-BYTE   CHARACTER   UDT 名
IXFCDEFL       003-BYTE   CHARACTER   デフォルト値の長さ
IXFCDEFV       254-BYTE   CHARACTER   デフォルト値
IXFCREF        001-BYTE   CHARACTER   参照タイプ
IXFCNDIM       002-BYTE   CHARACTER   次元数
IXFCDSIZ       varying    CHARACTER   各次元のサイズ
```

列記述子レコードには、以下のフィールドが入っています。

IXFCRECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。 C レコードは、そのすべての定義済みフィールドを入れるのに十分な長さでなければなりません。

IXFCRECT
IXF レコード・タイプ (このレコードの場合、C にセットされる)。

IXFCNAML
IXFCNAME フィールド内の列名の長さ (バイト単位)。

IXFCNAME
列の名前。

IXFCNULL
この列で NULL が可能かどうかを指定します。有効な設定は、Y または N です。

IXFCDEF
このフィールドのデフォルト値が定義されているかどうかを指定します。有効な設定は、Y または N です。

IXFCSLCT
データ中の列のサブセットの選択を可能にすることを目的としたフィールドで、現在は廃止されています。 PC/IXF ファイルへの書き込みを実行するプログラムでは、このフィールドに常に Y を保管しなければなりません。 PC/IXF ファイルを読むプログラムでは、このフィールドを無視しなければなりません。

IXFCKPOS
主キーの一部としての列の位置。有効値は 01 から 16 ですが、主キーの一部でない列の場合は N です。

IXFCCLAS
IXFCTYPE フィールドで使用されるデータ・タイプのクラス。データベース・マネージャーは、リレーショナル・タイプ (R) のみサポートします。

IXFCTYPE
列のデータ・タイプ。

IXFCSBCP

SBCS CPGID の 1 バイト文字表記。このフィールドは、この列の D レコードの IXFDCOLS フィールドに入れられる 1 バイト文字データの CPGID を指定します。

このフィールドのセマンティクスは、列のデータ・タイプ (IXFCTYPE フィールドで指定される) によって違います。

- 文字ストリング列の場合、このフィールドには、通常、H レコードの IXFHSBCP フィールドの値と等しいゼロ以外の値が入っていなければなりません。ただし、その他の値も許可されます。この値がゼロの場合、列はビット・ストリング・データが入ると解釈されます。
- 数値列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは 0 にセットされ、インポート・ユーティリティーでは無視されます。
- 日付列または時刻列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは IXFHSBCP フィールドの値にセットされ、インポート・ユーティリティーでは無視されます。
- GRAPHIC 列の場合、このフィールドは 0 でなければなりません。

IXFCDBCP

DBCS CPGID の 1 バイト文字表記。このフィールドは、この列の D レコードの IXFDCOLS フィールドに入れられる 2 バイト文字データの CPGID を指定します。

このフィールドのセマンティクスは、列のデータ・タイプ (IXFCTYPE フィールドで指定される) によって違います。

- 文字ストリング列の場合、このフィールドには、0 か、または H レコードの IXFHDBCP フィールドの値と等しい値が入っていなければなりません。ただし、その他の値も許可されます。 IXFCSBCP フィールドの値が 0 の場合、このフィールドの値は 0 でなければなりません。
- 数値列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは 0 にセットされ、インポート・ユーティリティーでは無視されます。
- 日付列または時刻列の場合、このフィールドには意味がありません。このフィールドは、エクスポート・ユーティリティーでは 0 にセットされ、インポート・ユーティリティーでは無視されます。
- GRAPHIC 列の場合、このフィールドの値は IXFHDBCP フィールドの値と同じでなければなりません。

IXFCLENG

記述されている列のサイズに関する情報を提供します。データ・タイプによっては、このフィールドは使用されず、ブランクを入れる必要があります。他のいくつかのデータ・タイプの場合、このフィールドには、列の長さを指定する整数の右寄せ文字表記が入れられます。また、さらに別のいくつかのデータ・タイプの場合、このフィールドは 2 つのサブフィールド (精度を表す 3 バイトと位取りを表す 2 バイト) に分割されます。これらのサブフィールドはいずれも整数の右寄せ文字表記です。

IXFCDRID

D レコード ID。このフィールドの内容は、整数値の右寄せ文字表記です。PC/IXF ファイルでは、各行のデータを入れるために複数の D レコードが使用されることがあります。このフィールドは、あるデータ行を表現する D レコードのうち、どの D レコードに列のデータが入っているかを指定します。値 1 (例えば 001) は、列のデータがデータ行の最初の D レコードに入っていることを示します。最初の C レコードの IXFCDRID 値は 1 でなければなりません。それ以降のすべての C レコードの IXFCDRID 値は、その直前の C レコードと等しい値か、または 1 大きい値でなければなりません。

IXFCPOSN

このフィールドの値は、表のあるデータ行を表現する D レコードの 1 つの中で、列のデータを見つけるのに使用されます。これは、D レコードの IXFDCOLS フィールド内でのこの列のデータの開始位置です。列が NULL 可能なら IXFCPOSN は NULL 標識を指し、それ以外の場合にはデータ自体を指します。列に可変長データがある場合、データ自体が現行長標識で始まります。 D レコードの IXFDCOLS フィールド内の最初のバイトに対応する IXFCPOSN 値は 1 です (0 ではありません)。列が新しい D レコードに入っている場合、IXFCPOSN 値は 1 になります。それ以外の場合、IXFCPOSN 値はデータ値が重なり合わない範囲で列ごとに増加します。

IXFCDESC

列に関する記述情報。これは単なる情報フィールドです。ファイルに書き込むプログラムにこのフィールドに書き込むデータがない場合、推奨される充てん値はブランクです。ファイルを読み取るプログラムでは、このフィールドを印刷または表示したり、情報フィールドに保管したりできますが、このフィールドの内容に基づく演算は信頼性がありません。

IXFCLOBL

この列内で定義される long または LOB の長さ (バイト単位)。この列が long または LOB でない場合、このフィールドの値は 000 になります。

IXFCUDTL

IXFCUDTN フィールド内のユーザー定義タイプ (UDT) 名の長さ (バイト単位)。この列のタイプが UDT でない場合、このフィールドの値は 000 になります。

IXFCUDTN

この列のデータ・タイプとして使用されるユーザー定義タイプの名前。

IXFCDEFL

IXFCDEFV フィールド内のデフォルト値の長さ (バイト単位)。この列にデフォルト値がない場合、このフィールドの値は 000 になります。

IXFCDEFV

この列にデフォルト値が定義されている場合に、それを指定します。

IXFCREF

列が階層の一部を成している場合、このフィールドは、その列がデータ列 (D) または参照列 (R) のどちらであるかを指定します。

IXFCNDIM

列の次元の数。配列は、このバージョンの PC/IXF ではサポートされていません。したがって、このフィールドの内容は整数値 0 の文字表記でなければなりません。

IXFCDSIZ

各次元のサイズまたは範囲。このフィールドの長さは、1 次元当たり 5 バイトです。配列はサポートされない (次元の数は 0 でなければならない) ため、このフィールドは長さ 0 であり、実際には存在しません。

データ・レコード

フィールド名	長さ	タイプ	備考
IXFDRECL	06-BYTE	CHARACTER	レコード長
IXFDRECT	01-BYTE	CHARACTER	レコード・タイプ = 'D'
IXFDRID	03-BYTE	CHARACTER	'D' レコード ID
IXFDFIL1	04-BYTE	CHARACTER	予約済み
IXFDCOLS	varying	variable	縦欄データ

データ・レコードには、以下のフィールドが入っています。

IXFDRECL

レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 D レコードは、レコードに保管される最後のデータ列の現行オカレンスのすべての有効なデータを入れるのに十分な長さでなければなりません。

IXFDRECT

IXF レコード・タイプ (このレコードの場合、D にセットされる)。これは、このレコードに表のデータ値が入っていることを示します。

IXFDRID

レコード ID。これは、あるデータ行を表現する一連の D レコードの中で特定の D レコードを識別します。あるデータ行の最初の D レコードの場合、このフィールドの値は 1 になります。第 2 の D レコードの場合、このフィールドの値は 2 になり、以下同様です。各データ行の中には、C レコードの中で指定されているすべての D レコード ID が実際に存在していなければなりません。

IXFDFIL1

ホスト IXF ファイルの中で、予約フィールドに対応するブランク 4 個に設定されるスペア・フィールドで、使用される可能性があるシフトアウト文字のプレースホルダーとなるもの。

IXFDCOLS

縦欄データ用の領域。データ・レコード (D レコード) のデータ域は、1 つまたは複数の列項目で構成されます。その D レコードと同じ D レコード ID の列記述子レコードごとに、1 つの列項目があります。 D レコードにおける列項目の開始位置は、C レコードの IXFCPOSN 値によって指示されます。

列項目データのフォーマットは、列が NULL 可能であるかどうかによって異なります。

- 列が NULL 可能である場合 (IXFCNULL フィールドが Y にセットされている場合)、列項目データには NULL 標識が組み込まれます。列が NULL でない場合は、標識の後にデータ・タイプ固有の情報 (実際のデータベース値を含む) が続きます。 NULL 標識は、NULL でない場合は x'0000' にセットされ、NULL の場合は x'FFFF' にセットされる 2 バイトの値です。
- 列が NULL 可能でない場合、列項目データの内容はデータ・タイプ固有の情報 (実際のデータベース値を含む) だけです。

可変長データ・タイプの場合のデータ・タイプ固有の情報には、現行長標識が組み込まれます。現行長標識は、IXFTMFRM フィールドによって指定される形式の 2 バイトの整数です。

D レコードのデータ域の長さは、32,771 バイトを超えてはなりません。

アプリケーション・レコード

フィールド名	長さ	タイプ	備考
IXFARECL	06-BYTE	CHARACTER	レコード長
IXFARECT	01-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	12-BYTE	CHARACTER	アプリケーション ID
IXFADATA	varying	variable	アプリケーション固有のデータ

アプリケーション・レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションを識別します。データベース・マネージャーによって作成された PC/IXF ファイルの A レコードの場合、このフィールドの最初の 6 文字を、データベース・マネージャーを識別する定数に設定して、最後の 6 文字を、データベース・マネージャーまたは A レコードを作成している別のアプリケーションのリリースまたはバージョンを識別する定数に設定することができます。

IXFADATA
このフィールドには、アプリケーションに依存する補足データが入れられます。その形式と内容は、A レコードを作成するプログラムと、A レコードを処理する他のアプリケーションによってのみ認識されます。

DB2 索引レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長

```
IXFARECT       001-BYTE   CHARACTER   レコード・タイプ = 'A'
IXFAPPID       012-BYTE   CHARACTER   アプリケーション ID =
                                        'DB2    02.00'
IXFAITYP       001-BYTE   CHARACTER   アプリケーション固有のデータ・タイプ =
                                        'I'
IXFADATE       008-BYTE   CHARACTER   'H' レコードから書き込まれた日付
IXFATIME       006-BYTE   CHARACTER   'H' レコードから書き込まれた時刻
IXFANDXL       002-BYTE   SHORT INT   索引名の長さ
IXFANDXN       256-BYTE   CHARACTER   索引名
IXFANCL        002-BYTE   SHORT INT   索引作成者名の長さ
IXFANCN        256-BYTE   CHARACTER   索引作成者名
IXFATABL       002-BYTE   SHORT INT   表名の長さ
IXFATABN       256-BYTE   CHARACTER   表名
IXFATCL        002-BYTE   SHORT INT   表作成者名の長さ
IXFATCN        256-BYTE   CHARACTER   表作成者名
IXFAUNIQ       001-BYTE   CHARACTER   ユニーク規則
IXFACCNT       002-BYTE   CHARACTER   列カウント
IXFAREVS       001-BYTE   CHARACTER   逆スキャン・フラグの許可
IXFAPCTF       002-BYTE   CHARACTER   空き PCT の量
IXFAPCTU       002-BYTE   CHARACTER   最小使用 PCT の量
IXFAEXTI       001-BYTE   CHARACTER   予約済み
IXFACNML       002-BYTE   SHORT INT   列名の長さ
IXFACOLN       varying    CHARACTER   索引内の列の名前
```

ユーザー定義索引ごとに、このタイプのレコードが 1 つずつ指定されます。このレコードは、表のすべての C レコードの後に置かれます。 DB2 索引レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAITYP
これがサブタイプ ″I″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFANDXL
IXFANDXN フィールド内の索引名の長さ (バイト単位)。

IXFANDXN
索引の名前。

IXFANCL
IXFANCN フィールド内の索引作成者名の長さ (バイト単位)。

IXFANCN
索引作成者の名前。

IXFATABL
IXFATABN フィールド内の表名の長さ (バイト単位)。

IXFATABN
表の名前。

IXFATCL
IXFATCN フィールド内の表作成者名の長さ (バイト単位)。

IXFATCN
表作成者の名前。

IXFAUNIQ
索引のタイプを指定します。有効値は、P (主キー)、U (ユニーク索引)、および D (非ユニーク索引) です。

IXFACCNT
索引定義内の列の数を指定します。

IXFAREVS
この索引で逆スキャンを行えるかどうかを指定します。有効値は、 Y (逆スキャン可) と N (逆スキャン不可) です。

IXFAPCTF
空き状態にしておく索引ページのパーセントを指定します。有効値は -1 から 99 です。 -1 またはゼロの値を指定すると、システム・デフォルト値が使用されます。

IXFAPCTU
2 つの索引ページを組み合わせる場合に、あらかじめ空きになっている必要のある索引ページの最小パーセントを指定します。有効値は 00 から 99 の範囲です。

IXFAEXTI
将来の使用のために予約されています。

IXFACNML
IXFACOLN フィールド内の列名の長さ (バイト単位)。

IXFACOLN
この索引の一部を成す列の名前。有効値は *+name -name...* の形式です。 + は列の昇順ソートを指定し、 - は列の降順ソートを指定します。

DB2 階層レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション ID = 'DB2 02.00'

IXFAXTYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'X'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻
IXFAYCNT	010-BYTE	CHARACTER	この階層の 'Y' レコード・カウント
IXFAYSTR	010-BYTE	CHARACTER	この階層の開始列

階層を記述する際、このタイプのレコードが 1 つ使用されます。すべての副表レコード (以下を参照) は階層レコードの直後に置かれなければならず、階層レコードは表のすべての C レコードの後に置かれます。 DB2 階層レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAXTYP
これがサブタイプ ″X″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFAYCNT
この階層レコードの後の副表レコードの予想数を指定します。

IXFAYSTR
エクスポート・データの先頭における副表レコードの索引を指定します。階層のエクスポートがルート以外の副表から開始された場合、その副表のすべての親表がエクスポートされます。このフィールドには、IXF ファイル内のこの副表の位置も格納されます。最初の X レコードは、ゼロの索引をもつ列を表します。

DB2 副表レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション ID = 'DB2 02.00'

```
IXFAYTYP       001-BYTE   CHARACTER   アプリケーション固有のデータ・タイプ =
                                         'Y'
IXFADATE       008-BYTE   CHARACTER   'H' レコードから書き込まれた日付
IXFATIME       006-BYTE   CHARACTER   'H' レコードから書き込まれた時刻
IXFASCHL       003-BYTE   CHARACTER   タイプ・スキーマ名の長さ
IXFASCHN       256-BYTE   CHARACTER   タイプ・スキーマ名
IXFATYPL       003-BYTE   CHARACTER   タイプ名の長さ
IXFATYPN       256-BYTE   CHARACTER   タイプ名
IXFATABL       003-BYTE   CHARACTER   表名の長さ
IXFATABN       256-BYTE   CHARACTER   表名
IXFAPNDX       010-BYTE   CHARACTER   親表の副表索引
IXFASNDX       005-BYTE   CHARACTER   現行の表の開始列索引
IXFAENDX       005-BYTE   CHARACTER   現行の表の終了列索引
```

階層の一部として副表を記述する際、このタイプのレコードが 1 つ使用されます。階層に属するすべての副表レコードは、対応する階層レコードの直後に、一緒に格納されている必要があります。副表は 1 つ以上の列で構成され、おのおのの列は列レコード内で記述されます。副表内の各列は、連続した C レコード・セット内で記述しなければなりません。 DB2 副表レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAYTYP
これがサブタイプ ″Y″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFASCHL
IXFASCHN フィールド内の副表スキーマ名の長さ (バイト単位)。

IXFASCHN
副表スキーマの名前。

IXFATYPL
IXFATYPN フィールド内の副表名の長さ (バイト単位)。

IXFATYPN
副表の名前。

IXFATABL
IXFATABN フィールド内の表名の長さ (バイト単位)。

IXFATABN
表の名前。

IXFAPNDX
親副表の副表レコード索引。この副表が階層のルートである場合、このフィールドには値 -1 が入ります。

IXFASNDX
この副表を構成する列レコードの開始索引。

IXFAENDX
この副表を構成する列レコードの終了索引。

DB2 継続レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション ID = 'DB2 02.00'
IXFACTYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'C'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻
IXFALAST	002-BYTE	SHORT INT	最後のディスケット・ボリューム番号
IXFATHIS	002-BYTE	SHORT INT	このディスケット・ボリューム番号
IXFANEXT	002-BYTE	SHORT INT	次のディスケット・ボリューム番号

ファイルが最終ボリュームでない限り、このレコードは、マルチボリュームの IXF ファイルの一部を成す各ファイルの末尾にあります。また、ファイルが先頭ボリュームでない限り、このレコードは、マルチボリュームの IXF ファイルの一部を成す各ファイルの先頭にもあります。このレコードの目的は、ファイル順序を記録しておくことです。 DB2 継続レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFACTYP
これがサブタイプ ″C″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFALAST
このフィールドは、リトル・エンディアン・フォーマットの 2 進数フィールドです。値は、IXFATHIS の値より 1 小さくなければなりません。

IXFATHIS
このフィールドは、リトル・エンディアン・フォーマットの 2 進数フィールドです。連続ボリューム上では、このフィールドの値も連続している必要があります。最初のボリュームでは 1 の値を持ちます。

IXFANEXT
このフィールドは、リトル・エンディアン・フォーマットの 2 進数フィールドです。レコードがファイルの先頭にない場合、値は、IXFATHIS 内の値よりも 1 大きくなければなりません。先頭にある場合、値はゼロでなければなりません。

DB2 終了レコード

フィールド名	長さ	タイプ	備考
IXFARECL	006-BYTE	CHARACTER	レコード長
IXFARECT	001-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	012-BYTE	CHARACTER	アプリケーション ID = 'DB2 02.00'
IXFAETYP	001-BYTE	CHARACTER	アプリケーション固有のデータ・タイプ = 'E'
IXFADATE	008-BYTE	CHARACTER	'H' レコードから書き込まれた日付
IXFATIME	006-BYTE	CHARACTER	'H' レコードから書き込まれた時刻

このレコードは、IXF ファイルの末尾にあるファイル終了マーカーです。 DB2 終了レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFAETYP
これがサブタイプ ″E″ の DB2 アプリケーション・レコードであることを指定します。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

DB2 ID レコード

フィールド名	長さ	タイプ	備考
IXFARECL	06-BYTE	CHARACTER	レコード長
IXFARECT	01-BYTE	CHARACTER	レコード・タイプ = 'A'
IXFAPPID	12-BYTE	CHARACTER	アプリケーション ID
IXFATYPE	01-BYTE	CHARACTER	アプリケーション固有のレコード・タイプ = 'S'
IXFADATE	08-BYTE	CHARACTER	アプリケーション・レコードの作成日
IXFATIME	06-BYTE	CHARACTER	アプリケーション・レコードの作成時刻
IXFACOLN	06-BYTE	CHARACTER	ID 列の列番号
IXFAITYP	01-BYTE	CHARACTER	常に生成される ('Y' または 'N')
IXFASTRT	33-BYTE	CHARACTER	ID 列の START AT 値
IXFAINCR	33-BYTE	CHARACTER	ID 列の INCREMENT BY 値
IXFACACH	10-BYTE	CHARACTER	ID 列の CACHE 値
IXFAMINV	33-BYTE	CHARACTER	ID 列の MINVALUE
IXFAMAXV	33-BYTE	CHARACTER	ID 列の MAXVALUE
IXFACYCL	01-BYTE	CHARACTER	ID 列の CYCLE ('Y' または 'N')
IXFAORDR	01-BYTE	CHARACTER	ID 列の ORDER ('Y' または 'N')
IXFARMRL	03-BYTE	CHARACTER	ID 列の注釈の長さ
IXFARMRK	254-BYTE	CHARACTER	ID 列の注釈の値

DB2 識別レコードには、以下のフィールドが入っています。

IXFARECL
レコード長標識。 PC/IXF レコードのうちこのレコード長標識より後の部分の長さをバイト単位で指定する整数値の 6 バイトの右寄せ文字表記 (合計レコード・サイズ - 6 バイト)。各 A レコードは、少なくとも IXFAPPID フィールドの全体を入れるのに十分な長さでなければなりません。

IXFARECT
IXF レコード・タイプ (このレコードの場合、A にセットされる)。これがアプリケーション・レコードであることを示します。アプリケーション ID によって暗黙指定されるデータの内容とフォーマットに関する特別の知識がないプログラムでは、このレコードは無視されます。

IXFAPPID
アプリケーション ID。これは、A レコードを作成したアプリケーションとして DB2 を識別します。

IXFATYPE
アプリケーション固有のレコード・タイプ。このフィールドの値は常に ″S″ でなければなりません。

IXFADATE
ファイルの作成日付 (*yyyymmdd* の形式)。このフィールドの値は、IXFHDATE と同じでなければなりません。

IXFATIME
ファイルの作成時刻 (*hhmmss* の形式)。このフィールドの値は、IXFHTIME と同じでなければなりません。

IXFACOLN
表内の ID 列の列番号。

IXFAITYP
ID 列のタイプ。 ″Y″ の値は、ID 列が常に GENERATED であることを示します。他のすべての値は、列が GENERATED BY DEFAULT タイプであることを意味すると解釈されます。

IXFASTRT
表の作成時に CREATE TABLE ステートメントで指定された ID 列の START AT® 値。

IXFAINCR
表の作成時に CREATE TABLE ステートメントで指定された ID 列の INCREMENT BY 値。

IXFACACH
表の作成時に CREATE TABLE ステートメントで指定された ID 列の CACHE 値。 ″1″ の値は、NO CACHE オプションに対応します。

IXFAMINV
表の作成時に CREATE TABLE ステートメントで指定された ID 列の MINVALUE。

IXFAMAXV
表の作成時に CREATE TABLE ステートメントで指定された ID 列の MAXVALUE。

IXFACYCL
表の作成時に CREATE TABLE ステートメントで指定された ID 列の CYCLE 値。値 ″Y″ は CYCLE オプションに対応し、それ以外の値は NO CYCLE に対応します。

IXFAORDR
表の作成時に CREATE TABLE ステートメントで指定された ID 列の ORDER 値。値 ″Y″ は ORDER オプションに対応し、それ以外の値は NO ORDER に対応します。

IXFARMRL
IXFARMRK フィールド内の注釈の長さ (バイト単位)。

IXFARMRK
これは ID 列に関連付けられたユーザー入力の注釈です。これは単なる情報フィールドです。データベース・マネージャーは、データのインポート時にこのフィールドを使用しません。

PC/IXF データ・タイプ

表 52. PC/IXF データ・タイプ

名前	IXFCTYPE の値	説明
BIGINT	492	IXFTMFRM によって指定される形式の 8 バイトの整数。これは -9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 の範囲の整数を表します。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 IXFCLENG は使用されず、ブランクを入れる必要があります。
BLOB、CLOB	404、408	可変長文字ストリング。ストリングの最大長は、列記述子レコードの IXFCLENG フィールドに入れられ、それは 32,767 バイト以下です。ストリング自体の前には現行長標識が付きます。これは、ストリングの長さをバイト単位で指定する 4 バイト整数です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 次の記述は BLOB にのみ適用されます。 IXFCSBCP が 0 の場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。 次の記述は CLOB にのみ適用されます。 IXFCDBCP がゼロ以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も組み込むことができます。
BLOB_LOCATION_ SPECIFIER および DBCLOB_ LOCATION_ SPECIFIER	960、964、968	固定長フィールド。255 バイトを超えてはなりません。 LOB ロケーション指定子 (LLS) は、 IXFCSBCP によって指定されるコード・ページにあります。 IXFCSBCP が 0 の場合、LLS はビット・データであり、変換プログラムによって変換しないようにしてください。 IXFCDBCP がゼロ以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も組み込むことができます。 LLS の長さが IXFCLENG に保管されるため、元の LOB の実際の長さは失われます。 LOB は LLS の長さで作成されるため、このタイプの列の入った PC/IXF ファイルを使用して LOB フィールドを再作成しないようにしてください。

表 52. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
BLOB_FILE、 CLOB_FILE、 DBCLOB_FILE	916、920、924	*name_length* および *name* フィールドにデータが入れられた SQLFILE 構造体を収める固定長フィールド。ストリングの長さは、列記述子レコードの IXFCLENG フィールドに入れられ、それは 255 バイト以下です。ファイル名のコード・ページは、IXFCSBCP によって指定されるコード・ページです。IXFCDBCP がゼロ以外の場合、ファイル名には IXFCDBCP によって指定されるコード・ページの 2 バイト文字も組み込むことができます。 IXFCSBCP が 0 の場合、ファイル名はビット・データであり、変換プログラムによって変換しないようにしてください。 構造体の長さが IXFCLENG に保管されるため、元の LOB の実際の長さは失われます。LOB は *sql_lobfile_len* の長さで作成されるため、タイプ BLOB_FILE、 CLOB_FILE、または DBCLOB_FILE の列をもつ IXF ファイルを使用して LOB フィールドを再作成しないようにしてください。
CHAR	452	固定長文字ストリング。ストリングの長さは、列記述子レコードの IXFCLENG フィールドに入れられ、それは 254 バイト以下です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 IXFCDBCP がゼロ以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も組み込むことができます。 IXFCSBCP がゼロの場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。
DATE	384	グレゴリオ暦に準拠したポイント・イン・タイム。それぞれの日付は、国際標準化機構規格 (ISO) フォーマット (*yyyy-mm-dd*) の 10 バイトの文字ストリングです。年の部分の範囲は `0001` から `9999` です。月の部分の範囲は `01` から `12` です。日の部分の範囲は `01` から *n* です。ここで、*n* は月によって異なり、月の日数とうるう年に関する一般的な規則に従います。どの部分においても、先行ゼロは省略できません。 IXFCLENG は使用されず、ブランクを入れる必要があります。DATE 内の有効な文字は、すべての PC ASCII コード・ページで不変です。そのため、IXFCSBCP および IXFCDBCP は有効ではなく、0 でなければなりません。

表 52. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
DBCLOB	412	2 バイト文字の可変長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の最大文字数を指定するもので、16,383 以下です。ストリング自体の前に現行長標識が付きます。これは、ストリングの長さを 2 バイト文字の文字数で指定する 4 バイト整数です (つまりこの整数の値は、バイト単位のストリング長の半分の値です)。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。
DECIMAL	484	精度が P (列記述子レコードの IXFCLENG の最初の 3 バイトによって指定される) で、位取りが S (IXFCLENG の最後の 2 バイトによって指定される) であるパック 10 進数。パック 10 進数の長さは (P+2)/2 です (バイト単位)。精度は 1 から 31 の範囲の奇数でなければなりません。パック 10 進数は、IXFTMFRM によって指定される内部フォーマットになっています。 IXFTMFRM では、PC のパック 10 進数が System/370 のパック 10 進数と同じになるように定義されます。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。
DECFLOAT	996	10 進浮動小数点値は、小数点付きの IEEE 754r 数です。小数点の位置は、それぞれの 10 進浮動小数点値に格納されます。 10 進浮動小数点数の範囲は、精度が 16 桁または 34 桁で、指数範囲はそれぞれ 10-383 から 10+384 までと 10-6143 から 10+6144 までです。16 桁値の格納される長さは 8 バイトで、34 桁値の格納される長さは 16 バイトです。
FLOATING POINT	480	長精度 (8 バイト) または短精度 (4 バイト) 浮動小数点数。これは、IXFCLENG が 8 または 4 のどちらにセットされているかによって決まります。データは、IXFTMFRM によって指定される内部マシン・フォーマットです。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 4 バイトの浮動小数点数は、データベース・マネージャーではサポートされません。

表 52. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
GRAPHIC	468	2 バイト文字の固定長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の文字数を指定するもので、127 以下です。ストリングの実際の長さ (バイト単位) は、IXFCLENG フィールドの値の 2 倍です。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。
INTEGER	496	IXFTMFRM によって指定される形式の 4 バイト整数。これは -2,147,483,648 から +2,147,483,647 の範囲の整数を表します。IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 IXFCLENG は使用されず、ブランクを入れる必要があります。
LONGVARCHAR	456	可変長文字ストリング。ストリングの最大長は、列記述子レコードの IXFCLENG フィールドに入れられ、それは 32,767 バイト以下です。ストリング自体の前には現行長標識が付きます。これは、ストリングの長さをバイト単位で指定する 2 バイトの整数です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。IXFCDBCP がゼロ以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も組み込むことができます。 IXFCSBCP がゼロの場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。

表 52. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
LONG VARGRAPHIC	472	2 バイト文字の可変長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリングの 2 バイト文字の最大文字数を指定するもので、16,383 以下です。ストリング自体の前に現行長標識が付きます。これは、ストリングの長さを 2 バイト文字の文字数で指定する 2 バイトの整数です (つまりこの整数の値は、バイト単位のストリング長の半分の値です)。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。
SMALLINT	500	IXFTMFRM によって指定される形式の 2 バイトの整数。これは -32,768 から +32,767 の範囲の整数を表します。 IXFCSBCP および IXFCDBCP は無効であり 0 でなければなりません。 IXFCLENG は使用されず、ブランクを入れる必要があります。
TIME	388	24 時間制によるポイント・イン・タイム。それぞれの時刻は、ISO フォーマット (*hh.mm.ss*) の 8 バイトのストリングです。時の部分の範囲は `00` から `24` で、その他の部分の範囲は `00` から `59` です。時が `24` の場合、その他の部分は `00` です。最小の時刻は `00.00.00`、最大の時刻は `24.00.00` です。どの部分においても、先行ゼロは省略できません。 IXFCLENG は使用されず、ブランクを入れる必要があります。 TIME 内の有効な文字は、すべての PC ASCII コード・ページで不変です。そのため、IXFCSBCP および IXFCDBCP は有効ではなく、0 でなければなりません。
TIMESTAMP	392	マイクロ秒の精度の日時。各タイム・スタンプは、*yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒) の形式の文字ストリングです。 IXFCLENG は使用されず、ブランクを入れる必要があります。 TIMESTAMP 内の有効な文字は、すべての PC ASCII コード・ページで不変です。そのため、IXFCSBCP および IXFCDBCP は有効ではなく、0 でなければなりません。

表 52. PC/IXF データ・タイプ (続き)

名前	IXFCTYPE の値	説明
VARCHAR	448	可変長文字ストリング。ストリングの最大長 (バイト単位) は、列記述子レコードの IXFCLENG フィールドに入れられ、それは 254 バイト以下です。ストリング自体の前には現行長標識が付きます。これは、ストリングの長さをバイト単位で指定する 2 バイトの整数です。ストリングのコード・ページは、IXFCSBCP によって指定されるコード・ページです。 IXFCDBCP がゼロ以外の場合、ストリングには IXFCDBCP によって指定されるコード・ページの 2 バイト文字も組み込むことができます。 IXFCSBCP がゼロの場合、ストリングはビット・データであり、変換プログラムによって変換しないようにしてください。
VARGRAPHIC	464	2 バイト文字の可変長ストリング。列記述子レコードの IXFCLENG フィールドは、ストリング内の 2 バイト文字の最大文字数を指定するもので、127 以下です。ストリング自体の前に現行長標識が付きます。これは、ストリングの長さを 2 バイト文字の文字数で指定する 2 バイトの整数です (つまりこの整数の値は、バイト単位のストリング長の半分の値です)。ストリングのコード・ページは、C レコードの IXFCDBCP によって指定される DBCS コード・ページです。ストリングの内容は 2 バイト文字データだけなので、IXFCSBCP は 0 でなければなりません。ストリングを囲むシフトインまたはシフトアウト文字はありません。

PC/IXF 文字または GRAPHIC 列では、IXFCSBCP 値と IXFCDBCP 値の一部の組み合わせは無効です。 IXFCSBCP と IXFCDBCP の組み合わせが無効である PC/IXF 文字または GRAPHIC 列は、無効なデータ・タイプです。

表 53. 有効な PC/IXF データ・タイプ

PC/IXF データ・タイプ	有効な (IXFCSBCP,IXFCDBCP) 対	無効な (IXFCSBCP,IXFCDBCP) 対
CHAR、VARCHAR、または LONG VARCHAR	(0,0)、(x,0)、または (x,y)	(0,y)
BLOB	(0,0)	(x,0)、(0,y)、または (x,y)
CLOB	(x,0)、(x,y)	(0,0)、(0,y)
GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、または DBCLOB	(0,y)	(0,0)、(x,0)、または (x,y)
注: x および y は 0 ではありません。		

PC/IXF のデータ・タイプの説明

表 54. PC/IXF ファイル・フォーマットで受け入れられるデータ・タイプの形式

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
BIGINT	データベース列と同じ BIGINT 列が作成されます。	すべての数値タイプ (SMALLINT、INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。 -9,223,372,036,854,775,808 から 9,223,372,036,854,775,807 の範囲外の値があれば、その特定の値に関してはリジェクトされます。
BLOB	PC/IXF BLOB 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	次の場合、 PC/IXF CHAR、 VARCHAR、LONG VARCHAR、 BLOB、BLOB_FILE、または BLOB_LOCATION_SPECIFIER 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC の BLOB 列も受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。

表 54. PC/IXF ファイル・フォーマットで受け入れられるデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
CHAR	PC/IXF CHAR 列が作成されます。データベース列の長さ、SBCS CPGID 値、および DBCS CPGID 値が PC/IXF 列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、 VARCHAR、または LONG VARCHAR 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 データベース列が FOR BIT DATA としてマークされている場合、PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC 列も受け入れ可能です。どちらの場合でも、PC/IXF 列が固定長なら、その長さはデータベース列の長さと互換性がなければなりません。データは、必要な場合右側に 1 バイト・スペース (x'20') が埋められます。
CLOB	PC/IXF CLOB 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の単一バイト・コード・ページの値がデータベース列の SBCS CPGID と等しく、 PC/IXF 列の 2 バイト・コード・ページの値がゼロか、データベース列の DBCS CPGID と等しい場合には、PC/IXF CHAR、VARCHAR、 LONG VARCHAR、 CLOB、CLOB_FILE、または CLOB_LOCATION_SPECIFIER 列が受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。
DATE	データベース列と同じ DATE 列が作成されます。	タイプ DATE の PC/IXF 列は通常の入力です。インポート・ユーティリティーは、長さに互換性がないものを除くすべての文字タイプの列を受け入れようとします。 PC/IXF ファイル内の文字タイプの列の内容は、ターゲット・データベースの地域コードと互換性のあるフォーマットの日付でなければなりません。

表 54. PC/IXF ファイル・フォーマットで受け入れられるデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
DBCLOB	PC/IXF DBCLOB 列が作成されます。データベース列の最大長、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と等しい場合は、 PC/IXF GRAPHIC、 VARGRAPHIC、 LONG VARGRAPHIC、 DBCLOB、 DBCLOB_FILE、または DBCLOB_LOCATION_SPECIFIER 列が受け入れ可能です。 PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。
DECIMAL	データベース列と同じ DECIMAL 列が作成されます。列の精度と位取りが列記述子レコードに保管されます。	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。インポート先の DECIMAL 列の範囲外の値があれば、その特定の値に関してはリジェクトされます。
DECFLOAT	データベース列と同じ DECFLOAT 列が作成されます。列の精度が列記述子レコードに保管されます。	SMALLINT、INTEGER、BIGINT (DECFLOAT(34) のみ)、 DECIMAL、FLOAT、REAL、DOUBLE、または DECFLOAT(16) (DECFLOAT(34) のみ) 型の列が受け入れられます。DECFLOAT の場合は他の数値列タイプも有効ですが、ターゲットの精度に適合しない場合は、丸められます。
FLOAT	データベース列と同じ FLOAT 列が作成されます。	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。すべての値は範囲内です。
GRAPHIC (DBCS のみ)	PC/IXF GRAPHIC 列が作成されます。データベース列の長さ、SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、 PC/IXF GRAPHIC、 VARGRAPHIC、または LONG VARGRAPHIC 列が受け入れ可能です。PC/IXF 列が固定長である場合、その長さはデータベース列の長さと互換性がなければなりません。データは、必要な場合右側に 2 バイト・スペース (x'8140') が埋められます。
INTEGER	データベース列と同じ INTEGER 列が作成されます。	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。 -2,147,483,648 から 2,147,483,647 の範囲外の整数があれば、それに関してはリジェクトされます。

表 54. PC/IXF ファイル・フォーマットで受け入れられるデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
LONG VARCHAR	PC/IXF LONG VARCHAR 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、 VARCHAR、または LONG VARCHAR 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 データベース列が FOR BIT DATA としてマークされている場合、PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC 列も受け入れ可能です。どちらの場合でも、PC/IXF 列が固定長なら、その長さはデータベース列の最大長と互換性がなければなりません。
LONG VARGRAPHIC (DBCS のみ)	PC/IXF LONG VARGRAPHIC 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、 PC/IXF GRAPHIC、 VARGRAPHIC、または LONG VARGRAPHIC 列が受け入れ可能です。PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。
SMALLINT	データベース列と同じ SMALLINT 列が作成されます。	すべての数値タイプ (SMALLINT、 INTEGER、 BIGINT、 DECIMAL、または FLOAT) の列が受け入れられます。 -32,768 から 32,767 の範囲外の値があれば、その特定の値に関してはリジェクトされます。
TIME	データベース列と同じ TIME 列が作成されます。	タイプ TIME の PC/IXF 列は通常の入力です。インポート・ユーティリティーは、長さに互換性がないものを除くすべての文字タイプの列を受け入れようとします。 PC/IXF ファイル内の文字タイプの列の内容は、ターゲット・データベースの地域コードと互換性のあるフォーマットの時刻データでなければなりません。

表 54. PC/IXF ファイル・フォーマットで受け入れられるデータ・タイプの形式 (続き)

データ・タイプ	エクスポート・ユーティリティーによって作成されるファイルの形式	インポート・ユーティリティーにとって受け入れ可能な形式
TIMESTAMP	データベース列と同じ TIMESTAMP 列が作成されます。	タイプ TIMESTAMP の PC/IXF 列は通常の入力です。インポート・ユーティリティーは、長さに互換性がないものを除くすべての文字タイプの列を受け入れようとします。PC/IXF ファイルの文字タイプの列に入っているデータは、タイム・スタンプの入力フォーマットになっていなければなりません。
VARCHAR	データベース列の最大長が = 254 の場合は、 PC/IXF VARCHAR 列が作成されます。データベース列の最大長が > 254 の場合は、 PC/IXF LONG VARCHAR 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	次の場合、PC/IXF CHAR、 VARCHAR、または LONG VARCHAR 列が受け入れ可能です。 • データベース列が FOR BIT DATA としてマークされている場合。 • PC/IXF 列の 1 バイト・コード・ページ値がデータベース列の SBCS CPGID と等しく、PC/IXF 列の 2 バイト・コード・ページ値が 0 またはデータベース列の DBCS CPGID と等しい場合。 データベース列が FOR BIT DATA としてマークされている場合、PC/IXF GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC 列も受け入れ可能です。どちらの場合でも、PC/IXF 列が固定長なら、その長さはデータベース列の最大長と互換性がなければなりません。
VARGRAPHIC (DBCS のみ)	データベース列の最大長が = 127 の場合は、 PC/IXF VARGRAPHIC 列が作成されます。データベース列の最大長が > 127 の場合は、 PC/IXF LONG VARGRAPHIC 列が作成されます。データベース列の最大長、 SBCS CPGID 値、および DBCS CPGID 値が列記述子レコードにコピーされます。	PC/IXF 列の 2 バイト・コード・ページ値がデータベース列と同じである場合は、 PC/IXF GRAPHIC、 VARGRAPHIC、または LONG VARGRAPHIC 列が受け入れ可能です。PC/IXF 列が固定長である場合、その長さはデータベース列の最大長と互換性がなければなりません。

PC/IXF ファイルのデータベースへのインポートを制御する一般規則

データベース・マネージャーのインポート・ユーティリティーでは、SBCS または DBCS 環境での PC/IXF ファイルのインポート時に、以下の規則が適用されます。

- インポート・ユーティリティーは、PC/IXF フォーマットのファイル (IXFHID = 'IXF') のみ受け入れます。その他のフォーマットの IXF ファイルはインポートできません。
- インポート・ユーティリティーは、1024 個より多くの列が入っている PC/IXF ファイルをリジェクトします。
- IXF 形式にエクスポートするときに、ID が、IXF 形式でサポートされている最大サイズを超えていると、エクスポート操作は成功しますが、その結果として生成されるデータ・ファイルを後から CREATE モードのインポート操作で使用することはできません。SQL27984W が戻されます。

  注: IMPORT コマンドの CREATE オプションと REPLACE_CREATE オプションは非推奨で、将来のリリースでは廃止される可能性があります。
- PC/IXF H レコードの IXFHSBCP の値は、SBCS CPGID と等しくなければなりません。あるいは、IXFHSBCP/IXFHDBCP と、ターゲット・データベースの SBCS/DBCS CPGID との間の変換表がなければなりません。 IXFHDBCP の値は '00000' か、またはターゲット・データベースの DBCS CPGID と等しくなければなりません。これらの条件のいずれも満たされない場合、インポート・ユーティリティーは、 FORCEIN オプションが指定されない限り、PC/IXF ファイルをリジェクトします。
- 無効なデータ・タイプ - 新しい表

  PC/IXF ファイルの*新しい* 表へのインポートは、IMPORT コマンドの CREATE または REPLACE_CREATE キーワードによって指定されます。新しい表へのインポートにおいて無効なデータ・タイプの PC/IXF 列が選択されると、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体がリジェクトされ、表は作成されず、データはインポートされません。
- 無効なデータ・タイプ - 既存の表

  PC/IXF ファイルの*既存の* 表へのインポートは、IMPORT コマンドの INSERT、INSERT_UPDATE、REPLACE、または REPLACE_CREATE キーワードによって指定されます。既存の表へのインポートにおいて無効なデータ・タイプの PC/IXF 列が選択されると、次のいずれかの処理が実行されます。
  - ターゲット表の列が NULL 可能である場合は、無効な PC/IXF 列のすべての値が無視され、表の列の値は NULL にセットされます。
  - ターゲット表の列が NULL 可能でない場合、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体がリジェクトされ、データはインポートされません。既存の表は未変更のままです。
- 新しい表へのインポートにおいて、NULL 可能な PC/IXF 列は NULL 可能なデータベース列を生成し、NULL 可能でない PC/IXF 列は NULL 可能でないデータベース列を生成します。
- NULL 可能でない PC/IXF 列を、NULL 可能なデータベース列にインポートすることができます。
- NULL 可能 PC/IXF 列を、NULL 可能でないデータベース列にインポートすることができます。 PC/IXF 列内で NULL 値が検出されると、インポート・ユーティリティーは NULL 値を備えた PC/IXF 行のすべての列の値をリジェクトし (行全体がリジェクトされ)、次の PC/IXF 行から処理が継続されます。つまり、

NULL のターゲット表の列が NULL 可能でない場合、その NULL 値の入った PC/IXF 行からはデータがインポートされません。

- 互換性のない列 - 新しい表

  新しい データベース表へのインポート中に、ターゲット・データベース列と互換性がない PC/IXF 列が選択されると、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体がリジェクトされ、表は作成されず、データはインポートされません。

  注: IMPORT の FORCEIN オプションを使用すると、互換性のある列の範囲が広がります。

- 互換性のない列 - 既存の表

  既存の データベース表へのインポート中に、ターゲット・データベース列と互換性がない PC/IXF 列が選択されると、次の 2 つの処理のうちいずれかが実行されます。

  – ターゲット表の列が NULL 可能である場合は、PC/IXF 列のすべての値が無視され、表の列の値は NULL にセットされます。
  – ターゲット表の列が NULL 可能でない場合、インポート・ユーティリティーは終了します。 PC/IXF ファイル全体がリジェクトされ、データはインポートされません。既存の表は未変更のままです。

  注: IMPORT の FORCEIN オプションを使用すると、互換性のある列の範囲が広がります。

- 無効な値

  インポート中に、ターゲット・データベース列にとって無効な PC/IXF 列値が検出されると、インポート・ユーティリティーは、無効な値の入った PC/IXF 行のすべての列の値をリジェクトし (行全体がリジェクトされ)、次の PC/IXF 行から処理が継続されます。

PC/IXF ファイルのデータベースへのインポートを制御するデータ・タイプ固有の規則

- 有効な PC/IXF 数値列は、互換性のある任意のデータベース数値列にインポートできます。 4 バイトの浮動小数点データが入っている PC/IXF 列は無効なデータ・タイプであるため、インポートできません。
- データベースの日付/時刻列は、対応する PC/IXF 日付/時刻列 (DATE、TIME、および TIMESTAMP) からの値、および列の長さおよび値の互換性制限に従っている PC/IXF 文字タイプ列 (CHAR、VARCHAR、および LONG VARCHAR) からの値を受け入れることができます。
- 有効な PC/IXF 文字タイプ列 (CHAR、VARCHAR、または LONG VARCHAR) は、FOR BIT DATA としてマークされている*既存の* データベース文字タイプ列に常にインポートできます。それ以外の場合、
  – IXFCSBCP と SBCS CPGID は一致していなければなりません。
  – IXFCSBCP/IXFCDBCP と SBCS/DBCS のための変換表がなければなりません。
  – 1 つのセットがすべて 0 (FOR BIT DATA) でなければなりません。

IXFCSBCP が 0 でない場合、IXFCDBCP の値は 0、またはターゲット・データベース列の DBCS CPGID の値と等しくなければなりません。

これらの条件のいずれかが満たされていない場合、PC/IXF とデータベース列には互換性がありません。

有効な PC/IXF 文字タイプ列を*新しい* データベース表にインポートする場合、IXFCSBCP の値は 0 またはデータベースの SBCS CPGID と等しくなければなりません。あるいは変換表がなければなりません。 IXFCSBCP が 0 の場合は、IXFCDBCP も 0 でなければなりません (そうでなければ、PC/IXF 列は無効なデータ・タイプです)。この場合、IMPORT は、新しい表の中に FOR BIT DATA としてマークされた文字タイプの列を作成します。 IXFCSBCP が 0 でなくデータベースの SBCS CPGID と等しい場合、 IXFCDBCP の値は 0 またはデータベースの DBCS CPGID でなければなりません。この場合、ユーティリティーは、新しい表の中に SBCS および DBCS CPGID 値がデータベースと等しい文字タイプの列を作成します。これらの条件が満たされていない場合、PC/IXF とデータベース列には互換性がありません。

FORCEIN オプションを使用すると、コード・ページの同等性チェックをオーバーライドすることができます。しかし、IXFCSBCP が 0 で IXFCDBCP が 0 でない PC/IXF 文字タイプ列は無効なデータ・タイプであり、FORCEIN が指定されていてもインポートできません。

- 有効な PC/IXF 文字タイプ列 (GRAPHIC、VARGRAPHIC、または LONG VARGRAPHIC) は、 FOR BIT DATA としてマークされている*既存の* データベース文字タイプ列に常にインポートできます。 FORCEIN オプションを使用すると、この制限を緩和することができます。しかし、IXFCSBCP が 0 でないか、または IXFCDBCP が 0 の PC/IXF GRAPHIC 列は無効なデータ・タイプであり、FORCEIN が指定されていてもインポートできません。

  有効な PC/IXF GRAPHIC 列をデータベースの GRAPHIC 列にインポートする場合、 IXFCDBCP の値はターゲット・データベース列の DBCS CPGID と等しくなければなりません (つまり 2 つの列の 2 バイト・コード・ページが一致していなければなりません)。
- PC/IXF ファイルの既存のデータベース表へのインポート中に固定長ストリング列 (CHAR または GRAPHIC) が選択され、その長さがターゲット列の最大長より長い場合、それらの列には互換性がありません。
- PC/IXF ファイルの既存のデータベース表へのインポート中に、可変長ストリング列 (VARCHAR、LONG VARCHAR、VARGRAPHIC、または LONG VARGRAPHIC) が選択され、その長さがターゲット列の最大長より長い場合、それらの列には互換性があります。個々の値は、データベース・マネージャー INSERT 互換規則に従って、ターゲット・データベース列にとって長すぎる PC/IXF 値は無効です。
- 固定長のデータベースの文字 タイプの列 (つまり CHAR 列) にインポートされる PC/IXF 値は、必要な場合値の長さがデータベース列と等しくなるよう右側に 1 バイト・スペース (0x20) が埋められます。固定長のデータベースの *GRAPHIC* タイプの列 (つまり GRAPHIC 列) にインポートされる PC/IXF 値は、必要な場合値の長さがデータベース列と等しくなるよう右側に 2 バイト・スペース (0x8140) が埋められます。

- PC/IXF VARCHAR 列の最大長は 254 バイトであるため、最大長が *n* (254 *n* 4001) の PC/IXF VARCHAR 列は、最大長が *n* のデータベース LONG VARCHAR 列にエクスポートしなければなりません。
- PC/IXF LONG VARCHAR 列の最大長は 32,767 バイトで、データベース LONG VARCHAR 列には 32,700 バイトの最大長制限がありますが、長さが 32,700 バイトを超える (ただし 32,768 バイトよりも小さい) PC/IXF LONG VARCHAR 列は有効であり、データベース LONG VARCHAR 列にインポートできます。ただし、データが失われる可能性があります。
- PC/IXF VARGRAPHIC 列の最大長は 127 バイトであるため、最大長が *n* (127 *n* 2001) の PC/IXF VARGRAPHIC 列は、最大長が *n* のデータベース LONG VARGRAPHIC 列にエクスポートしなければなりません。
- PC/IXF LONG VARGRAPHIC 列の最大長は 16,383 バイトで、データベース LONG VARGRAPHIC 列には 16,350 バイトの最大長制限がありますが、長さが 16,350 バイトを超える (ただし 16,384 バイトよりも小さい) PC/IXF LONG VARGRAPHIC 列は有効であり、データベース LONG VARGRAPHIC 列にインポートできます。ただし、データが失われる可能性があります。

FORCEIN オプションを使用せずに PC/IXF ファイルを新しいデータベース表または既存のデータベース表にインポートする場合について、表 55 および表 56 にまとめます。

表 55. FORCEIN オプションを使用しない場合の PC/IXF ファイルのインポートのサマリー - 数値タイプ

	データベースの列データ・タイプ					
PC/IXF の列データ・タイプ	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**DFP**	**FLT**
-SMALLINT	N					
	E	E	E	E[a]	E	E
-INTEGER		N				
	E[a]	E	E	E[a]	E	E
-BIGINT			N			
	E[a]	E[a]	E	E[a]	E	E
-DECIMAL				N		
	E[a]	E[a]	E[a]	E[a]	E	E
-DECFLOAT						
	E[a]	E[a]	E[a]	E[a]	E	E[a]
-FLOAT						N
	E[a]	E[a]	E[a]	E[a]	E	E

[a] ターゲットの数値データ・タイプの範囲内にない値は、その特定の値に関してリジェクトされます。

表 56. FORCEIN オプションを使用しない場合の PC/IXF ファイルのインポートのサマリー - 文字、グラフィック、および日時の各タイプ

	データベースの列データ・タイプ						
PC/IXF の列データ・タイプ	**(0,0)**	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
-(0,0)	N						

表 56. FORCEIN オプションを使用しない場合の PC/IXF ファイルのインポートのサマリー - 文字、グラフィック、および日時の各タイプ (続き)

	データベースの列データ・タイプ						
PC/IXF の列データ・タイプ	**(0,0)**	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
	E				E[c]	E[c]	E[c]
-(SBCS,0)		N	N				
	E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)			N		E[c]	E[c]	E[c]
	E		E				
-GRAPHIC				N			
	E			E			
-DATE					N		
					E		
-TIME						N	
						E	
-TIME STAMP							N
							E

[b] データ・タイプは DBCS 環境でのみ使用可能です。

[c] 有効な日付または時刻の値でない値は、その特定の値に関してリジェクトされます。

[d] データ・タイプは DBCS 環境では使用不可です。

注:

1. この表は、すべての有効な PC/IXF およびデータベース・マネージャー・データ・タイプの一覧表です。 PC/IXF 列をデータベース列にインポートできる場合は、 PC/IXF データ・タイプの行とデータベース・マネージャー・データ・タイプの列が交差する位置のセルに文字が示されています。 'N' は、ユーティリティーが新しいデータベース表を作成することを示します (示されているデータ・タイプのデータベース列が作成されます)。 'E' は、ユーティリティーが既存のデータベース表にデータをインポートすることを示します (示されているデータ・タイプのデータベース列は有効なターゲットです)。
2. 文字ストリング・データ・タイプは、コード・ページ属性によって区別されます。これらの属性は、(SBCS,DBCS) の順序対として示されています。ここで、
   - SBCS は 0、または文字データ・タイプの 1 バイト・コード・ページ属性の非 0 値を示します。
   - DBCS は 0、または文字データ・タイプの 2 バイト・コード・ページ属性の非 0 値を示します。
3. この表で文字タイプの PC/IXF 列が文字タイプのデータベース列にインポートできることが示されている場合、それらのコード・ページ属性の対の値はコード・ページの同等性を制御する規則を満たします。

PC/IXF およびバージョン 0 の System/370 IXF の相違

以下に、PC/IXF (データベース・マネージャーによって使用される) と、バージョン 0 System/370 IXF (いくつかのホスト・データベース製品によって使用される) の間の相違点について説明します。

- PC/IXF ファイルは、EBCDIC 指向ではなく ASCII です。 PC/IXF ファイルでは、H レコードの新しいコード・ページ ID や列記述子レコードでの実際のコード・ページ値の使用を含め、コード・ページ識別機能が大幅に拡張されています。さらに、文字データの列を FOR BIT DATA としてマークするためのメカニズムもあります。 PC/IXF ファイル・フォーマットと、その他の IXF またはデータベース・ファイル・フォーマットとの間の変換では、FOR BIT DATA 列内の値のコード・ページ変換ができないため、 FOR BIT DATA 列には特別な意義があります。
- マシン・データ・フォーマットのみ可能です。つまり、IXFTFORM フィールドの内容は常に `M` でなければなりません。さらに、マシン・データは PC 形式でなければなりません。つまり、IXFTMFRM フィールドの値は `PC` でなければなりません。したがって、PC/IXF データ・レコードのデータ部分の整数、浮動小数点数、および 10 進数は PC 形式でなければなりません。
- PC/IXF ファイルにおいて、H レコードの後であればどこにでもアプリケーション (A) レコードを入れることができます。それらは、IXFHHCNT フィールドの値の計算ではカウントされません。
- すべての PC/IXF レコードはレコード長標識で始まります。これは、PC/IXF レコードのうちレコード長標識自体を除く長さの整数値を 6 バイトの文字で表記したものです (つまり合計レコード長 - 6 バイト)。レコード長フィールドの目的は、PC プログラムがレコード境界を識別できるようにすることです。
- 可変長データによるストレージの節約を活用するため、またフィールドが複数のレコードに分割されている場合の複雑な処理を回避するため、 PC/IXF では、バージョン 0 の IXF X レコードはサポートされませんが、D レコード ID はサポートされます。可変長フィールドまたは NULL 可能フィールドがデータの D レコードの最後のフィールドである場合には、そのフィールドの最大長の全体を PC/IXF ファイルに書き込む必要はありません。

FORCEIN オプション

`forcein` ファイル・タイプ修飾子を使用すると、PC/IXF ファイルとターゲット・データベースとでデータのコード・ページが違っていても、PC/IXF ファイルをインポートすることができます。このオプションにより、互換性のある列を定義する上でさらに柔軟性が高くなります。

forcein の一般的なセマンティクス

SBCS または DBCS 環境で `forcein` ファイル・タイプ修飾子を使用する場合、下記の一般的なセマンティクスが適用されます。

- `forcein` ファイル・タイプ修飾子を使用する際には、注意が必要です。普通は、このオプションを使用可能にせずにインポートを試みるようにしてください。しかし、PC/IXF データ交換アーキテクチャーの一般的な性質のために、一部の PC/IXF ファイルには、介入なしではインポートできないデータ・タイプまたは値が収められている可能性があります。

- forcein を使用して*新しい* 表へのインポートを実行する場合、既存の表へのインポートとは異なる結果が生じる可能性があります。既存の表には、PC/IXF の各データ・タイプごとに事前定義のターゲット・データ・タイプが対応しています。
- lobsinfile ファイル・タイプ修飾子を使用して LOB データがエクスポートされている場合に、ファイルがコード・ページの異なる別のクライアントに移動されると、それぞれ別個のファイル内の CLOB および DBCLOB は、その他のデータの場合とは異なり、データベースへのインポートまたはロード時にクライアントのコード・ページに変換されません。

forcein のコード・ページのセマンティクス

SBCS または DBCS 環境で forcein ファイル・タイプ修飾子を使用する場合、下記のコード・ページのセマンティクスが適用されます。

- forcein ファイル・タイプ修飾子を指定すると、インポート・ユーティリティーのすべてのコード・ページ比較が使用できなくなります。

  この規則は、新しいデータベース表または既存のデータベース表へのインポート時に、列レベルとファイル・レベルのコード・ページ比較に適用されます。列 (例えばデータ・タイプ) レベルでは、この規則は次のデータベース・マネージャーおよび PC/IXF データ・タイプにのみ適用されます。つまり、文字 (CHAR、VARCHAR、および LONG VARCHAR) と GRAPHIC (GRAPHIC、VARGRAPHIC、および LONG VARGRAPHIC)。この制限は、その他のデータ・タイプのコード・ページ属性がデータ・タイプ値の解釈に関係しないという事実に基づいています。
- forcein を指定しても、データ・タイプを判別するためのコード・ページ属性の検査は使用不可にされません。

  例えば、データベース・マネージャーでは、FOR BIT DATA 属性を使用して CHAR 列を宣言することができます。このような宣言により、その列の SBCS CPGID と DBCS CPGID の両方が 0 に設定されます。それらの CPGID の値が 0 の場合、列値は文字ストリングではなくビット・ストリングとして識別されます。
- forcein は、コード・ページ変換を暗黙指定しません。

  forcein ファイル・タイプ修飾子の影響を受けるデータ・タイプの値は、「元のまま」コピーされます。コード・ページ環境の変更に対応するためのコード・ポイント・マッピングは使用されません。ターゲット列が固定長の場合には、インポートされた値にスペースを埋めることが必要になることがあります。
- forcein を使用して*既存の* 表にデータをインポートする場合には、
  - ターゲット・データベース表および列のコード・ページ値が常に優先されます。
  - PC/IXF ファイルおよび列のコード・ページ値は無視されます。

  この規則は、forcein が使用されるかどうかに関係なく適用されます。データベース・マネージャーは、データベースの作成後にデータベースまたは列のコード・ページ値の変更を許可しません。
- forcein を使用して*新しい* 表へのインポートを実行する場合には、
  - ターゲット・データベースのコード・ページ値が優先されます。

– IXFCSBCP = IXFCDBCP = 0 の文字タイプの PC/IXF 列は、FOR BIT DATA としてマークされた表の列を生成します。
– その他のすべての PC/IXF 文字タイプ列は、 SBCS および DBCS CPGID 値がデータベースと等しい文字タイプの表の列を生成します。
– PC/IXF GRAPHIC 列は、SBCS CPGID が「未定義」で、DBCS CPGID がデータベースと等しい表 GRAPHIC 列を生成します (DBCS 環境のみ)。

forcein の例

IXFCSBCP = '00897' および IXFCDBCP = '00301' である PC/IXF CHAR 列を考察してみます。この列を、SBCS CPGID = '00850' および DBCS CPGID = '00000' のデータベース CHAR 列にインポートするとします。 forcein を指定しない場合、ユーティリティーは終了し、データがインポートされないか、あるいは PC/IXF 列値が無視され、データベース列に NULL が入れられます (データベース列が NULL 可能である場合)。 forcein を指定した場合、ユーティリティーは、コード・ページの非互換性を無視して処理を継続します。データ・タイプにその他 (長さなど) の非互換性がなければ、PC/IXF 列の値は「元のまま」インポートされ、データベース列のコード・ページ環境で解釈できるようになります。

以下の 2 つの表には次の点が示されています。

- PC/IXF ファイルのうち指定されたコード・ページ属性を持つデータ・タイプがインポートされる場合に、*新しい* データベース表で作成される列のコード・ページ。
- PC/IXF データ・タイプが無効であるか、または互換性がない場合に、それらがインポート・ユーティリティーによってリジェクトされること。

表 57. インポート・ユーティリティーのコード・ページに関するセマンティクスのサマリー (新しい表) (SBCS 用): この表では、a と x の間の変換表がないことを想定しています。もしそれがある場合、項目 3 および 4 は、forcein が使用されなくても正常に機能します。

PC/IXF データ・タイプのコード・ページ属性	データベース表の列のコード・ページ属性	
	forcein なし	forcein あり
(0,0)	(0,0)	(0,0)
(a,0)	(a,0)	(a,0)
(x,0)	リジェクト	(a,0)
(x,y)	リジェクト	(a,0)
(a,y)	リジェクト	(a,0)
(0,y)	リジェクト	(0,0)
注: 1. 表 58 の注を参照してください。		

表 58. インポート・ユーティリティーのコード・ページに関するセマンティクスのサマリー (新しい表) (DBCS 用): この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	データベース表の列のコード・ページ属性	
	forcein なし	forcein あり
(0,0)	(0,0)	(0,0)
(a,0)	(a,b)	(a,b)

表 58. インポート・ユーティリティーのコード・ページに関するセマンティクスのサマリー (新しい表) (DBCS 用) (続き): この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	データベース表の列のコード・ページ属性	
	forcein なし	forcein あり
(x,0)	リジェクト	(a,b)
(a,b)	(a,b)	(a,b)
(x,y)	リジェクト	(a,b)
(a,y)	リジェクト	(a,b)
(x,b)	リジェクト	(a,b)
(0,b)	(-,b)	(-,b)
(0,y)	リジェクト	(-,b)

注:

1. PC/IXF データ・タイプのコード・ページ属性は順序対として示されています。 x はゼロ以外の 1 バイト・コード・ページ値、y はゼロ以外の 2 バイト・コード・ページ値を表します。 '-' は未定義のコード・ページ値を表します。
2. 各種のコード・ページ属性で、あえて異なる文字を使用しています。異なる文字は値が異なることを暗示しています。例えば、PC/IXF データ・タイプが (x,y) として示されており、データベース列が (a,y) として示されている場合、x は a と等しくありませんが、 PC/IXF ファイルとデータベースの 2 バイト・コード・ページ値は同じ y です。
3. forcein のコード・ページのセマンティクスによって影響を受けるのは、文字および GRAPHIC データ・タイプだけです。
4. 新しい表の入ったデータベースのコード・ページ属性は (a,0) であることが想定されています。このため、新しい表のうち文字タイプのすべての列のコード・ページ属性は (0,0) または (a,0) でなければなりません。

   DBCS 環境において、新しい表を収めたデータベースのコード・ページ属性は (a,b) であることが想定されています。そのため、新しい表のうちすべての GRAPHIC 列のコード・ページ属性は (-,b) でなければならず、文字タイプのすべての列のコード・ページは (a,b) でなければなりません。 SBCS CPGID は、GRAPHIC データ・タイプの場合は未定義であるため、'-' と示されています。
5. 結果のデータ・タイプは、『forcein のデータ・タイプのセマンティクス』で説明されている規則によって決まります。
6. リジェクトの結果は、無効なまたは互換性のないデータ・タイプに関する規則を反映したものです。

以下の 2 つの表には次の点が示されています。

- インポート・ユーティリティーが、さまざまなコード・ページ属性を持つ PC/IXF データ・タイプを、指定されたコード・ページ属性を持つ*既存の* 表の列 (*ターゲット* 列) に受け入れること。
- インポート・ユーティリティーが、特定のコード・ページ属性を持つ PC/IXF データ・タイプを、指定されたコード・ページ属性を持つ*既存の* 表の列にインポートするのを許可しないこと。ユーティリティーは、PC/IXF データ・タイプが無効であるか、または互換性がない場合、それらをリジェクトします。

表 59. インポート・ユーティリティーのコード・ページに関するセマンティクスのサマリー (既存の表) (SBCS 用): この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	ターゲット・データベース列のコード・ページ属性	インポートの結果	
		forcein なし	forcein あり
(0,0)	(0,0)	受け入れ	受け入れ
(a,0)	(0,0)	受け入れ	受け入れ
(x,0)	(0,0)	受け入れ	受け入れ
(x,y)	(0,0)	受け入れ	受け入れ
(a,y)	(0,0)	受け入れ	受け入れ
(0,y)	(0,0)	受け入れ	受け入れ
(0,0)	(a,0)	NULL またはリジェクト	受け入れ
(a,0)	(a,0)	受け入れ	受け入れ
(x,0)	(a,0)	NULL またはリジェクト	受け入れ
(x,y)	(a,0)	NULL またはリジェクト	受け入れ
(a,y)	(a,0)	NULL またはリジェクト	受け入れ
(0,y)	(a,0)	NULL またはリジェクト	NULL またはリジェクト

注:

1. 492 ページの表 57 の注を参照してください。
2. NULL またはリジェクトの結果は、無効なまたは互換性のないデータ・タイプに関する規則を反映したものです。

表 60. インポート・ユーティリティーのコード・ページに関するセマンティクスのサマリー (既存の表) (DBCS 用): この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	ターゲット・データベース列のコード・ページ属性	インポートの結果	
		forcein なし	forcein あり
(0,0)	(0,0)	受け入れ	受け入れ
(a,0)	(0,0)	受け入れ	受け入れ
(x,0)	(0,0)	受け入れ	受け入れ
(a,b)	(0,0)	受け入れ	受け入れ
(x,y)	(0,0)	受け入れ	受け入れ
(a,y)	(0,0)	受け入れ	受け入れ
(x,b)	(0,0)	受け入れ	受け入れ
(0,b)	(0,0)	受け入れ	受け入れ
(0,y)	(0,0)	受け入れ	受け入れ

表 60. インポート・ユーティリティーのコード・ページに関するセマンティクスのサマリー (既存の表) (DBCS 用) (続き): この表では、a と x の間の変換表がないことを想定しています。

PC/IXF データ・タイプのコード・ページ属性	ターゲット・データベース列のコード・ページ属性	インポートの結果	
		forcein なし	forcein あり
(0,0)	(a,b)	NULL またはリジェクト	受け入れ
(a,0)	(a,b)	受け入れ	受け入れ
(x,0)	(a,b)	NULL またはリジェクト	受け入れ
(a,b)	(a,b)	受け入れ	受け入れ
(x,y)	(a,b)	NULL またはリジェクト	受け入れ
(a,y)	(a,b)	NULL またはリジェクト	受け入れ
(x,b)	(a,b)	NULL またはリジェクト	受け入れ
(0,b)	(a,b)	NULL またはリジェクト	NULL またはリジェクト
(0,y)	(a,b)	NULL またはリジェクト	NULL またはリジェクト
(0,0)	(-,b)	NULL またはリジェクト	受け入れ
(a,0)	(-,b)	NULL またはリジェクト	NULL またはリジェクト
(x,0)	(-,b)	NULL またはリジェクト	NULL またはリジェクト
(a,b)	(-,b)	NULL またはリジェクト	NULL またはリジェクト
(x,y)	(-,b)	NULL またはリジェクト	NULL またはリジェクト
(a,y)	(-,b)	NULL またはリジェクト	NULL またはリジェクト
(x,b)	(-,b)	NULL またはリジェクト	NULL またはリジェクト
(0,b)	(-,b)	受け入れ	受け入れ
(0,y)	(-,b)	NULL またはリジェクト	受け入れ

注:

1. 492 ページの表 57 の注を参照してください。
2. NULL またはリジェクトの結果は、無効なまたは互換性のないデータ・タイプに関する規則を反映したものです。

forcein のデータ・タイプのセマンティクス

forcein ファイル・タイプ修飾子を使用すると、特定の PC/IXF 列を、データ・タイプが違う、またはその他の点で互換性がないデータ・タイプのターゲット・データベース列にインポートすることができます。SBCS または DBCS 環境で forcein を使用する場合、下記のデータ・タイプのセマンティクスが適用されます (一部の例外を除く)。

- SBCS 環境では、forcein により、次のインポートが可能になります。
  - PC/IXF BIT データ・タイプ (PC/IXF の文字タイプの列で IXFCSBCP = 0 = IXFCDBCP) を、文字タイプのデータベース列 (SBCS CPGID がゼロ以外、DBCS CPGID = 0) にインポートすること。既存の表のみ。
  - PC/IXF MIXED データ・タイプ (IXFCSBCP および IXFCDBCP がゼロ以外) を文字タイプのデータベース列にインポートすること。新しい表と既存の表の両方。
  - PC/IXF GRAPHIC データ・タイプを、データベース FOR BIT DATA 列 (SBCS CPGID = 0 = DBCS CPGID) にインポートすること。新しい表のみ (既存の表では常に可能)。
- forcein ファイル・タイプ修飾子は、有効な PC/IXF データ・タイプの範囲を拡張しません。

  有効な PC/IXF データ・タイプとして定義されていないデータ・タイプの PC/IXF 列は、 forcein が指定されているかどうかにかかわりなく、インポートでは無効です。
- DBCS 環境では、forcein により、次のインポートが可能になります。
  - PC/IXF BIT データ・タイプを文字タイプのデータベース列にインポートすること。
  - PC/IXF BIT データ・タイプをデータベース GRAPHIC 列にインポートすること。ただし、PC/IXF BIT 列が固定長の場合、長さは偶数でなければなりません。長さが奇数である固定長の PC/IXF BIT 列は、データベース GRAPHIC 列と互換性がありません。可変長の PC/IXF BIT 列は、長さが奇数であっても偶数であっても、互換性があります。ただし、可変長列の奇数の長さの値は、データベース GRAPHIC 列へのインポートでは無効です。
  - PC/IXF MIXED データ・タイプを文字タイプのデータベース列にインポートすること。

表 61 は、forcein を指定して PC/IXF ファイルを新しいデータベース表または既存のデータベース表にインポートする操作をまとめたものです。

表 61. forcein を使用する場合の PC/IXF ファイルのインポートのサマリー

	データベースの列データ・タイプ											
PC/IXF の列データ・タイプ	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
-SMALLINT	N											
	E	E	E	E[a]	E							
-INTEGER		N										

表 61. forcein を使用する場合の PC/IXF ファイルのインポートのサマリー (続き)

	データベースの列データ・タイプ											
PC/IXF の列データ・タイプ	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
	E[a]	E	E	E[a]	E							
-BIGINT			N									
	E[a]	E[a]	E	E[a]	E							
-DECIMAL				N								
	E[a]	E[a]	E[a]	E[a]	E							
-FLOAT					N							
	E[a]	E[a]	E[a]	E[a]	E							
-(0,0)						N						
						E	E w/F	E w/F	E w/F	E[c]	E[c]	E[c]
-(SBCS,0)							N	N				
						E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)							N w/F[d]	N		E[c]	E[c]	E[c]
						E	E w/F	E				
-GRAPHIC						N w/F[d]			N			
						E			E			
-DATE										N		
										E		
-TIME											N	
											E	
-TIME STAMP												N
												E

注: forcein を使用しなければ PC/IXF 列をデータベース列にインポートできない場合は、'N' または 'E' に 'w/F' を付けて示しています。 'N' は、ユーティリティーが新しいデータベース表を作成することを示します。'E' は、ユーティリティーが既存のデータベース表にデータをインポートすることを示します。 forcein ファイル・タイプ修飾子は、文字および GRAPHIC データ・タイプの互換性にのみ影響を与えます。

[a] ターゲットの数値データ・タイプの範囲内にない値は、その特定の値に関してリジェクトされます。

[b] データ・タイプは DBCS 環境でのみ使用可能です。

[c] 有効な日付または時刻の値でない値は、その特定の値に関してリジェクトされます。

[d] ソース PC/IXF データ・タイプがターゲット・データベースによってサポートされない場合にのみ適用されます。

[e] データ・タイプは DBCS 環境では使用不可です。

ワークシート・ファイル・フォーマット (WSF)

Lotus 1-2-3 と Symphony の製品では、同じ基本フォーマットが使用されており、それぞれの新しいリリースでの追加の機能があります。データベース・マネージャーでは、すべての Lotus 製品で同じであるワークシート・レコードのサブセットがサポートされます。つまり、データベース・マネージャーによってサポートされる Lotus 1-2-3 および Lotus Symphony 製品のリリースでは、3 文字の拡張子 (例えば WKS、WK1、WRK、WR1、WJ2) の付いたすべてのファイル名が受け入れられます。

それぞれの WSF ファイルは 1 つのワークシートを表現します。データベース・マネージャーは、次の規則を使用してワークシートを解釈し、そのエクスポート操作によって生成されるワークシートに整合性を提供します。

- 最初の行 (ROW 値 0) のセルは、ワークシート全体に関する記述情報のために予約されています。この行のすべてのデータはオプションです。それらは、インポート時には無視されます。
- 2 番目の行 (ROW 値 1) のセルは、列のラベルとして使用されます。
- 残りの行は、データ行 (レコードまたは表のデータ行) です。
- ある列見出しの下にあるセル値は、その特定の列またはフィールドの値です。
- NULL 値は、セル内容レコードのうち特定の行の中で、特定の列に対応する実際のセル内容レコードがないことによって示されます (例えば整数、数値、ラベル、または公式レコードがない場合)。

  注: NULL で構成される行は、インポートもエクスポートもされません。

エクスポート操作中に、WSF フォーマットに準拠したファイルを作成すると、いくらかのデータが失われることがあります。

WSF ファイルでは、Lotus のコード・ポイント・マッピングが使用されます。それは、 DB2 データベースによってサポートされる既存のコード・ページと必ずしも同じではありません。その結果、WSF ファイルのインポートまたはエクスポート時に、Lotus のコード・ポイントと、アプリケーションのコード・ページによって使用されるコード・ポイントの間でデータ変換が実行されます。 DB2 では、Lotus のコード・ポイントと、コード・ページ 437、819、850、860、863、および 865 によって定義されるコード・ポイントとの間の変換がサポートされています。

注: マルチバイト文字セット・ユーザーの場合、変換は実行されません。

データの移動に関する Unicode のための考慮事項

エクスポート、インポート、およびロード・ユーティリティーは、非 Unicode データベースに接続されている Unicode クライアントで使用される場合にはサポートされません。

この項で説明されているとおり、 Unicode データベースには DEL、ASC、および PC/IXF ファイル・フォーマットがサポートされます。 WSF フォーマットはサポートされません。

Unicode データベースから ASCII 区切り文字付き (DEL) ファイルにエクスポートする際に、すべての文字データはアプリケーション・コード・ページに変換されます。文字ストリングと GRAPHIC ストリング・データの両方が、クライアントの同じ SBCS または MBCS コード・ページに変換されます。区切り文字付き ASCII ファイル全体でコード・ページは 1 つしかないため、これがどのデータベースのエクスポートの場合でも所定の動作となり、変更できません。したがって、区切り文字付き ASCII ファイルにエクスポートする場合には、ご使用のアプリケーション・コード・ページに存在する UCS-2 文字だけが保管されます。その他の文字は、アプリケーション・コード・ページのデフォルト置換文字に置き換えられます。 UTF-8 クライアント (コード・ページ 1208) の場合、 UTF-8 クライアントによりすべての UCS-2 文字がサポートされているため、データの脱落はありません。

ASCII ファイル (DEL または ASC) から Unicode データベースにインポートする場合、文字ストリング・データはアプリケーション・コード・ページから UTF-8 に変換され、 GRAPHIC ストリング・データはアプリケーション・コード・ページから UCS-2 に変換されます。データの脱落はありません。異なるコード・ページに保管された ASCII データをインポートする場合には、 IMPORT コマンドを発行する前にデータ・ファイル・コード・ページを変更しなければなりません。
DB2CODEPAGE レジストリー変数を ASCII データ・ファイルのコード・ページに設定するか、または `codepage` ファイル・タイプ修飾子を使用することにより、データ・ファイルのコード・ページを指定できます。

SBCS および MBCS クライアントで有効な ASCII 区切り文字の範囲は、これらのクライアント用に IBM DB2 V9.1 が現在サポートしているものと同じです。
UTF-8 クライアントの有効な区切り文字の範囲は X'01' から X'7F' で、通常の制約事項が適用されます。

Unicode データベースから PC/IXF ファイルにエクスポートする際には、文字ストリング・データはクライアントの SBCS/MBCS コード・ページに変換されます。 GRAPHIC ストリング・データは変換されず、 UCS-2 (コード・ページ 1200) に保管されます。データの脱落はありません。

PC/IXF ファイルから Unicode データベースにインポートする際には、文字ストリング・データは PC/IXF ヘッダーに格納される SBCS/MBCS コード・ページに、 GRAPHIC ストリング・データは PC/IXF ヘッダーに格納される DBCS コード・ページにあるものと見なされます。文字ストリング・データは、インポート・ユーティリティーにより、 PC/IXF ヘッダーで指定されるコード・ページからクライアントのコード・ページに変換され、その後 (INSERT ステートメントによって) クライアント・コード・ページから UTF-8 に変換されます。 GRAPHIC ストリング・データは、インポート・ユーティリティーによって、 PC/IXF ヘッダーで指定される DBCS コード・ページから UCS-2 (コード・ページ 1200) に直接変換されます。

ロード・ユーティリティーは、データをデータベースに直接入れ、デフォルトでは、ASC または DEL ファイル内のデータは、データベースのコード・ページにあるものと見なします。したがって、デフォルトでは、ASCII ファイルのコード・ページ変換は実行されません。データ・ファイルのコード・ページが (`codepage` 修飾子を使用して) 明示的に指定された場合、ロード・ユーティリティーは、データをロードする前にこの情報を使用して、指定されるコード・ページからデータベース・コード・ページに変換します。 PC/IXF ファイルでは、ロード・ユーティリテ

ィーは、常に IXF ヘッダーに指定されるコード・ページからデータベース・コード・ページ (CHAR の場合は 1208、 GRAPHIC の場合は 1200) に変換します。

DBCLOB ファイルのコード・ページは常に UCS-2 では 1200 です。 CLOB ファイルのコード・ページは、インポート、ロード、あるいはエクスポートされるデータ・ファイルのコード・ページと同じです。例えば、PC/IXF フォーマットを使用してデータをロードまたはインポートする際には、 CLOB ファイルは、PC/IXF ヘッダーによって指定されるコード・ページにあるものと見なされます。 DBCLOB ファイルが ASC または DEL フォーマットの場合には、ロード・ユーティリティーは、CLOB データがデータベースのコード・ページにあるものと見なし、インポート・ユーティリティーは、これがクライアント・アプリケーションのコード・ページにあるものと見なします。

以下の理由で、Unicode データベースには nochecklengths 修飾子が常に指定されます。

- DBCS コード・ページのないデータベースには任意の SBCS を接続できます。
- UTF-8 フォーマットの文字ストリングは通常、クライアント・コード・ページにあるものとは異なる長さになります。

コード・ページ 1394、1392、および 5488 に関する考慮事項

インポート、エクスポート、およびロード・ユーティリティーを使用して、中国語コード・ページ GB 18030 (コード・ページ ID 1392 および 5488) および日本語コード・ページ ShiftJISX 0213 (コード・ページ ID 1394) から DB2 Unicode データベースにデータを転送できます。加えて、エクスポート・ユーティリティーを使用して、DB2 Unicode データベースから GB 18030 または ShiftJIS X0213 コード・ページ・データにデータを転送することもできます。

例えば、以下のコマンドは、リモートで接続されるクライアントに常駐する Shift_JISX0213 データ・ファイル u/jp/user/x0213/data.del を MYTABLE にロードします。

```
db2 load client from /u/jp/user/x0213/data.del
of del modified by codepage=1394 insert into mytable
```

ここで、MYTABLE は DB2 Unicode データベース内にあります。

Unicode クライアントと Unicode サーバーとの間の接続しかサポートされないため、 Unicode クライアントを使用するか、あるいはロード、インポート、またはエクスポート・ユーティリティーを使用する前に、 DB2 レジストリー変数 **DB2CODEPAGE** を 1208 に設定する必要があります。

コード・ページ 1394、1392、または 5488 から Unicode に変換すると、拡張が行われます。例えば、2 バイト文字は、GRAPHIC 列で 2 つの 16 ビット Unicode 文字として格納されます。 Unicode データベースのターゲット列は、拡張された Unicode バイトを入れるのに十分な幅であることを確認する必要があります。

非互換性

Unicode データベースに接続されているアプリケーションの場合、 GRAPHIC ストリング・データは常に UCS-2 (コード・ページ 1200) にあります。非 Unicode データベースに接続されるアプリケーションの場合、 GRAPHIC ストリング・データ

は、アプリケーションの DBCS コード・ページにあるか、またはアプリケーション・コード・ページが SBCS の場合には許可されません。例えば、日本語の非 Unicode データベースに 932 クライアントが接続される場合、 GRAPHIC ストリング・データはコード・ページ 301 に入れられます。 Unicode データベースに接続されている 932 クライアント・アプリケーションの場合、 GRAPHIC ストリング・データは UCS-2 エンコード方式になります。

文字セットと各国語サポート

DB2 データ移動ユーティリティーには、次のような各国語サポート (NLS) が備わっています。

- インポートおよびエクスポート・ユーティリティーには、クライアント・コード・ページからサーバー・コード・ページへの自動コード・ページ変換が備わっています。
- ロード・ユーティリティーでは、 `codepage` 修飾子とともに DEL および ASC ファイルを指定して、任意のコード・ページからサーバー・コード・ページにデータを変換することができます。
- どのユーティリティーでも、IXF データは、元のコード・ページ (IXF ファイルに保管されているもの) からサーバーのコード・ページに自動的に変換されます。

場合によって、コード・ページが異なるために文字データの長さに変化が生じることがあります。例えば、日本語または中国語 (繁体字) の拡張 UNIX コード (EUC) と 2 バイト文字セット (DBCS) では、同じ文字が別々の長さでエンコードされていることがあります。普通、入力データの長さとターゲット列の長さの比較は、まだどのデータも読み取らないうちに実行されます。入力の長さがターゲットの長さを超えている場合、列が NULL 可能であれば NULL がその列に挿入されます。そうでない場合、要求はリジェクトされます。 `nochecklengths` ファイル・タイプ修飾子を指定した場合は、初期の比較を実行しないで、データのインポートまたはロードを実行しようとします。変換完了後にデータが長すぎることが明らかになった場合、その行はリジェクトされます。そうでなければ、データはインポートされるかロードされます。

XML データ移動

XML データ移動のサポートは、ロード、インポート、およびエクスポート・ユーティリティーによって提供されます。

XML データのインポート

インポート・ユーティリティーを使用して、 XML 文書を通常のリレーショナル表に挿入できます。インポートできるのは、整形式 XML 文書だけです。

IMPORT コマンドの XML FROM オプションを使用して、インポートする XML 文書の場所を指定します。 XMLVALIDATE オプションは、インポートされた文書を妥当性検査する方法を指定します。インポートされた XML データが、IMPORT コマンドで指定されたスキーマに対して、ソース XML 文書内のスキーマ・ロケーション・ヒントによって識別されるスキーマに対して、またはメイン・データ・ファイル内の XML Data Specifier によって識別されるスキーマに対して、妥当性検査

されるように選択できます。また、XMLPARSE オプションを使用して、XML 文書がインポートされるときの空白文字の処理方法を指定できます。 xmlchar および xmlgraphic ファイル・タイプ修飾子によって、インポートされる XML データのエンコード特性を指定できます。

XML データのロード

ロード・ユーティリティーは、大量の XML データを表に挿入するための効果的な方法を提供します。このユーティリティーはまた、ユーザー定義のカーソルからロードする機能など、インポート・ユーティリティーでは使用できない特定のオプションを使用することができます。

IMPORT コマンドと同じように LOAD コマンドでも、ロードする XML データの場所、XML データの妥当性検査オプション、および空白文字の処理方法を指定できます。 IMPORT と同様に、xmlchar および xmlgraphic ファイル・タイプ修飾子を使用して、ロードされた XML データのエンコード特性を指定できます。

XML データのエクスポート

データは、XML データ・タイプの列を 1 列以上含む表からエクスポートできます。エクスポートされた XML データは、エクスポートされたリレーショナル・データを含むメイン・データ・ファイルとは別個のファイルに格納されます。各エクスポート XML 文書についての情報は、エクスポートされたメインのデータ・ファイル内で XML データ指定子 (XDS) によって表されます。 XDS は、XML 文書が保管されるシステム・ファイルの名前、このファイル内の XML 文書の正確な場所と長さ、および XML 文書の妥当性検査に使用される XML スキーマを指定するストリングです。

EXPORT コマンドの XMLFILE、XML TO、および XMLSAVESCHEMA パラメーターを使用することにより、エクスポートされた XML 文書の格納方法についての詳細を指定できます。 xmlinsepfiles、xmlnodeclaration、xmlchar、および xmlgraphic ファイル・タイプ修飾子により、エクスポートされた XML データの保管場所およびエンコード方式に関する詳細を指定できます。

XML データの移動に関する重要な考慮事項

XML データをインポートまたはエクスポートする際には、注意すべきいくつかの考慮事項があります。

- エクスポートされた XML データは、エクスポートされたリレーショナル・データを含むメイン・データ・ファイルとは別個の場所に常に保管されます。
- デフォルトで、エクスポート・ユーティリティーは XML データを Unicode で書き込みます。 xmlchar ファイル・タイプ修飾子を使用して、XML データが文字コード・ページで記述されるようにすることができます。 xmlgraphic ファイル・タイプ修飾子を使用すると、XML データは、アプリケーション・コード・ページとは関係なく UTF-16 のグラフィック・コード・ページで記述されます。
- バージョン 9.5 以降は、挿入前にデータをデータベース・コード・ページから UTF-8 に変換し、 XML データを非 Unicode データベースに保管できるようになりました。 XML 構文解析中に文字が置換されてしまうことがないようにするため、挿入する文字データはデータベース・コード・ページに含まれるコード・ポイントのみを使用して構成する必要があります。 enable_xmlchar 構成パラメ

ーターを no に設定すると、XML 構文解析中に文字データ・タイプの挿入がブロックされて、BIT DATA、BLOB や pureXML™ など、コード・ページ変換を実施しないデータ・タイプの挿入が制限されます。

- インポートおよびロード・ユーティリティーでは、インポートする XML 文書にエンコード属性を含む宣言タグが含まれていない限り、その文書は Unicode であると推定されます。 xmlchar ファイル・タイプ修飾子を使用して、インポートする XML 文書が文字コード・ページでエンコードされるように指定できます。一方、xmlgraphic ファイル・タイプ修飾子は、インポートする XML 文書が UTF-16 でエンコードされることを指定します。
- インポートおよびロード・ユーティリティーでは、整形式ではない文書を含む行は拒否されます。
- XMLVALIDATE オプションがインポート・ユーティリティーまたはロード・ユーティリティーに指定されている場合、マッチング・スキーマに対する妥当性検査が成功した文書は、表に挿入されるときにスキーマ情報のアノテーションが付けられます。対応するスキーマによる妥当性検査に失敗した文書が含まれている行はリジェクトされます。
- XQuery 指定のエクスポート・ユーティリティーを使用して、整形式 XML 文書ではない Query および XPath データ・モデル (XDM) のインスタンスをエクスポートできます。ただし、XML データ・タイプの定義された列には完全な XML 文書だけしか含めることができないので、エクスポートされた整形式ではない XML 文書は XML 列に直接インポートできません。
- ロード中の **CPU_PARALLELISM** は、統計が収集されている場合、1 に縮小されます。
- XML ロード操作を続行するには、共有ソート・メモリーを使用することが必要です。そのため、**SHEAPTHRES_SHR** または **INTRA_PARALLEL** を有効にするか、または接続コンセントレーターをオンにする必要があります。デフォルトでは **SHEAPTHRES_SHR** に値が設定されているので、共有ソート・メモリーはデフォルト構成で使用できる状態であることに注意してください。
- XML 列を含む表へのロード操作には、SOURCEUSEREXIT オプション、**SAVECOUNT** パラメーター、または anyorder ファイル・タイプ修飾子を指定できません。
- LOB ファイルの場合と同様に、XML ファイルもサーバー・サイドに存在する必要があります。

インポートおよびエクスポート時の LOB および XML ファイルの振る舞い

LOB および XML ファイルは、データをインポートおよびエクスポートする時に使用できる特定の性質および互換性を共有します。

エクスポート

データをエクスポートするときに LOBS TO オプションを付けて 1 つ以上の LOB パスを指定する場合、エクスポート・ユーティリティーはパスの間を循環し、それぞれの連続した LOB 値を適切な LOB ファイルに書き込みます。同様に、1 つ以上の XML パスが XML TO オプションで指定された場合、エクスポート・ユーティリティーはそれらのパスの間で循環して、それぞれの連続した XQuery および XPath データ・モデル (XDM) インス

タンスを該当する XML ファイルに書き込みます。デフォルトでは、LOB 値および XDM インスタンスは、エクスポートされるリレーショナル・データが書き込まれているパスと同じパスに書き込まれます。
LOBSINSEPFILES または XMLINSEPFILES ファイル・タイプ修飾子が設定されているのでないかぎり、LOB ファイルと XML ファイルは両方とも複数の値を同じファイルに連結できます。

LOBFILE オプションを指定すると、エクスポート・ユーティリティーによって生成される LOB ファイルのベース名を指定することができます。同様に、XMLFILE オプションを指定すると、エクスポート・ユーティリティーによって生成される XML ファイルのベース名を指定することができます。エクスポート・データ・ファイルの名前に拡張子 .lob を付けたものが、デフォルトの LOB ファイルのベース名になります。エクスポート・データ・ファイルの名前に拡張子 .xml を付けたものが、デフォルトの XML ファイルのベース名になります。したがって、エクスポート LOB ファイルまたは XML ファイルの完全名は、ベース名、3 桁の数値の拡張子、そして拡張子 .lob または .xml をこの順番でつなぎ合わせたもので構成されます。

インポート

データをインポートするとき、LOB ロケーション指定子 (LLS) には XML ターゲット列との互換性があり、XML データ指定子 (XDS) には LOB ターゲット列との互換性があります。 LOBS FROM オプションが指定されない場合、インポートする LOB ファイルは、入力リレーショナル・データ・ファイルと同じパスにあると見なされます。同様に、XML FROM オプションが指定されない場合、インポートする XML ファイルは、入力リレーショナル・データ・ファイルと同じパスにあると見なされます。

エクスポートの例

以下の例では、LOB 値はすべて /mypath/t1export.del.001.lob ファイルに書き込まれ、XDM インスタンスはすべて /mypath/t1export.del.001.xml ファイルに書き込まれます。

```
EXPORT TO /mypath/t1export.del OF DEL MODIFIED BY LOBSINFILE
SELECT * FROM USER.T1
```

以下の例では、最初の LOB 値は /lob1/t1export.del.001.lob ファイルに書き込まれ、2 番目は /lob2/t1export.del.002.lob ファイルに書き込まれ、3 番目は /lob1/t1export.del.001.lob に付加され、4 番目は /lob2/t1export.del.002.lob に付加され、以降このパターンで付加されていきます。

```
EXPORT TO /mypath/t1export.del OF DEL LOBS TO /lob1,/lob2
MODIFIED BY LOBSINFILE SELECT * FROM USER.T1
```

以下の例では、最初の XDM インスタンスは /xml1/xmlbase.001.xml ファイルに書き込まれ、2 番目は /xml2/xmlbase.002.xml ファイル、3 番目は /xml1/xmlbase.003.xml、4 番目は /xml2/xmlbase.004.xml、というパターンで書き込まれていきます。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /xml1,/xml2 XMLFILE xmlbase
   MODIFIED BY XMLINSEPFILES SELECT * FROM USER.T1
```

インポートの例

XML 列を 1 つ含む「mytable」という表があり、以下の IMPORT コマンドがあるとします。

```
IMPORT FROM myfile.del of del LOBS FROM /lobpath XML FROM /xmlpath
MODIFIED BY LOBSINFILE XMLCHAR replace into mytable
```

「myfile.del」に以下のデータが含まれるとします。

```
mylobfile.001.lob.123.456/
```

この場合、インポート・ユーティリティーは、/lobpath/mylobfile.001.lob ファイルから、ファイル・オフセットを 123 から開始し、長さ 456 バイトで XML 文書のインポートを試行します。

ファイル「mylobfile.001.lob」は XML パスではなく LOB パスにあると見なされます。値が XML データ指定子 (XDS) ではなく LOB ロケーション指定子 (LLS) によって参照されているからです。

XMLCHAR ファイル・タイプ修飾子が指定されているので、文書は文字コード・ページでエンコードされるものと見なされます。

XML データ指定子

エクスポート、インポート、およびロード・ユーティリティーを使用して移動される XML データは、メイン・データ・ファイルとは別のファイルに格納する必要があります。 XML データは、メイン・データ・ファイル内で XML データ指定子 (XDS) を使用して表されます。

XDS は XDS という名前の XML タグによって表されるストリングであり、列内の実際の XML データについての情報を記述する属性が付随しています。そのような情報には、実際の XML データが含まれているファイルの名前、およびそのファイル内の XML データのオフセットおよび長さが含まれます。 XDS の属性について、以下で説明します。

FIL XML データが入っているファイルの名前。

OFF FIL 属性で名前指定されているファイル中の XML データのバイト・オフセット。このオフセットは 0 から始まります。

LEN FIL 属性で名前指定されているファイル中の XML データのバイト単位の長さ。

SCH この XML 文書の妥当性検査に使用する XML スキーマの完全修飾 SQL ID。この SQL ID のスキーマと名前のコンポーネントは、それぞれこの XML スキーマに対応する SYSCAT.XSROBJECTS カタログ表の行の「OBJECTSCHEMA」および「OBJECTNAME」値として保管されます。

XDS はデータ・ファイル中で文字フィールドとして解釈され、ファイル形式の文字の列に関する構文解析動作の対象になります。例えば、区切り文字付き ASCII ファイル形式 (DEL) の場合、区切り文字が XDS 中にあれば、二重にしなければなりません。属性値の中の特殊文字 <、>、&、'、″ は常にエスケープしなければなりません。大文字と小文字の区別のあるオブジェクト名は、" 文字エンティティーで囲まなければなりません。

例

値が abc&"def".del の FIL 属性について考えてみましょう。区切り文字付き ASCII ファイルにこの XDS を組み込むには、区切り文字が ″ 文字であれば、″ 文字を二重にして、特殊文字をエスケープします。

```
<XDS FIL=""abc&amp;&quot;def&quot;.del"" />
```

以下の例は、区切り文字付き ASCII データ・ファイル中にある XDS を示しています。XML データはファイル xmldocs.xml.001 に保管され、バイト・オフセットは 100 で始まり長さは 300 バイトになります。この XDS は二重引用符で区切られる ASCII ファイル中にあるので、XDS タグ自体の中の二重引用符は二重にしなければなりません。

```
"<XDS FIL = ""xmldocs.xml.001"" OFF=""100"" LEN=""300"" />"
```

以下の例は、完全修飾 SQL ID ANTHONY.purchaseOrderTest を示しています。XDS 中で、ID の大文字と小文字の区別がある部分は " 文字エンティティーで囲まなければなりません。

```
"<XDS FIL='/home/db2inst1/xmlload/a.xml' OFF='0' LEN='6758'
   SCH='ANTHONY.&quot;purchaseOrderTest&quot;' />"
```

Query および XPath のデータ・モデル

SQL で使用できる XQuery 関数を使用するか、または XQuery を直接呼び出すことにより、データベース表内で XML データにアクセスできます。 Query および XPath データ・モデル (XDM) のインスタンスは、整形式 XML 文書、ノードのシーケンス、原子値のシーケンス、またはノードと原子値の任意の組み合わせとすることができます。

個々の XDM インスタンスは、EXPORT コマンドによって 1 つ以上の XML ファイルに書き込むことができます。

第 2 部 付録

付録 A. インポート・ユーティリティーとロード・ユーティリティーの相違点

下記の表に、 DB2 ロード・ユーティリティーとインポート・ユーティリティーの主要な相違点を要約します。

インポート・ユーティリティー	ロード・ユーティリティー
大量のデータを移動する場合、処理速度が遅くなります。	大量のデータを移動する場合、インポート・ユーティリティーより処理が速くなります。ロード・ユーティリティーは、フォーマット設定されたページをデータベースに直接書き込むためです。
パーティション内並列処理の利用が制限されます。パーティション内並列処理は、ALLOW WRITE ACCESS モードでインポート・ユーティリティーを並行して呼び出すことによりのみ、行うことができます。	パーティション内並列処理を利用します。一般にこれには対称マルチプロセッサー (SMP) マシンが必要です。
FASTPARSE はサポートされません。	FASTPARSE サポートによって、ユーザー提供データのデータ・チェックが簡略化されます。
階層データがサポートされます。	階層データはサポートされません。
PC/IXF フォーマットでの表、階層、および索引の作成がサポートされます。	表および索引は存在していなければなりません。
マテリアライズ照会表へのインポートはサポートされません。	マテリアライズ照会表へのロードがサポートされます。
WSF フォーマットがサポートされます。	WSF フォーマットはサポートされません。
BINARYNUMERICS はサポートされません。	BINARYNUMERICS はサポートされます。
PACKEDDECIMAL はサポートされません。	PACKEDDECIMAL はサポートされます。
ZONEDDECIMAL はサポートされません。	ZONEDDECIMAL はサポートされます。
GENERATED ALWAYS として定義された列をオーバーライドできません。	`generatedoverride` および `identityoverride` ファイル・タイプ修飾子を使用して、 GENERATED ALWAYS 列をオーバーライドできます。
表、ビュー、およびニックネームへのインポートがサポートされます。	表へのロードのみサポートされます。
すべての行がログに記録されます。	最小限のロギングが実行されます。
トリガー・サポートがあります。	トリガー・サポートはありません。

インポート・ユーティリティー	ロード・ユーティリティー
インポート操作が中断された場合に、*commitcount* が指定されていた場合、表は使用可能であり、最後の COMMIT までにロードされた行が収められています。ユーザーはインポート操作を再開するか、表を現状のまま受け入れることができます。	ロード操作が中断された場合に、*savecount* が指定されていると、表はロード・ペンディング状態のままになります。その表は、ロード操作が再開されるか、ロード終了操作が呼び出されるか、またはロード操作の試行前に作成されたバックアップ・イメージから表スペースがリストアされるまで、使用できません。
必要なスペースは、最大の索引のサイズ + 10% とほぼ同じです。このスペースは、データベース内の TEMPORARY 表スペースから取得されます。	必要なスペースは、表に関して定義されているすべての索引のサイズの合計とほぼ同じであり、そのサイズの 2 倍まで大きくなる可能性があります。このスペースは、データベース内の一時スペースから取得されます。
インポート操作中にすべての制約が妥当性検査されます。	ロード・ユーティリティーは、固有性をチェックし、生成される列値を計算するが、他のすべての制約 SET INTEGRITY を使用してチェックしなければなりません。
キー値は、インポート操作中に一度に 1 つずつ索引に挿入されます。	データがロードされた後で、キー値がソートされ、索引が作成されます。
統計情報を更新する必要がある場合は、インポート操作の後で RUNSTATS ユーティリティーを実行しなければなりません。	表内のすべてのデータを置換する場合は、ロード操作中に統計情報を収集できます。
DB2 Connect によってホスト・データベースへのインポートが可能です。	ホスト・データベースへのロードはできません。
インポート・ファイルは、インポート・ユーティリティーが呼び出されるクライアントになければなりません。	指定するオプションに応じて、ロード・ファイルまたはパイプは、データベースを備えたデータベース・パーティション上か、またはロード・ユーティリティーを呼び出すリモート接続クライアント上のどちらかに置くことができます。 **注:** LOB および XML データはサーバー・サイドからのみ読み取ることができます。
バックアップ・イメージは不要です。インポート・ユーティリティーでは SQL の挿入が使用されるため、アクティビティーがログに記録され、障害時にそれらの操作をリカバリーするのにバックアップは不要です。	ロード操作中にバックアップ・イメージを作成することができます。

付録 B. エクスポート・ユーティリティー、インポート・ユーティリティー、ロード・ユーティリティーで使用するバインド・ファイル

以下の表は、バインド・ファイルとそのデフォルト分離レベルをリストし、さらにどのユーティリティーがどんな目的で使用するかを説明しています。

バインド・ファイル (デフォルト分離レベル)	ユーティリティー/目的
db2ueiwi.bnd (CS)	インポート/エクスポート。表の列および索引に関する情報を照会するために使用されます。
db2uexpm.bnd (CS)	エクスポート。エクスポート操作に指定される照会をフェッチするのに使用されます。
db2uimpm.bnd (RS)	インポート。 INSERT、REPLACE または REPLACE_CREATE オプションが使用されたときに、ソース・データ・ファイルからターゲット表にデータを挿入するために使用されます。 **注:** IMPORT コマンドの CREATE オプションと REPLACE_CREATE オプションは非推奨で、将来のリリースでは廃止される可能性があります。
db2uipkg.bnd (CS)	インポート。 BIND オプションをチェックするために使用されます。
db2uiici.bnd (RR)	インポート。 IXF CREATE オプションが指定されたときに索引を作成するために使用されます。
db2ucktb.bnd (CS)	ロード。ロード操作で一般初期化プロセスを実行するために使用されます。
db2ulxld.bnd (CS)	ロード。カーソル操作によってロード時に提供される照会を処理するために使用されます。
db2uigsi.bnd (UNIX ベースのシステムでは RS、その他のプラットフォームでは RR)	インポート/エクスポート。索引をドロップし、REPLACE オプションでのインポートで参照制約をチェックするために使用されます。また、 IXF ファイルをエクスポートするための ID 列情報を検索するのにも使用されます。
db2uiict.bnd (RR)	インポート。 IXF CREATE オプションが指定される際に表を作成するために使用されます。
db2uqtpd.bnd (RR)	インポート/エクスポート。階層表の処理を実行するために使用されます。

バインド・ファイル (デフォルト分離レベル)	ユーティリティー/目的
db2uqtnm.bnd (RR)	インポート。 IXF CREATE オプションが指定されたときに階層表の処理を実行するために使用されます。
db2uimtb.bnd (RS)	インポート。インポート操作で一般初期化プロセスを実行するために使用されます。
db2uImpInsUpdate.bnd (RS)	インポート。 INSERT_UPDATE オプションが使用されたときに、ソース・データ・ファイルからターゲット表にデータを挿入するために使用されます。 INSERT BUF オプションを使用してバインドすることはできません。

付録 C. 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。

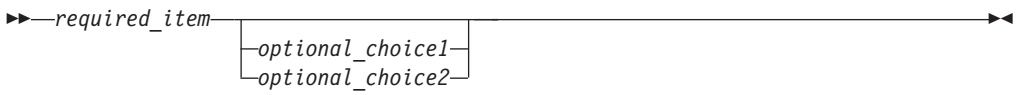
オプショナル項目は、メインパスの下に示されます。

オプショナル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。

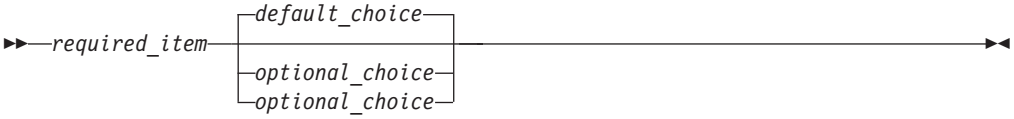
複数の項目からの選択が可能な場合、それらの項目を縦に並べて (スタックに) 示しています。

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。

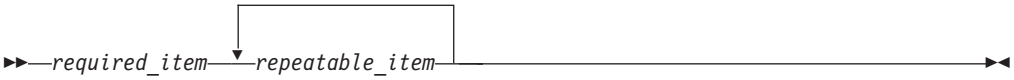
項目から 1 つをオプションで選択できる場合、スタック全体がメインパスよりも下に示されます。



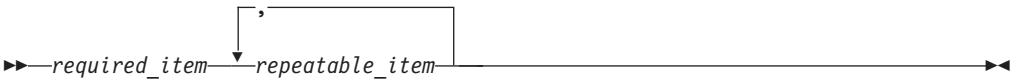
項目の 1 つがデフォルト値の場合、その項目はメインパスより上に示され、残りの選択項目はメインパスよりも下に示されます。



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 つ以上のブランク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。

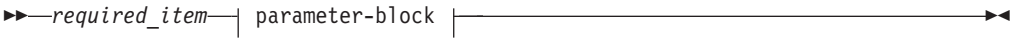


スタックの上部の反復の矢印の記号は、そのスタックの中から複数の項目を選択できること、または 1 つの選択項目を繰り返して選択できることを示します。

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
```

付録 D. データ移動問題についてのデータの収集

データ移動コマンドの実行中に問題が発生し、その問題の原因が判別できない場合は、問題を診断して解決するために、あなた自身または IBM ソフトウェア・サポートが使用できる診断データを収集してください。

- db2move コマンドに関連した問題のデータを収集するには、コマンドを発行したディレクトリーに移動してください。コマンドで指定したアクションに応じて、以下のファイルを見つけます。
  - COPY アクションの場合、COPY.*timestamp*.ERR および COPYSCHEMA.*timestamp*.MSG というファイルを探します。さらに LOAD_ONLY または DDL_AND_LOAD モードのいずれかを指定した場合、LOADTABLE.*timestamp*.MSG というファイルも探してください。
  - EXPORT アクションの場合、EXPORT.out というファイルを探してください。
  - IMPORT アクションの場合、IMPORT.out というファイルを探してください。
  - LOAD アクションの場合、LOAD.out というファイルを探してください。
- EXPORT、IMPORT、または LOAD コマンドに関連した問題のデータを収集するには、コマンドに MESSAGES パラメーターが含まれているかどうかを判別します。含まれている場合、出力ファイルを収集します。これらのユーティリティーは、現行ディレクトリーおよびデフォルト・ドライブ以外を指定しない場合にはそれらを宛先として使用します。
- REDISTRIBUTE コマンドに関連した問題のデータを収集するには、*databasename.database_partition_groupname.timestamp* (Linux および UNIX の場合) および *databasename.database_partition_groupname.date.time* (Windows の場合) というファイルを探してください。それは、*$HOME*/sqllib/db2dump ディレクトリーまたは $DB2PATH¥sqllib¥redist にそれぞれ置かれています (*$HOME* はインスタンス所有者のホーム・ディレクトリー)。

付録 E. 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® 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 資料などのその他の DB2 技術情報には、オンライン (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 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ー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 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 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が示されていますが、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 62. 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ー 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・リカバリーと高可用性 ガイドおよびリファレンス	SC88-4423-02	入手可能	2009 年 4 月
データ・サーバー、データベース、およびデータベース・オブジェクトのガイド	SC88-4259-02	入手可能	2009 年 4 月
データベース・セキュリティー・ガイド	SC88-4418-02	入手可能	2009 年 4 月
ADO.NET および OLE DB アプリケーションの開発	SC88-4425-02	入手可能	2009 年 4 月

表 62. DB2 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
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 管理およびユーザーズ・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 管理およびユーザーズ・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 ユーザーズ・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

表 62. DB2 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
システム・モニター ガイドおよびリファレンス	SC88-4422-02	入手可能	2009 年 4 月
Text Search 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・パフォーマンスのチューニング	SC88-4417-02	入手可能	2009 年 4 月
Visual Explain チュートリアル	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・マネージャー ガイドおよびリファレンス	SC88-4446-02	入手可能	2009 年 4 月
pureXML ガイド	SC88-4447-02	入手可能	2009 年 4 月
XQuery リファレンス	SC88-4448-02	入手不可	2009 年 4 月

表 63. 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かどうか	最終更新
DB2 Connect Personal Edition 概説およびインストール	GC88-4443-02	入手可能	2009 年 4 月
DB2 Connect サーバー機能 概説およびインストール	GC88-4444-02	入手可能	2009 年 4 月
DB2 Connect ユーザーズ・ガイド	SC88-4442-02	入手可能	2009 年 4 月

表 64. Information Integration 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
Information Integration: フェデレーテッド・システム 管理ガイド	SC88-4166-01	入手可能	2008 年 3 月
Information Integration: レプリケーションおよびイベント・パブリッシングのための ASNCLP プログラム・リファレンス	SC88-4167-02	入手可能	2008 年 3 月

表 64. 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、以下のようにします。
  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)
     - 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、 520 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ーから 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ー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、ブラウザー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。

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、以下のようにします。
  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。

ブラウザー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、

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ー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、以下のようにします。

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。ここには、DB2 診断ツールおよびユーティリティーを使用して、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/) 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で提供され、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 F. 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で提供し、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が、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証できません。 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、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 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]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

［ワ行］

A

B

C

D

E

F

G

I

K

L

M

N

P

R

S

T

U

V

W

X

Printed in Japan

SC88-4421-02

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

データ移動ユーティリティー ガイドおよびリファレンス