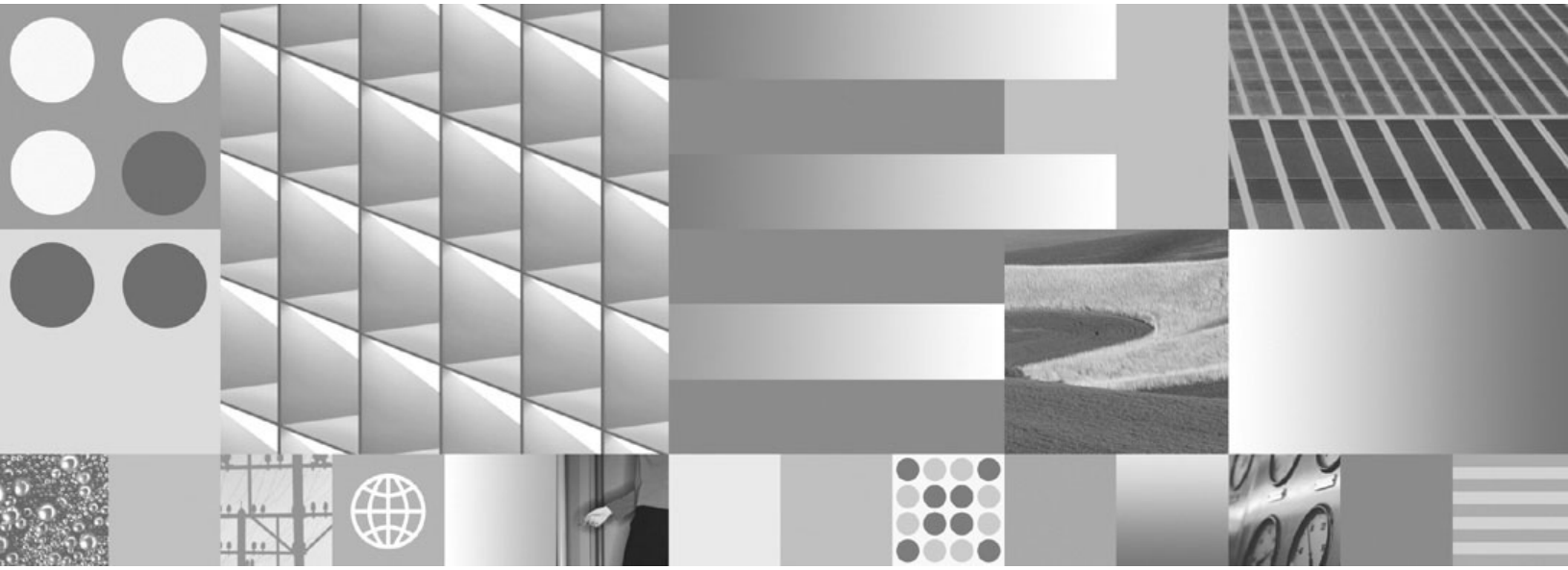
DB2 Version 9.5
for Linux, UNIX, and Windows

バージョン 9 リリース 5



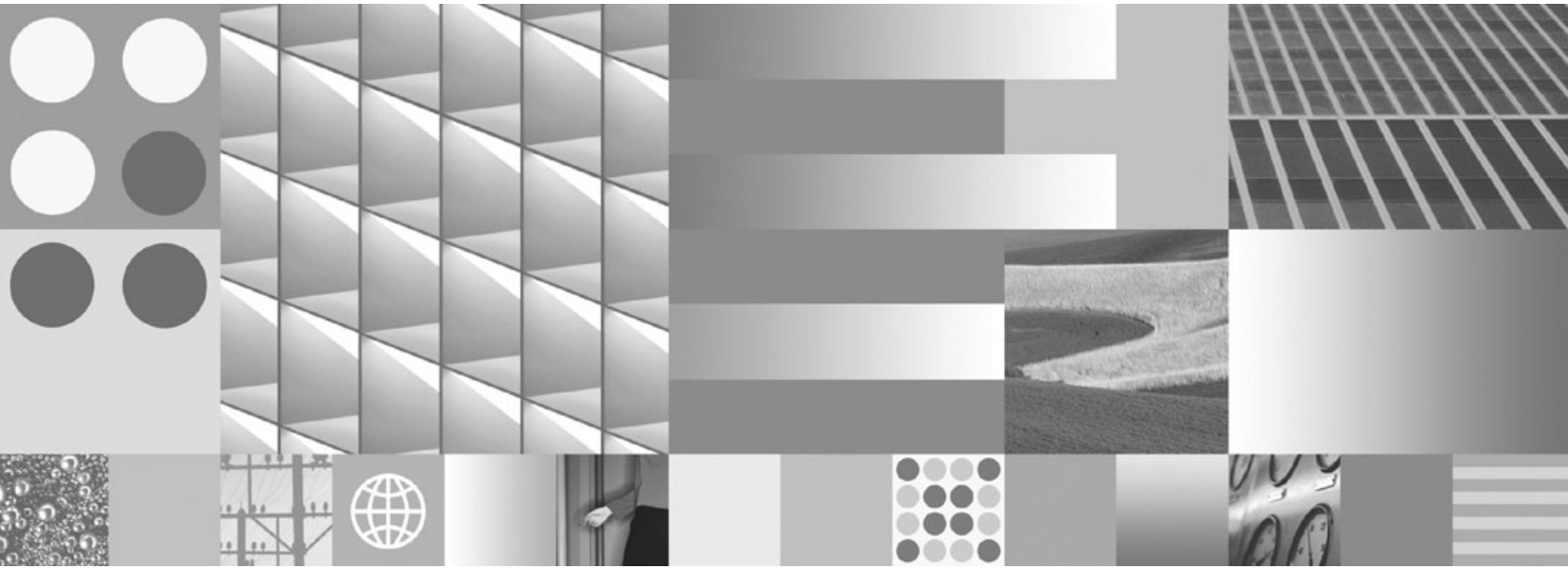
SQL リファレンス 第 1 巻

最終更新: 2010 年 12 月

SC88-4436-03
(英文原典：SC23-5861-03)

DB2 Version 9.5
for Linux, UNIX, and Windows

バージョン 9 リリース 5



SQL リファレンス 第 1 巻

最終更新: 2010 年 12 月

SC88-4436-03
(英文原典：SC23-5861-03)

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、 1037 ページの『付録 O. 特記事項』に記載されている情報をお読みください。

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

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

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

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

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

原典： SC23-5861-03
DB2 Version 9.5
for Linux, UNIX, and Windows
Version 9 Release 5
SQL Reference, Volume 1
Updated December, 2010

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 2010.12

© Copyright IBM Corporation 1993, 2010.

目次

本書について ix
本書の対象読者 ix
本書の構成. ix
構文図の見方 xi
本書の表記規則 xiii
エラー条件 xiii
強調表記規則 xiii
関連資料 xiii

第 1 章 概念. 1
データベース 1
構造化照会言語 (SQL) 1
照会と表式 2
DB2 コール・レベル・インターフェース (CLI) と Open Database Connectivity (ODBC). 2
JDBC (Java Database Connectivity) と組み込み SQL for Java (SQLJ) プログラム 3
スキーマ. 3
表 5
制約 5
索引 5
トリガー. 8
ビュー 9
表またはビューの別名. 11
パッケージ 11
許可、特権、およびオブジェクト所有権. 11
システム・カタログ・ビュー 18
アプリケーションのプロセス、並行性、およびリカバリー 18
分離レベル 20
表スペース 24
文字変換 26
各国語サポートと SQL ステートメント. 29
分散リレーショナル・データベースへの接続 . . . 30
イベント・モニター 31
複数のデータベース・パーティションにまたがるデータベース・パーティション 32
パーティション表でのラージ・オブジェクトの動作 34
DB2 フェデレーテッド・システム. 35
フェデレーテッド・システム 35
データ・ソースとは? 36
フェデレーテッド・データベース 36
SQL コンパイラー 37
ラッパーおよびラッパー・モジュール 37
サーバー定義およびサーバー・オプション . . . 38
ユーザー・マッピング. 39
ニックネームとデータ・ソース・オブジェクト . 40
ニックネーム列オプション 41
データ・タイプ・マッピング 42
フェデレーテッド・サーバー 42
サポートされるデータ・ソース. 44
フェデレーテッド・データベース・システム・カタログ 48
照会オプティマイザー. 49
照合シーケンス 51

第 2 章 言語エレメント. 53
文字. 54
トークン 55
ID 57
データ・タイプ 82
データ・タイプ・リスト 84
データ・タイプのプロモーション. 104
データ・タイプ間のキャスト 106
代入と比較 115
結果データ・タイプの規則 130
ストリング変換の規則 135
Unicode データベースでのストリング比較. . . 136
データベース・パーティション互換データ・タイプ 138
定数 140
特殊レジスター. 145
CURRENT CLIENT_ACCTNG 148
CURRENT CLIENT_APPLNAME 149
CURRENT CLIENT_USERID 150
CURRENT CLIENT_WRKSTNNAME. 151
CURRENT DATE 152
CURRENT DBPARTITIONNUM 153
CURRENT DECFLOAT ROUNDING MODE . . 154
CURRENT DEFAULT TRANSFORM GROUP 155
CURRENT DEGREE 156
CURRENT EXPLAIN MODE 157
CURRENT EXPLAIN SNAPSHOT 158
CURRENT FEDERATED ASYNCHRONY . . . 159
CURRENT IMPLICIT XMLPARSE OPTION . . 160
CURRENT ISOLATION 161
CURRENT LOCK TIMEOUT 162
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 163
CURRENT MDC ROLLOUT MODE 164
CURRENT OPTIMIZATION PROFILE 165
CURRENT PACKAGE PATH 166
CURRENT PATH 167
CURRENT QUERY OPTIMIZATION. 168
CURRENT REFRESH AGE 169
CURRENT SCHEMA 170
CURRENT SERVER 171
CURRENT TIME 172
CURRENT TIMESTAMP. 173
CURRENT TIMEZONE 174
CURRENT USER 175
SESSION_USER 176

© Copyright IBM Corp. 1993, 2010

SYSTEM_USER 177
USER 178
グローバル変数 179
関数 181
メソッド 190
式 199
日付/時刻演算と期間 210
CASE 式 215
CAST 指定 218
XMLCAST 指定 223
ARRAY エレメント仕様 225
間接参照操作 226
OLAP 仕様 228
メソッドの呼び出し 237
サブタイプの扱い 239
シーケンス参照 240
ROW CHANGE 式 244
述部 246
照会の述部の処理 247
検索条件 250
基本述部 253
比較述部 254
BETWEEN 述部 257
EXISTS 述部 258
IN 述部 259
LIKE 述部 261
NULL 述部 267
TYPE 述部 268
VALIDATED 述部 270
XMLEXISTS 述部 273

第 3 章 関数 277

関数の概要 277
サポートされている関数および管理 SQL ルーチンとビュー 278
集約関数 307
ARRAY_AGG 309
AVG 311
CORRELATION 313
COUNT 314
COUNT_BIG 315
COVARIANCE 317
GROUPING 318
MAX 320
MIN 322
回帰関数 323
STDDEV 326
SUM 327
VARIANCE 328
XMLAGG 329
XMLGROUP 331
スカラー関数 334
ABS または ABSVAL 335
ACOS 336
ASCII 337
ASIN 338
ATAN 339
ATAN2 340
ATANH 341
BIGINT 342
BITAND、BITANDNOT、 BITOR、BITXOR、および BITNOT 344
BLOB 346
CARDINALITY 347
CEILING 348
CHAR 349
CHARACTER_LENGTH 354
CHR 356
CLOB 357
COALESCE 358
COLLATION_KEY_BIT 359
COMPARE_DECFLOAT 361
CONCAT 363
COS 364
COSH 365
COT 366
DATAPARTITIONNUM 367
DATE 368
DAY 369
DAYNAME 370
DAYOFWEEK 371
DAYOFWEEK_ISO 372
DAYOFYEAR 373
DAYS 374
DBCLOB 375
DBPARTITIONNUM 376
DECFLOAT 378
DECIMAL 380
DECODE 384
DECRYPT_BIN および DECRYPT_CHAR . . . 386
DEGREES 388
DEREF 389
DIFFERENCE 390
DIGITS 391
DOUBLE 392
ENCRYPT 394
EVENT_MON_STATE 396
EXP 397
FLOAT 398
FLOOR 399
GENERATE_UNIQUE 400
GETHINT 402
GRAPHIC 403
GREATEST 405
HASHEDVALUE 406
HEX 408
HOUR 410
IDENTITY_VAL_LOCAL 411
INSERT 415
INTEGER 419
JULIAN_DAY 421
LCASE 422

LCASE (ロケール依存) 423
LEAST 424
LEFT 425
LENGTH 429
LN 431
LOCATE スカラー関数 432
LOG10. 436
LONG_VARCHAR 437
LONG_VARGRAPHIC 438
LOWER 439
LOWER (ロケール依存) 440
LTRIM 442
MAX 443
MAX_CARDINALITY. 444
MICROSECOND 445
MIDNIGHT_SECONDS 446
MIN 447
MINUTE 448
MOD 449
MONTH 450
MONTHNAME 451
MULTIPLY_ALT 452
NORMALIZE_DECFLOAT 454
NULLIF 455
NVL 456
OCTET_LENGTH 457
OVERLAY 458
PARAMETER 462
POSITION スカラー関数. 463
POSSTR スカラー関数 466
POWER 468
QUANTIZE 469
QUARTER 471
RADIANS 472
RAISE_ERROR 473
RAND 475
REAL 476
REC2XML 477
REPEAT 482
REPLACE 483
RID_BIT および RID 485
RIGHT. 487
ROUND 491
RTRIM 493
SECLABEL 494
SECLABEL_BY_NAME 495
SECLABEL_TO_CHAR 496
SECOND 498
SIGN 499
SIN. 500
SINH 501
SMALLINT 502
SOUNDEX 503
SPACE 504
SQRT 505
STRIP 506
SUBSTR 507
SUBSTRING. 510
TABLE_NAME 513
TABLE_SCHEMA 514
TAN 516
TANH 517
TIME 518
TIMESTAMP 519
TIMESTAMP_FORMAT 521
TIMESTAMP_ISO 525
TIMESTAMPDIFF 526
TO_CHAR 528
TO_DATE 529
TOTALORDER 530
TRANSLATE スカラー関数. 532
TRIM 535
TRUNCATE 537
TYPE_ID 539
TYPE_NAME 540
TYPE_SCHEMA 541
UCASE 542
UCASE (ロケール依存) 543
UPPER 544
UPPER (ロケール依存) 545
VALUE 547
VARCHAR 548
VARCHAR_BIT_FORMAT 550
VARCHAR_FORMAT. 551
VARCHAR_FORMAT_BIT 554
VARGRAPHIC 555
WEEK. 557
WEEK_ISO 558
XMLATTRIBUTES. 559
XMLCOMMENT 561
XMLCONCAT 562
XMLDOCUMENT 564
XMLELEMENT. 566
XMLFOREST 574
XMLNAMESPACES 578
XMLPARSE 580
XMLPI 583
XMLQUERY. 585
XMLROW 588
XMLSERIALIZE 590
XMLTEXT 592
XMLVALIDATE 594
XMLXSROBJECTID 599
XSLTRANSFORM 600
YEAR 604
表関数. 604
XMLTABLE 605
ユーザー定義関数 609

第 4 章 プロシージャー 611
プロシージャーの概要 611
XSR_ADDSCHEMADOC プロシージャー 611

XSR_COMPLETE プロシージャー 612
XSR_DTD プロシージャー 613
XSR_EXTENTITY プロシージャー 615
XSR_REGISTER プロシージャー 616
XSR_UPDATE プロシージャー 618

第 5 章 SQL 照会 621
副選択. 622
全選択. 663
Select-statement 668

付録 A. SQL と XML の制限 679

付録 B. SQLCA (SQL 連絡域) 689

付録 C. SQLDA (SQL 記述子域) . . . 695

付録 D. システム・カタログ・ビュー 707
カタログ・ビューのロードマップ. 709
SYSCAT.ATTRIBUTES 714
SYSCAT.AUDITPOLICIES 716
SYSCAT.AUDITUSE 718
SYSCAT.BUFFERPOOLDBPARTITIONS. 719
SYSCAT.BUFFERPOOLS. 720
SYSCAT.CASTFUNCTIONS. 721
SYSCAT.CHECKS 722
SYSCAT.COLAUTH 723
SYSCAT.COLCHECKS 724
SYSCAT.COLDIST. 725
SYSCAT.COLGROUPCOLS 726
SYSCAT.COLGROUPDIST 727
SYSCAT.COLGROUPDISTCOUNTS 728
SYSCAT.COLGROUPS 729
SYSCAT.COLIDENTATTRIBUTES 730
SYSCAT.COLOPTIONS 731
SYSCAT.COLUMNS 732
SYSCAT.COLUSE 737
SYSCAT.CONSTDEP 738
SYSCAT.CONTEXTATTRIBUTES. 739
SYSCAT.CONTEXTS 740
SYSCAT.DATAPARTITIONEXPRESSION 741
SYSCAT.DATAPARTITIONS 742
SYSCAT.DATATYPES 744
SYSCAT.DBAUTH. 747
SYSCAT.DBPARTITIONGROUPDEF 749
SYSCAT.DBPARTITIONGROUPS 750
SYSCAT.EVENTMONITORS 751
SYSCAT.EVENTS 753
SYSCAT.EVENTTABLES 754
SYSCAT.FULLHIERARCHIES 755
SYSCAT.FUNCMAPOPTIONS 756
SYSCAT.FUNCMAPPARMOPTIONS 757
SYSCAT.FUNCMAPPINGS 758
SYSCAT.HIERARCHIES 759
SYSCAT.HISTOGRAMTEMPLATEBINS. 760
SYSCAT.HISTOGRAMTEMPLATES 761
SYSCAT.HISTOGRAMTEMPLATEUSE 762
SYSCAT.INDEXAUTH 763
SYSCAT.INDEXCOLUSE. 764
SYSCAT.INDEXDEP 765
SYSCAT.INDEXES. 766
SYSCAT.INDEXEXPLOITRULES 772
SYSCAT.INDEXEXTENSIONDEP 773
SYSCAT.INDEXEXTENSIONMETHODS. 774
SYSCAT.INDEXEXTENSIONPARMS 775
SYSCAT.INDEXEXTENSIONS 776
SYSCAT.INDEXOPTIONS 777
SYSCAT.INDEXXMLPATTERNS 778
SYSCAT.KEYCOLUSE 779
SYSCAT.NAMEMAPPINGS 780
SYSCAT.NICKNAMES 781
SYSCAT.PACKAGEAUTH 784
SYSCAT.PACKAGEDEP 785
SYSCAT.PACKAGES 787
SYSCAT.PARTITIONMAPS 794
SYSCAT.PASSTHRUAUTH 795
SYSCAT.PREDICATESPECS 796
SYSCAT.REFERENCES 797
SYSCAT.ROLEAUTH 798
SYSCAT.ROLES 799
SYSCAT.ROUTINEAUTH 800
SYSCAT.ROUTINEDEP 802
SYSCAT.ROUTINEOPTIONS 803
SYSCAT.ROUTINEPARMOPTIONS 804
SYSCAT.ROUTINEPARMS 805
SYSCAT.ROUTINES 807
SYSCAT.ROUTINESFEDERATED 816
SYSCAT.SCHEMAAUTH. 818
SYSCAT.SCHEMATA. 819
SYSCAT.SECURITYLABELACCESS 820
SYSCAT.SECURITYLABELCOMPONENTELEMENTS 821
SYSCAT.SECURITYLABELCOMPONENTS. . . . 822
SYSCAT.SECURITYLABELS 823
SYSCAT.SECURITYPOLICIES 824
SYSCAT.SECURITYPOLICYCOMPONENTRULES 825
SYSCAT.SECURITYPOLICYEXEMPTIONS. . . . 826
SYSCAT.SEQUENCEAUTH 827
SYSCAT.SEQUENCES 828
SYSCAT.SERVEROPTIONS 830
SYSCAT.SERVERS 831
SYSCAT.SERVICECLASSES 832
SYSCAT.STATEMENTS 834
SYSCAT.SURROGATEAUTHIDS 835
SYSCAT.TABAUTH 836
SYSCAT.TABCONST 838
SYSCAT.TABDEP 839
SYSCAT.TABDETACHEDDEP 841
SYSCAT.TABLES 842
SYSCAT.TABLESPACES. 849
SYSCAT.TABOPTIONS 851
SYSCAT.TBSPACEAUTH 852
SYSCAT.THRESHOLDS 853

SYSCAT.TRANSFORMS 855
SYSCAT.TRIGDEP 856
SYSCAT.TRIGGERS 857
SYSCAT.TYPEMAPPINGS 859
SYSCAT.USEROPTIONS 862
SYSCAT.VARIABLEAUTH 863
SYSCAT.VARIABLEDEP 864
SYSCAT.VARIABLES 865
SYSCAT.VIEWS 866
SYSCAT.WORKACTIONS 867
SYSCAT.WORKACTIONSETS 870
SYSCAT.WORKCLASSES 871
SYSCAT.WORKCLASSSETS 872
SYSCAT.WORKLOADAUTH 873
SYSCAT.WORKLOADCONNATTR 874
SYSCAT.WORKLOADS 875
SYSCAT.WRAPOPTIONS 877
SYSCAT.WRAPPERS 878
SYSCAT.XDBMAPGRAPHS 879
SYSCAT.XDBMAPSHREDTREES 880
SYSCAT.XSROBJECTAUTH 881
SYSCAT.XSROBJECTCOMPONENTS 882
SYSCAT.XSROBJECTDEP 883
SYSCAT.XSROBJECTHIERARCHIES 884
SYSCAT.XSROBJECTS 885
SYSIBM.SYSDUMMY1 886
SYSSTAT.COLDIST 887
SYSSTAT.COLGROUPDIST 888
SYSSTAT.COLGROUPDISTCOUNTS 889
SYSSTAT.COLGROUPS 890
SYSSTAT.COLUMNS 891
SYSSTAT.INDEXES 892
SYSSTAT.ROUTINES 896
SYSSTAT.TABLES 897

付録 E. フェデレーテッド・システム 899
SQL ステートメントで有効なサーバーのタイプ 900
フェデレーテッド・システムの関数マッピング・オプション 901
デフォルトの順方向データ・タイプ・マッピング 902
DB2 Database for Linux, UNIX, and Windows データ・ソース 903
DB2 for System i データ・ソース 904
DB2 for VM and VSE データ・ソース 905
DB2 for z/OS データ・ソース 906
Informix データ・ソース 907
Microsoft SQL Server データ・ソース 909
ODBC データ・ソース 911
Oracle NET8 データ・ソース 912
Sybase データ・ソース 913
Teradata データ・ソース 915
デフォルトの逆方向データ・タイプ・マッピング 916
DB2 Database for Linux, UNIX, and Windows データ・ソース 917
DB2 for System i データ・ソース 918
DB2 for VM and VSE データ・ソース 919
DB2 for z/OS データ・ソース 920
Informix データ・ソース 921
Microsoft SQL Server データ・ソース 922
Oracle NET8 データ・ソース 923
Sybase データ・ソース 924
Teradata データ・ソース 925

付録 F. SAMPLE データベース 927

付録 G. 予約済みスキーマ名と予約語 955

付録 H. トリガーと参照制約の間の相互作用の例 959

付録 I. Explain 表 963
ADVISE_INDEX 表 964
ADVISE_INSTANCE 表 968
ADVISE_MQT 表 969
ADVISE_PARTITION 表 971
ADVISE_TABLE 表 973
ADVISE_WORKLOAD 表 974
EXPLAIN_ARGUMENT 表 975
EXPLAIN_DIAGNOSTIC 表 982
EXPLAIN_DIAGNOSTIC_DATA 表 984
EXPLAIN_INSTANCE 表 986
EXPLAIN_OBJECT 表 989
EXPLAIN_OPERATOR 表 993
EXPLAIN_PREDICATE 表 996
EXPLAIN_STATEMENT 表 1000
EXPLAIN_STREAM 表 1003

付録 J. Explain レジスター値 1005

付録 K. 例外表 1011

付録 L. ルーチンで使用可能な SQL ステートメント 1015

付録 M. コンパイル済みステートメントから呼び出される CALL 1019

付録 N. DB2 技術情報の概説 1025
DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式) 1026
DB2 の印刷資料の注文方法 1029
コマンド行プロセッサーから SQL 状態ヘルプを表示する 1030
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 1030
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 1030
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 1031
DB2 チュートリアル 1033

DB2 トラブルシューティング情報 1034
ご利用条件 1034

付録 O. 特記事項 1037

索引 1041

本書について

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

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

本書の対象読者

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

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

本書の構成

SQL リファレンス 第 1 巻には、リレーショナル・データベースの概念、言語エレメント、関数、および照会の形式に関する情報が含まれています。この巻に含まれる特定の章や付録について、以下で簡潔に説明します。

- 『概念』では、リレーショナル・データベースおよび SQL の基本的な概念を説明しています。
- 『言語エレメント』では、SQL の基本的な構文と多くの SQL ステートメントに共通する言語エレメントについて説明しています。
- 『関数』には、SQL 集約関数とスカラー関数の構文図、セマンティックの説明、規則、および使用例があります。
- 『プロシージャー』には、プロシージャーの構文図、セマンティックの説明、規則、および使用例があります。
- 『SQL 照会』では、さまざまな照会形式について説明しています。
- 『SQL と XML 制限値』には、SQL の制限をリストしています。
- 『SQLCA (SQL 連絡域)』では、SQLCA 構造について説明しています。
- 『SQLDA (SQL 記述域)』では、SQLDA 構造について説明しています。
- 『システム・カタログ・ビュー』では、システム・カタログ・ビューについて説明しています。
- 『フェデレーテッド・システム』では、フェデレーテッド・システムのオプションとタイプ・マッピングについて説明しています。
- 『サンプル・データベース』では、サンプル・データベースについて概説しています。ここにある表は、多くの例で使用されています。

© Copyright IBM Corp. 1993, 2010

- 『予約スキーマ名および予約語』には、IBM® SQL および ISO/ANSI SQL2003 標準の予約スキーマ名および予約語があります。
- 『トリガーと参照制約の間の相互作用の例』では、トリガーと参照制約の相互作用について説明しています。
- 『Explain 表』では、Explain 表について説明しています。
- 『Explain レジスター値』では、CURRENT EXPLAIN MODE 特殊レジスター値と CURRENT EXPLAIN SNAPSHOT 特殊レジスター値の相互作用について、またこれらの特殊レジスター値と PREP および BIND コマンドとの相互作用について説明しています。
- 『例外表』には、SET INTEGRITY ステートメントと共に使用するユーザー作成表に関する情報があります。
- 『ルーチンで使用可能な SQL ステートメント』では、種々の SQL データ・アクセス・コンテキストを含むルーチンで実行できる SQL ステートメントをリストしています。
- 『コンパイル済みステートメントから呼び出される CALL』では、コンパイル済みステートメントから呼び出すことができる CALL ステートメントについて説明しています。

構文図の見方

構文図の構造を理解するために、以下の情報を参考にできます。

構文図は、左から右、上から下に、線に沿って読みます。

記号 ▶▶── は、構文図の始まりを示します。

記号 ──▶ は、構文が次の行に続くことを示します。

記号 ▶── は、構文が前の行から続いていることを示します。

記号 ──▶◀ は、構文図の終わりを示します。

構文フラグメントは、記号 ├── で始まり、記号 ──┤ で終わります。

必須項目は、横線 (メインパス) 上に示されます。

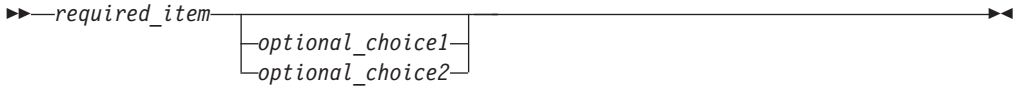
オプショナル項目は、メインパスの下に示されます。

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。

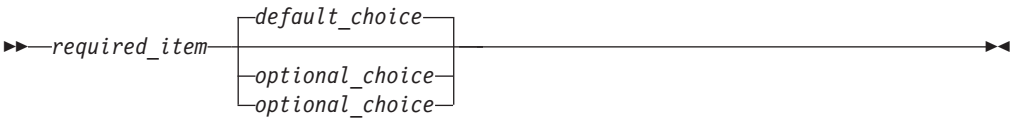
複数の項目からの選択が可能な場合、それらの項目を縦に並べて (スタックに) 示しています。

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。

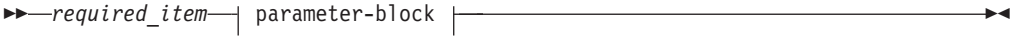
繰り返しの矢印にコンマが示されている場合は、繰り返し項目をコンマで区切らなければなりません。

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。

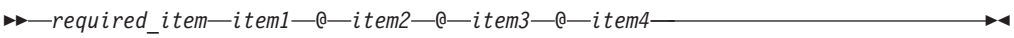
句読点、括弧、算術演算子、その他の記号が示されている場合には、それらを構文の一部として入力する必要があります。

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。例えば次の図で、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:

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



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

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

本書の表記規則

エラー条件

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

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

強調表記規則

本書では、以下の表記規則を使用しています。

太字	コマンド、キーワード、および名前がシステムによって事前定義されている他の項目を表します。
イタリック	以下のいずれかを示します。 • ユーザーが指定する必要のある名前または値 (変数) • 一般的な強調 • 新しい用語の紹介 • 他の情報源の参照
モノスペース	以下のいずれかを示します。 • ファイルおよびディレクトリー • コマンド・プロンプトまたはウィンドウでユーザーが入力する必要のある情報 • 特定のデータ値の例 • システムで表示されるものと類似したテキストの例 • システム・メッセージの例

関連資料

以下の資料は、アプリケーションを準備する際に役立つ可能性があります。

- *データベース・アプリケーション開発の基礎*
  - DB2 アプリケーション開発の概要を示します。これにはプラットフォーム前提条件、サポートされる開発ソフトウェア、およびサポートされているプログラミング API の利点と制約事項についてのガイダンスが含まれます。
- *DB2 for i5/OS SQL リファレンス*
  - この資料では、DB2 Query Manager and SQL Development Kit on System i® によってサポートされる SQL が定義されています。この資料にはシステム管理のタスク、データベース管理、アプリケーション・プログラミング、および操作のタスクに関する参照情報が含まれています。このマニュアルには、構文、使用上の注意、キーワード、および DB2 を実行する i5/OS® システム上で使用される各 SQL ステートメントの例が含まれます。
- *DB2 for z/OS SQL リファレンス*

  - この資料では、DB2 for z/OS® で使用される SQL を定義しています。この資料では、DB2 を実行する z/OS システムでの照会書式、SQL ステートメント、SQL プロシージャー・ステートメント、DB2 の制約事項、 SQLCA、SQLDA、カタログ表、および SQL 予約語について説明しています。
- *DB2 Spatial Extender ユーザーズ・ガイドおよびリファレンス*
  - この資料では、地理情報システム (GIS) を作成および使用するアプリケーションの作成方法を説明しています。GIS の作成および使用には、データベースにリソースを提供すること、またデータの照会を行って位置、距離、および領域内での分布などの情報を取得することが含まれます。
- *IBM SQL リファレンス*
  - この資料には、IBM のデータベース製品に関係したすべての共通 SQL エレメントを収録しています。この資料では、IBM データベースを使用する移植可能プログラムを準備する際に参照できる、制約事項や規則について説明しています。このマニュアルでは、SQL 拡張機能、および各種の規格と製品 (SQL92E、XPG4-SQL、IBM-SQL、および IBM リレーショナル・データベース製品) 間における非互換性のリストを示しています。
- *American National Standard X3.135-1992, Database Language SQL*
  - SQL の ANSI 規格定義があります。
- *ISO/IEC 9075:1992, Database Language SQL*
  - SQL の 1992 ISO 標準定義があります
- *ISO/IEC 9075-2:2003, Information technology -- Database Languages -- SQL -- Part 2: Foundation (SQL/Foundation)*
  - SQL の 2003 ISO 標準定義の大部分がここにあります。
- *ISO/IEC 9075-4:2003, Information technology -- Database Languages -- SQL -- Part 4: Persistent Stored Modules (SQL/PSM)*
  - SQL プロシージャー制御ステートメントの 2003 ISO 標準定義があります。

第 1 章 概念

データベース

DB2 データベースは、*リレーショナル・データベース* です。このデータベースは、相互に関連する表にすべてのデータを格納します。データが共有され、重複が最小限にとどめられるように、表間のリレーションシップが確立されます。

リレーショナル・データベース は、1 つの表集合として扱われ、データのリレーショナル・モデルに従って操作されます。このデータベースには、データの保存、管理、およびアクセスに使用されるオブジェクトが一式揃っています。そのようなオブジェクトの例として、表、ビュー、索引、関数、トリガー、およびパッケージがあります。オブジェクトには、システムで定義するもの (システム定義オブジェクト) とユーザーが定義するもの (ユーザー定義オブジェクト) があります。

分散リレーショナル・データベース は、相互接続された異なるコンピューター・システムに分散している表集合とほかのオブジェクトで構成されています。各コンピューター・システムには、その環境で表を管理するリレーショナル・データベース・マネージャーが 1 つあります。 これらのデータベース・マネージャーは、特定のデータベース・マネージャーが SQL ステートメントを別のコンピューター・システムで実行することができるような仕方で、相互に通信および調整を行います。

パーティション・リレーショナル・データベース は、データが複数のデータベース・パーティションにまたがって管理されるリレーショナル・データベースのことです。データベース・パーティション間のデータの分離は、ほとんどの SQL ステートメントではユーザーに認識されません。ただし、一部のデータ定義言語 (DDL) ステートメントでは、データベース・パーティション情報が考慮されます (CREATE DATABASE PARTITION GROUP など)。 DDL は、同じデータベース内のデータのリレーションシップを記述するために使用される SQL ステートメントのサブセットです。

フェデレーテッド・データベース は、データが複数のデータソース (分離リレーショナル・データベースなど) に保存されるリレーショナル・データベースのことです。データはあたかも単一の大容量のデータベースにあるかのように見え、従来の SQL 照会でアクセスすることができます。データに対する変更は、該当するデータ・ソースへ明示的に送られます。

構造化照会言語 (SQL)

SQL は、リレーショナル・データベースのデータの定義と操作を行うための標準化された言語です。データのリレーショナル・モデルに従って、データベースは表の集まりとして扱うことができ、リレーションシップは表の中の各値で表され、データは 1 つまたは複数の基本表から派生する結果表を指定することによって検索されます。

© Copyright IBM Corp. 1993, 2010

SQL ステートメントは、データベース・マネージャーによって実行されます。データベース・マネージャーの機能の 1 つは、結果表の仕様を、データ検索を最適化する一連の内部命令に変換することです。この変換は、準備処理およびバインドの 2 つのフェーズで行われます。

実行可能な SQL ステートメントはすべて、その実行に先立って準備しておく必要があります。その準備の結果は、ステートメントの実行可能形式または操作可能形式です。 SQL ステートメントを準備する方式とその操作可能形式の持続の程度の違いによって、静的 SQL と動的 SQL とがあります。

照会と表式

照会 は、 (一時的な) 結果表を指定するための特定の SQL ステートメントからなるコンポーネントです。

表式 は、単純な照会から一時的な結果表を作成します。節を使うと、その結果表がさらに詳細なものになります。たとえば、表式を照会として使用して、複数の部門からすべての管理者を選択し、さらに管理者が 15 年以上の実務経験があり、ニューヨーク支社に配属されていなければならないことを指定することができます。

共通表式 は、複雑な照会内の一時ビューのようなものです。それは照会内のほかの場所から参照することができ、ビューの代わりに使用できます。複雑な照会の中で特定の共通表式を使用する場合、それぞれが同じ一時ビューを共用することになります。

1 つの照会の中で 1 つの共通表式を再帰的に使用することにより、航空座席予約システム、部品表 (BOM) 生成プログラム、ネットワーク計画などのアプリケーションのサポートのために利用できます。

DB2 コール・レベル・インターフェース (CLI) と Open Database Connectivity (ODBC)

DB2 コール・レベル・インターフェースは、アプリケーション・プログラムに動的 SQL ステートメントを処理する機能を提供するアプリケーション・プログラミング・インターフェースです。 CLI プログラムは Open database connectivity Software Developer's Kit (Microsoft® または他のベンダーから入手可能) を使用してコンパイルすることもできます。これにより、ODBC データ・ソースへアクセスできるようになります。組み込み SQL と違い、このインターフェースではプリコンパイルは必要ありません。種々のデータベースに対して実行できますが、それぞれのためにコンパイルする必要はありません。アプリケーションは実行時にプロシージャー呼び出しを使用して、データベースへの接続、SQL ステートメントの発行、およびデータや状況情報の入手を行うことができます。

DB2 CLI インターフェースには、組み込み SQL では使用できない多くの機能があります。以下に例を示します。

- CLI では、データベース・システム・カタログ情報を照会する DB2 ファミリーを通じて一貫した方法をサポートする関数呼び出しが用意されています。これにより、特定のデータベース・サーバーに合わせてカタログ照会を作成する必要が少なくなります。

- CLI には、カーソルを使った、次のようなスクロール機能があります。
  - 1 行以上のフォワード・スクロール
  - 1 行以上のリバース・スクロール
  - 最初の行からの 1 行以上のフォワード・スクロール
  - 最後の行からの 1 行以上のリバース・スクロール
  - カーソルの直前保管位置からのスクロール
- CLI を使用して作成されたアプリケーション・プログラムから呼び出されるストアード・プロシージャーからは、結果セットをプログラムに戻すことができます。

JDBC (Java Database Connectivity) と組み込み SQL for Java (SQLJ) プログラム

DB2 は Java™ database connectivity (JDBC) と組み込み SQL for Java (SQLJ) という 2 つの標準の Java プログラミング API を実装します。どちらを使用しても、DB2 にアクセスする Java アプリケーションおよびアプレットを作成することができます。

- JDBC 呼び出しは Java 固有のメソッドによって DB2 CLI 呼び出しに変換されます。 JDBC は、DB2 CLI を介した DB2 クライアントから DB2 サーバーへの流れを要求します。 JDBC で静的 SQL を使用することはできません。
- SQLJ アプリケーションは、データベースへの接続や SQL エラーの処理といったタスクの基盤として JDBC を使用しますが、 SQLJ ソース・ファイルに組み込み静的 SQL ステートメントを組み入れることもできます。 SQLJ 変換プログラムを使って SQLJ ソース・ファイルを変換してからでないと、生成される Java ソース・コードをコンパイルすることはできません。

スキーマ

スキーマ とは、名前を持つオブジェクトの集合のことです。これにより、それらのオブジェクトを論理的にグループ化できます。スキーマは、名前修飾子でもあります。これにより、複数のオブジェクトに対して同じ自然名を使用しながら、それらのオブジェクトに対するあいまい参照を防ぐことができます。

例えば、「INTERNAL」および「EXTERNAL」というスキーマ名によって、2 つの異なる SALES 表を識別することが容易になります (INTERNAL.SALES、EXTERNAL.SALES)。

スキーマによって、複数のアプリケーションがネーム・スペースの衝突を生じることなく、単一のデータベースにデータを保管できるようにもなります。

スキーマと *XML* スキーマ とは別個のものなので混同しないでください。後者は、XML 文書の構造を記述し、その内容を妥当性検査するための標準です。

表、ビュー、ニックネーム、トリガー、関数、パッケージ、および他のオブジェクトをスキーマに入れることができます。スキーマ自体が 1 つのデータベース・オブジェクトです。現行ユーザーを指定するか、またはスキーマ所有者として記録された指定の許可 ID を指定した CREATE SCHEMA ステートメントを使用して、スキ

ーマは明示的に作成されます。また、ユーザーが IMPLICIT_SCHEMA 権限を持っている場合には、他のオブジェクトを作成する際に暗黙的に作成することもできます。

スキーマ名 は、 2 つの部分から成るオブジェクト名の高位の部分として使用されます。オブジェクトを作成する際にスキーマを使用して固有に修飾すると、オブジェクトはこのスキーマに割り当てられます。オブジェクトを作成する際にスキーマ名を指定しないと、デフォルトのスキーマ名 (CURRENT SCHEMA 特殊レジスターで指定されたもの) が使用されます。

例えば、DBADM 権限を有するユーザーが、ユーザー A に対して C と呼ばれるスキーマを作成するとします。

```
CREATE SCHEMA C AUTHORIZATION A
```

次にユーザー A は、以下のステートメントを出して、スキーマ C 内に X という名前の表を作成することができます (ただし、ユーザー A が CREATETAB データベース権限をもつことを前提とします)。

```
CREATE TABLE C.X (COL1 INT)
```

予約済みのスキーマ名があります。例えば、組み込み関数は SYSIBM スキーマに属し、プリインストールされたユーザー定義関数は SYSFUN スキーマに属します。

データベースが作成される場合に、それが RESTRICTIVE オプションを使用して作成されるのではない場合は、すべてのユーザーが IMPLICIT_SCHEMA 権限を持ちます。この権限を使用して、ユーザーは、まだ存在していないスキーマ名を持つオブジェクトを作成するときに、常に暗黙にスキーマを作成します。スキーマが暗黙的に作成されるときは、CREATEIN 特権が付与されます。この特権により、どのようなユーザーも、そのスキーマに他のオブジェクトを作成することができます。別名、特殊タイプ、関数、およびトリガーなどのオブジェクトの作成能力は、暗黙的に作成されるスキーマにまで拡張されます。暗黙的に作成されるスキーマについてのデフォルトの特権には、旧バージョンとの後方互換性があります。

IMPLICIT_SCHEMA 権限が PUBLIC から取り消される場合、スキーマは、CREATE SCHEMA ステートメントを使用して明示的に作成されるか、または IMPLICIT_SCHEMA 権限が与えられているユーザー (例えば、 DBADM 権限のあるユーザー) によって暗黙的に作成されます。 PUBLIC から IMPLICIT_SCHEMA 権限を取り消すことは、スキーマ名の使用に対する制御が増す一方で、既存のアプリケーションがオブジェクトの作成を試みる時に許可エラーが生じる可能性があります。

スキーマには特権もあるので、スキーマ所有者がその特権を使用すれば、どのようなユーザーがスキーマ中のオブジェクトを作成、変更、コピー、およびドロップする権限をもつかを制御することができます。これにより、データベース内にあるオブジェクトのサブセットの操作を制御できます。当初、スキーマの所有者にはスキーマに関するこれらのすべての特権が与えられ、それらの特権を他のユーザーに付与することもできます。暗黙的に作成されたスキーマはシステムによって所有され、当初、そのようなスキーマにオブジェクトを作成する特権がすべてのユーザーに与えられます。 SYSADM または DBADM 権限を有するユーザーは、任意のスキーマでユーザーが保持する特権を変更することができます。したがって、任意の

スキーマ (暗黙的に作成されたスキーマであっても) のオブジェクトを作成、変更、コピー、およびドロップするためのアクセスを制御することができます。

表

表とは、データベース・マネージャーによって維持される論理構造です。表は列と行で構成されます。

列と行の交点すべてには、*値* と呼ばれる特定のデータ項目があります。*列* は、同一のタイプまたはそのいずれかのサブタイプの値の集合です。*行* は、*n* 番目の値が、表の *n* 番目の列の値であるような、一続きに配置された値です。

アプリケーション・プログラムでは、行が表に登録された順序を判別できますが、行の実際の順序はデータベース・マネージャーによって判別され、通常制御することはできません。マルチディメンション・クラスタリング (MDC) では、行間の実際の順序ではなく、ある種のクラスタリング (群/集合) の感覚が提供されます。

制約

どの業務でも、データが特定の制限または規則に従っていなければならない場合があります。例えば、従業員番号が固有でなければならない、などです。データベース・マネージャーは、このような規則を強制する手段として*制約* を提供します。

以下のタイプの制約が使用できます。

- NOT NULL 制約
- ユニーク (またはユニーク・キー) 制約
- 主キー制約
- 外部キー (または参照整合性) 制約
- (表) チェック制約
- インフォメーショナル制約

制約は表のみに関連付けられ、表の作成プロセスの一部として (CREATE TABLE ステートメントを使用して) 定義されるか、または表の作成後に (ALTER TABLE ステートメントを使用して) 表の定義に追加されます。ALTER TABLE ステートメントを使用して、制約を変更することができます。たいていの場合、既存の制約はいつでもドロップできます。この操作は、表の構造や、そこに格納されているデータには影響を与えません。

注: 表オブジェクトに関連付けられるのはユニーク制約とプライマリー制約のみで、これらはしばしば 1 つ以上のユニーク索引または主キー索引を使用することによって強制されます。

索引

索引 は 1 つ以上のキーのセットであり、それぞれのキーは表の行を指しています。 SQL *オプティマイザー* は表のデータにアクセスする効率的な方法を自動的に選択します。オプティマイザーはデータへの一番速いアクセス・パスを判別するときに、索引を考慮に入れます。

注: すべての索引が表の行を指示するわけではありません。MDC ブロック索引は、データのエクステント (または、ブロック) を指します。 XML データ用の XML 索引では、特定の XML パターン式を使用して、単一の列に格納された XML 文書内にあるパスおよび値の索引付けを行います。その列のデータ・タイプは XML でなければなりません。 MDC ブロック索引と XML 索引の両方が、システム生成の索引です。

索引は、以下の目的でデータベース・マネージャーによって使用されます。

- パフォーマンスを改善する。ほとんどの場合、索引を使うとデータへのアクセスが速くなります。索引をビューに対して作成することはできませんが、ビューのベースとなる表に対して索引を作成することで、そのビューに対する操作のパフォーマンスを改善できる場合があります。
- 固有性を確保する。ユニーク索引を持つ表に、同一のキーを持つ行を含めることはできません。

表にデータを追加すると、その表や追加データに対して別の操作が実行されていないかぎり、データは表の下部に追加されます。データに順序はありません。 特定の行データを検索するときは、表の各行の最初から最後までチェックする必要があります。 特定の順序で表内のデータにアクセスするには、索引を使用します。

データの行に列値を使用することにより、その行全体を識別することができます。行を識別するため、1 つ以上の列が必要な場合があります。そのような列のことを *キー* といいます。 1 つの列を複数のキーで使用できます。

索引はキー内の値によって配列されます。キーはユニーク・キーか非ユニーク・キーのどちらかになります。各表には少なくとも 1 つのユニーク・キーが必要ですが、他にも非ユニーク・キーをいくつか含めることができます。各索引のキーは 1 つだけです。 例えば、索引に従業員 ID 番号 (固有) を使用し、別の索引に部門番号 (非固有) を使用できます。

例

7 ページの図 1 の表 A には、表の従業員番号に基づいた索引があります。このキー値は、表の行を指すポインターの機能を提供します。例えば、従業員番号 19 は、従業員 KMP を指しています。 索引では、ポインターを介してデータへのパスを作成できるので、効率よく表の行にアクセスできます。

ユニーク索引は、索引キーが必ず固有になるようにするために作成できます。*索引キー* は、索引が定義されている列または列の順序付き集合です。ユニーク索引を使用すると、索引になっている列にある索引キーごとの値または列の値が必ず固有のものとなります。

7 ページの図 1 は、索引と表のリレーションシップを示しています。

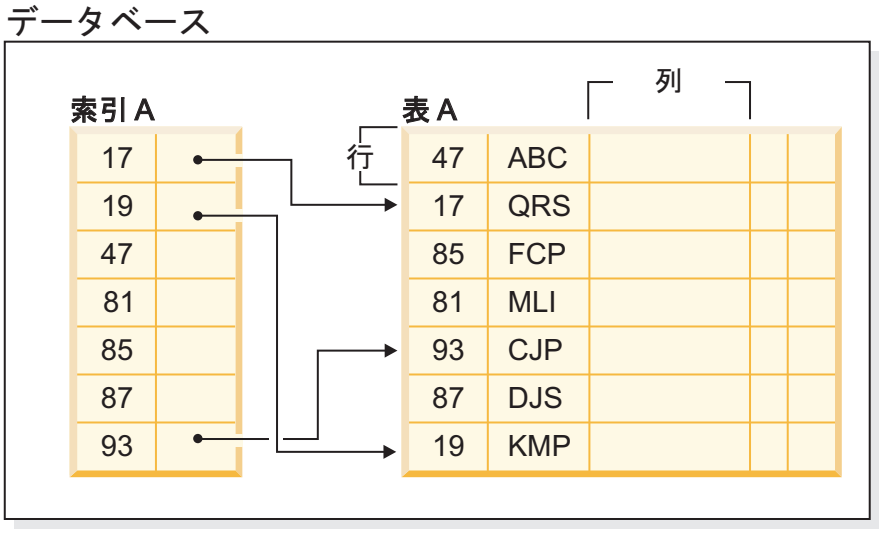


図 1. 索引と表の関係

図 2 では、データベース・オブジェクトのいくつかの関連を図示しています。この図はまた、表、索引、ロング・データが表スペースに保管されている様子も示しています。

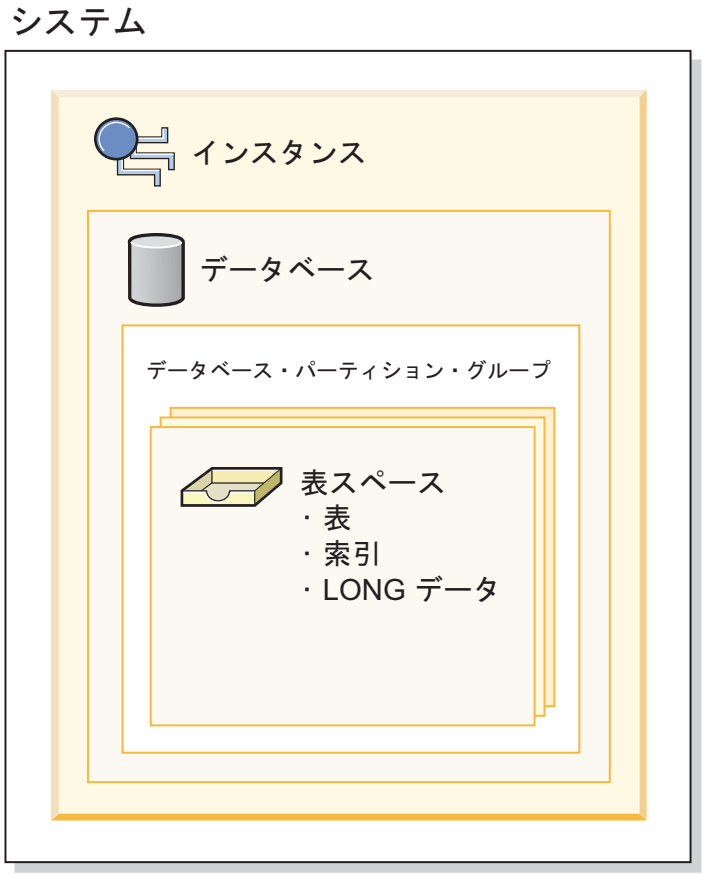


図 2. 一部のデータベース・オブジェクトの相互関係

トリガー

トリガー は、指定した表に対する挿入、更新、または削除操作への応答として実行されるアクションのセットを定義します。このような SQL 操作が実行されるとき、トリガーが*起動*されるといいます。トリガーはオプションであり、 CREATE TRIGGER ステートメントを使用して定義されます。

データ整合性規則を実施するために、参照制約およびチェック制約とともにトリガーを使用できます。また、トリガーを使用して、他の表への更新を行ったり、挿入または更新される行の値を自動的に生成またはトランスフォームできます。あるいは、関数を呼び出してタスク (アラートを発するなど) を実行することもできます。

トリガーは、*移り変わる* ビジネス規則を定義および実施するための便利な機構です。この規則は、さまざまな状態のデータ (例えば、昇給率が 10 % を超えることのできない給与など) を扱う規則です。

トリガーを使用すると、ビジネス規則を実施する論理をデータベース内に置くことができます。つまり、アプリケーションがそれらの規則の実施を担当しないということです。すべての表に対してロジックを一カ所に集中すれば、ロジックの変更時にアプリケーション・プログラムへの変更が必要ないため、簡単に保守を行えるようになります。

トリガーを作成する際に、以下を指定します。

- *サブジェクト表*。これは、トリガーが定義される表を指定します。
- *トリガー・イベント*。これは、サブジェクト表を変更する特定の SQL 操作を定義します。イベントには、挿入、更新、または削除操作があります。
- *トリガー起動時間*。これは、トリガー・イベントが発生する前か後のどちらで、トリガーを活動化するかを指定します。

トリガーを活動化するステートメントには、*影響を受ける行のセット* が組み込まれています。これらは、挿入、更新、または削除されるサブジェクト表の行です。*トリガー細分性* では、トリガーのアクションを、ステートメントで 1 回実行するか、または影響を受ける行ごとに 1 回実行するかを指定します。

トリガー・アクション は、オプションの検索条件、およびトリガーが起動されると必ず実行される SQL ステートメントのセットで構成されます。 SQL ステートメントが実行されるのは、検索条件が true と評価された場合のみです。トリガー起動時間がトリガー・イベントの前の場合、トリガー・アクションに、SELECT ステートメント、遷移変数を設定するステートメント、および SQL 状態をシグナル通知するステートメントを組み入れることができます。トリガー起動時間がトリガー・イベントの後の場合、トリガー・アクションに、SELECT、INSERT、UPDATE、DELETE ステートメント、または SQL 状態をシグナル通知するステートメントを組み入れることができます。

トリガー・アクションでは、*遷移変数* を使用して、影響を受ける行のセット内の値を参照できます。遷移変数は、サブジェクト表の列の名前を使用します。この名前は、参照するのが古い値 (更新前) か新しい値 (更新後) かを識別するために指定された名前によって修飾されます。 BEFORE、INSERT、または UPDATE トリガーで、 SET Variable ステートメントを使用して新しい値を変更することもできます。

影響を受ける行のセット内の値を参照する別の方法は、*遷移表* を使用することです。遷移表では、サブジェクト表の列の名前も使用しますが、名前を指定することにより、影響を受ける行の完全なセットを表として扱うことができます。遷移表は、AFTER トリガーでしか使用できません (つまり、BEFORE および INSTEAD OF トリガーでは使用できません)。また、古い値と新しい値に別々の遷移表を定義することができます。

表、イベント (INSERT、UPDATE、DELETE)、または起動時間 (BEFORE、AFTER、INSTEAD OF) の組み合わせに対して複数のトリガーを指定することができます。特定の表、イベント、および起動時間に対して複数のトリガーが存在する場合、トリガーが活動化される順序は、作成された順序と同じです。そのため、一番あとに作成されたトリガーが、最後に活動化されます。

トリガーの活動化では、*トリガー・カスケード* が行われる場合があります。これは、SQL ステートメントを実行するあるトリガーを活動化することにより、そのステートメントによって、他のトリガーが活動化されるか、または同じトリガーが再度活動化された結果です。トリガー・アクションによって、削除の参照整合性規則の適用の結果である更新が行われることもあります。これにより、今度は、追加トリガーの活動化が行われる場合があります。トリガー・カスケードでは、トリガーおよび参照整合性の削除規則のチェーンが活動化され、単一の INSERT、UPDATE、または DELETE ステートメントの結果として、データベースへの大幅な変更が行われる場合があります。

複数のトリガーが同じオブジェクトに対する挿入、更新、または削除操作を行う場合、アクセスの競合を避けるために一時表が使用されます。これは、パーティション・データベース環境では特に、パフォーマンスに大きな影響を与える可能性があります。

ビュー

ビュー は、データを保守せずに表すための効率的な方法です。ビューは実際の表ではなく、また永続ストレージを必要とすることもありません。「仮想表」が作成され、使用されます。

ビュー により、1 つ以上の表にあるデータをさまざまな方法で見ることができます。つまり、ビューとは、結果表に名前を付けて指定したものです。この指定は、ビューが SQL ステートメントで参照されるときにいつも実行される SELECT ステートメントのことです。ビューには表と同じく列と行があります。ビューはすべて、データ・リトリーブにおいて表と同じように使用することができます。挿入、更新、または削除の操作がビューに対して実行できるかどうかは、その定義により異なります。

ビューには、ベースとなっている表の列または行のすべてまたは一部を含めることができます。例えば、ビューの中で部署表と従業員表を結合して、特定の部署の従業員をすべてリストすることができます。

10 ページの図 3 は、表とビューの関連を示しています。

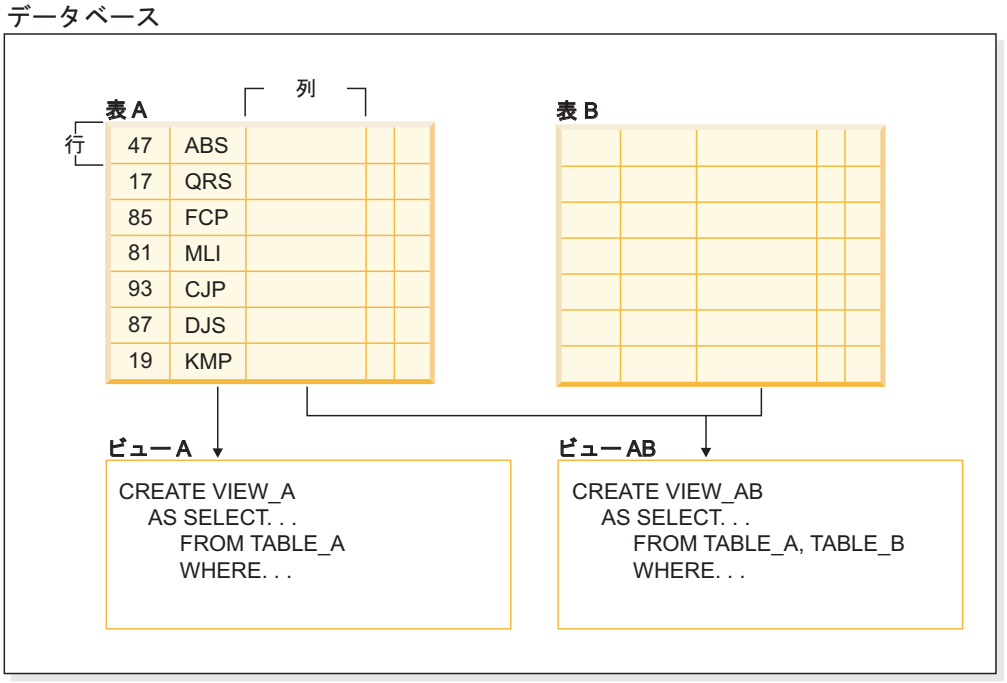


図 3. 表とビューの関係

ビューを使用して、機密データへのアクセスを制御することができます。なぜなら、ビューによって複数のユーザーが同じデータを異なる表示で見ることができるからです。例えば、数人のユーザーが、従業員に関するデータの表にアクセスしているとします。管理職は、自分の部門の従業員のデータは見ることができますが、他の部門の従業員のデータは見ることができません。人事部のユーザーは、すべての従業員の雇用日付を見ることができますが給料は見えません。経理課のユーザーは、給料を見ることはできますが雇用日付は見えません。こうしたユーザーはそれぞれ表から派生したビューで作業します。各ビューは、1 つの表のように表示され、それぞれ固有の名前があります。

ビューの列が基本表の列から直接に派生している場合、そのビューの列は表の列に適用されるあらゆる制約を継承します。例えば、ビューにその表の外部キーが入っている場合、そのビューを使用する挿入および更新操作は表と同じ参照制約に従います。また、ビューの表が親表である場合、そのビューを使用する削除および更新操作は、表の削除および更新操作と同じ規則に従います。

ビューでは、列ごとにデータ・タイプを結果表から派生させる (つまり、型をユーザー定義の構造化タイプの属性に基づいたものにする) ことができます。このようなビューを*型付きビュー* と呼びます。型付き表と同様に、型付きビューもビュー階層の一部になることができます。*サブビュー* は、*スーパービュー* から列を継承します。 *サブビュー* という語は、型付きビュー、およびビュー階層でその下にあるすべての型付きビューに当てはまります。ビュー V の*厳密な意味*でのサブビューとは、型付きビュー階層で V の下にあるビューのことです。

ビューが作動不能になる場合があります (表がドロップされた場合など)。これが発生すると、そのビューは SQL 操作では使えなくなります。

表またはビューの別名

別名 とは、表またはビューの代替名です。既存の表またはビューを参照できる 場合に、これを使用して表またはビューを参照することができます。

別名は、どのようなコンテキストでも使用できるというわけではありません。例えば、チェック制約のチェック条件では使用できません。別名は、宣言済み一時表を参照することはできません。

表やビューのように、別名も作成やドロップができ、関連するコメントを付けることができます。ただし、表とは異なり、別名はチェーニング と呼ばれるプロセスの中で互いを参照できます。別名は公開して参照される名前ですので、これを使用するために特別な権限や特権を必要とするわけではありません。しかし、別名によって参照される表やビューにアクセスするためには、これらオブジェクトに関連する許可が必要です。

データベース別名およびネットワーク別名などといった、他のタイプの別名もあります。別名は、フェデレーテッド・システム上のデータ表またはビューを参照するニックネーム に対して作成することもできます。

パッケージ

パッケージ とは、プログラム準備中に作成される、単一のソース・ファイル内にあるすべてのセクションが入ったオブジェクトのことです。*セクション* とは、コンパイル済み形式の SQL ステートメントのことです。セクションは必ず 1 つのステートメントに対応しますが、すべてのステートメントにセクションがあるわけではありません。静的 SQL 用に作成されたセクションは、バインド形式または操作可能形式の SQL ステートメントに相当します。動的 SQL 用に作成されたセクションは、実行時に使用されるプレースホルダー制御構造に相当します。

許可、特権、およびオブジェクト所有権

ユーザー (許可 ID で識別される) は、指定された関数を実行する権限を持っている場合にのみ、SQL または XQuery ステートメントを正常に実行することができます。表を作成するには、ユーザーに表作成の許可が必要であり、表を変更するには、表変更の許可が必要となります。その他も同様です。

許可には、この後に解説されているとおり、*管理権限*、*特権* および *LBAC 信用証明情報* の 3 つの形式があります。

データベース・マネージャーでは、特定のタスクを実行するのに必要なデータベース機能を使用するために、各ユーザーが特定の許可を暗黙または明示的に与えられていなければなりません。*明示的* な権限あるいは特権は、ユーザーに対して付与されます (データベース・カタログでは GRANTEETYPE が U)。*暗黙的な* 権限あるいは特権は、各ユーザーが所属するグループに対して付与されるか、あるいは、ユーザー、グループ、または別のロールをメンバーとして持つロールに対して付与されます。 (グループの場合、データベース・カタログでは GRANTEETYPE が G、ロールの場合は、R ととなります。)

管理権限

管理権限のある 1 人以上の担当者は、データベース・マネージャーを制御するタスクに携わり、データの安全と整合性に対する責任を持ちます。 SYSADM および DBADM レベルの管理権限のある担当者は、データベース・セキュリティーに関連するオブジェクト以外のすべてのオブジェクトについてすべての特権が暗黙的にあり、だれがデータベース・マネージャーにアクセスするか、およびこのアクセスの程度を制御します。

権限レベル によって、特権のグループ分けの方法、およびより高いレベルのデータベース・マネージャーの保守とユーティリティー操作が得られます。データベース *権限* は、ユーザーがデータベース・レベルのアクティビティーを実行できるようにします。ユーザー、グループ、またはロールは、以下のような 1 つ以上の権限を持つことができます。

- インスタンス・レベルで機能する管理権限レベル、SYSADM (システム管理者)

  SYSADM 権限レベルは、データベース・マネージャーによって作成および保守されるすべてのリソースに対する制御を可能にします。システム管理者は DBADM、SYSCTRL、SYSMAINT、および SYSMON 権限をすべて所有し、DBADM 権限および SECADM 権限を付与または取り消す権限を持っています。

  SYSADM 権限を持つユーザーは、データベース・マネージャーの制御、およびデータの保護と整合性を担当します。 SYSADM 権限はデータベース内で暗黙の DBADM 権限を与えますが、データベース内で暗黙の SECADM 権限は与えません。
- データベース・レベルで機能する管理権限レベル:
  - DBADM (データベース管理者)

    DBADM 権限レベルはデータベース・レベルで適用され、1 つのデータベースに対する管理権限を与えます。このデータベース管理者は、オブジェクトの作成、データベース・コマンドの発行、および表データへのアクセスに必要な権限を所有します。また、データベース管理者は、 CONTROL や個々の特権を付与または取り消すことができます。
  - SECADM (セキュリティー管理者)

    SECADM 権限レベルはデータベース・レベルで適用されます。これは表を保護するために使用されるロール、トラステッド・コンテキスト、監査ポリシー、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、およびセキュリティー・ラベルの作成、変更 (該当する場合)、およびドロップを行うのに必要な権限です。また、ロール、セキュリティー・ラベル、および免除の認可および取り消しのためと、SETSESSIONUSER 特権の認可および取り消しのために必要な権限でもあります。 SECADM 権限を持つユーザーは、所有していないオブジェクトの所有権を移行することができます。また、このようなユーザーは、AUDIT ステートメントを使用して、サーバー側の特定のデータベースまたはデータベース・オブジェクトに監査ポリシーを関連付けることもできます。

    SECADM 権限には表に格納されたデータにアクセスする固有の特権はなく、他の追加の固有の特権もありません。この権限の付与が行えるのは、SYSADM

権限をもつユーザーだけとなります。SECADM 権限をユーザーに付与することはできますが、グループ、ロール、または PUBLIC には付与できません。

- インスタンス・レベルで機能するシステム制御権限レベル:
  - SYSCTRL (システム制御)

    SYSCTRL 権限レベルは、システム・リソースに影響を与える操作に対する制御を可能にします。例えば、SYSCTRL 権限を持つユーザーは、データベースの作成、更新、開始、停止、またはドロップを行うことができます。さらに、このユーザーはインスタンスの開始または停止を行うことができますが、表データへのアクセスはできません。 SYSCTRL 権限を持つユーザーには、SYSMON もまた与えられます。
  - SYSMAINT (システム保守)

    SYSMAINT 権限レベルは、インスタンスに関連したすべてのデータベースに対する保守操作を実行するのに必要な権限を与えます。 SYSMAINT 権限を持つユーザーは、データベースの更新と構成、データベースまたは表スペースのバックアップ、既存のデータベースのリストア、およびデータベースのモニターを行うことができます。 SYSCTRL と同様に、SYSMAINT は表データへのアクセス権限を与えません。 SYSMAINT 権限を持つユーザーには、SYSMON 権限もまた与えられます。
- SYSMON (システム・モニター) 権限レベル

  SYSMON は、データベース・システム・モニターの使用に必要な権限を与えます。インスタンス・レベルで機能します。
- データベース権限

  表やルーチンの作成、表へのデータのロードなどのアクティビティーを実行するには、特定のデータベース権限が必要です。例えば、ロード・ユーティリティーを使ってデータを表にロードするには、LOAD データベース権限が必要です (その表に対する INSERT 特権も必要です)。

14 ページの図 4 は、権限とその制御の範囲 (データベース、データベース・マネージャー) の間の関係を示します。

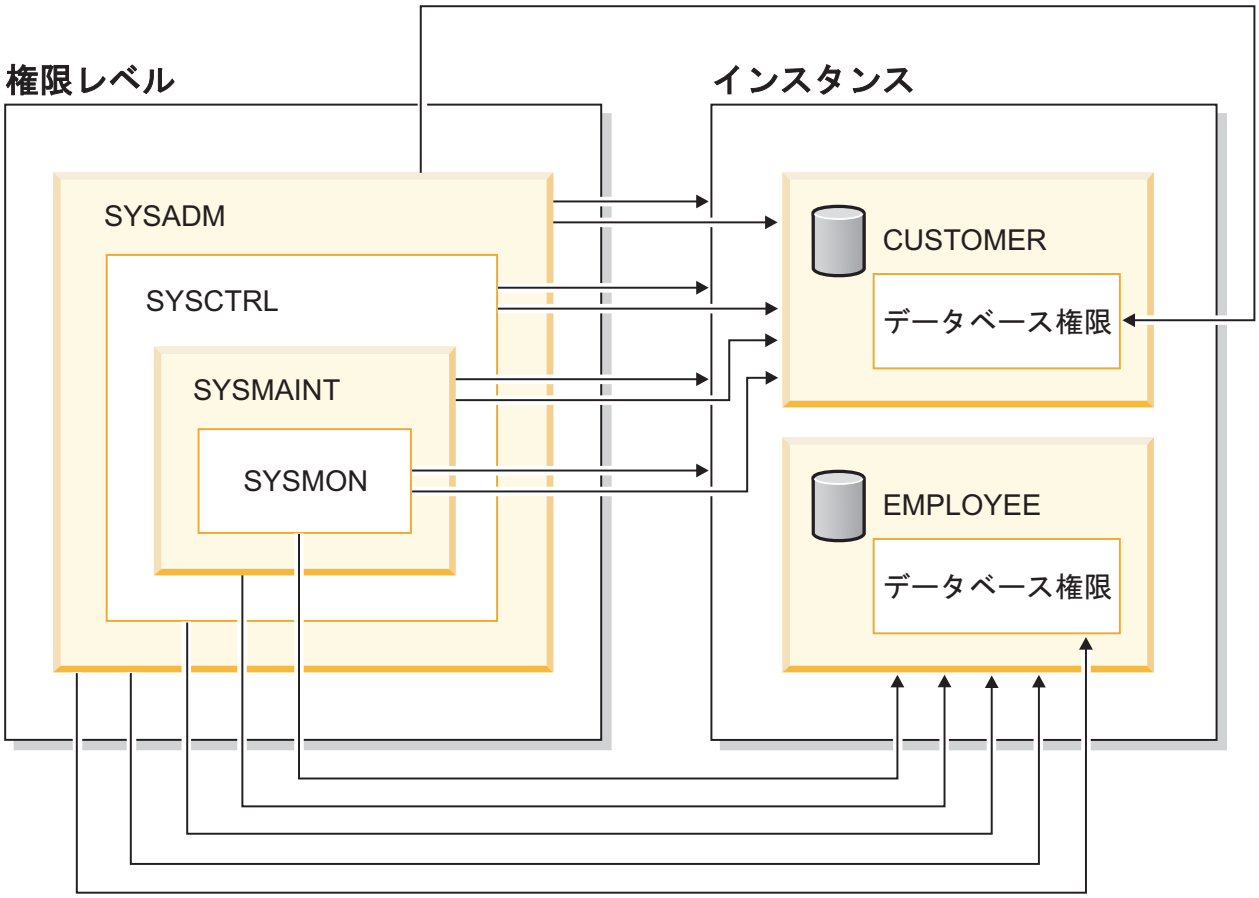


図 4. 権限の階層

特権

特権 とは、ユーザーが実行を許可されているアクティビティーです。許可ユーザーは、オブジェクトを作成することができ、所有しているオブジェクトにアクセス権を持ち、GRANT ステートメントを使用することによって、所有オブジェクトに対する特権を他のユーザーに渡すことができます。

特権は、個々のユーザー、グループ、または PUBLIC に付与できます。 PUBLIC は、将来のユーザーを含むすべてのユーザーで構成される特殊グループです。グループのメンバーであるユーザーは、グループがサポートされている場合は、グループに付与された特権を間接的に利用できます。

CONTROL 特権: オブジェクトに対する CONTROL 特権を持っているユーザーは、そのデータベース・オブジェクトにアクセスでき、そのオブジェクトに対する他のユーザーの特権を付与または取り消すことができます。

注: CONTROL 特権は、表、ビュー、ニックネーム、索引、およびパッケージにのみ適用されます。

別のユーザーがそのオブジェクトに対する CONTROL 特権を必要とする場合、SYSADM または DBADM 権限を持つユーザーが、そのオブジェクトに対する CONTROL 特権を付与することができます。

場合によっては、オブジェクトの作成者がそのオブジェクトに対する CONTROL 特権を自動的に取得します。

個別特権: ユーザーが特定オブジェクトに対して特定のタスクを実行できるようにするために、個別特権を与えることができます。管理権限 (SYSADM または DBADM) を持つユーザー、または CONTROL 特権を持つユーザーは、他のユーザーの特権を付与または取り消すことができます。

個別特権およびデータベース権限は特定の機能の実行を許可しますが、同じ特権または権限を他のユーザーに与えることはできません。 GRANT ステートメントで WITH GRANT OPTION を使用すれば、表、ビュー、スキーマ、パッケージ、ルーチン、シーケンスに関する特権を他のユーザーに対して GRANT できる権利を、他のユーザーに拡張して与えることができます。ただし、WITH GRANT OPTION を使用する場合、特権を GRANT する人が、いったん GRANT された特権を取り消すことはできません。特権を取り消すためには、SYSADM 権限、DBADM 権限、または CONTROL 特権を持っていなければなりません。

パッケージまたはルーチン内のオブジェクトに対する特権: ユーザーにパッケージまたはルーチンを実行する特権があると、パッケージまたはルーチン内で使用されるオブジェクトに対する特定の特権が必ずしも必要とされません。パッケージまたはルーチンに静的 SQL または XQuery ステートメントが含まれる場合、パッケージの所有者の特権がそれらのステートメントに使用されます。パッケージまたはルーチンに動的 SQL または XQuery ステートメントが含まれる場合、特権の検査に使用される許可 ID は、動的照会ステートメントを発行するパッケージの DYNAMICRULES バインド・オプションの設定と、パッケージがルーチン (AUDIT_ARCHIVE、AUDIT_LIST_LOGS、AUDIT_DELIM_EXTRACT の監査ルーチンは除く) のコンテキストで使用される際にそれらのステートメントが発行されるかどうかによって異なります。

1 つのユーザーまたはグループに対して、個々の特権または権限をいくつか組み合わせて許可することもできます。特権をオブジェクトに関連付ける場合、そのオブジェクトはすでに存在していなければなりません。例えば、表がそれ以前に作成されているのでなければ、その表についての SELECT 特権をユーザーに与えることはできません。

注: ユーザーまたはグループを表す許可名が権限と特権を付与され、しかもその許可名で作成されたユーザーまたはグループがいない場合には、注意が必要です。後で、その許可名を使用してユーザーまたはグループが作成され、その許可名に関連するすべての権限と特権を自動的に受け取る可能性があります。

すでに付与された特権を取り消すには、REVOKE ステートメントを使用します。 1 つの許可名から特権を取り消すと、すべての許可名によって付与された特権が取り消されます。

ある許可名から特権を取り消しても、その許可名によって特権を付与された他の許可名からその同じ特権が取り消されることはありません。例えば、ユーザー CLAIRE が SELECT WITH GRANT OPTION をユーザー RICK に与えた後、RICK が SELECT を BOBBY および CHRIS に与えたとします。もし CLAIRE が SELECT 特権を RICK から取り消しても、 BOBBY と CHRIS は引き続き SELECT 特権を保持します。

LBAC 信用証明情報

セキュリティー管理者は、ラベル・ベースのアクセス制御 (LBAC) を使用して、個々の行および個々の列ごとに、どのユーザーに書き込みアクセスがあり、どのユーザーに読み取りアクセスがあるのかを厳密に決定することができます。セキュリティー管理者は、セキュリティー・ポリシーを作成して LBAC システムを構成します。セキュリティー・ポリシー では、どのデータに誰がアクセスできるかの決定で使用される基準が記述されます。任意の 1 つの表を保護するためには、1 つのセキュリティー・ポリシーしか使用できませんが、異なるテーブルには、異なるセキュリティー・ポリシーを用いて保護することができます。

セキュリティー・ポリシーを作成した後、セキュリティー管理者は、そのポリシーの一部分となる、セキュリティー・ラベルおよび免除と呼ばれるデータベース・オブジェクトを作成します。セキュリティー・ラベルは一連のセキュリティー基準を表現したものとなります。免除は、作成したセキュリティー・ポリシーで保護されたデータにアクセスする場合に、これを保有するユーザーがセキュリティー・ラベルの比較について、定められた規則を免れることが出来ますものとなります。

作成が完了すると、セキュリティー・ラベルを表の個々の列と行に関連付けてそこに保持されているデータを保護することができます。セキュリティー・ラベルにより保護されるデータは、保護データと呼ばれます。セキュリティー管理者は、ユーザーにセキュリティー・ラベルを付与することにより、保護データへのアクセスを許可します。ユーザーが保護データへのアクセスを試行すると、そのユーザーのセキュリティー・ラベルが、データを保護しているセキュリティー・ラベルと比較されます。セキュリティー・ラベルには、保護ラベルによってブロックされるものと、されないものがあります。

オブジェクトの所有権

オブジェクトが作成されるとき、1 つの許可 ID に対して、そのオブジェクトの*所有権* が割り当てられます。所有権を与えられているユーザーは、任意の適用できる SQL または XQuery ステートメントを使ってそのオブジェクトを参照することを許可されます。

スキーマ内でオブジェクトを作成するとき、ステートメントの許可 ID は、暗黙的または明示的に指定されるスキーマ内でオブジェクトを作成するのに必要な特権を持っていなければなりません。つまり、許可名がスキーマの所有者であるか、スキーマに対する CREATEIN 特権を持っている必要があります。

注: 表スペース、バッファー・プール、またはデータベース・パーティション・グループを作成するときには、この要件は適用されません。これらのオブジェクトはスキーマ内には作成されません。

オブジェクトが作成されるとき、ステートメントの許可 ID がそのオブジェクトの定義者になり、オブジェクトの作成後にデフォルトでオブジェクトの所有者になります。

注: ただし、1 つの例外があります。 CREATE SCHEMA ステートメントで AUTHORIZATION オプションを指定した場合、 CREATE SCHEMA 操作の一部として作成されるすべてのオブジェクトは、 AUTHORIZATION オプションが指定する許可 ID によって所有されます。ただし、最初の CREATE SCHEMA 操作の後で

スキーマ内で作成されるすべてのオブジェクトは、特定の CREATE ステートメントに関連した許可 ID によって所有されます。

例えば、ステートメント CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C1 INT) によって、スキーマ SCOTTSTUFF および表 SCOTTSTUFF.T1 が作成され、このどちらもユーザー SCOTT によって所有されます。ここで、ユーザー BOBBY に対して SCOTTSTUFF スキーマに対する CREATEIN 特権が与えられ、BOBBY が表 SCOTTSTUFF.T1 への索引を作成するとします。索引はスキーマの後で作成されるため、 SCOTTSTUFF.T1 への索引を所有するのは BOBBY です。

特権は、作成されるオブジェクトのタイプに応じて、以下のようにオブジェクト所有者に割り当てられます。

- CONTROL 特権は、新しく作成される表、索引、およびパッケージに対して暗黙的に付与されます。この特権を持つオブジェクト作成者は、そのデータベース・オブジェクトにアクセスでき、そのオブジェクトに対する他のユーザーの特権を付与または取り消すことができます。他のユーザーがそのオブジェクトに対する CONTROL 特権を要求した場合、 SYSADM または DBADM 権限を持つユーザーが、そのオブジェクトに対する CONTROL 特権を付与する必要があります。オブジェクト所有者は、CONTROL 特権を取り消すことができません。
- ビュー定義によって参照されるすべての表、ビュー、およびニックネームに対する CONTROL 特権をオブジェクト所有者が持っている場合、新しく作成されるビューに対して CONTROL 特権が暗黙的に付与されます。
- 他のオブジェクト (トリガー、ルーチン、シーケンス、表スペース、バッファー・プールなど) には、 CONTROL 特権が関連付けられません。オブジェクト所有者は、オブジェクトに関連付けられるすべての特権を自動的に受け取ります (さらに所有者は、サポートされている場合、GRANT ステートメントで WITH GRANT オプションを使用することで、これらの特権を他のユーザーに与えることができます)。また、オブジェクト所有者は、オブジェクトの変更、コメントの追加、およびオブジェクトのドロップを行うことができます。これらの許可はオブジェクト所有者に暗黙的に与えられ、取り消すことはできません。

表の変更など、オブジェクトに対する特定の特権は、所有者によって付与できます。また SYSADM または DBADM 権限を持つユーザーによって所有者から取り消せます。 表にコメントするなど、オブジェクトに対する特定の特権は、所有者によって付与できません。また所有者から取り消せません。TRANSFER OWNERSHIP ステートメントを使用してこれらの特権を別のユーザーに移動します。オブジェクトが作成されるとき、ステートメントの許可 ID がそのオブジェクトの定義者になり、オブジェクトの作成後にデフォルトでオブジェクトの所有者になります。 ただし、パッケージが作成され、OWNER バインド・オプションが指定されている場合、パッケージ内の静的 SQL ステートメントによって作成されたオブジェクトの所有者は OWNER バインド・オプションの値となります。 さらに、AUTHORIZATION 節が CREATE SCHEMA ステートメントに指定される場合、AUTHORIZATION キーワードの後に指定される許可名はスキーマの所有者です。

セキュリティー管理者またはオブジェクト所有者は、TRANSFER OWNERSHIP ステートメントを使用してデータベース・オブジェクトの所有権を変更することができます。そこで、許可 ID を修飾子として使用してオブジェクトを作成してから TRANSFER OWNERSHIP ステートメントを使用して管理者オブジェクトに持つ所

有権を許可 ID に移動することで、管理者は許可 ID のためにオブジェクトを作成できます。

システム・カタログ・ビュー

データベース・マネージャーは、その制御下のデータに関する情報の組み込まれた一連の表とビューを管理しています。これらの表とビューをまとめて、*システム・カタログ* と呼びます。

このシステム・カタログには、表、ビュー、索引、パッケージ、および関数といったデータベース・オブジェクトの論理および物理構造に関する情報が含まれています。統計情報もあります。データベース・マネージャーは、システム・カタログの情報が常に正確であるように管理します。

システム・カタログ・ビューは、ほかのデータベースのビューと類似しています。システム・カタログ・ビューのデータを照会するために、SQL ステートメントを使用することができます。更新可能なシステム・カタログ・ビューのセットを使用して、そのシステム・カタログの特定の値を変更することができます。

アプリケーションのプロセス、並行性、およびリカバリー

すべての SQL プログラムは、*アプリケーション・プロセス* またはエージェントの一部として実行されます。アプリケーション・プロセスには、1 つ以上のプログラムの実行が関係しており、データベース・マネージャーがリソースを割り当てたりロックしたりする場合の単位となります。異なるいくつかのプログラムの実行、または同じプログラムの複数の異なる実行には、異なる複数のアプリケーション・プロセスが関係しています。

同時に複数のアプリケーション・プロセスが同じデータへのアクセスを要求することがあります。このような状況でデータ保全性を維持するためのメカニズムとして *ロッキング* があります。これは、たとえば 2 つのアプリケーション・プロセスが同時にデータの同じ行を更新するのを防ぐ、などの処理を行います。

データベース・マネージャーは、あるアプリケーション・プログラムが行った変更でまだコミットされていないものが、誤って他のプロセスに認識されることのないよう、ロックを獲得します。プロセスが終了すると、データベース・マネージャーは、アプリケーション・プロセスのためにデータベース・マネージャーが獲得し保持していたロックをすべて解放します。もっと早い時期にロックを解放するには、アプリケーション・プロセス自体で明示的に要求する必要があります。この操作は *コミット* と呼ばれ、これにより作業単位中に獲得していたロックが解放され、作業単位中に加えられた変更がデータベースにコミットされます。

データベース・マネージャーには、アプリケーション・プロセスが行った変更で、まだコミットされていないものを取り消す手段が用意されています。これは、アプリケーション・プロセス側に障害が発生したとき、またはデッドロックやロック・タイムアウト状態などで必要になります。アプリケーション・プロセスで、自分の行ったデータベースへの変更を取り消すように明示的に要求することができます。これは*ロールバック* 操作を使って行います。

作業単位 とは、アプリケーション・プロセス内の、リカバリー可能な一連の操作のことです。作業単位は、アプリケーション・プロセスが開始されたときと、アプリケーションの終了以外の理由で直前の作業単位が終了したときに、開始します。作業単位は、コミット操作、ロールバック操作、またはアプリケーション・プロセスの終了によって終了します。コミットまたはロールバック操作は、それによって終了する作業単位の中で行われたデータベースへの変更内容にしか影響しません。

このような変更がコミットされないまま残っている間は、他のアプリケーション・プロセスはそれらの変更を認識することはできませんし、変更をバックアウトすることも可能です。ただし、分離レベルが非コミット読み取り (UR) である場合にはこの限りではありません。データベースの変更内容がコミットされると、他のアプリケーション・プロセスからその変更内容にアクセスできるようになり、ロールバックによってバックアウトすることはできなくなります。

DB2 コール・レベル・インターフェース (CLI) および組み込み SQL を使用すると、*並行トランザクション* と呼ばれる接続モードを使用できます。これは、それぞれが独立したトランザクションである複数の接続をサポートします。 1 つのアプリケーションが同じデータベースに対して複数の接続を並行して行うことができます。

データベース・マネージャーがアプリケーション・プロセスのために獲得したロックは、作業単位が終了するまで保持されます。ただし、分離レベルがカーソル固定 (CS、カーソルが行から行に移動されるとロックは解放される) か非コミット読み取り (UR、ロックは取得されない) の場合はこの限りではありません。

アプリケーション・プロセスが自分自身のロックのために操作できなくなるということは決してありません。しかしながら、アプリケーションが並行してトランザクションを使用する場合、一方のトランザクションによるロックのために、他方のトランザクションの運用が影響を受ける可能性があります。

作業単位の開始と終了によって、アプリケーション・プロセス内の整合点が定義されます。たとえば、銀行業務のトランザクションで、ある口座から別の口座へ資金を振り込むことがあります。このようなトランザクションでは、その資金を第 1 の口座から減算してから、第 2 の口座に加算する、ということが必要になります。減算のステップの直後の段階では、データに矛盾が生じています。資金が第 2 の口座に加算して初めて、整合性が取り戻されるわけです。両方のステップが完了したときに、コミット操作を実行して作業単位を終了させれば、他のアプリケーション・プロセスが変更内容を利用できるようになります。 1 つの作業単位が終わる前に障害が発生すると、データベース・マネージャーはコミットされていない変更内容をロールバックし、その作業単位の開始時点でのデータ整合性をリストアします。

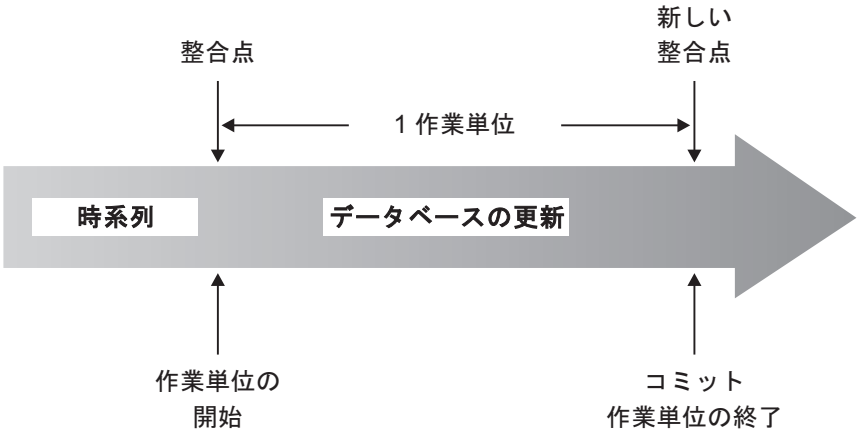


図 5. COMMIT ステートメントの作業単位

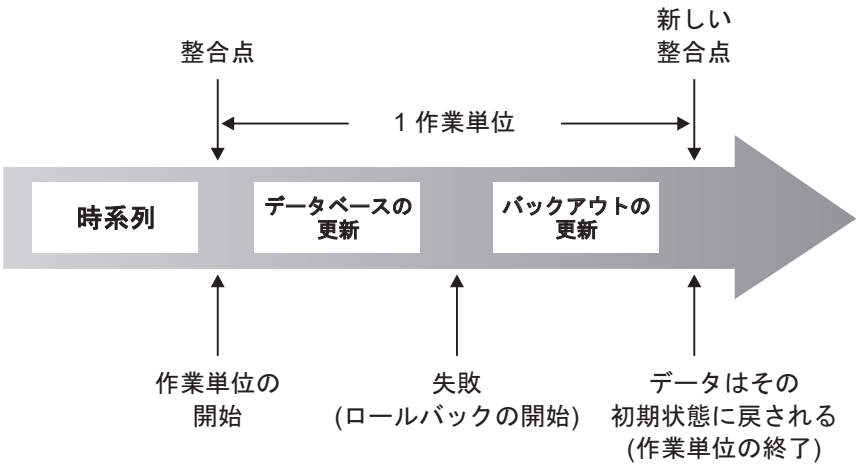


図 6. ROLLBACK ステートメントの作業単位

分離レベル

アプリケーション・プロセスに関連する*分離レベル* は、並行して実行している他のアプリケーション・プロセスからそのアプリケーション・プロセスを分離する度合いを定義します。したがって、アプリケーション・プロセスの分離レベルは、以下を指定します。

- アプリケーションによって読み取られ更新される行を、並行して実行される他のアプリケーション・プロセスから使用できる度合い。
- 並行して実行される他のアプリケーション・プロセスの更新活動によってアプリケーションが影響を受ける度合い。

静的 SQL ステートメントの分離レベルは、パッケージの属性として指定され、そのパッケージを使用するアプリケーション・プロセスに適用されます。分離レベルは、ISOLATION バインドまたはプリコンパイル・オプションを設定することにより、プログラム準備処理で指定されます。動的 SQL ステートメントの場合、デフォルトの分離レベルは、ステートメントを作成するパッケージに指定された分離レベルです。 SET CURRENT ISOLATION ステートメントを使用すると、セッション内で発行される動的 SQL に対して代替分離レベルを指定できます。詳しくは、

『CURRENT ISOLATION 特殊レジスター』を参照してください。静的 SQL ステートメントと動的 SQL ステートメントのどちらの場合でも、select-statement (SELECT ステートメント) 内の isolation-clause (分離節) は、特殊レジスター (設定されている場合) と BIND オプションの両方の値をオーバーライドします。詳しくは、『Select-statement』を参照してください。

これによって、並行アプリケーション・プロセスによるデータ・アクセスは、ロックの種類に応じて制限または禁止されます。宣言済み一時表とその行は、宣言したアプリケーションしかアクセスできないので、ロックされることはありません。

データベース・マネージャーでは、大きく分けて次の 3 つのロック・カテゴリーがサポートされています。

共用 並行アプリケーション・プロセスの操作を、読み取り専用のデータ操作のみに制限します。

更新 並行アプリケーション・プロセスは行の更新を宣言したのではない限り、データへの読み取り専用操作に限定されます。データベース・マネージャーは、行を現在見ているプロセスがそれを更新する可能性があると想定します。

排他 同時アプリケーション・プロセスがどのような形であれ、そのデータにアクセスできないようにします。読み取りはできてもデータの変更はできない*非コミット読み取り* の分離レベルのアプリケーション・プロセスにはあてはまりません。

ロッキングは基本表の行について行われます。しかし、データベース・マネージャーが複数の行ロックを単一の表ロックに置き換えることがあります。これを*ロック・エスカレーション* といいます。アプリケーション・プロセスでは、最低限必要なロック・レベルが確保されます。

データベース・マネージャーは 4 つの分離レベルをサポートしています。分離レベルとは関係なく、データベース・マネージャーは、挿入、更新、または削除の対象となる行のすべてに排他ロックをかけます。このため、どの分離レベルでも、このアプリケーション・プロセスが 1 作業単位中に変更する行は、その作業単位が完了するまで他のアプリケーション・プロセスから変更されることは決してありません。分離レベルには、以下のものがあります。

- 反復可能読み取り (RR)

  このレベルでは、以下の点が確実に守られます。

  – 作業単位の中で読み取られた行は、その作業単位が完了するまで、他のアプリケーション・プロセスから変更されません。行は、対応する OPEN ステートメントと同じ作業単位で読み取られます。 CLOSE ステートメントにオプションの WITH RELEASE 節を使用することの意義は、カーソルが再オープンされた場合に、反復不能読み取りおよび幻像読み取りの防止は、以前アクセスした行には適用されないことにあります。
  – 他のアプリケーション・プロセスによって変更される行は、そのアプリケーション・プロセスがコミットするまで読み取ることができません。

  反復可能読み取りレベルでは、幻像読み取り行の出現は許されません (「読み取り固定」の説明を参照)。

RR レベルで実行しているアプリケーション・プロセスは、排他ロック以外に、参照するすべての行に対して共有ロックを獲得します。さらに、アプリケーション・プロセスが並行アプリケーション・プロセスの影響から完全に分離されるようにロッキングが実行されます。

- 読み取り固定 (RS)

  反復可能読み取りレベルと同様に、読み取り固定レベルでは、以下の点が確実に守られます。

  – 作業単位の中で読み取られた行は、その作業単位が完了するまで、他のアプリケーション・プロセスから変更されません。行は、対応する OPEN ステートメントと同じ作業単位で読み取られます。 CLOSE ステートメントにオプションの WITH RELEASE 節を使用することの意義は、カーソルが再オープンされた場合に、反復不能読み取りの防止は、以前アクセスした行には適用されないことにあります。
  – 他のアプリケーション・プロセスによって変更される行は、そのアプリケーション・プロセスがコミットするまで読み取ることができません。

  反復可能読み取りとは異なり読み取り固定では、アプリケーション・プロセスは、他の並行アプリケーション・プロセスの影響から完全には分離されません。RS レベルでは、同じ照会を複数回発行するアプリケーション・プロセスで行が追加されていくという場合があります。これらの行は、データベースに新しい情報を付加する他のアプリケーション・プロセスによって作成されたものです。このような付加行は*幻像読み取り行* と呼ばれます。

  幻像読み取り行は、たとえば次のような状況で発生します。

  1. アプリケーションのプロセス P1 が、検索条件を満たす一連の *n* 個の行を読み取る。
  2. 次にアプリケーション・プロセス P2 が、検索条件を満たす 1 つまたは複数の行を挿入し、それらの挿入行をコミットする。
  3. P1 が同じ検索条件で一連の行を再度読み取り、元の行と P2 によって挿入された行を両方とも獲得する。

  RS 分離レベルで実行しているアプリケーション・プロセスは、排他ロックに加えて、条件に合うすべての行に対して少なくとも共用ロックを獲得します。

- カーソル固定 (CS)

  反復可能読み取りレベルと同様に、カーソル固定レベルでは、他のアプリケーション・プロセスによって変更された行は、そのアプリケーション・プロセスによってコミットされるまではまったく読み取り不能になります。

  反復可能読み取りとは異なりカーソル固定では、すべての更新可能なカーソルの現在行が、他のアプリケーション・プロセスによって変更されないことだけが確実になります。このため、作業単位の中で読み取られた行が、他のアプリケーション・プロセスによって変更される可能性があります。

  CS 分離レベルで実行しているアプリケーション・プロセスは、排他ロックに加えて、すべてのカーソルの現在行に対して少なくとも共用ロックを獲得します。

- 非コミット読み取り (UR)

SELECT INTO、読み取り専用カーソルによる FETCH、INSERT で使用される全選択、 UPDATE での行の全選択、またはスカラーの全選択なら、非コミット読み取りレベルでは以下のことが可能です。

- 作業単位の中で読み取られた行は、すべて他のアプリケーション・プロセスから変更できます。
- 他のアプリケーション・プロセスで変更された行は、そのアプリケーション・プロセスで変更をコミットしていなくても、すべて読み取ることができます。

これ以外の操作では、CS レベルの規則が適用されます。

分離レベルの比較

次の表は、分離レベルについて要約しています。

	UR	CS	RS	RR
アプリケーションは、他のアプリケーション・プロセスによって行われた非コミットの変更を見ることができるか ?	はい	いいえ	いいえ	いいえ
アプリケーションは、他のアプリケーション・プロセスによって行われた非コミットの変更を更新できるか ?	いいえ	いいえ	いいえ	いいえ
ステートメントをもう一度実行した場合、他のアプリケーション・プロセスによる影響があるか ? *下記の現象 P3 (幻像) を参照。*	はい	はい	はい	いいえ[3]
「更新された」行を他のアプリケーション・プロセスで更新することができるか ? 下記の注 1 を参照。	いいえ	いいえ	いいえ	いいえ
「更新された」行を、UR 以外の分離レベルの他のアプリケーション・プロセスで読み取れるか ?	いいえ	いいえ	いいえ	いいえ
「更新された」行を UR の分離レベルの他のアプリケーション・プロセスで読み取れるか ?	はい	はい	はい	はい
「アクセスされた」行を、他のアプリケーション・プロセスによって更新することが可能か ? *下記の現象 P2 (反復不能読み取り) を参照。*	はい	はい	いいえ	いいえ
「アクセスされた」行を他のアプリケーション・プロセスが読み取ることができるか ?	はい	はい	はい	はい
「現在」行を、他のアプリケーション・プロセスによって更新または削除することが可能か ? *下記の現象 P1 (ダーティー読み取り) を参照。*	下記の注 2 を参照。	下記の注 2 を参照。	いいえ	いいえ

	UR	CS	RS	RR

注:

1. アプリケーションが表に対する読み取りと書き込みの両方を行う場合、分離レベルはアプリケーションのための保護を提供しません。たとえば、アプリケーションは表でカーソルをオープンし、それからその同じ表に挿入、更新、または削除の操作を実行します。オープン・カーソルでもっと行を取り出してゆくにつれて、アプリケーションが矛盾するデータを見つける場合があります。
2. カーソルが更新可能でない場合、CS では、現在行を他のアプリケーション・プロセスによって更新または削除できる場合もあります。たとえば、バッファリングによってクライアントの現在の行が、サーバーの実際の現在行の値と違うということが引き起こされる場合があります。
3. 読み取りを行ってから次の読み取りを行うまでの間に、ラベル・ベースのアクセス制御 (LBAC) クレデンシャルが変化した場合、アクセス可能な行が異なるために、2 度目の読み取りの結果は異なるものとなることがあります。

現象の例:

P1 *ダーティー読み取り*。作業単位 UW1 が行を変更するとします。 UW1 が COMMIT を実行する前に、作業単位 UW2 がその行を読み取るとします。次に UW1 が ROLLBACK を実行したとすると、 UW2 は実行しない行を読み取ったことになります。

P2 *反復不能読み取り*。作業単位 UW1 が行を読み取るとします。作業単位 UW2 がその行を変更してから、COMMIT を実行するとします。 UW1 がもう一度その行を読み取ると、値が修正されていることがあります。

P3 *幻像*。作業単位 UW1 が、ある検索条件を満たしている *n* 個の行を読み取るとします。次に作業単位 UW2 が、その検索条件を満たしている 1 つまたは複数の行を挿入して、 COMMIT を実行します。 UW1 が同じ検索条件でもう一度最初の読み取りを実行すると、元の行のほかに挿入された行が追加されていることになります。

表スペース

表スペース は、表、索引、ラージ・オブジェクト、およびロング・データを含む、ストレージ構造です。表スペースは、データベース・パーティション・グループの中に常駐します。表スペースによって、データベースと表データのロケーションをコンテナーに直接割り当てることができます。 (コンテナーとしては、ディレクトリー名、装置名、ファイル名があります。) これによってパフォーマンスが改善され、構成の柔軟性が高くなります。

表スペースはデータベース・パーティション・グループの中にあるので、表の保持のために選択された表スペースは、その表のデータがデータベース・パーティション・グループ内の複数のデータベース・パーティションにわたって分配される方法を定義します。 1 つの表スペースが複数のコンテナーにわたる場合もあります。(1 つまたは複数の表スペースからの) 複数のコンテナーを、同じ物理ディスク (またはドライブ) 上に作成することも可能です。自動ストレージの表スペースを使用している場合は、この表スペースはデータベース・マネージャーによって処理されます。自動ストレージの表スペースを*使用していない* 場合、パフォーマンスを向上させるためには、各コンテナーごとに異なるディスクを使用する必要があります。

図 7 は、データベース内の表および表スペースと、そのデータベースに関連するコンテナーのリレーションシップを示しています。

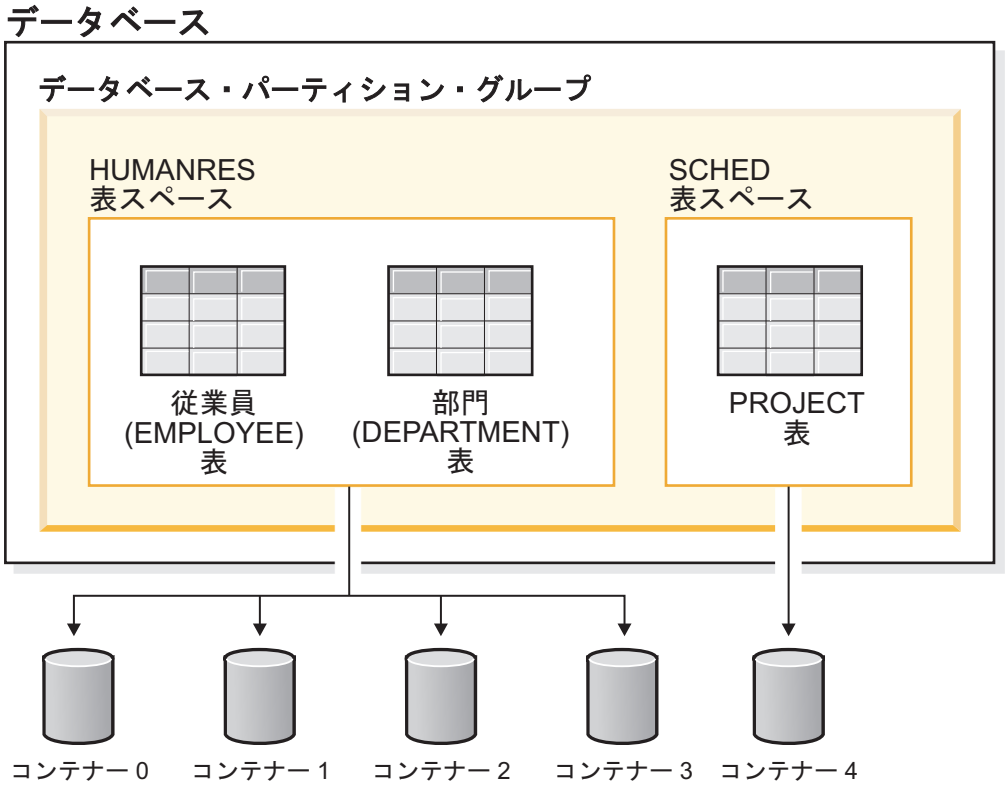


図 7. データベース内の表スペースと表

EMPLOYEE および DEPARTMENT 表は HUMANRES 表スペースにあり、これはコンテナー 0、1、2、および 3 にわたっています。 PROJECT 表は SCHED 表スペースにあり、コンテナー 4 に入っています。この例では、各コンテナーが別のディスクにあることを示しています。

データベース・マネージャーは、コンテナー間でデータ・ロードの平衡を取ろうとします。結果として、データを格納するのにすべてのコンテナーが使われます。別のコンテナーを使用する前に、データベース・マネージャーがコンテナーに書き込むページ数は、エクステント・サイズ と呼ばれます。データベース・マネージャーは、毎回最初のコンテナーから表データを格納し始めるとは限りません。

26 ページの図 8 は、エクステント・サイズが 4 KB ページ 2 つ分の HUMANRES 表スペースを表しています。それぞれのページには、割り振りエクステントが小さく設定されているコンテナーが 4 つずつあります。 DEPARTMENT 表と EMPLOYEE 表は、どちらも 7 ページあり、 4 つのコンテナーすべてにわたっています。

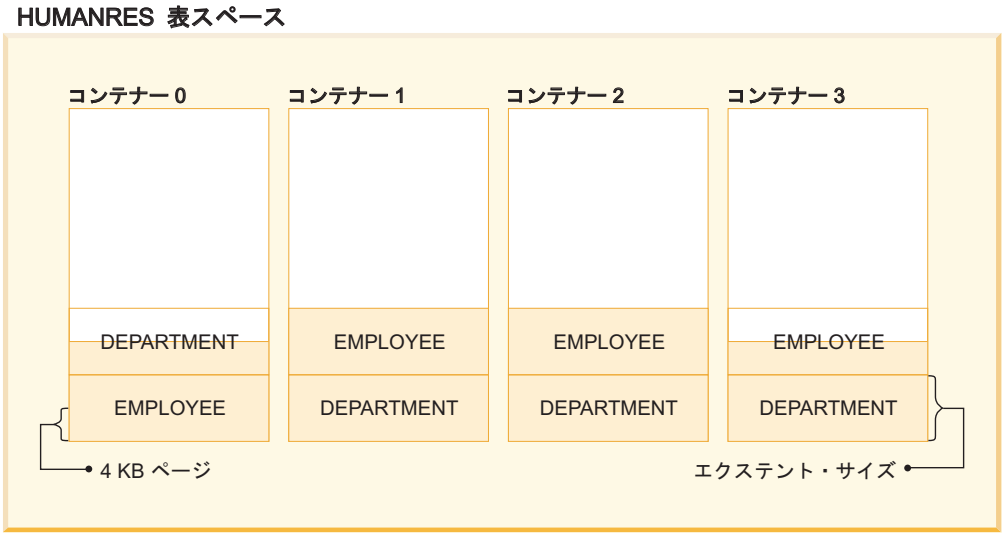


図 8. 表スペースの中のコンテナーとエクステント

文字変換

ストリング は、文字を表す一連のバイトです。ストリング内のすべての文字は共通のコード化表現を持っています。場合によっては、このような文字を別のコード化表現に変換しなければならないことがあります。これは文字変換 という処理です。文字変換が必要な場合は自動的に実行されますが、正常に終了すればその実行はアプリケーションからは認識されません。

文字変換は、SQL ステートメントがリモートで実行される場合に発生する可能性があります。たとえば、送信システムと受信システムでコード化表現が異なるかもしれない以下のシナリオを考えてみてください。

- ホスト変数の値が、アプリケーション・リクエスターからアプリケーション・サーバーに送信された。
- 結果列の値が、アプリケーション・サーバーからアプリケーション・リクエスターに送信された。

以下が文字変換を説明する際に使用する用語のリストです。

文字セット
定義済みの文字の集まり。たとえば、いくつかのコード・ページには次の文字セットが出現します。

- A から Z の 26 個の文字 (アクセント記号なし)
- a から z の 26 個の文字 (アクセント記号なし)
- 0 から 9 の数字
- . , : ; ? ( ) ' " / - _ & + % * = < >

コード・ページ
コード・ポイントに対する一連の文字の割り当て。たとえば、コード・ページ 850 の ASCII エンコード・スキーマでは、 "A" にはコード・ポイント X'41' が割り当てられ、 "B" にはコード・ポイント X'42' が割り当てられています。 1 つのコード・ページの中では、それぞれのコード・ポイントはただ 1 つの特定の意味をもちます。コード・ページはデータベースの 1 つ

の属性です。アプリケーション・プログラムがデータベースに接続している場合、データベース・マネージャーがそのアプリケーションのコード・ページを判別します。

コード・ポイント
文字を表すユニークなビット・パターン。

コード化スキーム
文字データを表現するために使用する規則の集まり。たとえば、次のとおりです。

- 1 バイト ASCII
- 1 バイト EBCDIC
- 2 バイト ASCII
- 1 バイト / 2 バイト混合 ASCII

以下の図は、典型的な文字セットが、 2 つの異なるコード・ページの異なるコード・ポイントにどのようにマップされるかを示しています。エンコード・スキーマが同じでも多くの異なるコード・ページがあり、同じコード・ポイントであってもコード・ページが異なれば異なる文字を表す場合があります。さらに、文字ストリングの中の 1 バイトは、 1 バイト文字セット (SBCS) の文字を表すとは限りません。文字ストリングは、混合およびビット・データにも使用されます。*混合データ*は、1 バイト文字、2 バイト文字、またはマルチバイト文字の混合です。*ビット・データ* (FOR BIT DATA か BLOB、またはバイナリー・ストリングと定義されている列) は、どの文字セットにも関連していません。

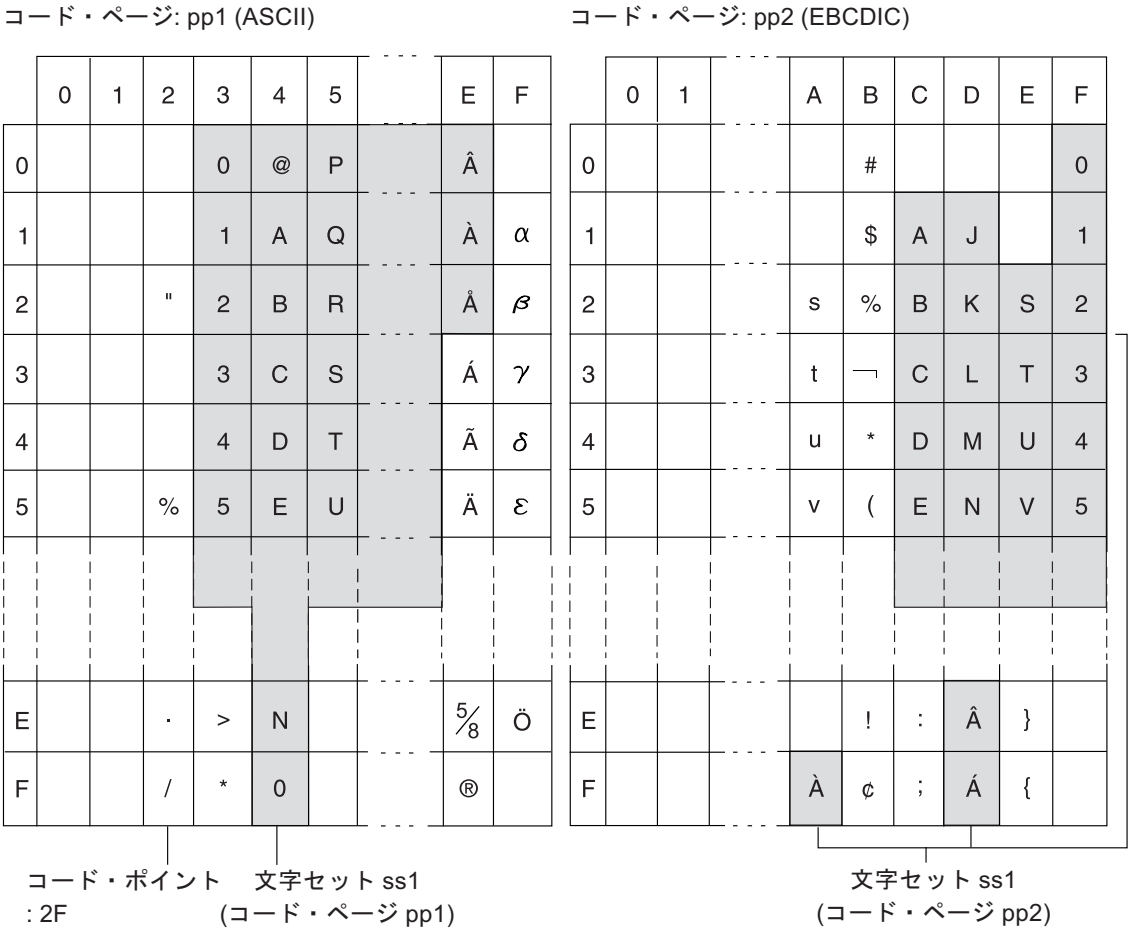


図 9. 異なるコード・ページにおける文字セットのマッピング

データベース・マネージャーは、アプリケーションがデータベースにバインドされるときに、すべての文字ストリングのコード・ページ属性を判別します。可能なコード・ページ属性には以下のものがあります。

データベース・コード・ページ
データベース・コード・ページは、データベース構成ファイルに保管されています。値はデータベースの作成時に指定され、その後の変更は不可能です。

アプリケーション・コード・ページ
このコード・ページの下でアプリケーションが実行されます。これはアプリケーションがバインドされたときのコード・ページと同じであるとは限りません。

セクション・コード・ページ
このコード・ページの下で SQL ステートメントが実行されます。通常、セクション・コード・ページはデータベース・コード・ページです。ただし、次の場合には Unicode コード・ページ (UTF-8) が使用されます。

- 非 Unicode データベース内で Unicode エンコード・スキームを使って作成された表をステートメントが参照する場合。
- 非 Unicode データベース内で PARAMETER CCSID UNICODE を使って定義された表関数をステートメントが参照する場合。

コード・ページ 0
これは、FOR BIT DATA の値または BLOB の値の入った式から派生したストリングを表すものです。

文字ストリング・コード・ページ属性は次のとおりです。

- 列は、データベース・コード・ページ、Unicode コード・ページ (UTF-8)、またはコード・ページ 0 (FOR BIT DATA または BLOB と定義されている場合) のいずれでもかまいません。
- 定数と特殊レジスター (たとえば USER、CURRENT SERVER、SERVER) は、セクション・データベース・コード・ページです。 SQL ステートメントがデータベースにバインドされるときに、必要があれば定数はアプリケーション・コード・ページからデータベース・コード・ページに変換されてから、セクション・コード・ページに変換されます。
- 入力ホスト変数は、アプリケーション・コード・ページです。バージョン 8 以降は、入力ホスト変数内のストリング・データは必要に応じて使用前にアプリケーション・コード・ページからセクション・コード・ページに変換されます。ホスト変数がビット・データとして解釈される状況で使用されると例外が発生します。たとえば、ホスト変数が FOR BIT DATA と定義された列に割り当てられる場合です。

スカラー操作、セット演算、または連結のように、ストリング・オブジェクトを結合する操作のコード・ページ属性は、一連の規則を使用して判別されます。実行時には、コード・ページ属性を使用して、ストリングのコード・ページ変換の要件が判別されます。

各国語サポートと SQL ステートメント

SQL ステートメントのコーディングは、言語に依存していません。SQL キーワードは、示されているとおりに入力する必要があります。大文字の場合、小文字の場合、大/小文字混合の場合があります。SQL ステートメントに記述するデータベース・オブジェクト、ホスト変数、プログラム・ラベルの名前は、アプリケーション・コード・ページでサポートされている文字でなければなりません。

サーバーは、ファイル名を変換しません。ファイル名のコーディングには、ASCII インバリアント・セットを使用するか、ファイル・システムに物理的に格納されている 16 進値でパスを指定します。

マルチバイト環境では、インバリアント文字セットに属していない特殊文字と見なされる文字が 4 つあります。それらの文字は、以下のとおりです。

- 2 バイトのパーセント文字と 2 バイトの下線文字。LIKE 処理で使用します。
- 2 バイトのスペース文字。特に、GRAPHIC ストリングのブランクの埋め込みに使用します。
- 2 バイトの置換文字。ソース・コード・ページとターゲット・コード・ページの間にマッピングが存在しない場合に、コード・ページ変換時の置換文字として使用します。

これらの各文字のコード・ポイントをコード・ページごとにまとめます。

表 1. 特殊 2 バイト文字のコード・ポイント

コード・ページ	2 バイトのパーセント	2 バイトの下線	2 バイトのスペース	2 バイトの置換文字
932	X'8193'	X'8151'	X'8140'	X'FCFC'
938	X'8193'	X'8151'	X'8140'	X'FCFC'
942	X'8193'	X'8151'	X'8140'	X'FCFC'
943	X'8193'	X'8151'	X'8140'	X'FCFC'
948	X'8193'	X'8151'	X'8140'	X'FCFC'
949	X'A3A5'	X'A3DF'	X'A1A1'	X'AFFE'
950	X'A248'	X'A1C4'	X'A140'	X'C8FE'
954	X'A1F3'	X'A1B2'	X'A1A1'	X'F4FE'
964	X'A2E8'	X'A2A5'	X'A1A1'	X'FDFE'
970	X'A3A5'	X'A3DF'	X'A1A1'	X'AFFE'
1381	X'A3A5'	X'A3DF'	X'A1A1'	X'FEFE'
1383	X'A3A5'	X'A3DF'	X'A1A1'	X'A1A1'
13488	X'FF05'	X'FF3F'	X'3000'	X'FFFD'
1363	X'A3A5'	X'A3DF'	X'A1A1'	X'A1E0'
1386	X'A3A5'	X'A3DF'	X'A1A1'	X'FEFE'
5039	X'8193'	X'8151'	X'8140'	X'FCFC'
1392	X'A3A5'	X'A3DF'	-	-

EXPORT、IMPORT、および LOAD ユーティリティーの場合、コード・ページ 5488 は 1392 と同等です。

Unicode データベースの GRAPHIC スペースは X'0020' であり、euc-Japan データベースと euc-Taiwan データベースで使用する GRAPHIC スペースの X'3000' とは異なります。X'0020' も X'3000' も、Unicode 標準のスペース文字です。これらの EUC データベースのデータと Unicode データベースを比較するときには、GRAPHIC スペースのコード・ポイントの違いを考慮に入れる必要があります。

分散リレーショナル・データベースへの接続

分散リレーショナル・データベースは、正式なリクエスター/サーバー・プロトコルと機能に基づいて構築されます。

アプリケーション・リクエスター は、接続の両端のうち、アプリケーション側をサポートするものです。アプリケーション・リクエスターは、アプリケーションからのデータベース要求を分散データベース・ネットワークでの使用に適した通信プロトコルに変換します。これらの要求は、接続のもう一方の端のデータベース・サーバー によって受信され、処理されます。アプリケーション・リクエスターとデータベース・サーバーは連携して通信とロケーションに関する考慮事項を処理し、アプリケーションがローカル・データベースにアクセスしているのと変わりなく稼働できるようにします。

表やビューを参照する SQL ステートメントを実行できるようにするためには、その前に、データベース・マネージャーのアプリケーション・サーバーにアプリケー

ション・プロセスを接続しておく必要があります。 CONNECT ステートメントにより、アプリケーション・プロセスとそのサーバーの接続が確立されます。

CONNECT ステートメントには、次の 2 つのタイプがあります。

- CONNECT (タイプ 1) では、作業単位 (リモート作業単位) セマンティクスごとに 1 つのデータベースがサポートされます。
- CONNECT (タイプ 2) では、作業単位 (アプリケーション制御の分散作業単位) セマンティクスごとに複数のデータベースがサポートされます。

DB2 コール・レベル・インターフェース (CLI) および組み込み SQL は、*並行トランザクション* と呼ばれる接続モードに対応しています。このモードでは、複数の接続が可能で、各接続が独立したトランザクションになります。1 つのアプリケーションが同じデータベースに対して複数の接続を並行して行うことができます。

アプリケーション・サーバーは、プロセスが開始される環境に対してローカルでもリモートでもかまいません。アプリケーション・サーバーは、分散リレーショナル・データベースを使用していない環境でも存在しています。この環境には、CONNECT ステートメントに指定されるアプリケーション・サーバーを記述するローカル・ディレクトリーが組み込まれています。

アプリケーション・サーバーは、表やビューを参照するバインドされた形式の静的 SQL ステートメントを実行します。このバインドされたステートメントは、データベース・マネージャーがバインド操作でそれ以前に作成したパッケージから取り出されます。

ほとんどの場合、アプリケーション・サーバーに接続しているアプリケーションは、そのアプリケーション・サーバーのデータベース・マネージャーでサポートされているステートメントや節を使用できます。このことは、一部のステートメントや節をサポートしないデータベース・マネージャーのアプリケーション・リクエスターによってアプリケーションが実行される場合でも当てはまります。

イベント・モニター

イベント・モニターを使用して、指定されたイベントの発生時に、データベースおよび接続されたアプリケーションに関する情報を収集します。イベントは、接続、デッドロック、ステートメント、トランザクションなどの、データベース・アクティビティーの遷移を表します。モニターするイベント (1 つ以上) のタイプごとにイベント・モニターを定義することができます。例えば、デッドロック・イベント・モニターは、デッドロックが発生するのを待機します。発生すると、関係するアプリケーションおよび競合するロックに関する情報を収集します。

デフォルトでは、すべてのデータベースについて DB2DETAILDEADLOCK という名前のイベント・モニターが定義されています。このモニターは、デッドロック・イベントに関する詳細情報を記録します。 DB2DETAILDEADLOCK イベント・モニターは、データベースの開始時に自動的に開始されます。

スナップショット・モニターは一般に、予防的な保守および問題分析のために使用されますが、イベント・モニターは、現時点の問題について管理者に警告し、また今にも起こりそうな問題を追跡するするために使用されます。

イベント・モニターを作成するには、CREATE EVENT MONITOR SQL ステートメントを使用します。イベント・モニターは、それらがアクティブなときにだけイベント・データを収集します。イベント・モニターを活動化または非活動化するには、 SET EVENT MONITOR STATE SQL ステートメントを使用します。イベント・モニターの状況 (アクティブか非アクティブか) は、 SQL 関数 EVENT_MON_STATE によって判別することができます。

CREATE EVENT MONITOR SQL ステートメントを実行すると、それが作成するイベント・モニターの定義が、以下のデータベース・システム・カタログ表に保管されます。

- SYSCAT.EVENTMONITORS: データベースについて定義されたイベント・モニター
- SYSCAT.EVENTS: データベースについてモニターされるイベント
- SYSCAT.EVENTTABLES: 表イベント・モニターのためのターゲット表

それぞれのイベント・モニターには、モニター・エレメント内のインスタンスのデータの、独自の専用論理ビューがあります。特定のイベント・モニターが非活動化された後、再活動化されると、これらのカウンターのビューがリセットされます。リセットは、新たに活動化されたイベント・モニターだけで行われます。他のすべてのイベント・モニターは、引き続きカウンター値の独自のビューを使用し続けます (追加があればそのカウンター値に追加します)。

イベント・モニターの出力は、非パーティション SQL 表、ファイル、または Named PIPE に送ることができます。

複数のデータベース・パーティションにまたがるデータベース・パーティション

データベース・マネージャーは、パーティション・データベースの複数のデータベース・パーティション (ノード) にまたがってデータを柔軟に拡散させることができます。ユーザーは、分散キーを宣言することによってデータを分散する方法を選択することができ、また、データを保管するデータベース・パーティション・グループおよび表スペースを選択することによって、いくつの、そしてどのデータベース・パーティションに表データを分散できるかを決定することができます。

さらに、分散マップ (更新可能) は、分散キー値のデータベース・パーティションへのマッピングを指定します。これにより、大きな表では 1 つのパーティション・データベース全体にまたがってワークロードを柔軟に均等化することができる一方、小さな表の場合はアプリケーション設計者の選択しだいで、1 つまたは少数のデータベース・パーティションに保管することもできます。ローカルの各データベース・パーティションに、保管するデータのローカル索引を備えると、ローカル・データへのアクセス性能が向上します。

パーティション・データベースで、分散キーは一連のデータベース・パーティションに表データを分散するために使用されます。索引データも、それに対応する表とともにパーティション化され、各データベース・パーティションにローカル保管されます。

複数のデータベース・パーティションにまたがるデータベース・パーティション

データベース・パーティションを使用してデータを保管するには、事前にパーティションをデータベース・マネージャーに対して定義しておく必要があります。データベース・パーティションは、db2nodes.cfg というファイルに定義されます。

パーティション・データベース・パーティション・グループの表スペースの表の分散キーは、CREATE TABLE ステートメント、または ALTER TABLE ステートメントに指定されます。指定されていない場合、デフォルト解釈によって、表の分散キーは、主キーの最初の列から作成されます。主キーが定義されていない場合、デフォルトの分散キーは、その表で定義されている、データ・タイプが long または LOB 以外の最初の列になります。パーティション・データベース内の表には、データ・タイプが long でも LOB でもない列が少なくとも 1 つは必要になります。単一パーティション・データベース・パーティション・グループの表スペースの表は、明示的に指定されている場合に限り、分散キーを持ちます。

行は、以下のようにデータベース・パーティション内に配置されます。

1. ハッシュ・アルゴリズム (データベース・パーティション機能) が分散キーのすべての列に適用され、その結果として分散マップの索引の値が生成されます。
2. 分散マップで、その索引の値にあるデータベース・パーティション番号は、行が保管されるデータベース・パーティションを識別します。

データベース・マネージャーは、*部分デクラスタリング* をサポートします。これは、システム内のデータベース・パーティションのサブセット (つまりデータベース・パーティション・グループ) に表を配分できることを意味しています。システム内のすべてのデータベース・パーティションにわたって表を配分する必要はありません。

データベース・マネージャーは、結合や副照会でアクセスされているデータが同じデータベース・パーティション・グループ内の同じデータベース・パーティションにある場合に、それを認識する能力を備えています。これを、*表コロケーション* といいます。同一の分散キー値を使用して連結されている表の行は、同一のデータベース・パーティションに置かれます。データベース・マネージャーは、データが保管されているデータベース・パーティションでの結合処理や副照会処理の実行を選択できます。これによって、大幅なパフォーマンスの改善が得られる場合もあります。

連結する表は、以下の条件を満たしている必要があります。

- 同一のデータベース・パーティション・グループにあり、再配分されていない。(再分散されると、データベース・パーティション・グループ内の表は別の分散マップを使用する可能性があります。このような表はコロケーションされません。)
- 分散キーの列の数が同数である。
- 分散キーの対応する列に、データベース・パーティションの面で互換性がある。
- 同一のデータベース・パーティションに定義されている単一のパーティション・データベース・パーティション・グループにある。

パーティション表でのラージ・オブジェクトの動作

パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

パーティション表のラージ・オブジェクトは、デフォルトでは、対応するデータ・オブジェクトと同じ表スペースに保管されます。このことは、表スペースを 1 つだけ使用するパーティション表にも、複数の表スペースを使用するパーティション表にも当てはまります。パーティション表のデータが複数の表スペースに保管される場合は、ラージ・オブジェクト・データも複数の表スペースに保管されます。

このデフォルト動作をオーバーライドするには、CREATE TABLE ステートメントの LONG IN 節を使用します。表の LONG データが保管される表スペースのリストを指定できます。デフォルト動作をオーバーライドするようにする場合、LONG IN 節で指定する表スペースは LARGE 表スペースでなければなりません。 1 つ以上のデータ・パーティションの LONG データが別個の表スペースに保管されるように指定する場合は、その表のすべてのデータ・パーティションについてそうする必要があります。つまり、一部のデータ・パーティションについてはリモート側で LONG データを保管し、他のデータ・パーティションについてはローカル側で LONG データを保管するということはできません。デフォルト動作を使用しても、LONG IN 節を使用してデフォルト動作をオーバーライドしても、各データ・パーティションに対応した LONG オブジェクトが作成されます。SMS 表スペースの場合、LONG データは、それが属するデータ・オブジェクトと同じ表スペースになければなりません。各データ・パーティションに対応する LONG データ・オブジェクトを保管するために使用されるすべての表スペースで、ページ・サイズ、エクステント・サイズ、ストレージ・メカニズム (DMS または SMS)、タイプ (REGULAR または LARGE) が、同じでなければなりません。リモート LARGE 表スペースは LARGE タイプである必要があります。また、SMS は使用できません。

例えば次の CREATE TABLE ステートメントは、各データ・パーティションの CLOB データのオブジェクトを、データと同じ表スペースに作成します。

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

LONG IN を使用することにより、データがある表スペースとは別個の 1 つ以上の LARGE 表スペースに CLOB データを配置できます。

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

注: LONG IN 節は、表レベルでデータ・パーティションごとに 1 つだけ使用できます。

DB2 フェデレーテッド・システム

フェデレーテッド・システム

フェデレーテッド・システムは、特殊なタイプの分散データベース管理システム (DBMS) です。 1 つのフェデレーテッド・システムは、フェデレーテッド・サーバーとして働く DB2 インスタンス、フェデレーテッド・データベースとして働くデータベース、1 つまたは複数のデータ・ソース、およびデータベースとデータ・ソースにアクセスするクライアント (ユーザーおよびアプリケーション) からなっています。

フェデレーテッド・システムを使用すると、1 つの SQL ステートメントで複数のデータ・ソースに分散要求を送信することができます。例えば、DB2 表、Oracle 表、および XML タグ付きファイルにあるデータを 1 つの SQL ステートメントで結合できます。次の図は、フェデレーテッド・システムのコンポーネントおよびアクセス可能なデータ・ソースの例を示しています。

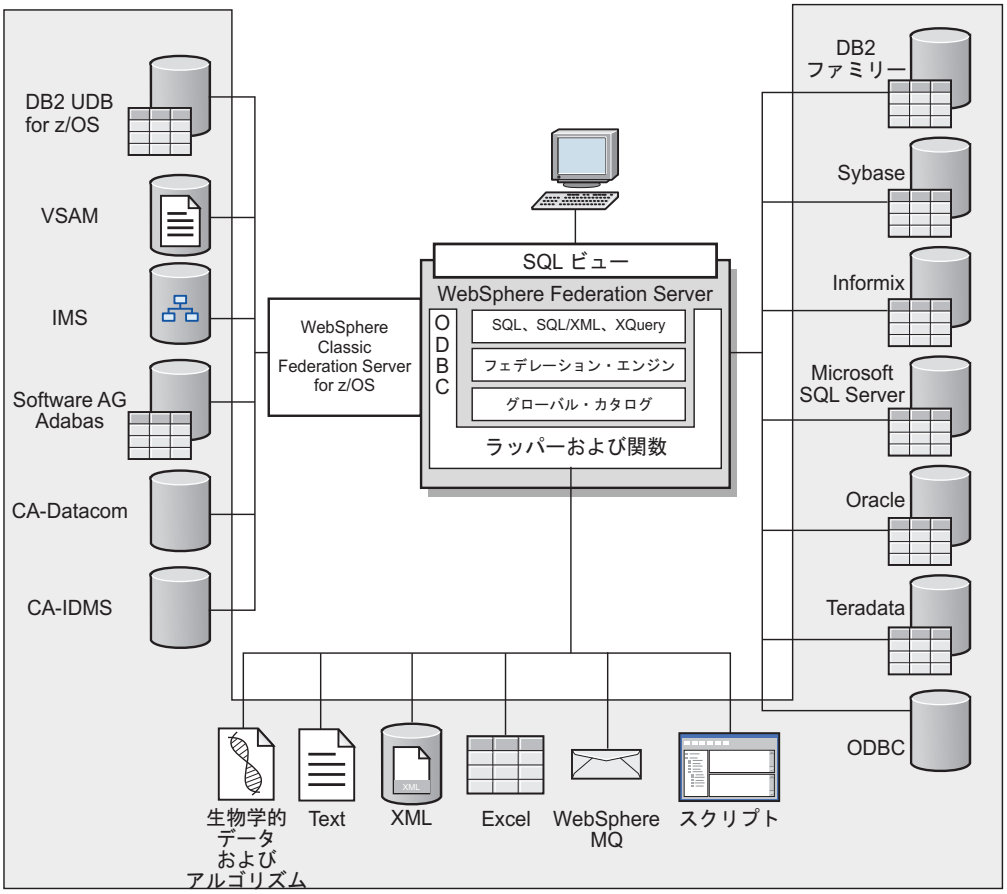


図 10. フェデレーテッド・システムのコンポーネント

フェデレーテッド・システムの持つ強力な機能により、以下のことが可能になります。

- ローカル表とリモート・データ・ソースのデータを、それらすべてがフェデレーテッド・データベースにローカルに保管されているかのように結合する。
- リレーショナル・データ・ソースのデータを、それらがフェデレーテッド・データベースに保管されているかのように更新する。
- リレーショナル・データ・ソースとの間で双方向にデータを移動する。
- データ・ソースに要求を送信して処理させることにより、データ・ソース側で処理するという利点を生かす。
- データ・ソース側での SQL の制約を補うため、分散要求の一部をフェデレーテッド・サーバー側で処理する。

データ・ソースとは?

フェデレーテッド・システムでは、リレーショナル・データベース (Oracle または Sybase など) または非リレーショナル・データ・ソース (BLAST 検索アルゴリズムまたは XML タグ付きファイル) をデータ・ソース にすることができます。

特定のデータ・ソースを介することで、他のデータ・ソースにアクセスすることも可能です。例えば、ODBC ラッパーを使用して、DB2 UDB for z/OS、IMS™、CA-IDMS、CA-Datacom、Software AG Adabas、VSAM などの、WebSphere® Classic Federation Server for z/OS のデータ・ソースにアクセスできます。

データ・ソースへのアクセスに使用される方式つまりプロトコルは、データ・ソースのタイプによって異なります。例えば、DRDA® は DB2 for z/OS のデータ・ソースにアクセスするために使用されます。

データ・ソースはオートノマス (自律的) です。例えば、フェデレーテッド・サーバーが Oracle データ・ソースに照会を送信しているときに、その同じデータ・ソースに Oracle アプリケーションがアクセスしてもかまいません。保全性およびロッキング制約が損なわれない限り、フェデレーテッド・システムが他のデータ・ソースへのアクセスを独占または制限することはありません。

フェデレーテッド・データベース

エンド・ユーザーおよびクライアント・アプリケーションにとって、データ・ソースは、 DB2 の単一の集合データベースに見えます。ユーザーとアプリケーションは、フェデレーテッド・サーバーが管理するフェデレーテッド・データベース とやり取りを行います。

フェデレーテッド・データベースにはデータの関する情報を保管するシステム・カタログが入っています。このフェデレーテッド・データベースのシステム・カタログには、データ・ソースとその特性を示すカタログ項目が入っています。フェデレーテッド・サーバーは、フェデレーテッド・データベース・システム・カタログに保管された情報およびデータ・ソース・ラッパーを検討した上で、SQL ステートメントを処理する最善のプランを決めます。

フェデレーテッド・システムは、データ・ソースからのデータがフェデレーテッド・データベース内の通常のリレーショナルの表またはビューであるかのように、SQL ステートメントを処理します。その結果、次のようになります。

- フェデレーテッド・システムはリレーショナル・データを非リレーショナルのフォーマットのデータと結合することができます。データ・ソースが異なる SQL ダイアレクトを使用していたり、あるいは SQL をまったくサポートしていなくても、あてはまります。
- フェデレーテッド・データベースの特性とデータ・ソースの特性に相違がある場合、フェデレーテッド・データベースの特性が優先されます。照会の結果は DB2 セマンティクスに準拠します。照会の結果の計算に他の DB2 以外のデータ・ソースからのデータが使用される場合でも同様です。

  例:

  – フェデレーテッド・サーバーが使用するコード・ページは、データ・ソースが使用するコード・ページと異なります。この場合、データ・ソースの文字データは、フェデレーテッド・ユーザーに戻される際、フェデレーテッド・データベースで使用されているコード・ページに基づいて変換されます。
  – フェデレーテッド・サーバーが使用する照合シーケンスは、データ・ソースが使用する照合シーケンスと異なります。この場合、文字データに対するソート操作はすべて、データ・ソースではなくフェデレーテッド・サーバーで行われます。

SQL コンパイラー

DB2 SQL コンパイラーは、照会の処理に役立つ情報を収集します。

データ・ソースからデータを入手するため、ユーザーおよびアプリケーションは SQL の照会をフェデレーテッド・データベースにサブミットします。照会をサブミットすると、DB2 SQL コンパイラーはグローバル・カタログ内の情報およびデータ・ソース・ラッパーを検討し、照会の処理に役立てます。この情報には、データ・ソースへの接続に関する情報、サーバー情報、マッピング、索引情報、および処理統計が含まれます。

ラッパーおよびラッパー・モジュール

ラッパー とは、フェデレーテッド・データベースがデータ・ソースと対話するためのメカニズムです。フェデレーテッド・データベースは、ライブラリーに保管されたルーチン (ラッパー・モジュール という) を使用してラッパーをインプリメントします。

これらのルーチンを使用することで、フェデレーテッド・データベースは、データ・ソースへの接続やデータ・ソースからのデータ検索の繰り返しなどの操作を実行できます。通常、フェデレーテッド・インスタンスの所有者は、CREATE WRAPPER ステートメントを使用して、ラッパーをフェデレーテッド・データベースに登録します。 DB2_FENCED オプションを使用すると、ラッパーを fenced またはトラステッドとして登録することができます。

ラッパーは、アクセスするデータ・ソースのタイプごとに 1 つ作成します。例えば、3 つの DB2 for z/OS データベース表、1 つの DB2 for System i 表、2 つの Informix® 表、および 1 つの Informix ビューにアクセスするとします。このとき作成する必要があるのは、DB2 データ・ソース・オブジェクト用のラッパーを 1 つと、 Informix データ・ソース・オブジェクト用のラッパーを 1 つです。これらのラッパーをフェデレーテッド・データベースに登録すれば、すぐにそれらのラッパ

ーを使用して対応するデータ・ソースから他のオブジェクトにアクセスすることが可能になります。例えば、DRDA ラッパーを使用すれば、すべての DB2 ファミリーのデータ・ソース・オブジェクト (DB2 Database for Linux, UNIX, and Windows、DB2 for z/OS、DB2 for System i、および DB2 Server for VM and VSE) からのデータ・ソースにアクセスできます。

各データ・ソース・オブジェクトを特定して識別 (名前やロケーションなど) するには、サーバー定義とニックネームを使用します。

ラッパーは多くの作業を行います。そのいくつかは次のようなものです。

- データ・ソースに接続します。ラッパーは、データ・ソースの標準の接続 API を使用します。
- データ・ソースに照会をサブミットします。
  - SQL をサポートするデータ・ソースの場合、照会は SQL でサブミットされます。
  - SQL をサポートしないデータ・ソースの場合、照会は、ソースに固有の照会言語に、または一連のソース API 呼び出しに変換されます。
- データ・ソースから結果セットを受け取ります。ラッパーは、データ・ソースの標準 API を使用して、結果セットを受信します。
- データ・ソースのデフォルトのデータ・タイプ・マッピングについてのフェデレーテッド・データベースの照会に応答します。ラッパーには、データ・ソース・オブジェクトにニックネームを作成する時に使用される、デフォルトのタイプ・マッピングが入っています。リレーショナル・ラッパーの場合、ユーザーが作成するデータ・タイプ・マッピングは、デフォルトのデータ・タイプ・マッピングをオーバーライドします。ユーザー定義のデータ・タイプ・マッピングは、グローバル・カタログに保管されます。
- データ・ソースのデフォルトの関数マッピングについてのフェデレーテッド・データベースの照会に応答します。フェデレーテッド・データベースは、照会の計画で使用するためのデータ・タイプのマッピング情報を必要とします。ラッパーには、DB2 関数がデータ・ソースの関数と対応付けられるかどうか、またどのように関数が対応付けられるかを、フェデレーテッド・データベースが判断する際に必要となる情報が含まれています。この情報は、データ・ソースが照会操作を実行できるかどうかを判断するために、SQL コンパイラーにより使用されます。リレーショナル・ラッパーの場合、ユーザーが作成する関数マッピングは、デフォルトの関数タイプ・マッピングをオーバーライドします。ユーザー定義の関数マッピングは、グローバル・カタログに保管されます。

ラッパー・オプション は、ラッパーを構成するため、またはWebSphere Federation Server がどのようにラッパーを使用するかを定義するために使用されます。

サーバー定義およびサーバー・オプション

データ・ソース用のラッパーを作成した後、フェデレーテッド・インスタンスの所有者はデータ・ソースをフェデレーテッド・データベースに定義します。

インスタンス所有者は、データ・ソースを識別するための名前を指定し、またデータ・ソースに関するその他の情報も指定します。この情報には、次のものが含まれます。

- データ・ソースのタイプおよびバージョン
- データ・ソースのデータベース名 (RDBMS のみ)
- データ・ソースに固有のメタデータ

例えば、DB2 ファミリーのデータ・ソースは複数のデータベースを持つことができます。そのため、フェデレーテッド・サーバーがどのデータベースに接続できるかを定義に指定しておく必要があります。それとは対照的に、Oracle データ・ソースが持つデータベースは 1 つなので、フェデレーテッド・サーバーは名前を知らなくてもそのデータベースに接続することができます。そのため、Oracle データ・ソースのフェデレーテッド・サーバー定義にデータベース名は含まれていません。

インスタンス所有者がフェデレーテッド・サーバーに提供する、名前およびその他の情報をまとめてサーバー定義 と呼びます。データ・ソースはデータを求める要求に応答し、それ自体がサーバーとして機能します。

サーバー定義の作成および変更には、CREATE SERVER および ALTER SERVER ステートメントを使用します。

サーバー定義内の情報の一部は、サーバー・オプション として保管されます。サーバー定義を作成するにあたって、サーバーに関して指定可能なオプションを理解しておくことは大切です。

サーバー・オプションは、データ・ソースへの接続が次々に続く間は持続されるように設定するか、または 1 つの接続が継続している間のみ持続されるように設定することができます。

ユーザー・マッピング

ユーザー・マッピングは、フェデレーテッド・サーバー上の 許可 ID とリモート・データ・ソースに接続するために必要な情報との間の関連です。

ユーザー・マッピングを作成するには、CREATE USER MAPPING ステートメントを使用します。ステートメント内に、ローカル許可 ID、サーバー定義に指定されたリモート・データ・ソース・サーバーのローカル名、およびリモート ID とパスワードを指定します。

例えば、リモート・サーバー用にサーバー定義を作成して、リモート・サーバーのローカル名に 'argon' を指定したとします。 Mary にリモート・サーバーへのアクセスを付与するには、次のユーザー・マッピングを作成します。

```
CREATE USER MAPPING FOR Mary
SERVER argon
OPTIONS (REMOTE_AUTHID 'remote_ID', REMOTE_PASSWORD 'remote_pw')
```

Mary が SQL ステートメントを発行してリモート・サーバーに接続するとき、フェデレーテッド・サーバーは以下のステップを実行します。

1. Mary のユーザー・マッピングを検索します。
2. リモート・サーバーに関連付けられたリモート・パスワード 'remote_pw' を暗号解読します。
3. リモート・サーバーに接続するためのラッパーを呼び出します。

4. リモート ID 'remote_ID' および暗号解読したリモート・パスワードをラッパーに渡します。
5. Mary のためのリモート・サーバーへの接続を作成します。

デフォルトでは、フェデレーテッド・サーバーはユーザー・マッピングをグローバル・カタログ内の SYSCAT.USEROPTIONS ビューに保管して、リモート・パスワードを暗号化します。代替方法として、ファイルまたは LDAP サーバーなどの外部リポジトリーを使用して、ユーザー・マッピングを保管することもできます。フェデレーテッド・サーバーと外部リポジトリーとの間のインターフェースを提供するには、ユーザー・マッピング・プラグインを作成します。

ユーザー・マッピングをどのように保管する場合でも、それらに対するアクセスを注意深く制限してください。ユーザー・マッピングで暗号漏えいが発生した場合、リモート・データベース内のデータは、無許可の活動に対してぜい弱になることがあります。

ニックネームとデータ・ソース・オブジェクト

ニックネーム とは、アクセス先のデータ・ソース・オブジェクトを識別するために使用する ID です。ニックネームによって識別されるオブジェクトを、データ・ソース・オブジェクト といいます。

別名が代替名であるのとは異なり、ニックネームはデータ・ソース・オブジェクトの代替名ではありません。ニックネームは、フェデレーテッド・サーバーがオブジェクトを参照するために使用するポインターです。ニックネームは通常、CREATE NICKNAME ステートメントに、特定のニックネーム列オプションとニックネーム・オプションを指定して定義されます。

クライアント・アプリケーションまたはユーザーが分散要求をフェデレーテッド・サーバーにサブミットする場合、その要求でデータ・ソースを指定する必要はありません。その代わりに、要求はデータ・ソース・オブジェクトをそのオブジェクトのニックネームで参照します。ニックネームはデータ・ソースの特定のオブジェクトにマップされます。このようにマッピング (対応付け) されることにより、ニックネームをデータ・ソース名で修飾する必要がなくなります。クライアント・アプリケーションまたはユーザーは、データ・ソース・オブジェクトのロケーションを意識する必要がありません。

ここで、ニックネーム *DEPT* が、 *NFX1.PERSON* という Informix データベース表を表すように定義するとします。SELECT * FROM *DEPT* というステートメントをフェデレーテッド・サーバーから使用できます。しかし、フェデレーテッド・サーバーに *NFX1.PERSON* というローカル表がなければ、フェデレーテッド・サーバーから SELECT * FROM *NFX1.PERSON* というステートメントを使用することはできません (パススルー・セッションは除く)。

データ・ソース・オブジェクトにニックネームを作成すると、オブジェクトについてのメタデータがグローバル・カタログに追加されます。照会オプティマイザーは、このメタデータとラッパー内の情報を使用して、データ・ソース・オブジェクトへのアクセスを容易にします。例えば、索引を持つ表にニックネームを作成する

と、グローバル・カタログにはその索引についての情報が入り、ラッパーには、DB2 のデータ・タイプとデータ・ソースのデータ・タイプとの間のマッピングが入ります。

ラベル・ベースのアクセス制御 (LBAC) を使用するオブジェクトのニックネームはキャッシュに入れられません。そのため、オブジェクトのデータの安全性は確保されます。例えば、Oracle (Net8) ラッパーを使用して、Oracle Label Security を使用する表に対してニックネームを作成する場合、その表の安全性は自動的に確認されます。その結果として生成されるニックネーム・データはキャッシュに入れることができません。したがって、そのデータに対してマテリアライズ照会表を作成することはできません。 LBAC を使用することにより、情報の表示が適切なセキュリティー特権を持つユーザーのみに制限されます。 LBAC がサポートされる前に作成されたニックネームについては、ALTER NICKNAME ステートメントを使用してキャッシングを使用不可にする必要があります。LBAC は、DRDA (DB2 for Linux, UNIX, and Windows バージョン 9.1 以降を使用するデータ・ソースに対応) および Net8 ラッパーの両方によってサポートされています。

ニックネーム列オプション

グローバル・カタログには、ニックネームが付けられたオブジェクトに関する追加のメタデータ情報を入れることができます。このメタデータは、データ・ソース・オブジェクトの特定の列の値を記述したものです。このメタデータを、ニックネーム*列オプション* というパラメーターに割り当てます。

ニックネーム列オプションは、列内のデータを通常の列とは異なる方法で処理するようラッパーに指示します。 SQL コンパイラーと照会オプティマイザーは、メタデータを使用して、データにアクセスするためのよりよいプランを作成します。

ニックネーム列オプションは、ラッパーにその他の情報を提供するためにも使用されます。例えば XML データ・ソースの場合、ニックネーム列オプションは、ラッパーが XML 文書から列を解析するときに使用する XPath 式をラッパーに指示するために使用されます。

フェデレーションを使用すると、DB2 サーバーはニックネームが参照するデータ・ソース・オブジェクトを、あたかもローカル DB2 表であるかのように扱います。したがって、ニックネームを作成するどのデータ・ソース・オブジェクトに対しても、ニックネーム列オプションをセットすることができます。ニックネーム列オプションの中には特定のタイプのデータ・ソース用に作られたものもあり、それらは該当するデータ・ソースにのみ適用できます。

フェデレーテッド・データベースの照合シーケンスとは異なる照合シーケンスを持つデータ・ソースがあるとします。フェデレーテッド・サーバーは通常、文字データを含む列をデータ・ソース側でソートすることはありません。データはフェデレーテッド・データベースに戻され、ローカルにソートが行われます。しかしここで、列が文字データ・タイプ (CHAR または VARCHAR) であり、数字 ('0'、'1'、...、'9') だけが含まれているとします。これは、NUMERIC_STRING ニックネーム列オプションに 'Y' を指定することにより明示できます。そうすることにより、DB2 照会オプティマイザーは、オプションでデータ・ソース側でソートを実行

できるようになります。ソートをリモート側で実行できれば、データをフェデレーテッド・サーバーに持ってきて、ソートをローカルで実行するというオーバーヘッドが避けられます。

ALTER NICKNAME ステートメントを使用することにより、リレーショナル・ニックネームにニックネーム列オプションを定義することもできます。非リレーショナル・ニックネームには、CREATE NICKNAME および ALTER NICKNAME ステートメントを使用してニックネーム列オプションを定義できます。

データ・タイプ・マッピング

フェデレーテッド・サーバーがデータ・ソースからデータを検索するには、データ・ソース側のデータ・タイプが、対応する DB2 のデータ・タイプに対応付けられて (マッピングされて) いなければなりません。

デフォルトのデータ・タイプ・マッピングの例として、以下のものがあります。

- Oracle タイプ FLOAT は DB2 タイプ DOUBLE にマップされます。
- Oracle タイプ DATE は DB2 タイプ TIMESTAMP にマップされます。
- DB2 for z/OS™ タイプ DATE は DB2 タイプ DATE にマップされます。

ほとんどのデータ・ソースの場合、ラッパー内にデフォルトのタイプ・マッピングがあります。 DB2 データ・ソース用のデフォルトのタイプ・マッピングは、DRDA ラッパーにあります。 Informix 用のデフォルトのタイプ・マッピングは INFORMIX ラッパーにあります。その他のタイプ・マッピングについても同様です。

非リレーショナルのデータ・ソースの中には、CREATE NICKNAME ステートメントでデータ・タイプ情報を指定しなければならないものがあります。ニックネームの作成時に、データ・ソース・オブジェクトの列ごとに、対応する DB2 データ・タイプを指定する必要があります。それぞれの列は、データ・ソース・オブジェクト内の特定のフィールドまたは列にマップされている必要があります。

リレーショナル・データ・ソースの場合は、デフォルトのデータ・タイプ・マッピングをオーバーライドできます。例えば、Informix INTEGER データ・タイプは、デフォルトでは DB2 INTEGER データ・タイプにマップします。デフォルト・マッピングをオーバーライドして、Informix の INTEGER データ・タイプを DB2 DECIMAL(10,0) データ・タイプにマップされるようにすることができます。

フェデレーテッド・サーバー

フェデレーテッド・システム内の DB2 サーバーのことを、フェデレーテッド・サーバーといいます。 DB2 インスタンスであればいくつでも、フェデレーテッド・サーバーとして機能するように構成することができます。既存の DB2 インスタンスをフェデレーテッド・サーバーとして使用したり、特にフェデレーテッド・システム専用として新しく作成したりできます。

フェデレーテッド・システムを管理する DB2 インスタンスのことをサーバー と呼びますが、それはこのインスタンスがエンド・ユーザーおよびクライアント・アプリケーションからの要求に応答するからです。フェデレーテッド・サーバーは受信した要求の各部を頻繁にデータ・ソースに送信して処理させます。プッシュダウン

操作は、リモート側で処理される操作です。フェデレーテッド・システムを管理する DB2 インスタンスは、要求をデータ・ソースにプッシュダウンする場合はクライアントとして働きますが、フェデレーテッド・サーバー と呼ばれます。

その他のアプリケーション・サーバーと同様に、フェデレーテッド・サーバーはデータベース・マネージャー・インスタンスです。アプリケーション・プロセスはフェデレーテッド・サーバーに接続し、フェデレーテッド・サーバー内のデータベースに要求をサブミットします。ただし、次の 2 つの主要な機能により、その他のアプリケーション・サーバーとは区別されます。

- フェデレーテッド・サーバーは、部分的または全面的にデータ・ソース向けの要求を受信するように構成されています。フェデレーテッド・サーバーは、これらの要求をデータ・ソースに配布します。
- その他のアプリケーション・サーバーと同様に、フェデレーテッド・サーバーは DRDA 通信プロトコル (over TCP/IP) を使用して、 DB2 ファミリーのインスタンスと通信します。ただし、他のアプリケーション・サーバーと異なり、フェデレーテッド・サーバーはデータ・ソースのネイティブ・クライアントを使用して、データ・ソースにアクセスします。例えば、フェデレーテッド・サーバーは Sybase Open Client を使用して Sybase データ・ソースにアクセスし、 Microsoft SQL Server ODBC ドライバーを使用して Microsoft SQL Server データ・ソースにアクセスします。

サポートされるデータ・ソース

フェデレーテッド・システムを使用してアクセスできるデータ・ソースはたくさんあります。

WebSphere Federation Server は、以下の表に示されているデータ・ソースをサポートしています。最初の表では、データ・クライアント・ソフトウェアの要件をリストしています。クライアント・ソフトウェアは、特に指定されていない限り、別個に入手する必要があります。

アクセスするデータ・ソースに対応するクライアント・ソフトウェアをインストールする必要があります。クライアント・ソフトウェアは、WebSphere Federation Server と同じシステムにインストールする必要があります。DB2 コントロール・センターなどのツールを使用する、および Java アプリケーション (ストアード・プロシージャーおよびユーザー定義関数を含む) を作成して実行するには、適切な Java SDK も必要です。

最新の情報については、Web 上の データ・ソース要件のページを参照してください。

表 2. サポートされるデータ・ソース、クライアント・ソフトウェアの要件、および 32 ビット・オペレーティング・システムによるサポート

			32 ビットのハードウェア・アーキテクチャーおよびオペレーティング・システム	
			X86-32	**X86-32**
データ・ソース	**サポートされているバージョン**	**クライアント・ソフトウェア**	**Linux、RedHat Enterprise Linux (RHEL)、SUSE**	**Windows**
BioRS	5.2	なし	Y	Y
Blast	2.2.1、2.2.3、2.2.4、2.2.9	Blast デーモン (提供される)	Y	Y
DB2 for Linux, UNIX, and Windows	8.1.x、8.2.x、9.1	なし	Y	Y
DB2 for z/OS	7.x、8.x	DB2 Connect™ V9.1	Y	Y
DB2 for iSeries®	5.2、5.3、5.4	DB2 Connect V9.1	Y	Y
DB2 Server for VSE and VM	7.2、7.4	DB2 Connect V9.1	Y	Y
Entrez		なし	Y	Y
フラット・ファイル		なし	Y	Y
HMMER	2.3、2.3.2	HMMER デーモン (提供される)	Y	Y

表 2. サポートされるデータ・ソース、クライアント・ソフトウェアの要件、および 32 ビット・オペレーティング・システムによるサポート (続き)

			32 ビットのハードウェア・アーキテクチャーおよびオペレーティング・システム	
			X86-32	**X86-32**
データ・ソース	**サポートされているバージョン**	**クライアント・ソフトウェア**	**Linux、RedHat Enterprise Linux (RHEL)、SUSE**	**Windows**
Informix	Informix XPS 8.40、8.50、8.51 および Informix IDS IDS 7.31、IDS 9.40、IDS 10.0	Informix Client SDK バージョン 2.81.TC2 以降	Y	Y
Microsoft Excel	2000、2002、2003	Excel 2000、2002 または 2003		Y
Microsoft SQL Server	Microsoft SQL Server 2000/SP3、2005	Windows の場合、Microsoft SQL Server Client ODBC 3.0 (またはそれ以降の) ドライバー UNIX の場合、DataDirect ODBC 5.0 *!ENTITY!* 5.1	Y	Y
ODBC	3.0	ODBC 3.0 に準拠する ODBC ドライバー**	Y	Y
OLE DB	2.7、2.8	OLE DB 2.0 以降	Y	Y
Oracle	8.1.7、9.2、10g、10gR2	Oracle NET クライアント 9.0 - 9.2.0.5 & 10.0 - 10.1.x または NET8 クライアント 8.1.7.x	Y	Y
Sybase	Sybase ASE 12.5、15.0	Sybase Open Client 12.5 - 15.0	Y	Y

表 2. サポートされるデータ・ソース、クライアント・ソフトウェアの要件、および 32 ビット・オペレーティング・システムによるサポート (続き)

			32 ビットのハードウェア・アーキテクチャーおよびオペレーティング・システム	
			X86-32	**X86-32**
データ・ソース	**サポートされているバージョン**	**クライアント・ソフトウェア**	**Linux、RedHat Enterprise Linux (RHEL)、SUSE**	**Windows**
Teradata	2.4、2.5、 2.6	Windows の場合、Teradata Client TTU 7.0 以降、および Teradata API ライブラリー CLIv2 4.7.0 以降。UNIX の場合、Teradata Call-Level Interface Version 2 CLIv2 Release 04.06、04.07 以降		Y
WebSphere Business Integration for PeopleSoft 2.6	PeopleSoft 8.x	なし	Y	Y
WebSphere Business Integration for SAP 2.6	SAP 3.x、4.x	なし	Y	Y
WebSphere Business Integration for Seibel 2.6	Seibel 7、7.5、2000	なし	Y	Y
Web サービス	WSDL 1.0, 1.1 SOAP 1.0、1.1	なし	Y	Y
XML	XML1.0、XML1.1	なし	Y	Y

** ODBC は、さまざまなデータ・ソースへのアクセスに使用できますが、特に RedBrick 6.20cu5 および WebSphere Information Integrator Classic Federation V8.2.x にアクセスするために使用できます。

表 3. 64 ビット・オペレーティング・システムによるサポート

64 ビットのハードウェア・アーキテクチャー	**X86-64**	**X86-64**	**Power**	**Itanium®**	**Power**	**Sparc**	**zSeries®**
オペレーティング・システム	**Linux RHEL SUSE**	**Windows**	**AIX®**	**HP-UX**	**Linux RHEL SUSE**	**Solaris**	**Linux RHEL SUSE**
データ・ソース							
BioRS	Y	Y	Y	Y	Y	Y	Y
Blast	Y	Y	Y	Y	Y	Y	Y
DB2 for Linux, UNIX, and Windows	Y	Y	Y	Y	Y	Y	Y
DB2 for z/OS	Y	Y	Y	Y	Y	Y	Y
DB2 for System i	Y	Y	Y	Y	Y	Y	Y
DB2 Server for VSE and VM	Y	Y	Y	Y	Y	Y	Y
Entrez	Y	Y	Y	Y	Y	Y	Y
HMMER	Y	Y	Y	Y	Y	Y	Y
Informix	Y		Y	Y	Y	Y	Y
Microsoft Excel							
Microsoft SQL Server	Y	Y	Y	Y		Y	
ODBC	Y	Y	Y***	Y		Y***	
OLE DB		Y		Y			
Oracle	Y	Y	Y	Y	Y	Y	Y
Script	Y	Y	Y	Y	Y	Y	Y
Sybase	Y		Y	Y	Y	Y	
Teradata			Y	Y		Y	

表 3. 64 ビット・オペレーティング・システムによるサポート (続き)

64 ビットのハードウェア・アーキテクチャー	**X86-64**	**X86-64**	**Power**	**Itanium®**	**Power**	**Sparc**	**zSeries®**
オペレーティング・システム	**Linux RHEL SUSE**	**Windows**	**AIX®**	**HP-UX**	**Linux RHEL SUSE**	**Solaris**	**Linux RHEL SUSE**
データ・ソース							
WebSphere Business Integration for PeopleSoft 2.6	Y	Y	Y	Y	Y	Y	Y
WebSphere Business Integration for SAP 2.6	Y	Y	Y	Y	Y	Y	Y
WebSphere Business Integration for Seibel 2.6	Y	Y	Y	Y	Y	Y	Y
Web サービス	Y	Y	Y	Y	Y	Y	Y
XML	Y	Y	Y	Y	Y	Y	Y

*** ODBC は、32 ビットおよび 64 ビットのクライアントを使用して RedBrick 6.20cu5 および WebSphere Information Integrator Classic Federation にアクセスするために使用できます。

フェデレーテッド・データベース・システム・カタログ

フェデレーテッド・データベース・システム・カタログには、フェデレーテッド・データベース内のオブジェクトの情報と、データ・ソース側のオブジェクトの情報が入っています。

フェデレーテッド・データベース内のカタログは、フェデレーテッド・システム全体についての情報が含まれているため、グローバル・カタログ と呼びます。 DB2 照会オプティマイザーは、グローバル・カタログ内の情報およびデータ・ソース・ラッパーを使用して、SQL ステートメントを処理する最善の方法を計画します。グローバル・カタログに保管される情報には、リモートとローカルの情報、例えば列名、列のデータ・タイプ、列のデフォルト値、索引情報、および統計情報などが含まれます。

リモート・カタログ情報は、データ・ソースが使用する情報または名前です。ローカル・カタログ情報は、フェデレーテッド・データベースが使用する情報または名前です。例えば、*EMPNO* という名前の列を持つリモート表があるとします。グローバル・カタログには、このリモートの列名が *EMPNO* として保管されます。別の名前を指定しないかぎり、ローカルの列名は *EMPNO* として保管されます。ローカルの列名を *Employee_Number* に変更することができます。この列を含む照会をサブミットするユーザーは、照会の中で *EMPNO* ではなく *Employee_Number* を使用します。データ・ソース列のローカル名を変更するには、ALTER NICKNAME ステートメントを使用します。

リレーショナル・データ・ソースおよび非リレーショナル・データ・ソースの場合、グローバル・カタログに保管される情報にはリモートとローカルの両方の情報が含まれます。

グローバル・カタログに保管されたデータ・ソース表の情報を見るには、フェデレーテッド・データベース内の SYSCAT.TABLES、SYSCAT.NICKNAMES、SYSCAT.TABOPTIONS、SYSCAT.INDEXES、SYSCAT.INDEXOPTIONS、SYSCAT.COLUMNS、および SYSCAT.COLOPTIONS カタログ・ビューを照会してください。

グローバル・カタログには、データ・ソースについてのその他の情報も入っています。例えば、フェデレーテッド・サーバーがデータ・ソースに接続したり、フェデレーテッド・ユーザー許可をデータ・ソースのユーザー権限にマップするために使用する情報が含まれます。グローバル・カタログには、明示的に設定されたデータ・ソースの属性 (サーバー・オプションなど) が入っています。

照会オプティマイザー

SQL コンパイラー処理の一部として、照会オプティマイザー は照会を分析します。コンパイラーは、アクセス・プラン と呼ばれる、照会を処理するための代替ストラテジーを作成します。

アクセス・プランは、照会を次のように処理することを要求します。

- 照会をデータ・ソースが処理する
- 照会をフェデレーテッド・サーバーが処理する
- 照会の一部をデータ・ソースが処理し、一部をフェデレーテッド・サーバーが処理する

照会オプティマイザーは、主にデータ・ソースの能力およびデータに関する情報を基にアクセス・プランを評価します。この情報はラッパーとグローバル・カタログにあります。照会オプティマイザーは照会を照会フラグメントと呼ばれるセグメントに分解します。通常、照会フラグメントをデータ・ソースにプッシュダウンした方が、より効率的です (データ・ソースがフラグメントを処理できる場合)。しかし、照会オプティマイザーは次のような他の要素も考慮します。

- 処理する必要のあるデータの量
- データ・ソースの処理速度
- フラグメントが戻すデータの量
- 通信の帯域幅

- 同じ照会結果を表す、使用可能なマテリアライズ照会表がフェデレーテッド・サーバーにあるかどうか

照会オプティマイザーは、照会フラグメントを処理するための代替アクセス・プランを生成します。フェデレーテッド・サーバーのローカルおよびリモート・データ・ソースで代替プランによって行われる作業量は一定ではありません。照会オプティマイザーはコスト・ベースであるため、リソースの消費コストを代替アクセス・プランに割り当てます。それから、照会オプティマイザーは最小のリソース消費コストで照会を処理するプランを選択します。

何らかのフラグメントをデータ・ソースで処理する場合、フェデレーテッド・データベースはそれらのフラグメントをデータ・ソースにサブミットします。データ・ソースがフラグメントを処理した後、結果が取り出されてフェデレーテッド・データベースに戻されます。フェデレーテッド・データベースが処理の一部を実行する場合、その処理結果とデータ・ソースから取り出した結果が組み合わされます。それから、フェデレーテッド・データベースはすべての結果をクライアントに戻します。

照合シーケンス

データベースでの文字データのソート順序は、データの構造や、データベースで定義されている照合シーケンスによって異なります。

データベース内のデータがすべて大文字で、数値や特殊文字が含まれないとします。データがデータ・ソースでソートされるか、フェデレーテッド・データベースでソートされるかに関係なく、データのソートは同じ出力になるはずです。各データベースで使用される照合シーケンスは、ソート結果に影響を与えません。データベース内のデータがすべて小文字か、すべて数字の場合も同様に、実際にソートが実行される場所に関係なく、データのソートは同じ結果を生成します。

データが次のいずれかの構造で構成されるとします。

- 文字と数字の組み合わせ
- 大文字と小文字の両方
- @ や # などの特殊文字

フェデレーテッド・データベースが使用する照合シーケンスとデータ・ソースが使用する照合シーケンスが異なる場合、このデータをソートしたときの出力結果は異なる可能性があります。

一般に、照合シーケンスという言葉は、特定の文字を別の文字より高い位置にソートするか、低い位置にソートするか、あるいは同じ位置にソートするかを決定する文字データの定義済みの順序付けを指します。

照合シーケンスによるソート順序の決定方法

照合シーケンスは、コード化文字セットの文字のソート順序を決定します。

文字セットは、コンピューター・システムまたはプログラミング言語で使用される文字の集合です。コード化文字セットの文字はそれぞれ、0 から 255 の範囲内の異なる数値 (またはそれに相当する 16 進数) に割り当てられます。その数値はコード・ポイントと呼ばれ、セット内の文字への数値の割り当ては集合的に、コード・ページと呼ばれます。

文字への割り当てに加え、コード・ポイントはソート順序の文字の位置にマップすることができます。専門的に言うと、照合シーケンスは、文字セットのコード・ポイントの、セットの文字のソート順序位置への集合的なマッピングです。文字の位置は数値によって表され、この数値を文字の重みといいます。最も単純な照合シーケンス (ID シーケンスという) での重みは、コード・ポイントと等しくなります。

例: データベース ALPHA は、デフォルト照合シーケンスの EBCDIC コード・ページを使用します。データベース BETA は、デフォルト照合シーケンスの ASCII コード・ページを使用します。以下のように、これら 2 つのデータベースの文字ストリングのソート順序は異なります。

```
SELECT.....
  ORDER BY COL2

EBCDIC-Based Sort           ASCII-Based Sort

COL2                    COL2
```

```
----                        ----
V1G                         7AB
Y2W                         V1G
7AB                         Y2W
```

例: 同様に、データベースの文字比較もそのデータベースで定義される照合シーケンスによって異なります。データベース ALPHA は、デフォルト照合シーケンスの EBCDIC コード・ページを使用します。データベース BETA は、デフォルト照合シーケンスの ASCII コード・ページを使用します。以下のように、これら 2 つのデータベースでの文字比較によって生成される結果は異なります。

```
SELECT.....
  WHERE COL2 > 'TT3'

EBCDIC-Based Results     ASCII-Based Results

COL2                     COL2
----                     ----
TW4                      TW4
X82                      X82
39G
```

照会最適化のためのローカル照合シーケンスの設定

管理者は、データ・ソースの照合シーケンスに一致する特定の照合シーケンスを持つフェデレーテッド・データベースを作成することができます。

それから、各データ・ソース・サーバー定義ごとに COLLATING_SEQUENCE サーバー・オプションを「Y」に設定します。この設定は、フェデレーテッド・データベースに、フェデレーテッド・データベースとデータ・ソースの照合シーケンスが一致していることを知らせます。

フェデレーテッド・データベースの照合シーケンスは、CREATE DATABASE コマンドの一部として設定します。このコマンドを通して、次のいずれかのシーケンスを指定できます。

- 一致シーケンス
- システム・シーケンス (データベースをサポートするオペレーティング・システムが使用するシーケンス)
- カスタマイズ・シーケンス (DB2 が用意する事前定義またはユーザー定義のシーケンス)

データ・ソースが DB2 for z/OS であるとします。 ORDER BY 文節で定義されるソートは、EBCDIC コード・ページに基づく照合シーケンスによってインプリメントされます。 ORDER BY 文節に従ってソートされた DB2 for z/OS データを取得するには、該当する EBCDIC コード・ページに基づく事前定義照合シーケンスを使用するようフェデレーテッド・データベースを構成します。

第 2 章 言語エレメント

© Copyright IBM Corp. 1993, 2010

文字

SQL 言語のキーワードと演算子で使う基本的な記号は、すべての IBM 文字セットの一部である 1 バイト文字です。言語の文字は、英字、数字、または特殊文字に分類されます。

文字 とは、26 個の大文字 (A から Z) および 26 個の小文字 (a から z)、さらに 3 個の文字 ($、#、および @) のいずれかです。これらの特殊文字は、ホスト・データベース製品との互換性を保つために備えられています。例えば、コード・ページ 850 では、$ は X'24'、# は X'23'、@ は X'40' にあります。英字には、拡張文字セットのアルファベット文字も入っています。拡張文字セットには、追加のアルファベット文字が入っています。例えば、発音区別符号 (´ は発音区別符号の一例です) の付いたアルファベット文字です。使用可能な文字は、使用するコード・ページによって異なります。

数字 は、0 から 9 のいずれかの文字です。

特殊文字 は、以下のいずれかの文字です。

文字	説明	文字	説明
	スペースまたはブランク	-	負符号
"	引用符または二重引用符	.	ピリオド
%	パーセント	/	スラッシュ
&	アンパーサンド	:	コロン
'	アポストロフィまたは単一引用符	;	セミコロン
(	左括弧	<	より小さい
)	右括弧	=	等しい
*	アスタリスク	>	より大きい
+	正符号	?	疑問符
,	コンマ	_	下線
|	縦バー[1]	^	脱字記号
!	感嘆符	[	左大括弧
{	左中括弧	]	右大括弧
}	右中括弧	¥	円記号[2]

[1] 縦バー (|) 文字を使用すると、IBM リレーショナル製品どうしのコード・ポータビリティーが妨げられることがあります。 || 演算子の代わりに、CONCAT 演算子を使用してください。

[2] いくつかのコード・ページには、円記号 (¥) 文字に対応するコード・ポイントがありません。 Unicode のストリング定数を入力する場合は、UESCAPE 節を使用して、円記号の代わりに Unicode のエスケープ文字を指定できます。

マルチバイト文字はすべて文字として扱われます。ただし、特殊文字である 2 バイト・ブランク文字は例外です。

トークン

トークンは、SQL の基本構成単位です。*トークン* は、1 つまたは複数の一連の文字です。ブランク文字を使用してもかまわないストリング定数または、区切り ID の場合を除いて、トークンにブランク文字を使用することはできません。

トークンは、通常トークンと区切り文字に分類されます。

- *通常トークン* とは、数値定数、通常 ID、ホスト ID、またはキーワードです。

  例

  ```
  1       .1       +2       SELECT       E       3
  ```

- *区切りトークン* とは、ストリング定数、区切り ID、演算子記号、または構文図に示される特殊文字です。パラメーター・マーカーとして機能する場合は疑問符も区切りトークンです。

  例

  ```
  ,       'string'       "fld1"       =       .
  ```

スペース: スペースは、1 つまたは複数の一連のブランク文字です。ストリング定数と区切り ID 以外のトークンには、スペースを使用することはできません。トークンの後にはスペースを続けることができます。すべての通常トークンの後には、スペースか、または構文で許されているなら区切りトークンを付ける必要があります。

コメント: SQL コメントは、囲んで示している (/* で始まり、*/ で終わる) か、単純 (2 つの連続するハイフンで始まり、行末で終わる) のいずれかです。静的 SQL ステートメントには、ホスト言語のコメントまたは SQL コメントを含めることができます。コメントは、スペースを指定できる場所であればどこにでも指定可能ですが、区切りトークンの中または EXEC と SQL キーワードの間には指定できません。

大文字小文字の区別: どのトークンにも小文字を使用することができますが、通常トークンでの小文字は大文字に変換されます。ただし、ID の大文字と小文字を区別する C 言語のホスト変数は例外です。区切りトークンが大文字に変換されることはありません。したがって、次のステートメントは、

```
select * from EMPLOYEE where lastname = 'Smith';
```

大文字に変換した後は、以下のステートメントと同等になります。

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = 'Smith';
```

マルチバイトの英文字は大文字変換されません。 1 バイト文字 (a から z) は大文字に変換されます。

Unicode 文字の場合:

- UTF-8 での大文字が、UTF-8 での小文字と同じ長さであれば、適用可能な場合、文字は大文字変換されます。たとえば、トルコ語の小文字のドットのない 'i' は、UTF-8 における値が X'C4B1' ですが、ドットのない大文字の 'I' の UTF-8 における値が X'49' なので、変換されません。

- 大文字への変換は、ロケールに配慮しない方法で行われます。たとえば、トルコ語の、ドットのある小文字の 'i' は、英語の (ドットのない) 大文字の 'I' に変換されます。
- 半角のアルファベットも全角のアルファベットも両方大文字に変換されます。たとえば、全角小文字の 'ａ' (U+FF41) は、全角大文字の 'Ａ' (U+FF21) に変換されます。

ID

ID とは、名前の形成に使用されるトークンです。 SQL ステートメントの ID は、SQL ID かホスト ID のいずれかです。

- SQL ID

  SQL ID には、通常 ID と区切り ID の 2 つのタイプがあります。

  – *通常 ID* は、後にゼロ個またはそれ以上の文字が続いている大文字です。各文字は、英大文字、数字、または下線文字です。なお、通常 ID は大文字に変換されることに注意してください。通常 ID は予約語であってはなりません。

    例

    ```
    WKLYSAL     WKLY_SAL
    ```

  – *区切り ID* は、1 文字以上の一連の文字を二重引用符で囲んだものです。区切り ID の中で 1 つの引用符を表す場合には、 2 つの連続した引用符を使用します。この方法で、ID に小文字を使用することができます。

    例

    ```
    "WKLY_SAL"     "WKLY SAL"     "UNION"     "wkly_sal"
    ```

  2 バイトのコード・ページで生成されているのに、使用するのはマルチバイトのコードのアプリケーションやデータベースであるという ID の文字変換の場合、次のような特別な配慮が必要な場合があります。つまり、このような ID の場合、文字変換後に ID の長さ制限を超えてしまう可能性があります。

- ホスト ID

  ホスト *ID* は、ホスト・プログラムで宣言されている名前です。ホスト ID の規則は、ホスト言語の規則に従います。ホスト ID は長さが 255 バイト以下でなければならず、また、 SQL または DB2 で始まっていてはなりません (大文字小文字のどちらでも)。

命名規則と暗黙オブジェクト名の修飾

オブジェクトを命名する規則は、オブジェクト・タイプによって異なります。データベース・オブジェクト名は、1 つの ID で構成されていても、または 2 つの ID で構成されるスキーマ修飾オブジェクトでも構いません。スキーマ修飾オブジェクト名は、スキーマ名なしで指定することができますが、その場合、スキーマ名は暗黙名になります。

動的 SQL ステートメントでは、スキーマ修飾オブジェクト名は、修飾子のないオブジェクト名参照の修飾子として CURRENT SCHEMA 特殊レジスター値を暗黙的に使用します。デフォルトでは、この値は現行の許可 ID に設定されています。バインド、定義、または起動の動作を備えたパッケージ内に動的 SQL ステートメントが入っている場合、 CURRENT SCHEMA 特殊レジスターは修飾には使用されません。バインド動作のパッケージでは、非修飾オブジェクト参照子の暗黙修飾の値として、パッケージのデフォルト修飾子が使用されます。定義動作のパッケージでは、ルーチン定義者の許可 ID が、そのルーチン内の非修飾オブジェクト参照子の暗黙修飾の値として使用されます。起動動作のパッケージでは、ルーチンの起動時に有効になっているステートメント許可 ID が、そのルーチン内の動的 SQL ステ

ートメント内の非修飾オブジェクト参照子の暗黙修飾の値として使用されます。詳しくは、 65 ページの『実行時における動的 SQL の特性』 を参照してください。

静的 SQL ステートメントでは、 QUALIFIER プリコンパイル/BIND オプションにより、非修飾のデータベース・オブジェクト名の修飾子が暗黙指定されます。デフォルトでは、この値はパッケージの許可 ID に設定されています。

以下のオブジェクト名は、SQL プロシージャーのコンテキスト内で使用される場合、それらの名前が区切られていたとしても、通常 ID で許可されている文字しか使用できません。

- condition-name
- label
- parameter-name
- procedure-name
- SQL-variable-name
- statement-name

構文図では、異なる種類の名前には異なる用語を使用しています。以下のリストでそのような用語を定義します。

alias-name
別名を指定するスキーマ修飾名。

attribute-name
構造化データ・タイプの属性を指定する ID。

authorization-name
ユーザー、グループ、またはロールを指定する ID。ユーザーまたはグループの場合:

- 有効な文字は、'A' から 'Z'、'a' から 'z'、'0' から '9'、'#'、'@'、'$'、'_'、'!'; ' '('、')'、'{'、'}'、'-'、'.'、'^' です。
- 以下の文字は、コマンド行プロセッサーを使用して入力する場合、引用符で区切る必要があります: '!'; ' '('、')'、'{'、'}'、'-'、'.'、'^'
- 名前は、文字 'SYS'、'IBM'、または 'SQL' で始めることはできません。
- 名前は、'ADMINS'、'GUESTS'、'LOCAL'、'PUBLIC'、または 'USERS' であってはなりません。
- 区切り許可 ID に小文字を使用することはできません。

bufferpool-name
バッファー・プールを指定する ID。

column-name
表またはビューの列を指定する修飾子付きまたは修飾子のない名前。修飾子は、表名、ビュー名、ニックネーム、または相関名です。

component-name
セキュリティー・ラベル・コンポーネントを指定する ID。

condition-name
SQL プロシージャーの条件を指定する ID。

constraint-name
参照制約、主キー制約、ユニーク制約、または表チェック制約を指定する ID。

correlation-name
結果表を指定する ID。

cursor-name
SQL カーソルを指定する ID。ホストとの互換性を保つため、この名前にハイフン文字を使用することもできます。

data-source-name
データ・ソースを指定する ID。この ID は、3 つの部分に分かれたリモート・オブジェクト名の最初の部分を成します。

db-partition-group-name
データベース・パーティション・グループを指定する ID。

descriptor-name
コロンの後に、SQL 記述子域 (SQLDA) を指定するホスト ID を付けたもの。ホスト ID の詳細は、74 ページの『ホスト変数の参照』を参照してください。記述子名には標識変数は使用されません。

distinct-type-name
特殊タイプを指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のない特殊タイプ名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

event-monitor-name
イベント・モニターを指定する ID。

function-mapping-name
関数マッピングを指定する ID。

function-name
関数を指定する、修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のない関数名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

global-variable-name
グローバル変数を指定する、修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないグローバル変数名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

group-name
構造化タイプ用に定義した変換グループを指定する、修飾子のない ID。

host-variable
ホスト変数を指定するトークンを連結したもの。 74 ページの『ホスト変数の参照』に説明されているように、ホスト変数内には少なくとも 1 つのホスト ID が使用されます。

index-name
索引または SPECIFICATION ONLY 指定の索引を指定するスキーマによって修飾された名前。

label SQL プロシージャーのラベルを指定する ID。

method-name
メソッドを指定する ID。メソッドのスキーマ・コンテキストは、そのメソッドのサブジェクト・タイプ (または、サブジェクト・タイプのスーパータイプ) のスキーマによって決まります。

nickname
フェデレーテッド・サーバーが表またはビューに参照することを指定する、スキーマによって修飾された名前。

package-name
パッケージを指定するスキーマ修飾名。空ストリングではないパッケージ ID がパッケージに付いている場合、 schema-id.package-id.version-id のフォームのバージョン ID もそのパッケージ名の末尾に付いている必要があります。

parameter-name
プロシージャー、ユーザー定義関数、メソッド、または索引拡張機能で参照できるパラメーターを指定する ID。

partition-name
パーティション表内のデータ・パーティションを指定する ID。

procedure-name
プロシージャーを指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないプロシージャー名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

remote-authorization-name
データ・ソースのユーザーを指定する ID。許可名に関する規則は、データ・ソースごとに異なります。

remote-function-name
データ・ソース・データベースに登録されている関数を指定する名前。

remote-object-name
データ・ソース表またはビューを指定するとともに、その表またはビューが置かれているデータ・ソースを識別する 3 つの部分に分かれた名前。この名前は、データ・ソース名、リモート・スキーマ名、およびリモート表名の各部分で構成されます。

remote-schema-name
データ・ソース表またはビューが属するスキーマを指定する名前。この名前は、3 つの部分に分かれたリモート・オブジェクト名の 2 番目の部分を成します。

remote-table-name
データ・ソースにある表またはビューを指定する名前。この名前は、3 つの部分に分かれたリモート・オブジェクト名の 3 番目の部分を成します。

remote-type-name
データ・ソース・データベースがサポートするデータ・タイプ。組み込まれたタイプの場合は、長形式を使用しないでください (たとえば、CHARACTER ではなく CHAR を使用してください)。

savepoint-name
セーブポイントを指定する ID。

schema-name
SQL オブジェクトを論理的にグループ化するための ID。オブジェクト名の修飾子として使用されるスキーマ名は、以下のものから暗黙的に決定されます。

- CURRENT SCHEMA 特殊レジスターの値
- QUALIFIER プリコンパイル/BIND オプションの値
- CURRENT PATH 特殊レジスターを使用する解決アルゴリズムに基づいて
- 同一の SQL ステートメントにある別のオブジェクトのスキーマ名に基づいて

混乱を避けるために、スキーマとして名前 SESSION を使わないようお勧めします。ただし、宣言済みのグローバル一時表のスキーマは除きます (この場合は、スキーマ名 SESSION を使用する必要があります)。

security-label-name
セキュリティー・ラベルを指定する、修飾子付きまたは修飾子のない名前。SQL ステートメントにおける修飾子のないセキュリティー・ラベル名は、該当する security-policy-name がある場合、それによって暗黙のうちに修飾されます。暗黙的に適用できる security-policy-name が存在しない場合、この名前は修飾されている必要があります。

security-policy-name
セキュリティー・ポリシーを指定する ID。

sequence-name
シーケンスを指定する ID。

server-name
アプリケーション・サーバーを指定する ID。フェデレーテッド・システムでは、サーバー名によってデータ・ソースのローカル名も指定します。

specific-name
ユニーク名を指定する、修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないユニーク名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

SQL-variable-name
SQL プロシージャー・ステートメントのローカル変数名。 SQL 変数名は、ホスト変数名が許可されている他の SQL ステートメントで使うこともできます。この名前に対しては、SQL 変数を宣言したコンパウンド・ステートメントのラベルで修飾できます。

statement-name
作成済み SQL ステートメントを指定する ID。

supertype-name
タイプのスーパータイプを指定する修飾子付きまたは修飾子のない名前。SQL ステートメントにおける修飾子のないスーパータイプ名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

table-name
表を指定するスキーマによって修飾された名前。

tablespace-name
表スペースを指定する ID。

trigger-name
トリガーを指定するスキーマによって修飾された名前。

type-mapping-name
データ・タイプ・マッピングを指定する ID。

type-name
型を指定する修飾子付きまたは修飾子のない名前。 SQL ステートメントにおける修飾子のないタイプ名は、コンテキストに応じてデータベース・マネージャーによって暗黙のうちに修飾されます。

typed-table-name
タイプ表を指定するスキーマによって修飾された名前。

typed-view-name
タイプ・ビューを指定するスキーマによって修飾された名前。

view-name
ビューを指定するスキーマによって修飾された名前。

wrapper-name
ラッパーを指定する ID。

XML-schema-name
XML スキーマを指定する、修飾子付きまたは修飾子のない名前。

xsrobject-name
XML スキーマ・リポジトリー内のオブジェクトを指定する、修飾子付きまたは修飾子のない名前。

別名

表の別名は、表またはビューの代替名と見なすことができます。このため、SQL ステートメントの中で表またはビューは、名前か表別名のどちらかで参照できます。

別名は、表名またはビュー名を使用できるロケーションであればどこででも使用できます。別名は、オブジェクトが存在していなくても作成することができます (ただし、オブジェクトを参照するステートメントがコンパイルされる時点では存在している必要があります)。別名チェーンの中に循環参照または反復参照がない限り、他の別名を別名によって参照することができます。別名で参照できるのは、同じデータベース内の表、ビュー、別名だけです。 CREATE TABLE や CREATE VIEW ステートメントのような、新しい表名またはビュー名が指定されるのが当然のロケーションでは別名を使用できません。たとえば、 PERSONNEL という別名を作成した後で CREATE TABLE PERSONNEL... のような使い方をするとエラーが戻されます。

構文図や SQL ステートメントの説明では、別名によって表またはビューを参照するというオプションは明示的には示されません。

修飾子なしの新しい別名に、既存の表、ビュー、または別名と同じ完全修飾名を付けることはできません。

SQL ステートメントで別名を使用する効果は、テキスト置換の効果に似ています。別名は SQL ステートメントのコンパイル時には定義されている必要があり、ステートメントのコンパイル時には修飾子付きの基本表名またはビュー名に置き換えられます。例えば、PBIRD.SALES が DSPN014.DIST4_SALES_148 の別名である場合に、コンパイル時には、

```
SELECT * FROM PBIRD.SALES
```

は、実際には次のようになります。

```
SELECT * FROM DSPN014.DIST4_SALES_148
```

フェデレーテッド・システムでは、上記のような使用法と制限は、表の別名だけでなく、ニックネームが表す別名にも適用されます。 したがって、ニックネームの代わりにニックネームの別名を SQL ステートメントで使用することも可能です。別名を参照するステートメントをコンパイルする前にニックネームを作成する場合、まだ存在していないニックネームに別名をあらかじめ作成しておくこともできます。あるニックネームの別名に、そのニックネームの他の別名を参照させることもできます。

他のリレーショナル・データベース管理システムのもとで実行するアプリケーションの構文を使えるようにするために、 CREATE ALIAS ステートメントと DROP ALIAS ステートメントにおいては、 ALIAS の代わりに SYNONYM を使用できるようになっています。

許可 ID と許可名

許可 ID とは、データベース・マネージャーとアプリケーション・プロセスとの間、またはデータベース・マネージャーとプログラム準備処理との間の接続が確立されるときに、データベース・マネージャーが獲得する文字ストリングのことです。これは、特権の集合を指定するものです。ユーザーやユーザー・グループを指す場合もありますが、この特性はデータベース・マネージャーからは制御されません。

許可 ID は、データベース・マネージャーにより以下の目的で使用されます。

- SQL ステートメントの許可検査
- QUALIFIER プリコンパイル/BIND オプションと CURRENT SCHEMA 特殊レジスターのデフォルト値。許可 ID は、デフォルトの CURRENT PATH 特殊レジスターと FUNCPATH プリコンパイル/BIND オプションにも入っています。

許可 ID はすべての SQL ステートメントに適用されます。静的 SQL ステートメントに適用される許可 ID は、プログラムのバインディングの過程で使用される許可 ID です。動的 SQL ステートメントに適用される許可 ID は、次のように、バインド時に指定した DYNAMICRULES オプションによってと、その動的 SQL ステートメントを発行したパッケージの現在のランタイム環境によって決まります。

- バインド動作をもつパッケージの場合に使用される許可 ID は、パッケージ所有者の許可 ID になります。
- 定義動作をもつパッケージの場合に使用される許可 ID は、それに対応するルーチンの定義者の許可 ID になります。
- 実行動作をもつパッケージの場合に使用される許可 ID は、パッケージを実行するユーザーの許可 ID になります。

- 起動動作をもつパッケージの場合に使用される許可 ID は、ルーチンの起動の時点で有効になっている許可 ID になります。これはランタイム許可 ID と呼ばれます。

詳しくは、 65 ページの『実行時における動的 SQL の特性』 を参照してください。

SQL ステートメントで指定される*許可名* を、そのステートメントの許可 ID と混同してはなりません。許可名は、種々の SQL ステートメントで使用される ID です。 許可名は、スキーマの所有者を指定するために CREATE SCHEMA ステートメントで使用されます。許可名は、付与または取り消しの操作の対象を指定するために GRANT および REVOKE ステートメントで使用されます。 *X* に特権を付与すると、それ以降、その特権を必要とするステートメントでは、 *X* (またはグループ *X* あるいはロール *X* のメンバー) が許可 ID になるということです。

例:

- ユーザー ID が SMITH であり、またデータベース・マネージャーがアプリケーション・プロセスとの接続を確立したときに獲得した許可 ID も SMITH であるとします。以下のステートメントは対話式に実行されます。

  ```
  GRANT SELECT ON TDEPT TO KEENE
  ```

  SMITH はこのステートメントの許可 ID です。したがって、動的 SQL ステートメントでの CURRENT SCHEMA 特殊レジスターのデフォルト値は SMITH になり、静的 SQL でのデフォルトの QUALIFIER プリコンパイル/BIND オプションも SMITH になります。ステートメントを実行できる権限は、SMITH につき合わせて検査され、SMITH が、 57 ページの『命名規則と暗黙オブジェクト名の修飾』で説明されている修飾規則に基づく *table-name* 暗黙修飾子となります。

  KEENE はこのステートメントで指定された許可名です。 KEENE には SMITH.TDEPT に対する SELECT 特権が付与されます。
- SMITH が管理権限を持っており、セッション中に SET SCHEMA ステートメントが発行されない、以下の動的 SQL ステートメントの許可 ID であるとします。

  ```
  DROP TABLE TDEPT
  ```

  これは、SMITH.TDEPT 表を削除します。

  ```
  DROP TABLE SMITH.TDEPT
  ```

  これは、SMITH.TDEPT 表を削除します。

  ```
  DROP TABLE KEENE.TDEPT
  ```

  これは、KEENE.TDEPT 表を削除します。 KEENE.TDEPT と SMITH.TDEPT は別の表であることに注意してください。

  ```
  CREATE SCHEMA PAYROLL AUTHORIZATION KEENE
  ```

  KEENE は、PAYROLL と呼ばれるスキーマを作成するステートメントで指定されている許可名です。 KEENE は、スキーマ PAYROLL の所有者であり、CREATEIN、 ALTERIN、および DROPIN 特権が与えられ、このような特権を他のユーザーに付与することができます。

実行時における動的 SQL の特性

BIND オプション DYNAMICRULES によって、動的 SQL ステートメントの処理時の許可検査に使用される許可 ID が決まります。さらにこのオプションは、修飾されるオブジェクト参照子に使用される暗黙修飾子などの他の動的 SQL 属性や、特定の SQL ステートメントを動的に呼び出せるかどうかも制御します。

許可 ID とその他の動的 SQL 属性の一連の値を動的 SQL ステートメント動作と呼びます。使用可能な 4 つの動作は、実行、バインド、定義、および起動です。以下の表で明らかなとおり、 DYNAMICRULES BIND オプションの値とランタイム環境の組み合わせで、使用する動作が決まります。実行動作を意味する DYNAMICRULES RUN がデフォルトです。

表 4. DYNAMICRULES とランタイム環境による動的 SQL ステートメントの動作の決定

	動的 SQL ステートメントの動作	
DYNAMICRULES 値	**スタンドアロン・プログラム環境**	**ルーチン環境**
BIND	バインド動作	バインド動作
RUN	実行動作	実行動作
DEFINEBIND	バインド動作	定義動作
DEFINERUN	実行動作	定義動作
INVOKEBIND	バインド動作	起動動作
INVOKERUN	実行動作	起動動作

実行動作

DB2 では、動的 SQL ステートメントの許可検査に使われる値と、動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾に使用される値用の初期値に使われる値として、パッケージを実行するユーザーの許可 ID (最初に DB2 に接続した ID) が使用されます。

バインド動作

実行時には DB2 では、静的 SQL に対して適用されるすべての規則が許可と修飾に対して適用されます。その場合、動的 SQL ステートメントの許可検査に使われる値と、動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾用のパッケージ・デフォルト修飾子として、パッケージ所有者の許可 ID が使用されます。

定義動作

定義動作が適用されるのは、ルーチン・コンテキストで実行されるパッケージ内に動的 SQL ステートメントがあって、しかもそのパッケージが DYNAMICRULES DEFINEBIND または DYNAMICRULES DEFINERUN でバインドされていた場合のみです。その場合、動的 SQL ステートメントの許可検査に使われる値と、ルーチン内の動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾に使われる値として、ルーチン定義者 (ルーチンのパッケージ・バインド・プログラムではなく) の許可 ID が DB2 で使用されます。

起動動作

起動動作が適用されるのは、ルーチン・コンテキストで実行されるパッケージ内に動的 SQL ステートメントがあって、しかもそのパッケージが

DYNAMICRULES INVOKEBIND または DYNAMICRULES INVOKERUN でバインドされていた場合のみです。動的 SQL の許可検査に使われる値と、ルーチン内の動的 SQL ステートメント内の非修飾オブジェクト参照子の暗黙修飾に使われる値として、ルーチンの起動時点で有効なステートメント許可 ID が DB2 で使用されます。これを以下の表に要約してあります。

起動環境	使用 ID
任意の静的 SQL	ルーチンを呼び出す SQL が所属するパッケージの所有者の暗黙または明示的な値。
ビューまたはトリガーの定義での使用	ビューまたはトリガーの定義者。
バインド動作パッケージに属する動的 SQL	ルーチンを呼び出す SQL が所属するパッケージの所有者の暗黙または明示的な値。
実行動作パッケージの動的 SQL	DB2 への初期接続を確立するのに使用する ID。
定義動作パッケージの動的 SQL	ルーチンを呼び出す SQL が所属するパッケージを使用するルーチンの定義者。
起動動作パッケージの動的 SQL	ルーチンを呼び出す現在の許可 ID

実行動作が適用されない場合の制限付きステートメント

バインド、定義、または起動の動作が有効になっている場合、動的 SQL ステートメント
GRANT、REVOKE、ALTER、CREATE、DROP、COMMENT、RENAME、SET INTEGRITY、SET EVENT MONITOR STATE と、ニックネームを参照する照会は使用できません。

DYNAMICRULES オプションに関する考慮事項

バインド、定義、または起動の動作パッケージから実行される動的 SQL ステートメント内の非修飾オブジェクト参照子を修飾するのに、 CURRENT SCHEMA 特殊レジスターを使用することはできません。これは、 CURRENT SCHEMA 特殊レジスターを変更するために SET CURRENT SCHEMA ステートメントを発行した後にもあてはまります。レジスター値は変更されますが、使用されることはありません。

単一の接続中に複数のパッケージが参照されると、それらのパッケージで準備されたすべての動的 SQL ステートメントは、個々のパッケージとその使用場所である環境用に DYNAMICRULES オプションで指定されている行動をとります。

パッケージがバインド動作をとるときは、パッケージのバインド・プログラムには、そのパッケージのユーザーには付与されるべきでない許可が付与されないように気を付けることが大切です。動的ステートメントは、パッケージ所有者の許可 ID を使用するからです。同様に、パッケージが定義動作をとるときは、ルーチンの定義者には、パッケージのユーザーには付与されるべきでない許可が付与されてはなりません。

許可 ID とステートメントの準備

BIND 時に VALIDATE BIND を指定する場合、バインド時に、表やビューを扱うときに必要な特権も存在していなければなりません。そのような特権や参照オブジェクトが存在しない場合に、SQLERROR NOPACKAGE が有効になっていると、バインド操作は失敗します。 SQLERROR CONTINUE オプションが指定されていると、バインド操作は正常に完了し、エラーを生じたステートメントにはすべてフラグが付けられます。そのようなステートメントを実行しようとすると、エラーが生じます。

VALIDATE RUN オプションでパッケージがバインドされると、通常のバインド処理はすべて完了しますが、アプリケーションで参照される表やビューを使うのに必要な特権は、この時点では存在していなくてもかまいません。必要な特権がバインドの実行時に存在しない場合に、アプリケーション内でステートメントを初めて実行すると、必ず増分バインドが実行されます。このときには、ステートメントに必要なすべての特権が存在していなければなりません。必要な特権が存在しない場合、ステートメントの実行は失敗します。

実行時の許可検査は、パッケージ所有者の許可 ID を使って行われます。

列名

列名 の意味はコンテキストによって異なります。 列名は以下の目的に使用できます。

- 列の名前を宣言する (CREATE TABLE ステートメントなどで)。
- 列を識別する (CREATE INDEX ステートメントなどで)。
- 列の値を指定する (以下に示すようなコンテキストで)。
  - 列関数においては、列名によって、その関数が適用されるグループまたは中間結果表の列のすべての値が指定されます。たとえば、MAX(SALARY) は、あるグループの SALARY 列の中のすべての値に関数 MAX が適用されることを表します。
  - GROUP BY または ORDER BY 節の中では、その節が適用される中間結果表の中のすべての値を、列名によって指定します。たとえば、ORDER BY DEPT と指定すると、 DEPT 列の値によって中間結果表が順序付けられます。
  - 式、検索条件、またはスカラー関数においては、列名によって、その構成項目が適用されるそれぞれの行またはグループの値が指定されます。たとえば、検索条件 CODE = 20 が何らかの行に適用される場合、列名 CODE によって指定される値は、その行の CODE 列の値です。
- FROM 節における *table-reference* の *correlation-clause* のように、列の名前を一時的に変更する。

修飾列名

列名の修飾子としては、表名、ビュー名、ニックネーム、別名、または相関名が使用されます。

列名が修飾されるかどうかは、コンテキストによって異なります。

- COMMENT ON ステートメントの形式に応じ、単一の列名に修飾の必要な場合があります。複数の列名は修飾してはなりません。

- 列名が列の値を指定している場合、修飾することができます。
- UPDATE ステートメントの割り当て節では、ユーザーのオプションで修飾することができます。
- 上記以外のコンテキストでは、列名を修飾してはなりません。

修飾子がオプションである場合、修飾子には 2 つの目的があります。これについては、70 ページの『あいまいさを避けるための列名修飾子』および 72 ページの『相関参照内の列名修飾子』で説明されています。

相関名

相関名 は、照会の FROM 節、および UPDATE または DELETE ステートメントの第 1 節で指定されます。たとえば、FROM X.MYTABLE Z という節では、Z が X.MYTABLE の相関名として確立されます。

```
FROM X.MYTABLE Z
```

X.MYTABLE の相関名として Z が定義されると、その SELECT ステートメントにおいては Z だけが X.MYTABLE インスタンスの列を参照する修飾子として使用できます。

相関名は、それが定義されているコンテキスト内でのみ、表、ビュー、ニックネーム、別名、ネストされた表の式、表関数、または表参照に関連付けられます。したがって、別の目的に使用するために、異なるステートメントの中や同じステートメントの異なる節の中で同じ相関名を定義できます。

修飾子としての相関名は、あいまいさの回避や、相関参照の確立に利用できます。また、表参照の単なる短縮名として利用することもできます。例では、X.MYTABLE と何度も入力するのを避けるためだけに使用されています。

相関名を表名、ビュー名、ニックネーム、または別名に対して指定する場合、その表、ビュー、ニックネーム、または別名のそのインスタンスでの列への修飾子付き参照は、その表名、ビュー名、ニックネーム、別名ではなく、相関名を使用しなければなりません。たとえば、以下の例の EMPLOYEE.PROJECT への参照は、EMPLOYEE に対する相関名がすでに指定されているため誤りです。

例

```
FROM EMPLOYEE E
  WHERE EMPLOYEE.PROJECT='ABC'      * incorrect*
```

PROJECT に対する修飾子付き参照では、以下のように、相関名「E」を代わりに使用する必要があります。

```
FROM EMPLOYEE E
  WHERE E.PROJECT='ABC'
```

FROM 節で指定する名前は、*直接的* か*間接的* のどちらかです。相関名が指定されていない場合の表名、ビュー名、ニックネーム、または別名は FROM 節の直接的な名前です。相関名は常に直接的な名前です。たとえば、以下の FROM 節では、EMPLOYEE には相関名が指定され、 DEPARTMENT には指定されていません。このため、DEPARTMENT は直接的な名前、EMPLOYEE は間接的な名前になります。

```
FROM EMPLOYEE E, DEPARTMENT
```

FROM 節の直接的な表名、ビュー名、ニックネーム、または別名は、その FROM 節での直接的なその他の表名、ビュー名、ニックネーム、または FROM 節の相関名のどれかと同じになる場合があります。これにより列名参照があいまいとなり、エラー (SQLSTATE 42702) となる可能性があります。

以下に示す最初の 2 つの FROM 節は、直接的な名前である EMPLOYEE をそれぞれ 2 回以上参照するような参照を行っていないので正しい FROM 節です。

1. 次の FROM 節が与えられているものとします。

   ```
   FROM EMPLOYEE E1, EMPLOYEE
   ```

   EMPLOYEE.PROJECT のような修飾子付き参照は、 FROM 節での EMPLOYEE の 2 番目のインスタンスの列を指すことになります。 EMPLOYEE の 1 番目のインスタンスに対する修飾子付き参照では、相関名 E1 を使用する (E1.PROJECT) 必要があります。
2. 次の FROM 節が与えられているものとします。

   ```
   FROM EMPLOYEE, EMPLOYEE E2
   ```

   EMPLOYEE.PROJECT のような修飾子付き参照は、 FROM 節での EMPLOYEE の 1 番目のインスタンスの列を指すことになります。 EMPLOYEE の 2 番目のインスタンスに対する修飾子付き参照では、相関名 E2 を使用する (E2.PROJECT) 必要があります。
3. 次の FROM 節が与えられているものとします。

   ```
   FROM EMPLOYEE, EMPLOYEE
   ```

   この節では、直接的な 2 つの表名 (EMPLOYEE と EMPLOYEE) が同じになっています。これ自体は可能ですが、特定の列名への参照があいまいになってしまいます (SQLSTATE 42702)。
4. 次のステートメントが与えられているものとします。

   ```
   SELECT *
     FROM EMPLOYEE E1, EMPLOYEE E2            * incorrect *
     WHERE EMPLOYEE.PROJECT = 'ABC'
   ```

   修飾子付き参照 EMPLOYEE.PROJECT は誤りです。これは、FROM 節の EMPLOYEE の 2 つのインスタンスの両方に相関名があるためです。そうするのではなく、PROJECT を参照するときは、どちらかの相関名 (E1.PROJECT または E2.PROJECT) で修飾する必要があります。
5. 次の FROM 節が与えられているものとします。

   ```
   FROM EMPLOYEE, X.EMPLOYEE
   ```

   EMPLOYEE の 2 番目のインスタンスの列を参照するときは、 X.EMPLOYEE を使用する (X.EMPLOYEE.PROJECT) 必要があります。 X が、動的 SQL では CURRENT SCHEMA 特殊レジスター値、静的 SQL では QUALIFIER プリコンパイル/BIND オプションである場合、そのような参照はあいまいなので列を参照することはできません。

FROM 節で相関名を使用することにより、結果表の列に関連付けられる列名のリストを指定することもできます。相関名の場合と同じように、このようにリストされ

た列名は、照会時に列の参照に使用する必要がある列の*直接的な* 名前になります。列名のリストを指定する場合、基礎表の列名は*間接的な名前* になります。

次の FROM 節が与えられているものとします。

```
FROM DEPARTMENT D (NUM,NAME,MGR,ANUM,LOC)
```

D.NUM などの修飾子の付いた参照は、 DEPTNO として表に定義されている DEPARTMENT 表の最初の列を表します。この FROM 節を使用した D.DEPTNO の参照は、列名 DEPTNO が間接的な列名であるため誤りです。

あいまいさを避けるための列名修飾子

関数、GROUP BY 節、ORDER BY 節、式、または検索条件のコンテキストでは、列名は、何らかの表、ビュー、ニックネーム、ネストされた表の式あるいは表関数の列の値を指します。列を収容する可能性のある表、ビュー、ニックネーム、ネストされた表の式および表関数は、そのコンテキストのオブジェクト表 と呼ばれます。複数の表が同じ名前の列を備えている場合があります。列名を修飾する理由の 1 つは、列がどの表のものかを指定することにあります。列名の修飾子は、SQL プロシージャーにおいて、列名と SQL ステートメントで使われる SQL 変数名を区別するときにも役立ちます。

ネストされた表の式または表関数は、 FROM 節で先行する *table-references* をオブジェクト表と見なします。後続の *table-references* はオブジェクト表とは見なされません。

表指定子

特定のオブジェクト表を指定する修飾子は、*表指定子* と呼ばれます。オブジェクト表を指定する節では、そのオブジェクト表に対する表指定子も設定します。以下の例は、SELECT 節の式のオブジェクト表を、直後の FROM 節で指定しています。

```
SELECT CORZ.COLA, OWNY.MYTABLE.COLA
  FROM OWNX.MYTABLE CORZ, OWNY.MYTABLE
```

FROM 節の表指定子は次のように設定されます。

- 表、ビュー、ニックネーム、別名、ネストされた表の式または表関数の後に続く名前は、相関名でもあり表指定子でもあります。したがって、CORZ は表指定子です。選択リストの中で、最初の列名を修飾するために CORZ が使用されています。
- 直接的な表名、ビュー名、ニックネーム、または別名は、表指定子です。したがって、OWNY.MYTABLE は表指定子です。選択リストの中で、第 2 の列名を修飾するために OWNY.MYTABLE が使用されています。

列を直接的な表名形式の表指定子で修飾するとき、直接的な表名は修飾の付いた形式でも付かない形式でも使用できます。修飾の付いた形式を使用する場合、修飾子は直接的な表名のデフォルト修飾子と同じでなければなりません。

例えば、現行スキーマが CORPDATA であると仮定します。その場合、

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT FROM EMPLOYEE
```

は有効です。 FROM 節で参照される EMPLOYEE 表は CORPDATA.EMPLOYEE を完全に修飾するためです。これは WORKDEPT 列の修飾子と一致します。さらに、

```
SELECT EMPLOYEE.WORKDEPT, REGEMP.WORKDEPT
  FROM CORPDATA.EMPLOYEE, REGION.EMPLOYEE REGEMP
```

も有効です。最初の選択リスト列は非修飾の直接的な表指定子 CORPDATA.EMPLOYEE (FROM 節にある) を参照し、2 番目の選択リスト列は、表オブジェクト REGION.EMPLOYEE (これも FROM 節にある) の相関名 REGEMP を参照するためです。

次に、現行スキーマが REGION であると仮定します。その場合、

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT FROM EMPLOYEE
```

は無効です。FROM 節で参照される EMPLOYEE 表は REGION.EMPLOYEE を完全に修飾し、WORKDEPT 列の修飾子は CORPDATA.EMPLOYEE 表を表すためです。

列へのあいまいな参照が生じる可能性を避けるため、各表指定子は、特定の FROM 節の中ではユニークでなければなりません。

未定義またはあいまいな参照の回避

列名が列の値を参照する場合、その名前の付いた列がただ 1 つのオブジェクト表の中になければなりません。以下の状態はエラーと見なされます。

- 指定された名前の列の入ったオブジェクト表がない。この参照は未定義になります。
- 列名が表指定子によって修飾されているが、指定された表に指定された名前の列が入っていない。この参照も未定義になります。
- 名前が修飾なしで、2 つ以上のオブジェクト表の中にその名前の列がある。この参照はあいまいです。
- 列名が表指定子で修飾されているが、その指定されている表が FROM 節の中でユニークでなく、指定されている表のどちらのオカレンスにもその列がある。この参照はあいまいです。
- 列名は、TABLE キーワードが先行しないネストされた表の式、もしくは右外部結合または全外部結合の右側のオペランドである表関数またはネストされた表の式にある。列名は、ネストされた表の式の全選択内の *table-reference* の列を指しません。この参照は未定義になります。

ユニークに定義された表指定子で列名を修飾することによって、あいまいな参照を避けてください。列が名前の異なる複数のオブジェクト表の中に入っている場合、その表名を指定子として使用することができます。また、相関名の後に続いて列名のリストを使用してオブジェクト表のいずれかの列にユニーク名を指定することによって、表指定子を使用しなくてもあいまいな参照を避けることができます。

列を直接的な表名形式の表指定子で修飾するとき、直接的な表名は修飾の付いた形式でも付かない形式でも使用できます。しかし、表名、ビュー名、またはニックネームと、表指定子を完全に修飾した後は、使用される修飾子と表が同じものでなければなりません。

1. たとえば、ステートメントの許可 ID が CORPDATA とすると、以下のステートメントは有効です。

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
  FROM EMPLOYEE
```

   は有効なステートメントです。

2. ステートメントの許可 ID が REGION の場合、以下は無効です。

```
SELECT CORPDATA.EMPLOYEE.WORKDEPT
  FROM EMPLOYEE                          * incorrect *
```

   これは、EMPLOYEE が表 REGION.EMPLOYEE を表しているのに対し、WORKDEPT の修飾子が別の表 CORPDATA.EMPLOYEE を表しているためです。

相関参照内の列名修飾子

全選択 とは、種々の SQL ステートメントのコンポーネントとして使用される照会の 1 つの形式です。任意のステートメントの検索条件で使用される全選択は、*副照会* と呼ばれます。ステートメントで式として単一値を検索するのに使用される全選択は、*スカラー全選択* または*スカラー副照会* と呼ばれます。照会の FROM 節で使用される全選択は、*ネストされた表の式* と呼びます。検索条件、スカラー副照会、およびネストされた表の式の副照会を、このトピックのこれ以降の部分では副照会と呼びます。

副照会にはそれ自身の副照会を収めることができます。その副照会の中に、さらに副照会が収められていてもかまいません。したがって、SQL ステートメントに副照会の階層が入ることになる場合があります。副照会を収容する階層のエレメントは、そこに収容された副照会よりも高いレベルとされます。

階層のあらゆるエレメントには、1 つ以上の表指定子が入っています。副照会は、階層中の自分のレベルで指定されている表の列だけでなく、階層中のそれより前のレベルで指定されている表の列から階層の最上位で識別される表の列まで参照できます。上位のレベルで指定される表の列への参照は、*相関参照* と呼ばれます。

既存の SQL 標準規格との互換性のため、修飾子付きと修飾子なしのどちらの列名も相関参照として認められています。ただし、副照会で使用されるすべての列参照を修飾することをお勧めします。そうしないと、同一の列名により予期しない結果が生じることがあります。たとえば、ある階層の表が相関参照として同じ列名を収めるように変更され、ステートメントが再度作成処理された場合、新たな参照は変更された表に対して適用されます。

副照会内の列名が修飾されているときは、修飾されているその列名が出現するのと同じ副照会から探索が始まり、修飾子に一致する表指定子が見つかるまで、階層の上位へ向かって階層の各レベルの探索が続けられます。該当するものが見つかると、その表に指定の列があるかどうかが調べられます。列名を収容しているレベルより高いレベルで表が見つかった場合、これは表指定子が見つかったレベルに対する相関参照となります。ネストされた表の式の全選択より上の階層を探索するためには、ネストされた表の式の前にオプションの TABLE キーワードを指定しなければなりません。

副照会に収容されている列名が修飾されていないときは、その列名が出現するのと同じ副照会から始めて、階層の各レベルで参照されている表が探索され、一致する列名が見つかるまで、階層の上位へ向かって探索が続けられます。列名を収容しているレベルより高いレベルの表で列が見つかった場合は、その列を収めた表が見つかったレベルに対する相関参照となります。列名が、特定のレベルの 2 つ以上の表で見つかった場合は、参照はあいまいになり、エラーと見なされます。

以下の例の T は、どの場合も、列 C の入った表指定子を参照しています。列名 T.C は、以下の条件がすべて満たされているときにのみ相関参照となります (この T は暗黙の修飾子か明示的な修飾子のいずれかを表します)。

- T.C は副照会の式で使用される。
- T が、その副照会の from 節で使用されている表を指していない。
- T が、副照会を収容している上位の階層レベルで使用されている表を示している。

同じ表、ビュー、またはニックネームが、多くのレベルで指定されていることがあるため、表指定子としてはユニークな相関名を使用するようお勧めします。 T が 2 つ以上のレベルで表の指定に使用される場合 (T は表名自体か重複の相関名)、 T.C は、T.C の入った副照会を最も直接的に収容するように T が使用されているレベルを参照することになります。上位レベルへの相関が必要な場合、ユニークな相関名を使用する必要があります。

相関参照 T.C は、2 つの検索条件が、検索条件 1 が副照会で、検索条件 2 が上位のレベルでそれぞれ適用されている T の行またはグループでの C の値を識別します。条件 2 が WHERE 節で使用される場合、副照会は条件 2 が適用される行ごとに評価されます。条件 2 が HAVING 節で使用される場合、副照会は条件 2 が適用されるグループごとに評価されます。

たとえば、次のステートメントにおいて、(最後の行の) 相関参照 X.WORKDEPT は、最初の FROM 節のレベルにある表 EMPLOYEE の WORKDEPT の値を指します。 (この節は X を EMPLOYEE の相関名として設定します。) このステートメントは、その部署の平均給与を下回る社員のリストを作成するものです。

```
SELECT EMPNO, LASTNAME, WORKDEPT
  FROM EMPLOYEE X
  WHERE SALARY < (SELECT AVG(SALARY)
                  FROM EMPLOYEE
                  WHERE WORKDEPT = X.WORKDEPT)
```

次の例は、THIS を相関名として使用しています。このステートメントは、社員のいない部門の行を削除します。

```
DELETE FROM DEPARTMENT THIS
   WHERE NOT EXISTS(SELECT *
                    FROM EMPLOYEE
                    WHERE WORKDEPT = THIS.DEPTNO)
```

変数の参照

SQL ステートメントの変数 は、SQL ステートメントの実行時に変更が可能な値を指定します。 SQL ステートメントで使用される変数にはさまざまな種類があります。

ホスト変数
ホスト変数は、ホスト言語のステートメントによって定義されます。ホスト変数を参照する方法については、『ホスト変数の参照』を参照してください。

遷移変数
遷移変数はトリガーで定義され、列の古い値または新しい値のいずれかを参照します。遷移変数を参照する方法については、*SQL リファレンス 第 2 巻*の『CREATE TRIGGER ステートメント』を参照してください。

SQL 変数
SQL 変数は、SQL 関数、SQL メソッド、SQL プロシージャー、トリガーの SQL コンパウンド・ステートメント、または動的 SQL ステートメントで定義されます。 SQL 変数について詳しくは、*SQL リファレンス 第 2 巻*の『SQL パラメーター、SQL 変数、およびグローバル変数の参照』を参照してください。

グローバル変数
グローバル変数は CREATE VARIABLE ステートメントによって定義されます。グローバル変数について詳しくは、『CREATE VARIABLE』および*SQL リファレンス 第 2 巻*の『SQL パラメーター、SQL 変数、およびグローバル変数の参照』を参照してください。

SQL パラメーター
SQL パラメーターは、CREATE FUNCTION、CREATE METHOD、または CREATE PROCEDURE ステートメントによって定義されます。 SQL パラメーターについて詳しくは、*SQL リファレンス 第 2 巻*の『SQL パラメーター、SQL 変数、およびグローバル変数の参照』を参照してください。

パラメーター・マーカー
パラメーター・マーカーは動的 SQL ステートメントで指定されます。ステートメントが静的 SQL ステートメントであれば本来、これに代わってホスト変数が指定されます。動的 SQL ステートメントの処理中に値をパラメーター・マーカーと関連付けるために、 SQL 記述子またはパラメーター・バインディングが使用されます。パラメーター・マーカーについて詳しくは、パラメーター・マーカーを参照してください。

ホスト変数の参照

ホスト変数 とは、以下のいずれかです。

- C の変数、C++ の変数、COBOL のデータ項目、FORTRAN の変数、または Java の変数など、ホスト言語の変数

または

- SQL 拡張機能を使って宣言された変数から SQL のプリコンパイラーによって生成されたホスト言語構成

これらは、SQL ステートメントで参照されています。ホスト変数はホスト言語のステートメントによって直接定義されるか、または SQL 拡張機能を使って間接的に定義されます。

SQL ステートメント内のホスト変数は、ホスト変数宣言規則に従ってプログラム内に記述されたホスト変数を識別する必要があります。

SQL ステートメントで使用されるホスト変数はすべて、REXX を除くすべてのホスト言語の SQL DECLARE セクションで宣言する必要があります。 SQL DECLARE セクションで宣言されている変数と同じ名前の変数を、 SQL DECLARE セクションの外部で宣言することはできません。 SQL DECLARE セクションは、BEGIN DECLARE SECTION で始まり、 END DECLARE SECTION で終わります。

メタ変数の *host-variable* (ホスト変数) が構文図の中で使われる場合、それはホスト変数への参照を示します。 VALUES INTO 節または、FETCH か SELECT INTO ステートメントの INTO 節のホスト変数は、行の中の列の値または式の値が割り当てられるホスト変数を識別するものです。その他のコンテキストでのホスト変数は、アプリケーション・プログラムからデータベース・マネージャーに渡される値を指定します。

動的 SQL におけるホスト変数

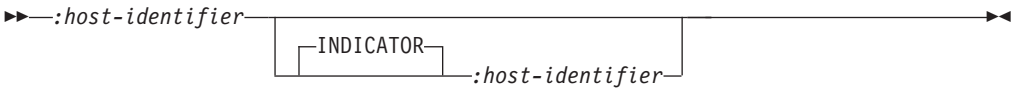
動的 SQL ステートメントにおいては、ホスト変数の代わりにパラメーター・マーカーが使用されます。 パラメーター・マーカーは疑問符 (?) で表されます。これは、動的 SQL ステートメントにおいてアプリケーションが値を提供する位置、すなわち、ステートメント・ストリングが静的 SQL ステートメントであるとすれば、ホスト変数が来ることになる位置を示します。以下に、ホスト変数を使った静的 SQL ステートメントの例を示します。

```
INSERT INTO DEPARTMENT
  VALUES (:hv_deptno, :hv_deptname, :hv_mgrno, :hv_admrdept)
```

次に、パラメーター・マーカーを使った動的 SQL ステートメントの例を示します。

```
INSERT INTO DEPARTMENT VALUES (?, ?, ?, ?)
```

構文図におけるメタ変数 *host-variable* (ホスト変数) は、一般に以下のように展開されます。



各 *host-identifier* (ホスト ID) は、ソース・プログラムの中で宣言される必要があります。 2 番目のホスト ID で指定される変数は、データ・タイプが短精度整数のものでなければなりません。

最初の host-identifier (ホスト ID) は、*メイン変数* を指定します。演算に応じて、このホスト ID はデータベース・マネージャーに値を提供したり、またはデータベース・マネージャーから提供される値を受け取ったりします。入力ホスト変数は、ランタイム・アプリケーション・コード・ページの値を提供します。出力ホスト変数には、データが出力アプリケーション変数にコピーされるときに、必要に応じてランタイム・アプリケーション・コード・ページに変換される値が提供されます。指定されるホスト変数は、同じプログラム内で入力変数と出力変数の両方として機能できます。

2 番目の host-identifier (ホスト ID) は、その*標識変数* を示します。標識変数の目的は以下のとおりです。

- NULL 値を指定する。標識変数の負の値は、NULL 値を指定するものとなります。 -2 の値は、結果を求める際に数値変換または演算式のエラーが発生したことを示します。
- 切り捨てられたストリングの元の長さを記録する (値のソースがラージ・オブジェクト・タイプでない場合)。
- ホスト変数に割り当てたときに時刻が切り捨てられた場合、その時刻の秒の部分を記録します。

たとえば、:HV1:HV2 を使用して挿入値または更新値を指定する場合に、 HV2 が負であると、指定される値は NULL 値になります。 HV2 が負でない場合、指定される値は HV1 の値です。

同様に、:HV1:HV2 が VALUES INTO 節、または FETCH あるいは SELECT INTO ステートメントに指定され、しかも戻された値が NULL 値である場合には、HV1 は変更されず、HV2 は負の値に設定されます。 DFT_SQLMATHWARN を yes にしてデータベースが構成されている場合 (または静的 SQL ステートメントのバインドの過程である場合)、 HV2 を -2 にすることができます。 HV2 が -2 である場合、HV1 の数値タイプへの変換エラー、または HV1 の値を判別するために使用される演算式の評価エラーにより、 HV1 の値を戻すことができません。 DB2 Universal Database™ のバージョン 5 より前のクライアント・バージョンを使用してデータベースにアクセスする場合、 HV2 は算術例外に対して -1 になります。戻された値が NULL 値でない場合は、その値が HV1 に割り当てられ、 HV2 はゼロに設定されます (ただし、 HV1 への割り当てに非 LOB ストリングのストリング切り捨てが必要になる場合を除きます。この場合 HV2 はストリングの元の長さに設定されます)。割り当て時に時刻の秒の部分の切り捨てが必要な場合、HV2 は秒数に設定されます。

2 番目のホスト ID が省略されている場合は、ホスト変数は標識変数を持たないことになります。ホスト変数参照 :HV1 によって指定される値は、常に HV1 の値であり、変数に NULL 値を割り当てることはできません。したがって、この形式は、対応する列で NULL 値を使えない場合以外は、INTO 節では使用しないでください。この形式が使用された場合に、列に NULL 値が入っていると、データベース・マネージャーは実行時にエラーを生成します。

ホスト変数を参照する SQL ステートメントは、対象のホスト変数の宣言の範囲内にある必要があります。カーソルの SELECT ステートメントで参照されるホスト変数の場合、この規則は DECLARE CURSOR ステートメントではなく、OPEN ステートメントに適用されます。

例

プロジェクト (PROJNO) 'IF1000' で、PROJECT 表を使用して、ホスト変数 PNAME (VARCHAR(26)) はプロジェクト名 (PROJNAME) に、ホスト変数 STAFF (dec(5,2)) はスタッフ配置の平均レベル (PRSTAFF) に、ホスト変数 MAJPROJ (char(6)) は主要プロジェクト (MAJPROJ) に設定します。 PRSTAFF と MAJPROJ 列は NULL 値である可能性があるため、標識変数 STAFF_IND (短精度整数) と MAJPROJ_IND (短精度整数) を使用します。

```
SELECT PROJNAME, PRSTAFF, MAJPROJ
  INTO :PNAME, :STAFF :STAFF_IND, :MAJPROJ :MAJPROJ_IND
  FROM PROJECT
  WHERE PROJNO = 'IF1000'
```

MBCS の考慮事項: ホスト変数名にマルチバイト文字を使用できるかどうかは、ホスト言語によって決まります。

BLOB、CLOB、および DBCLOB のホスト変数の参照

通常の BLOB、CLOB、および DBCLOB の変数、 LOB のロケーター変数 (『ロケーター変数の参照』を参照)、および LOB ファイル参照変数 (78 ページの『BLOB、CLOB、および DBCLOB ファイル参照変数の参照』を参照) は、すべてのホスト言語の中で定義可能です。 LOB が可能なロケーションでは、構文図の *host-variable* (ホスト変数) という用語は、通常のホスト変数、ロケーター変数、またはファイル参照変数を指します。これらはネイティブのデータ・タイプではないため、 SQL 拡張機能が使用され、それぞれの変数を表現するのに必要なホスト言語構成をプリコンパイラーが生成します。 REXX の場合、LOB はストリングにマップされます。

ラージ・オブジェクト値全体を保持できるほど大きい変数を定義できる場合もあります。このような場合で、サーバーからのデータ転送を据え置いてもパフォーマンス上のメリットが期待できない場合は、ロケーターを使用する必要はありません。しかし、ホスト言語やスペースの制限により、ラージ・オブジェクト全体を一度に一時記憶に保管するのが難しい場合がよくありますし、パフォーマンス上のメリットを考え合わせた上で、ラージ・オブジェクトはロケーターによって参照し、一度にラージ・オブジェクトの一部分だけを保持するホスト変数にオブジェクトの一部を選択して割り当て、そこで更新するという方法を採用することもできるかもしれません。

ロケーター変数の参照

ロケーター変数 は、アプリケーション・サーバーで LOB 値を表すロケーターの入ったホスト変数です。

SQL ステートメントにおけるロケーター変数は、ロケーター変数の宣言規則に従ってプログラムに記述されたロケーター変数を識別したものでなければなりません。これは常に SQL ステートメントによって間接的に行われます。

構文図で locator-variable (ロケーター変数) の語が使用される場合、それはロケーター変数への参照を表します。メタ変数 *locator-variable* (ロケーター変数) は、*host-variable* (ホスト変数) の場合と同じく、 *host-identifier* (ホスト ID) を組み込めるように拡張できます。

他のすべてのホスト変数と同様に、ラージ・オブジェクトのロケーター変数にも標識変数を対応させることができます。ラージ・オブジェクトのロケーター・ホスト変数に対応する標識変数は、他のデータ・タイプの標識変数と同じように動作します。データベースから NULL 値が戻されると、標識変数が設定され、ロケーター・ホスト変数は変更されません。つまり、ロケーターが NULL 値を指すことはないということです。

現時点で何の値も表していないロケーター変数が参照されると、エラー (SQLSTATE 0F001) になります。

トランザクションのコミット時、またはトランザクションの終了時に、そのトランザクションが獲得していたロケーターはすべて解放されます。

BLOB、CLOB、および DBCLOB ファイル参照変数の参照

BLOB、CLOB、および DBCLOB のファイル参照変数は、 LOB の直接のファイル入出力に使用されるもので、すべてのホスト言語で定義可能です。 これらはネイティブのデータ・タイプではないため、 SQL 拡張機能が使用され、それぞれの変数を表現するのに必要なホスト言語構成をプリコンパイラーが生成します。 REXX の場合、LOB はストリングにマップされます。

LOB ロケーターが LOB バイトを収容するのではなく LOB バイトを表すのと同じように、ファイル参照変数はファイルを収容するのではなくファイルを表します。データベースの照会、更新、および挿入では、ファイル参照変数を使用して 1 つの列値を保管したり検索したりすることができます。

ファイル参照変数には以下のプロパティーがあります。

データ・タイプ
BLOB、CLOB、または DBCLOB。この特性は、変数の宣言時に指定されます。

方向 これはアプリケーション・プログラムによって実行時に指定される必要があります (ファイル・オプション値の一部として)。方向は以下のどちらかです。

- 入力 (EXECUTE ステートメント、OPEN ステートメント、 UPDATE ステートメント、INSERT ステートメント、または DELETE ステートメントでのデータのソースとして使用されます)。
- 出力 (FETCH ステートメントまたは SELECT INTO ステートメントのデータの宛先として使用されます)。

ファイル名
これはアプリケーション・プログラムによって実行時に指定される必要があります。以下のいずれかです。

- ファイルの完全パス名 (推奨)。
- 相対ファイル名。相対ファイル名を指定した場合、それはクライアント・プロセスの現行パスに追加されます。

アプリケーション内では、ファイルは 1 つのファイル参照変数でのみ参照する必要があります。

ファイル名の長さ
これはアプリケーション・プログラムによって実行時に指定される必要があります。ファイル名の長さをバイト単位で表したものです。

ファイル・オプション
アプリケーションがファイル参照変数を使用するには、事前にいくつかのオプションの中の 1 つをその変数に割り当てる必要があります。オプション

の設定は、ファイル参照変数構造の中のフィールドの INTEGER 値によって行います。ファイル参照変数ごとに、以下の値の 1 つを指定する必要があります。

- 入力 (クライアントからサーバーへ)

  SQL_FILE_READ
  これは、オープン、読み取り、クローズの対象となる通常のファイルです。 (オプションは、COBOL では SQL-FILE-READ、FORTRAN では sql_file_read、REXX では READ です。)

- 出力 (サーバーからクライアントへ)

  SQL_FILE_CREATE
  新規ファイルを作成します。該当のファイルがすでに存在していると、エラーが戻されます。 (オプションは、COBOL では SQL-FILE-CREATE、FORTRAN では sql_file_create、REXX では CREATE です。)

  SQL_FILE_OVERWRITE (上書き)
  指定した名前のファイルが存在している場合には上書きされ、存在していない場合には新たなファイルが作成されます。(オプションは、COBOL では SQL-FILE-OVERWRITE、FORTRAN では sql_file_overwrite、REXX では OVERWRITE です。)

  SQL_FILE_APPEND
  指定した名前のファイルが存在している場合には出力がそれに追加されます。存在していない場合には新たなファイルが作成されます。(オプションは、COBOL では SQL-FILE-APPEND、FORTRAN では sql_file_append、REXX では APPEND です。)

データ長
これは入力では使用されません。出力のとき、ファイルに書き込まれる新規データの長さがこのデータ長に設定されます。このデータ長はバイト単位です。

他のすべてのホスト変数と同様に、ファイル参照変数にも標識変数を対応させることができます。

出力ファイル参照変数の例 (C の場合)

宣言部が以下のようにコーディングされているとします。

```
EXEC SQL BEGIN DECLARE SECTION
   SQL TYPE IS CLOB_FILE  hv_text_file;
   char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

これをプリプロセスした後は以下のようになります。

```
EXEC SQL BEGIN DECLARE SECTION
   /* SQL TYPE IS CLOB_FILE  hv_text_file; */
   struct {
       unsigned long  name_length; //  File Name Length
       unsigned long  data_length; //  Data Length
       unsigned long  file_options; // File Options
```

```
        char           name[255];   // File Name
    } hv_text_file;
    char  hv_patent_title[64];
EXEC SQL END DECLARE SECTION
```

その後、以下のコードを使って、データベースの CLOB の列から選択し、:hv_text_file で参照される新規ファイルに書き込むことができます。

```
strcpy(hv_text_file.name, "/u/gainer/papers/sigmod.94");
hv_text_file.name_length = strlen("/u/gainer/papers/sigmod.94");
hv_text_file.file_options = SQL_FILE_CREATE;

EXEC SQL SELECT content INTO :hv_text_file from papers
     WHERE TITLE = 'The Relational Theory behind Juggling';
```

入力ファイル参照変数の例 (C の場合)

前出の例と同じ宣言部を考えます。以下のコードは、:hv_text_file によって参照される通常ファイルからのデータを CLOB の列へ挿入するものです。

```
strcpy(hv_text_file.name, "/u/gainer/patents/chips.13");
hv_text_file.name_length = strlen("/u/gainer/patents/chips.13");
hv_text_file.file_options = SQL_FILE_READ:
strcpy(:hv_patent_title, "A Method for Pipelining Chip Consumption");

EXEC SQL INSERT INTO patents( title, text )
         VALUES(:hv_patent_title, :hv_text_file);
```

構造化タイプ・ホスト変数の参照

構造化タイプ変数は、FORTRAN、REXX、および Java を除く、すべてのホスト言語で定義できます。これらはネイティブのデータ・タイプではないため、 SQL 拡張機能が使用され、それぞれの変数を表現するのに必要なホスト言語構成をプリコンパイラーが生成します。

他のすべてのホスト変数と同様に、構造化タイプ変数にも標識変数を対応させることができます。構造化タイプ・ホスト変数に対応する標識変数は、他のデータ・タイプの標識変数と同じように動作します。データベースから NULL 値が戻されると、標識変数が設定され、構造化タイプ・ホスト変数は変更されません。

構造化タイプ用の実際のホスト変数は、組み込みデータ・タイプとして定義されます。構造化タイプと関連した組み込みデータ・タイプは、以下のようなアクセスが可能でなければなりません。

- プリコンパイル・コマンドで指定した TRANSFORM GROUP オプションによって定義したとおりの、構造化タイプの FROM SQL トランスフォーム関数の結果に基づくアクセス。
- プリコンパイル・コマンドで指定した TRANSFORM GROUP オプションによって定義したとおりの、構造化タイプの TO SQL トランスフォーム関数のパラメーターへのアクセス。

ホスト変数の代わりにパラメーター・マーカーを使用している場合、 SQLDA に適切なパラメーター・タイプの特性を指定する必要があります。この場合、SQLDA には SQLVAR 構造のセットが「2 つ」必要です。また、副次 SQLVAR の SQLDATATYPE_NAME フィールドには、構造化タイプのスキーマおよびタイプ名を入れなければなりません。 SQLDA 構造でスキーマを省略すると、エラーが発生します (SQLSTATE 07002)。

例

C プログラムで、(組み込みタイプの BLOB(1048576) を使用して、タイプ POLYGON の) ホスト変数 *hv_poly* と *hv_point* を定義します。

```
EXEC SQL BEGIN DECLARE SECTION;
      static SQL
         TYPE IS POLYGON AS BLOB(1M)
         hv_poly, hv_point;
EXEC SQL END DECLARE SECTION;
```

データ・タイプ

SQL で扱うことのできる一番小さいデータの単位は*値* です。値の解釈方法は、値の出所 (ソース) のデータ・タイプによって異なります。ソースには、以下のものがあります。

- 定数
- 列
- 関数
- 式
- 特殊レジスター
- 変数 (ホスト変数、SQL 変数、グローバル変数、パラメーター・マーカー、およびルーチンのパラメーターなど)

DB2 は、いくつかの組み込みデータ・タイプをサポートします。また、ユーザー定義のデータ・タイプもサポートします。 83 ページの図 11 は、サポートされる組み込みデータ・タイプを示しています。

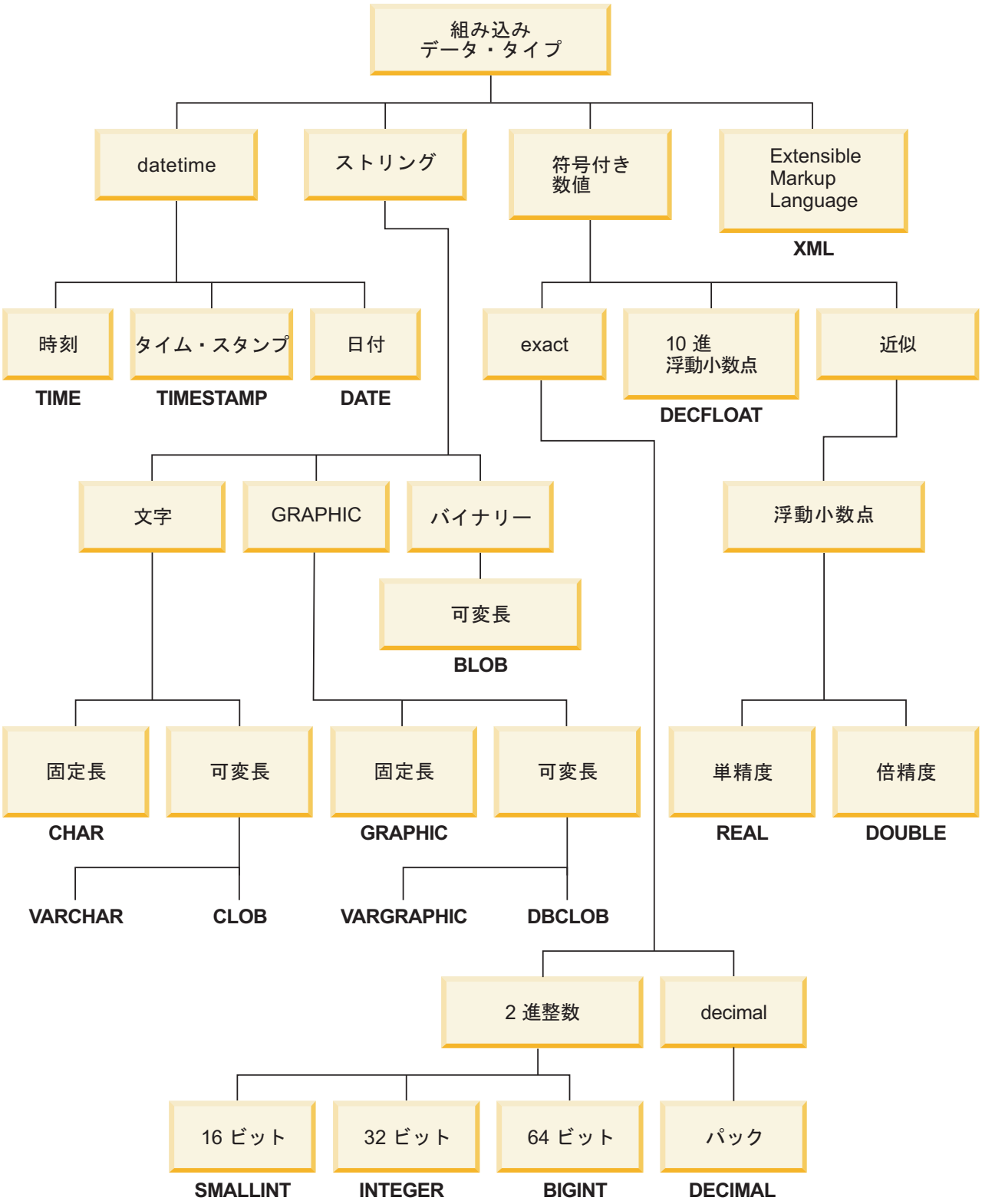


図 11. DB2 組み込みデータ・タイプ

すべてのデータ・タイプには、NULL 値が入っています。 NULL 値とは、すべての非 NULL 値と区別されていて、それによって (非 NULL) 値がないことを指し示

すための特殊値のことです。すべてのデータ・タイプには NULL 値が含まれますが、 NOT NULL として定義されている列に NULL 値を入れることはできません。

データ・タイプ・リスト

数値

数値データ・タイプは、整数、10 進数、浮動小数点数、および 10 進浮動小数点数です。

数値データ・タイプは、以下のように分類されます。

- 厳密な数: 整数および 10 進数
- 10 進浮動小数点数
- 近似数: 浮動小数点数

整数には、短精度整数、長精度整数、および 64 ビット整数 (big integer) が含まれます。整数の数値は、整数の厳密な表記です。 10 進数は、固定精度と位取りを持つ、数値の厳密な表記です。整数および 10 進数は、厳密な数値タイプと考えられます。

10 進浮動小数点数は、16 または 34 の精度を持つことができます。 10 進浮動小数点数は、実数の厳密な表記と実数の近似値の両方をサポートするため、厳密な数値タイプと近似値タイプのいずれでもないと考えられます。

浮動小数点数には、単精度および倍精度があります。浮動小数点数は、実数の近似値であり、近似値タイプと考えられます。

すべての数値には、*符号*、*精度*、および *位取り*があります。 10 進浮動小数点数以外のすべての数値の場合、列値がゼロなら、符号は正になります。10 進浮動小数点数には、負および正のゼロが含まれます。10 進浮動小数点数には、数値とさまざまな指数の付いた同じ数値に対してそれぞれ別個の値があります (例えば、0.0、0.00、0.0E5、1.0、1.00、1.0000)。精度は、符号を除いた 10 進数の桁数の合計です。 位取りは、小数点以下の小数桁数の合計です。小数点がない場合、位取りはゼロになります。

CREATE TABLE ステートメントの説明の中のデータ・タイプの項も参照してください。

短精度整数 (SMALLINT)

短精度整数 は、精度が 5 桁の 2 バイト整数です。短精度整数の範囲は -32768 から 32767 です。

長精度整数 (INTEGER)

長精度整数 は、精度が 10 桁の 4 バイトの整数です。長精度整数の範囲は -2,147,483,648 から +2,147,483,647 です。

64 ビット整数 (BIGINT)

64 ビット整数 は、精度が 19 桁の 8 バイトの整数です。 64 ビット整数の範囲は -9,223,372,036,854,775,808 から +9,223,372,036,854,775,807 です。

10 進数 (DECIMAL または NUMERIC)

10 進数値 は、暗黙的な小数点を持つパック 10 進数です。小数点の位置は、その数値の精度と位取りによって決定されます。 数値の小数部分の桁数である位取りが、負になったり精度数よりも大きくなったりすることはありません。最大精度は 31 桁です。

10 進数の列の値は、すべて同じ精度と位取りの値です。 10 進数の変数または 10 進数の列の数値の範囲は、 *-n* から *+n* です (絶対値 *n* は、適切な精度および 10 進数で表現できる最も大きな数値)。最大範囲は $-10^{31}+1$ から $10^{31}-1$ です。

単精度浮動小数点 (REAL)

単精度浮動小数点 数は、実数の 32 ビット近似値です。この数は、ゼロにするか、または -3.4028234663852886e+38 から -1.1754943508222875e-38 まで、または 1.1754943508222875e-38 から 3.4028234663852886e+38 までの範囲にすることができます。

倍精度浮動小数点 (DOUBLE または FLOAT)

倍精度浮動小数点 数は、実数の 64 ビットの近似値です。この数は、ゼロにするか、または -1.7976931348623158e+308 から -2.2250738585072014e-308 まで、または 2.2250738585072014e-308 から 1.7976931348623158e+308 までの範囲にすることができます。

10 進浮動小数点数 (DECFLOAT)

10 進浮動小数点 値は、小数点の付いた IEEE 754r の数値です。小数点の位置は、各 10 進浮動小数点値に格納されます。最大精度は 34 桁です。 10 進浮動小数点数の範囲は、16 桁か 34 桁の精度のどちらかであり、それぞれ 10^{-383} から 10^{+384} または 10^{-6143} から 10^{+6144} の指数範囲です。 DECFLOAT 値の最小指数 E_{min} は、DECFLOAT(16) の場合 -383、DECFLOAT(34) の場合 -6143 です。 DECFLOAT 値の最大指数 E_{max} は、DECFLOAT(16) の場合 384、DECFLOAT(34) の場合 6144 です。

有限数に加えて、10 進浮動小数点数は、以下のいずれかの名前の 10 進浮動小数点特殊値を表すことができます。

- 無限大 - 絶対値が無限に大きい数を表す値
- 静止 NaN - 未定義の結果を表す値で、無効数値条件を引き起こさない値
- シグナリング NaN - 未定義の結果を表す値で、数値演算で使用される場合に無効数値条件を引き起こす値

数値がこれらの特殊値のいずれかであるとき、その数値の係数および指数は未定義です。 正の無限大と負の無限大があるので、無限大値の符号は重要です。NaN 値の符号は、算術演算では意味がありません。

非正規数およびアンダーフロー

調整された指数が E_{min} より小さいゼロ以外の数値は、非正規数と呼ばれています。これらの非正規数は、すべての演算のオペランドとして受け入れられ、どの演算の結果としても生じる可能性があります。

非正規の結果では、指数の最小値は E_{min} - (精度 -1) となり、E_{tiny} と呼ばれます。ここで、精度は処理精度です。必要な場合には、指数が E_{tiny} より小さくならないように結果が丸められます。丸め中に結果が正確でなくなる場合、アンダーフロー条件が戻されます。非正規の結果は常にアンダーフロー条件を戻すとは限りません。

計算中に数値がアンダーフローしてゼロになるとき、指数は E_{tiny} になります。指数の最大値は、影響を受けません。

非正規数の指数の最大値は、結果が非正規数にならない演算中に生じる指数の最小値と同じです。 これは、小数桁の係数の長さが精度と等しいときに起こります。

文字ストリング

文字ストリング は、一連のバイトです。ストリングの長さは、その一連のバイトのバイト数です。長さがゼロの場合、その値は空ストリング と呼ばれます。この値を NULL 値と混同しないようにしてください。

固定長文字ストリング (CHAR)

固定長ストリングの列の値は、すべて同じ長さです。この長さは、その列の長さ属性によって決定されます。長さ属性は、1 以上 254 以下でなければなりません。

可変長文字ストリング

可変長文字ストリングには、以下の 2 つのタイプがあります。

- VARCHAR 値は、最大 32 672 バイトまでの長さにすることができます。
- CLOB (文字ラージ・オブジェクト) 値は、最大 2 ギガバイト (2 147 483 647 バイト) までの長さにすることができます。 CLOB は、(単一文字セットで記述された文書などの) ラージ SBCS、または混合 (SBCS および MBCS) 文字ベース・データを保管するのに使用されます。したがって、それに関連する SBCS または混合コード・ページがあります。

結果が CLOB データ・タイプとなる式、および構造化タイプ列に対して、特殊な制限が適用されます。そのような式および列は、以下の場所では使用できません。

- DISTINCT 節が先行している SELECT リスト
- GROUP BY 節
- ORDER BY 節
- UNION ALL 以外のセット演算子の副選択
- 基本述部、多値比較述部、BETWEEN 述部、または IN 述部
- 列関数
- VARGRAPHIC、TRANSLATE、および日付/時刻スカラー関数
- LIKE 述部のパターン・オペランドまたは POSSTR 関数の検索ストリング・オペランド
- 日付/時刻値のストリング表記。

VARCHAR を引数として取る SYSFUN スキーマの関数は、 4 000 バイトよりも長い VARCHAR を引数として受け入れません。しかし、そのような関数の多くには、 CLOB(1M) を受け入れるための代替シグニチャーが用意されています。そのような代替シグニチャーが用意されている関数の場合は、ユーザーが 4 000 バイトよりも長い VARCHAR ストリングを明示的に CLOB へキャストし、結果が戻されたら任意の長さの VARCHAR へ再びキャストし直すという手順を取ります。

C の NULL 終了文字ストリングは、プリコンパイル・オプションの標準レベルに応じて、異なった方式で処理されます。

それぞれの文字ストリングは、さらに次のいずれかと定義されます。

ビット・データ
コード・ページに対応していないデータ。

1 バイト文字セット (SBCS) データ
それぞれの文字が 1 バイトで表現されるデータ。

混合データ
1 バイト文字セットとマルチバイト文字セット (MBCS) の文字の混合を納めたデータ。

注: データ・タイプ LONG VARCHAR は引き続きサポートされますが、非推奨になっており、将来のリリースで除去される可能性があります。

組み込み関数内の文字単位

個々の組み込み関数に対して文字単位を指定する機能を利用して、「バイト・ベース方式」ではなく、「文字ベース方式」でストリング・データを処理することができます。*文字単位* によって、実行される操作の長さが決まります。操作の文字単位として、CODEUNITS16、CODEUNITS32、または OCTETS を指定することができます。

CODEUNITS16
Unicode UTF-16 を操作の単位に指定します。 CODEUNITS16 が便利なのは、幅が 2 バイトのコード単位のデータをアプリケーションで処理する場合です。*補足文字* と呼ばれる一部の文字の場合、2 つの UTF-16 コード単位をエンコードする必要があることに注意してください。例えば、音楽のト音記号の場合、2 つの UTF-16 コード単位 (UTF-16BE の X'D834' および X'DD1E') が必要です。

CODEUNITS32
Unicode UTF-32 を操作の単位に指定します。 CODEUNITS32 が便利なのは、単純な固定長の形式のデータを処理し、データの保管形式 (ASCII、UTF-8、または UTF-16) に関係なく同じ応答を戻す必要のあるアプリケーションの場合です。

OCTETS
バイトを操作の単位に指定します。アプリケーションがバッファー・スペースを割り振ろうとしている場合や、単純なバイト処理を操作で使用する必要のある場合に、OCTETS が使用されることがよくあります。

OCTETS (バイト数) を使用して計算したストリングの算出長は、CODEUNITS16 または CODEUNITS32 を使用して計算したものとは異なることがあります。
OCTETS を使用した場合、ストリングの長さは、単純にストリング中のバイト数をカウントして判別されます。 CODEUNITS16 または CODEUNITS32 を使用した場合、ストリングの長さは、それぞれ UTF-16 または UTF-32 でストリングを表すのに必要な 16 ビットまたは 32 ビットのコード単位の数をカウントして判別されます。 CODEUNITS16 および CODEUNITS32 を使用して判別した長さは、補足文字がデータ内に入っていない限り同じです (90 ページの『CODEUNITS16 と CODEUNITS32 の相違』 を参照)。

例えば、Unicode UTF-8 でエンコードされた VARCHAR(128) 列である NAME の中に、値 'Jürgen' が入っていると想定します。それぞれ CODEUNITS16 および CODEUNITS32 でストリングの長さをカウントする以下の 2 つの照会は、同じ値 (6) を戻します。

```
SELECT CHARACTER_LENGTH(NAME,CODEUNITS16) FROM T1
  WHERE NAME = 'Jürgen'

SELECT CHARACTER_LENGTH(NAME,CODEUNITS32) FROM T1
  WHERE NAME = 'Jürgen'
```

次の照会は、OCTETS でストリングの長さをカウントしますが、これは値 7 を戻します。

```
SELECT CHARACTER_LENGTH(NAME,OCTETS) FROM T1
  WHERE NAME = 'Jürgen'
```

これらの値は、指定の文字単位で表現されたストリングの長さを表しています。

以下の表は、名前 'Jürgen' を UTF-8、UTF-16BE (ビッグ・エンディアン)、および UTF-32BE (ビッグ・エンディアン) で表現したものです。

```
形式       名前 'Jürgen' の表現
--------  ------------------------------------
UTF-8   X'4AC3BC7267656E'
UTF-16BE  X'004A00FC007200670065006E'
UTF-32BE  X'0000004A000000FC00000072000000670000006500000006E'
```

文字 'ü' の表現方法は、次のように、3 つの文字単位でそれぞれ異なります。

- 文字 'ü' の UTF-8 表現は X'C3BC' です。
- 文字 'ü' の UTF-16BE 表現は X'00FC' です。
- 文字 'ü' の UTF-32BE 表現は X'000000FC' です。

組み込み関数で文字単位を指定しても、関数の結果のデータ・タイプまたはコード・ページには影響を与えません。 CODEUNITS16 または CODEUNITS32 を指定すると、必要に応じて、評価の目的でデータが DB2 によって Unicode に変換されます。

LOCATE または POSITION 関数に対して OCTETS を指定した場合に、ストリング引数のコード・ページがそれぞれ異なっていると、データは DB2 によって *source-string* 引数のコード・ページに変換されます。この場合、関数の結果は、*source-string* 引数のコード・ページのものになります。単一のストリング引数をとる関数に対して OCTETS を指定した場合、データは、そのストリング引数のコード・ページで評価され、関数の結果は、そのストリング引数のコード・ページのものになります。

CODEUNITS16 と CODEUNITS32 の相違

CODEUNITS16 または CODEUNITS32 を指定すると、Unicode 補足文字がデータ中に入っていないかぎり、結果は同じになります。その理由は、2 つの UTF-16 コード単位または 1 つの UTF-32 コード単位で補足文字が表されるからです。 UTF-8 では、非補足文字は 1 から 3 バイトまでで表され、補足文字は、4 バイトで表されます。 UTF-16 では、非補足文字は、1 つの CODEUNITS16 コード単位つまり 2 バイトで表され、補足文字は、2 つの CODEUNITS16 コード単位つまり 4 バイトで表されます。 UTF-32 では、文字は、1 つの CODEUNITS32 コード単位つまり 4 バイトで表されます。

例えば、以下の表は、数学の太字の大文字 A と、ローマ字の大文字 A の 16 進値を示しています。数学の太字の大文字 A は、UTF-8、UTF-16、および UTF-32 では 4 バイトで表される補足文字です。

文字	UTF-8 で表現した場合	UTF-16BE で表現した場合	UTF-32BE で表現した場合
Unicode 値 X'1D400' - 'A'。数学の太字の大文字 A	X'F09D9080'	X'D835DC00'	X'0001D400'
Unicode 値 X'0041' - 'A'。ローマ字の大文字 A。	X'41'	X'0041'	X'00000041'

C1 は Unicode UTF-8 でエンコードされた VARCHAR(128) の列であり、表 T1 には 1 つの行が入っていて、これに数学の太字の大文字 A (X'F09D9080') の値が入っているものとします。 以下の照会は、それぞれ異なる結果を戻します。

```
照会                                                  戻り
-----                                                 -------
SELECT CHARACTER_LENGTH(C1,CODEUNITS16) FROM T1       2

SELECT CHARACTER_LENGTH(C1,CODEUNITS32) FROM T1       1

SELECT CHARACTER_LENGTH(C1,OCTETS) FROM T1            4
```

GRAPHIC ストリング

GRAPHIC ストリング は、 2 バイト文字データを表す一連のバイトです。ストリングの長さは、その一連のバイトの 2 バイト文字の数です。長さがゼロの場合、その値は空ストリング と呼ばれます。この値を NULL 値と混同しないようにしてください。

GRAPHIC ストリングの値に 2 バイト文字コード・ポイント以外の値が入っていないかどうかを調べる妥当性検査は行われません。 (この規則の例外は、WCHARTYPE CONVERT オプションを指定してプリコンパイルされたアプリケーションです。このオプションを指定した場合、妥当性検査が行われます。) データベース・マネージャーは、 2 バイト文字データが GRAPHIC データ・フィールドに入っていることを想定しています。データベース・マネージャーは、GRAPHIC ストリング値の長さが偶数バイトであることを検査します。

C の NULL 終了 GRAPHIC ストリングは、プリコンパイル・オプションの標準レベルに応じて、異なった方式で処理されます。このデータ・タイプは表内に作成することはできません。データをデータベースに挿入するときやデータベースから検索するときにのみ使用可能です。

固定長 GRAPHIC ストリング (GRAPHIC)

固定長 GRAPHIC ストリングの列の値は、すべて同じ長さです。この長さは、その列の長さ属性によって決定されます。長さ属性は、1 以上 127 以下でなければなりません。

可変長 GRAPHIC ストリング

可変長 GRAPHIC ストリングには、以下の 2 つのタイプがあります。

- VARGRAPHIC 値は、最長 16 336 個の 2 バイト文字とすることができます。
- DBCLOB (2 バイト文字ラージ・オブジェクト) 値は、最長 1 073 741 823 個の 2 バイト文字とすることができます。 DBCLOB は、(単一文字セットで記述された文書のような) 大規模な DBCS 文字ベースのデータの保管に使用されます。したがって、 DBCLOB にはそれに関連する DBCS コード・ページがあります。

最大長が 127 バイトを超える可変長 GRAPHIC ストリングが結果となる式には、特別な制限が適用されます。この制限は、 88 ページの『可変長文字ストリング』で指定されているものと同じです。

注: データ・タイプ LONG VARGRAPHIC は引き続きサポートされますが、非推奨になっており、将来のリリースで除去される可能性があります。

バイナリー・ストリング

バイナリー・ストリング は、一連のバイトです。通常はテキスト・データの入った文字ストリングとは異なり、バイナリー・ストリングは従来型ではないデータ、例えば画像、音声、混合メディアなどを保持します。 FOR BIT DATA サブタイプの文字ストリングも似たような目的で使用されることがありますが、この 2 つのデータ・タイプは互換ではありません。 BLOB スカラー関数を使用すると、 FOR BIT DATA 文字ストリングをバイナリー・ストリングにキャスト (タイプ変換) することができます。バイナリー・ストリングはコード・ページに対応していません。文字ストリングと同じ制限があります (詳細については、 88 ページの『可変長文字ストリング』を参照)。

バイナリー・ラージ・オブジェクト (BLOB)

バイナリー・ラージ・オブジェクト (BLOB) は、最長 2 ギガバイト (2,147,483,647 バイト) の可変長バイナリー・ストリングです。 BLOB は、ユーザー定義タイプおよびユーザー定義関数で活用するために構造化データを保持します。 FOR BIT DATA 文字ストリングと同じように、 BLOB ストリングに対応するコード・ページはありません。

ラージ・オブジェクト (LOB)

ラージ・オブジェクト および総称頭字語である LOB は、 BLOB、CLOB、または DBCLOB のデータ・タイプを参照するときに使用されます。 LOB 値は、 88 ページの『可変長文字ストリング』に説明されている LONG VARCHAR 値に適用される制限に従います。このような制限は、LOB ストリングの長さ属性が 254 バイト以下であっても適用されます。

LOB 値は非常に大きいので、この値をデータベース・サーバーからクライアント・アプリケーション・プログラムのホスト変数に転送するには多くの時間がかかります。アプリケーションが一度に処理するのは通常は LOB 値の全体ではなく小さな部分だけなので、アプリケーションはラージ・オブジェクト・ロケーターを使用して LOB を参照できます。

ラージ・オブジェクト・ロケーター つまり LOB ロケーターは、データベース・サーバーの単一 LOB 値を表す値を伴うホスト変数です。

アプリケーション・プログラムは LOB ロケーターに LOB 値を選択できます。その後、アプリケーション・プログラムは LOB ロケーターを使用して、そのロケーター値を入力として指定することによって、その LOB 値に対するデータベース操作 (スカラー関数 SUBSTR、 CONCAT、VALUE、LENGTH の適用、割り当ての実行、 LIKE または POSSTR による LOB の探索、LOB に対するユーザー定義関数の適用など) を要求することができます。出力結果 (クライアントのホスト変数に割り当てられるデータ) は、多くの場合、入力 LOB 値の小さいサブセットとなります。

LOB ロケーターは、基本値以外のものを表現する場合もあり、 LOB 式に対応する値を表現することができます。たとえば、LOB ロケーターで、次の式に対応する値を表現できます。

```
SUBSTR( <lob 1> CONCAT <lob 2> CONCAT <lob 3>, <start>, <length> )
```

そのホスト変数に NULL 値が選択されている場合、標識変数は値が NULL 値であることを示す -1 に設定されます。しかし、LOB ロケーターの場合は、標識変数の意味が少し違います。ロケーター・ホスト変数自体は NULL 値にすることができないので、標識変数の負の値は、その LOB ロケーターが表す LOB 値が NULL 値であることを示します。標識変数の値により、NULL 値情報はクライアントにとってローカルに保持されます。サーバー側では有効なロケーターによって NULL 値を追跡しません。

LOB ロケーターが表すのは 1 つの値であって、データベースの行やロケーションを表すわけではない、ということは重要です。 値がロケーターに選択されると、ロケーターが参照する値に影響を及ぼすような操作を、元の行や表に対して実行することはできません。ロケーターに対応する値は、トランザクションが終了するか、ロケーターが明示的に解放されるか、どちらかが行われるまで有効です。ロケーターでは、この機能を実現するために、追加でデータのコピーなどを行ったりはしません。その代わりに、ロケーター・メカニズムに基本 LOB 値の内容が保管されます。 LOB 値 (または、上記のように式) のマテリアライズは、 LOB 値が実際に何らかの位置を割り当てられるまで延期されます。すなわち、ホスト変数の形式でユーザー・バッファーを割り当てられるか、もしくはデータベースの別のレコードを割り当てられるまでです。

LOB ロケーターは、トランザクションの中で LOB 値を参照するための唯一のメカニズムです。 LOB ロケーターはそれが作成されたトランザクションを超えて存続することはありません。これはデータベース・タイプではなく、データベースに保管されることはありません。したがって、ビューやチェック制約には加わりません。しかし、LOB ロケーターは LOB タイプのクライアント側の表現なので、FETCH、OPEN、または EXECUTE ステートメントで使用される SQLDA 構造の中で記述されるよう、 LOB ロケーターの SQLTYPE が用意されています。

日付/時刻の値

日付/時刻のデータ・タイプには、DATE、TIME、および TIMESTAMP などがあります。日付/時刻の値は、特定の算術演算およびストリング操作で使用することができ、特定のストリングとは互換性がありますが、これはストリングでも数字でもありません。

日付

日付 (date) は、年、月、日の 3 つの部分からなる値です。年の部分の範囲は 0001 から 9999 です。月の部分の範囲は 1 から 12 です。日の部分の範囲は 1 から *x* です (*x* は月によって異なります)。

日付の内部表示は 4 バイトのストリングです。各バイトは、2 桁のパック 10 進数からなります。最初の 2 バイトは年、3 番目のバイトは月、最後のバイトは日です。

DATE 列の長さは、SQLDA の項で説明するように、10 バイトです。これは、日付の値を文字ストリングで表記するために適した長さになっています。

時刻

時刻 (time) は、時、分、秒の 3 つの部分からなる値であり、 24 時間制の時刻を表します。時の部分の範囲は 0 から 24。それ以外の部分の範囲は 0 から 59 です。時が 24 の場合、分と秒の指定はゼロになります。

時刻の内部表示は 3 バイトのストリングです。各バイトは、2 桁のパック 10 進数からなります。最初のバイトは時、2 番目のバイトは分、最後のバイトは秒です。

TIME 列の長さは、SQLDA の項で説明するように、8 バイトです。これは、時刻の値を文字ストリングで表記するために適した長さになっています。

タイム・スタンプ

タイム・スタンプ *(timestamp)* は、 7 つの部分 (年、月、日、時、分、秒、マイクロ秒) から成っており、時刻が小数部分でマイクロ秒を指定する以外は、上記の定義と同様に日時を示します。

タイム・スタンプの内部表示は 10 バイトのストリングです。各バイトは、2 桁のパック 10 進数からなります。最初の 4 バイトは日付、次の 3 バイトは時刻、最後の 3 バイトはマイクロ秒です。

SQLDA に記述されている TIMESTAMP 列の長さは 26 バイトです。これは、値の文字ストリング表示に適した長さです。

日付/時刻の値のストリング表記

データ・タイプが DATE、TIME、または TIMESTAMP の値は、ユーザーが意識することのない内部形式で表されます。ただし、日付、時刻、およびタイム・スタンプの値は、ストリングで表すこともできます。データ・タイプが DATE、TIME、または TIMESTAMP である定数や変数がないため、この表示方法は便利です。日付/時刻の値を取り出すには、この値をストリング変数に割り当てる必要があります。

CHAR または GRAPHIC 関数 (Unicode データベース用のみ) を使用すると、日付/時刻値をストリング表記に変更することができます。通常、ストリング表記は、プログラムがプリコンパイルされるときか、またはデータベースにバインドされるときに、 DATETIME オプションの指定によってオーバーライドされるのでない限り、アプリケーションのテリトリー・コードに関連する日付/時刻の値のデフォルトの形式になります。

ラージ・オブジェクト・ストリング、LONG VARCHAR 値、または LONG VARGRAPHIC は、その長さに関係なく、日付/時刻値を表すために使用することはできません (SQLSTATE 42884)。

日付/時刻の値の有効なストリング表記が内部の日付/時刻の値の操作に使用される場合、ストリング表記が日付、時刻、またはタイム・スタンプの値の内部形式に変換されてから、操作が実行されます。

日付、時刻、およびタイム・スタンプのストリングでは、文字と数字しか使用することができません。

日付ストリング

日付のストリング表示は、数字で始まり、長さが 8 バイト以上のストリングです。末尾のブランクを付けることができます。月と日の部分の先行ゼロは省略可能です。

日付を示す有効なストリング・フォーマットを、以下の表に示します。各フォーマットは、名前および関連する省略形によって識別されます。

表 5. 日付のストリング表記フォーマット

フォーマット名	省略形	日付 フォーマット	例
国際標準化機構	ISO	yyyy-mm-dd	1991-10-27
IBM USA 標準規格	USA	mm/dd/yyyy	10/27/1991
IBM 欧州標準規格	EUR	dd.mm.yyyy	27.10.1991
日本工業規格西暦	JIS	yyyy-mm-dd	1991-10-27
地域別定義	LOC	アプリケーションのテリトリー・コードに依存します。	—

時刻ストリング

時刻のストリング表記は、数字で始まり、長さが 4 バイト以上のストリングです。末尾にブランクを付けることができます。時刻の時部分の先行ゼロは省略可能であり、秒は完全に省略することができます。 秒が省略されている場合は、0 秒が指定されたと見なされます。したがって、13:30 は 13:30:00 に等しくなります。

時刻を示す有効なストリング・フォーマットを、以下の表に示します。各フォーマットは、名前および関連する省略形によって識別されます。

表 6. 時刻のストリング表記フォーマット

フォーマット名	省略形	時刻 フォーマット	例
国際標準化機構	ISO	hh.mm.ss	13.30.05
IBM USA 標準規格	USA	hh:mm AM または PM	1:30 PM
IBM 欧州標準規格	EUR	hh.mm.ss	13.30.05
日本工業規格西暦	JIS	hh:mm:ss	13:30:05
地域別定義	LOC	アプリケーションのテリトリー・コードに依存します。	—

注:

1. ISO、EUR、または JIS フォーマットでは、.ss (もしくは :ss) は省略可能です。
2. 国際標準化機構は、時刻フォーマットを日本工業規格 (西暦) フォーマットと同じフォーマットに変更しました。このため、アプリケーションで現行の国際標準化機構フォーマットが必要な場合は、 JIS を使用してください。
3. USA 時刻ストリング・フォーマットでは、分の指定を省略できます。その場合、暗黙のうちに 00 分と見なされます。したがって、1 PM は 1:00 PM に等しくなります。
4. USA 時刻フォーマットでは、時を 13 以上にすることはできず、00:00 AM という特殊な場合を除いて、0 にすることはできません。 AM および PM の前にはスペースが 1 個入れられます。 AM および PM は、小文字または大文字のどちらで表しても構いません。

   24 時間制の JIS フォーマットを使用した場合、 USA フォーマットと 24 時間制との対応は次のようになります。

   - 12:01 AM から 12:59 AM は、00:01:00 から 00:59:00 に対応します。
   - 01:00 AM から 11:59 AM は、01:00:00 から 11:59:00 に対応します。
   - 12:00 PM (正午) から 11:59 PM は、12:00:00 から 23:59:00 に対応します。
   - 12:00 AM (深夜) は 24:00:00 に対応し、00:00 AM (深夜) は 00:00:00 に対応します。

タイム・スタンプ・ストリング

タイム・スタンプのストリング表記は、数字で始まり、長さが 16 バイト以上のストリングです。 タイム・スタンプの完全なストリング表示は、*yyyy-mm-dd-hh.mm.ss.nnnnnn* という形式です。末尾のブランクを付けることができます。タイム・スタンプの月、日、および時の部分の先行ゼロは省略でき、マイクロ秒は切り捨てたり、完全に省略したりできます。マイクロ秒の部分で後続ゼロが省略されている場合、抜けている数字は 0 が指定されたと見なされます。したがって、 1991-3-2-8.30.00 は 1991-03-02-08.30.00.000000 に等しくなります。

また SQL ステートメントは、タイム・スタンプの ODBC ストリング表示を入力値としてのみサポートします。タイム・スタンプの ODBC ストリング表示の形式は、

yyyy-mm-dd hh:mm:ss.nnnnnn です。

XML 値

XML 値は、XML 文書、XML コンテンツ、または XML ノードのシーケンスの形を取った、整形式 XML を表します。 XML データ・タイプで定義された列の値として表に保管される XML 値は、整形式 XML 文書でなければなりません。 XML 値は、他のどのストリング値にも相当しない内部表記として処理されます。 XMLSERIALIZE 関数を使用して、XML 値を、XML 文書を表すシリアライズ化ストリング値にトランスフォームすることができます。同様に、XMLPARSE 関数を使用して、XML 文書を表すストリング値を XML 値にトランスフォームすることもできます。 XML 値は、アプリケーションのストリングおよびバイナリーのデータ・タイプとの交換時に、暗黙で解析またはシリアライズ化することができます。

結果が XML データ・タイプ値となる式に対しては、特殊な制限が適用されます。そのような式および列は、以下の場所では使用できません (SQLSTATE 42818)。

- DISTINCT 節が先行している SELECT リスト
- GROUP BY 節
- ORDER BY 節
- UNION ALL 以外のセット演算子の副選択
- 基本述部、多値比較述部、BETWEEN 述部、IN 述部、または LIKE 述部
- DISTINCT を指定した集約関数

ユーザー定義タイプ

ユーザー定義のデータ・タイプには次の 4 つのタイプがあります。

- 特殊タイプ
- 構造化タイプ
- 参照タイプ
- 配列タイプ

これらのそれぞれのタイプについて、次の項で説明します。

特殊タイプ

特殊タイプ とは、内部表記を既存のタイプ (その 『ソース』・タイプ) と共用するユーザー定義のデータ・タイプです。しかし、特殊タイプはほとんどの操作で、非互換の別個のタイプと見なされます。例えば、ピクチャー・タイプ、テキスト・タイプ、音声タイプを定義しようとする場合、これらのタイプのセマンティクスはどれも異なりますが、内部表記としては組み込みデータ・タイプ BLOB を使用します。

次に、AUDIO という名前の特殊タイプを作成する例を示します。

```
CREATE TYPE AUDIO AS BLOB (1M)
```

AUDIO は組み込みデータ・タイプの BLOB と内部表記は同じですが、別個のタイプと見なされます。これにより、AUDIO 用に特別に関数を設定できるようになり、そのような関数は他のどのデータ・タイプ (ピクチャー、テキストなど) の値にも決して適用されないようになります。

特殊タイプは、修飾子付き ID を持っています。 CREATE TYPE (Distinct)、DROP、または COMMENT ステートメント以外で特殊タイプ名が使用されるとき、スキーマ名によってそれが修飾されていない場合は、SQL パスを順に調べて、特殊タイプの一致する最初のスキーマが探索されます。

特殊タイプを使うと、そのインスタンスに対しては、明示的に特殊タイプに基づいて定義された関数や演算子しか適用されないようになるため、強力なタイプ識別機能が実現されます。そのため、特殊タイプはそのソース・タイプの関数や演算子を自動的には獲得しません。そのようなものは無意味である可能性があるためです。(例えば、AUDIO タイプの LENGTH 関数は、そのオブジェクトの長さをバイト単位ではなく秒単位で戻します。)

LONG VARCHAR、LONG VARGRAPHIC、または LOB の各タイプをソースとする特殊タイプは、ソース・タイプと同じ制限に従います。

しかし、ソース・タイプの特定の関数と演算子が特殊タイプに適用されるように明示的に指定することは可能です。これは、特殊タイプのソース・タイプに対して定義された関数をソースとするユーザー定義関数を定義することによって行うことができます。ソース・タイプとして LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、または DBCLOB を使用しているもの以外のユーザー定義特殊タイプについては、自動的に比較演算子が生成されます。さらに、ソース・タイプから特殊タイプへ、また特殊タイプからソース・タイプへのキャストをサポートする関数も生成されます。

構造化タイプ

構造化タイプ とは、データベースに定義されている構造を持つユーザー定義のデータ・タイプのことです。これには、名前が付けられている一連の*属性* が入っており、それぞれにデータ・タイプがあります。構造化タイプには、一連のメソッド仕様も組み込まれています。

構造化タイプは表、ビュー、または列のタイプとして使用することができます。表またはビューのタイプとして使用する場合、その表またはビューは、それぞれ*型付き表* または*型付きビュー* となります。型付き表および型付きビューの場合、構造化タイプの属性の名前およびデータ・タイプは、型付き表または型付きビューの列の名前およびデータ・タイプになります。型付き表または型付きビューの行は、構造化タイプのインスタンスの表示と考えることができます。列のデータ・タイプとして使用する場合、その列には該当構造化タイプの値 (または、下記のように、そのタイプにサブタイプがあれば、その値) が入ります。構造列オブジェクトの属性を取り出して処理するときには、メソッドを使います。

用語: *スーパータイプ* とは、*サブタイプ* という、他の構造化タイプが定義されている構造化タイプのことです。サブタイプはスーパータイプのすべての属性およびメソッドを継承し、さらに属性およびメソッドを定義することもできます。共通のスーパータイプに関連する構造化タイプのセットは*タイプ階層* と呼ばれ、それより上位のスーパータイプを持たないタイプをそのタイプ階層の*ルート・タイプ* と呼びます。

サブタイプという用語は、タイプ階層において 1 つのユーザー定義の構造化タイプおよびその下にあるすべてのユーザー定義の構造化タイプを指して用いられます。したがって、階層内における構造化タイプ **T** のサブタイプは、 **T** と、**T** の下にあるすべての構造化タイプになります。構造化タイプ **T** の*厳密な意味でのサブタイプ*とは、タイプ階層で **T** の下にある構造化タイプのことです。

タイプ階層内での再帰的タイプ定義に対しては、いくつかの制限があります。このため、許可されている特定タイプの再帰的定義を参照するために、簡単な方法を考える必要があります。以下の定義が使われます。

- *直接的な使用*: 以下のいずれか 1 つが当てはまる場合のみ、タイプ **A** は、別のタイプ **B** を直接使用します。
  1. タイプ **A** に、タイプ **B** の属性がある場合
  2. タイプ **B** が、**A** のサブタイプ、または **A** のスーパータイプである場合
- *間接的な使用*: 以下のいずれかが当てはまる場合、タイプ **A** は、タイプ **B** を間接的に使用します。
  1. タイプ **A** がタイプ **B** を直接に使う場合
  2. タイプ **A** が何らかのタイプ **C** を直接に使用し、タイプ **C** がタイプ **B** を間接的に使う場合

いずれかの属性タイプがそれ自体を直接的または間接的に使用するように、タイプを定義することはできません。そのような構成を作成する必要がある場合、参照を属性として使うことを考慮してください。例えば、構造化タイプ属性では、「管理

職」が属性タイプ「従業員」である場合に、「管理職」の属性を持つ「従業員」のインスタンスというものはあり得ません。しかし、REF (従業員) のタイプを持つ「管理職」の属性はあり得ます。

他の特定のオブジェクトが、あるタイプを直接または間接的に使っている場合、そのタイプをドロップすることはできません。例えば、表またはビューの列が、タイプを直接または間接的に使っている場合、タイプをドロップすることはできません。

参照タイプ

参照タイプ は構造化タイプと対になっているタイプです。特殊タイプに似て、参照タイプは組み込みデータ・タイプの 1 つと共通の表記を使用するスカラー・タイプです。この同じ表記はタイプ階層のすべてのタイプで共用されます。参照型付き表記は、タイプ階層のルート・タイプの作成時に定義されます。参照タイプを使用する場合、構造化タイプはタイプのパラメーターとして指定されます。このパラメーターを、参照の*ターゲット・タイプ* といいます。

参照のターゲットは、通常、型付き表または型付きビューの行です。参照タイプを使用する場合、*有効範囲* を定義することができます。有効範囲は、参照値のターゲット行がある表 (*ターゲット表* と呼ばれる) またはビュー (*ターゲット・ビュー* と呼ばれる) を指定します。ターゲット表またはターゲット・ビューは、参照タイプのターゲット・タイプと同じタイプでなければなりません。効力範囲を持つ参照タイプのインスタンスは、型付き表または型付きビューの行 (*ターゲット行* と呼ばれる) を固有識別します。

配列タイプ

*配列*とは、データ・エレメントの順序付きコレクションがあり、各エレメントはコレクションにおけるその順序位置で参照できるようになっている構造のことをいいます。 *N* が配列におけるカーディナリティー (エレメント数) である場合、各エレメントに関連付けられた順序位置は、1 以上 *N* 以下の整数値です。配列内のすべてのエレメントのデータ・タイプは同じです。

*配列タイプ*とは、別のデータ・タイプの配列として定義されているデータ・タイプのことをいいます。すべての配列タイプには最大カーディナリティーがあり、これは CREATE TYPE ステートメントで指定されます。 A が最大カーディナリティー *M* を持つ配列タイプである場合、タイプ A の値のカーディナリティーは 0 から *M* まで (両端を含む) の値となります。 C などのプログラミング言語における配列の最大カーディナリティーとは異なり、SQL 配列の最大カーディナリティーはそれらの物理表現とは関連していません。むしろ、最大カーディナリティーは、添え字が境界内にあることを確実にするため、実行時にシステムにより使用されます。配列の値を示すために必要なメモリー量は、通常、そのタイプの最大カーディナリティーにではなく、そのカーディナリティーに比例します。

配列が参照されている時、配列のすべての値はメイン・メモリーに格納されます。そのため、大量のデータを含む配列は、大量のメイン・メモリーを消費します。

データ・タイプのプロモーション

データ・タイプは、関連するいくつかのデータ・タイプからなるグループに分類されます。そのようなグループの中では、あるデータ・タイプを他のデータ・タイプより優先すると見なす優先順位が存在します。この優先順位を使用すると、あるデータ・タイプを、優先順位がそれより上のデータ・タイプにプロモートすること (*プロモーション*) が可能になります。たとえば、CHAR データ・タイプは VARCHAR に、 INTEGER は DOUBLE-PRECISION にプロモートできますが、CLOB を VARCHAR にプロモートすることはできません。

データ・タイプのプロモーションは、以下の場合に使用されます。

- 関数解決を実行する場合
- ユーザー定義タイプをキャストする場合
- ユーザー定義タイプを組み込みデータ・タイプに割り当てる場合

次の表 7 に、各データ・タイプの優先順位順のリストを示します。特定のデータ・タイプのプロモート先として可能なデータ・タイプを調べたいとき、この表を使うことができます。この表に示されているとおり、最適の選択は、別のデータ・タイプへプロモートすることではなく、常に同じデータ・タイプです。

表 7. データ・タイプの優先順位表

データ・タイプ	データ・タイプ優先順位リスト (高いものから順に)
CHAR	CHAR、VARCHAR、LONG VARCHAR、CLOB
VARCHAR	VARCHAR、LONG VARCHAR、CLOB
LONG VARCHAR	LONG VARCHAR、CLOB
GRAPHIC	GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、DBCLOB
VARGRAPHIC	VARGRAPHIC、LONG VARGRAPHIC、DBCLOB
LONG VARGRAPHIC	LONG VARGRAPHIC、DBCLOB
BLOB	BLOB
CLOB	CLOB
DBCLOB	DBCLOB
SMALLINT	SMALLINT、INTEGER、BIGINT、decimal、real、double、 DECFLOAT
INTEGER	INTEGER、BIGINT、decimal、real、double、DECFLOAT
BIGINT	BIGINT、decimal、real、double、DECFLOAT
10 進	decimal、real、double、DECFLOAT
実数	real、double、DECFLOAT
倍精度	double、DECFLOAT
DECFLOAT	DECFLOAT
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
udt	udt (同じ名前) または udt のスーパータイプ
REF(T)	REF(S) (S が T のスーパータイプの場合)

表 7. データ・タイプの優先順位表 (続き)

データ・タイプ	データ・タイプ優先順位リスト (高いものから順に)

注:

1. 上記の小文字で示したタイプは、以下のように定義されます。
   - decimal = DECIMAL(p,s) または NUMERIC(p,s)
   - real = REAL または FLOAT(*n*)。ここで、*n* は 24 を超えない値。
   - double = DOUBLE、DOUBLE-PRECISION、FLOAT、または FLOAT(*n*)。ここで、*n* は 25 以上。
   - udt = ユーザー定義タイプ

   リストの中のデータ・タイプの短形式および長形式の同義語は、リストの中の同義語と同じであると見なされます。
2. Unicode データベースの場合、以下は、等価のデータ・タイプと見なされます。
   - CHAR および GRAPHIC
   - VARCHAR および VARGRAPHIC
   - LONG VARCHAR および LONG VARGRAPHIC
   - CLOB および DBCLOB

   Unicode データベース内で関数を解決する際、任意の関数呼び出しにおいて、ユーザー定義関数および組み込み関数の両方が該当する場合、組み込み関数の方が通常呼び出されます。ユーザー定義関数が呼び出されるのは、そのスキーマが CURRENT PATH 特殊レジスター内で SYSIBM に先行し、Unicode のデータ・タイプが等価であるかどうかにかかわらず、その引数データ・タイプが、関数呼び出しの引数データ・タイプすべてと一致する場合に限られます。
3. データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

データ・タイプ間のキャスト

特定のデータ・タイプの値を別のデータ・タイプへキャストする必要や、データ・タイプは同じでも長さ、精度、または位取りの異なるデータ・タイプへキャスト する必要が生じることがよくあります。データ・タイプのプロモーションは、あるデータ・タイプから別のデータ・タイプへのプロモーションにおいて、値を新しいデータ・タイプへキャストすることが必要になる 1 つの例です。別のデータ・タイプへキャストできるデータ・タイプは、ソース・データ・タイプから宛先データ・タイプへキャスト*可能* であるといいます。

あるデータ・タイプから別のデータ・タイプへのキャストは、暗黙的に行われることもあれば、明示的に行うこともできます。関係するデータ・タイプによっては、cast 関数、CAST 仕様、または XMLCAST 仕様を使用して、データ・タイプを明示的に変更することができます。データベース・マネージャーは、特殊タイプが関係する割り当て中に、データ・タイプを暗黙的にキャストすることがあります。さらに、ソース関数から派生するユーザー定義関数を作成するときは、ソース関数のパラメーターのデータ・タイプが、作成しようとしている関数のデータ・タイプにキャスト可能でなければなりません。

組み込みデータ・タイプの間でサポートされているキャストを、 108 ページの表 8 に示します。第 1 列がキャスト・オペランドのデータ・タイプ (ソース・データ・タイプ) を表し、上部に横方向に示したデータ・タイプがキャスト操作の目的データ・タイプを表します。 Y は、ソースとターゲットのデータ・タイプの組み合わせに対して CAST 仕様を使用できることを示します。 XMLCAST 仕様のみを使用できるケースでは、その旨注記されています。

Unicode データベースでは、文字または GRAPHIC ストリングが別のデータ・タイプにキャストされるときに切り捨てが行われる場合、非ブランク文字が切り捨てられる場合に警告が戻されます。この切り捨て動作は、非ブランク文字が切り捨てられる場合にエラーが起こるときの、ターゲットへの文字または GRAPHIC ストリングの割り当てとは異なります。

特殊タイプに関する以下のキャストがサポートされています。(他に注意書きがなければ、CAST 仕様を使用しています。)

- 特殊タイプ *DT* から、そのソース・データ・タイプ *S* へのキャスト
- 特殊タイプ *DT* のソース・データ・タイプ *S* から、特殊タイプ *DT* へのキャスト
- 特殊タイプ *DT* から、それと同じ特殊タイプ *DT* へのキャスト
- データ・タイプ *A* から、特殊タイプ *DT* へのキャスト。ただし、 *A* は特殊タイプ *DT* のソース・データ・タイプ *S* へプロモート可能なもの
- INTEGER から、ソース・データ・タイプが SMALLINT である特殊タイプ *DT* へのキャスト
- DOUBLE から、ソース・データ・タイプが REAL である特殊タイプ *DT* へのキャスト
- DECFLOAT から、ソース・データ・タイプが CHAR である特殊タイプ DT へのキャスト

- VARCHAR から、ソース・データ・タイプが CHAR である特殊タイプ *DT* へのキャスト
- VARGRAPHIC から、ソース・データ・タイプが GRAPHIC である特殊タイプ *DT* へのキャスト
- Unicode データベースの場合、VARCHAR または VARGRAPHIC から、ソース・データ・タイプが CHAR または GRAPHIC である特殊タイプ *DT* へのキャスト
- ソース・データ・タイプが *S* である特殊タイプ *DT* から XML への、XMLCAST 仕様を使用したキャスト
- XML から、任意の組み込みデータ・タイプのソース・データ・タイプをもった特殊タイプ *DT* への、XMLCAST 仕様を使用したキャスト (XML 値の XML スキーマ・データ・タイプによる)

FOR BIT DATA 文字タイプを CLOB にキャストすることはできません。

構造化タイプの値を何か別のものにキャストすることはできません。 *ST* のスーパータイプに対するすべてのメソッドは、*ST* に当てはまるので、構造化タイプ *ST* を、そのスーパータイプのいずれかにキャストすべきではありません。必要な操作が *ST* のサブタイプだけに当てはまる場合、サブタイプ処理式を使用して、*ST* をサブタイプの 1 つとして扱います。

キャストに関与したユーザー定義データ・タイプがスキーマ名によって修飾されていない場合、 *SQL パス* が、ユーザー定義データ・タイプを組み入れられた最初のスキーマをその名前で検出するために使用されます。

参照タイプに関して、以下のキャストがサポートされています。

- 参照タイプ *RT* から、表記データ・タイプ *S* へのキャスト
- 参照タイプ *RT* の表記データ・タイプ *S* から、参照タイプ *RT* へのキャスト
- ターゲット・タイプが *T* である参照タイプ *RT* から、ターゲット・タイプが *S* である参照タイプ *RS* へのキャスト (*S* は *T* のスーパータイプ)
- データ・タイプ *A* から、参照タイプ *RT* へのキャスト (ただし *A* は、参照タイプ *RT* の表記データ・タイプ *S* へプロモート可能なもの)

キャストに関与した参照データ・タイプのターゲット・タイプがスキーマ名によって修飾されていない場合、 *SQL パス* が、ユーザー定義データ・タイプを組み入れられた最初のスキーマをその名前で検出するために使用されます。

データ・タイプ間のキャスト

表 8. 組み込みデータ・タイプ間のサポートされるキャスト

	ターゲット・データ・タイプ																			
ソース・データ・タイプ	SMALLINT	INTEGER	BIGINT	DECIMAL	REAL	DOUBLE	DECFLOAT	CHAR	CHAR FOR BIT DATA	VARCHAR	VARCHAR FOR BIT DATA	CLOB	GRAPHIC	VARGRAPHIC	DBCLOB	BLOB	DATE	TIME	TIMESTAMP	XML
SMALLINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	Y[3]
INTEGER	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	Y[3]
BIGINT	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	Y[3]
DECIMAL	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	Y[3]
REAL	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	Y[3]
DOUBLE	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	Y[3]
DECFLOAT	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-	-	-	-	-	-	-	-	-
CHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y	Y	Y	Y	Y[4]
CHAR FOR BIT DATA	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	Y[3]
VARCHAR	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y	Y	Y	Y	Y[4]
VARCHAR FOR BIT DATA	Y	Y	Y	Y	-	-	Y	Y	Y	Y	Y	-	-	-	-	Y	Y	Y	Y	Y[3]
CLOB	-	-	-	-	-	-	-	Y	-	Y	-	Y	Y[1]	Y[1]	Y[1]	Y	-	-	-	Y[4]
GRAPHIC	Y[1]	Y[1]	Y[1]	Y[1]	-	-	Y[1]	Y[1]	-	Y[1]	-	Y[1]	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y[3]
VARGRAPHIC	Y[1]	Y[1]	Y[1]	Y[1]	-	-	Y[1]	Y[1]	-	Y[1]	-	Y[1]	Y	Y	Y	Y	Y[1]	Y[1]	Y[1]	Y[3]
DBCLOB	-	-	-	-	-	-	-	Y[1]	-	Y[1]	-	Y[1]	Y	Y	Y	Y	-	-	-	Y[3]
BLOB	-	-	-	-	-	-	-	-	-	Y	Y	-	-	-	-	Y	-	-	-	Y[4]
DATE	-	Y	Y	Y	-	-	-	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	Y	-	-	Y[3]
TIME	-	Y	Y	Y	-	-	-	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	-	Y	-	Y[3]
TIMESTAMP	-	-	Y	Y	-	-	-	Y	Y	Y	Y	-	Y[1]	Y[1]	-	-	Y	Y	Y	Y[3]
XML	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y[5]	Y

表 8. 組み込みデータ・タイプ間のサポートされるキャスト (続き)

	ターゲット・データ・タイプ																			
ソース・データ・タイプ	SMALLINT	INTEGER	BIGINT	DECIMAL	NUMERIC	DOUBLE	REAL	DECFLOAT	CHAR	VARCHAR	LONG VARCHAR	CLOB	GRAPHIC	VARGRAPHIC	LONG VARGRAPHIC	DBCLOB	BLOB	DATE	TIME	TIMESTAMP	XML
SMALLINT																					
INTEGER																					
BIGINT																					
DECIMAL																					
NUMERIC																					
DOUBLE																					
REAL																					
DECFLOAT																					
CHAR																					
VARCHAR																					
LONG VARCHAR																					
CLOB																					
GRAPHIC																					
VARGRAPHIC																					
LONG VARGRAPHIC																					
DBCLOB																					
BLOB																					
DATE																					
TIME																					
TIMESTAMP																					
XML																					

注

- ユーザー定義タイプおよび参照タイプに関してサポートされているキャストについては、この表の前にある説明を参照してください。
- 構造化タイプの値を何か別のものにキャストすることはできません。
- データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

[1] キャストは、Unicode データベースの場合にのみサポートされます。

[2] FOR BIT DATA

[3] キャストは XMLCAST を使用しないと実行できません。

[4] ストリングを XML 列に割り当てる (INSERT または UPDATE) ときに、XMLPARSE 関数が暗黙的に処理されて、ストリングを XML に変換します。割り当てを正常に完了するには、そのストリングが整形式 XML 文書でなければなりません。

[5] キャストは XMLCAST を使用しないと実行できず、XML 値の基礎となる XML スキーマ・データ・タイプに依存します。詳細は、『XMLCAST』の項を参照してください。

表 9 は、識別されたターゲット・データ・タイプへキャストするときに適用する規則に関する情報を見つける場所を示しています。

表 9. データ・タイプへのキャストに関する規則

ターゲット・データ・タイプ	規則
SMALLINT	*SQL リファレンス 第 1 巻* の『SMALLINT スカラー関数』
INTEGER	*SQL リファレンス 第 1 巻* の『INTEGER スカラー関数』
BIGINT	*SQL リファレンス 第 1 巻* の『BIGINT スカラー関数』
DECIMAL	*SQL リファレンス 第 1 巻* の『DECIMAL スカラー関数』
NUMERIC	*SQL リファレンス 第 1 巻* の『NUMERIC スカラー関数』

表 9. データ・タイプへのキャストに関する規則 (続き)

ターゲット・データ・タイプ	規則
REAL	*SQL リファレンス 第 1 巻* の『REAL スカラー関数』
DOUBLE	*SQL リファレンス 第 1 巻* の『DOUBLE スカラー関数』
DECFLOAT	*SQL リファレンス 第 1 巻* の『DECFLOAT スカラー関数』
CHAR	*SQL リファレンス 第 1 巻* の『CHAR スカラー関数』
VARCHAR	*SQL リファレンス 第 1 巻* の『VARCHAR スカラー関数』
CLOB	*SQL リファレンス 第 1 巻* の『CLOB スカラー関数』
GRAPHIC	*SQL リファレンス 第 1 巻* の『GRAPHIC スカラー関数』
VARGRAPHIC	*SQL リファレンス 第 1 巻* の『VARGRAPHIC スカラー関数』
DBCLOB	*SQL リファレンス 第 1 巻* の『DBCLOB スカラー関数』
BLOB	*SQL リファレンス 第 1 巻* の『BLOB スカラー関数』
DATE	*SQL リファレンス 第 1 巻* の『DATE スカラー関数』
TIME	*SQL リファレンス 第 1 巻* の『TIME スカラー関数』
TIMESTAMP	ソース・タイプが文字ストリングの場合、*SQL リファレンス 第 1 巻* の『TIMESTAMP スカラー関数』を参照してください。そこでは、1 つのオペランドが指定されています。ソース・データ・タイプが DATE の場合、タイム・スタンプは指定された日付と時刻 00:00:00 から構成されます。ソース・データ・タイプが TIME の場合、タイム・スタンプは CURRENT DATE および指定された時刻から構成されます。

XML 以外の値から XML 値へのキャスト

表 10. XML 以外の値から XML 値への、サポートされているキャスト

	ターゲット・データ・タイプ	
ソース・データ・タイプ	XML	結果の XML スキーマ型
SMALLINT	Y	xs:short
INTEGER	Y	xs:int
BIGINT	Y	xs:long
DECIMAL または NUMERIC	Y	xs:decimal

表 10. XML 以外の値から XML 値への、サポートされているキャスト (続き)

	ターゲット・データ・タイプ	
ソース・データ・タイプ	**XML**	**結果の XML スキーマ型**
REAL	Y	xs:float
DOUBLE	Y	xs:double
DECFLOAT	N	-
CHAR	Y	xs:string
VARCHAR	Y	xs:string
CLOB	Y	xs:string
GRAPHIC	Y	xs:string
VARGRAPHIC	Y	xs:string
DBCLOB	Y	xs:string
DATE	Y	xs:date
TIME	Y	xs:time
TIMESTAMP	Y	xs:dateTime
BLOB	Y	xs:base64Binary
文字タイプ FOR BIT DATA	Y	xs:base64Binary
特殊タイプ		この図表は、特殊タイプのソース・タイプで使用してください。

注:

- データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

文字ストリング値を XML 値にキャストする場合、その結果の xs:string 原子値に、不正な XML 文字が入っていてはなりません (SQLSTATE 0N002)。入力文字ストリングが Unicode でない場合、入力文字は Unicode に変換されます。

SQL バイナリー形式へキャストすると、その結果は、タイプが xs:base64Binary の XQuery 原子値になります。

XML 値から XML 以外の値へのキャスト

XML 値から XML 以外の値への XMLCAST は、2 つのキャストに分かれます。つまり、ソースの XML 値を、SQL ターゲット・タイプに対応する XQuery タイプに変換する XQuery キャストと、その後に続く、対応する XQuery タイプから実際の SQL タイプへのキャストです。

XMLCAST がサポートされるのは、ターゲット・タイプに対応する、サポートされた XQuery ターゲット・タイプがあり、かつソース値のタイプから対応する XQuery ターゲット・タイプへの、サポートされた XQuery キャストがある場合です。XQuery キャストで使用されるターゲット・タイプは、対応する XQuery ターゲット・タイプを基にしたものであり、さらに別の制約事項を伴う場合があります。

以下の表は、そのような変換の結果の XQuery タイプを一覧で示しています。

表 11. XML 値から XML 以外の値への、サポートされているキャスト

	ソース・データ・タイプ	
ターゲット・データ・タイプ	**XML**	**対応する XQuery ターゲット・タイプ**
SMALLINT	Y	xs:short
INTEGER	Y	xs:int
BIGINT	Y	xs:long
DECIMAL または NUMERIC	Y	xs:decimal
REAL	Y	xs:float
DOUBLE	Y	xs:double
DECFLOAT	Y	一致するタイプがありません[1]
CHAR	Y	xs:string
VARCHAR	Y	xs:string
CLOB	Y	xs:string
GRAPHIC	Y	xs:string
VARGRAPHIC	Y	xs:string
DBCLOB	Y	xs:string
DATE	Y	xs:date
TIME (時間帯なし)	Y	xs:time
TIMESTAMP (時間帯なし)	Y	xs:dateTime
BLOB	Y	xs:base64Binary
CHAR FOR BIT DATA	N	キャスト不能
VARCHAR FOR BIT DATA	Y	xs:base64Binary
特殊タイプ		この図表は、特殊タイプのソース・タイプで使用してください。
行、参照、構造化されたデータ・タイプまたは抽象データ・タイプ (ADT)、その他	N	キャスト不能

注

[1] DB2 は XML スキーマ 1.0 をサポートしますが、これは DECFLOAT に一致する XML スキーマ・タイプを提供していません。XMLCAST の XQuery キャストの手順の処理は、以下のように処理されます。

- ソース値が XML スキーマの数値タイプで入力される場合、その数値タイプを使用します。
- ソース値が XML スキーマ・タイプ xs:boolean で入力される場合、xs:double を使用します。
- それ以外の場合、有効な数値形式の追加検査をして、xs:string を使用します。

以下の制約の場合、制約から派生する XML スキーマ・データ・タイプが、XQuery キャストのターゲット・データ・タイプとして効果的に使用されます。

- ストリング・タイプに変換される XML 値は、文字またはバイトの切り捨てなしに、DB2 の該当タイプの長さ制限に収まらなければなりません。派生する XML スキーマ・タイプに使用される名前は、大文字の SQL タイプ名の後に、下線文

字とストリングの最大長が続いたものになります。例えば、XMLCAST ターゲット・データ・タイプが VARCHAR(20) の場合は VARCHAR_20 となります。

- DECIMAL 値に変換される XML 値は、指定された DECIMAL 値の精度内に収まらなければならず、小数点の後に位取りより多い非ゼロ数字が付いていてはなりません。派生する XML スキーマ・タイプに使用される名前は、DECIMAL_*precision_scale* となります。ただし、*precision* は、ターゲットの SQL データ・タイプの精度であり、*scale* は、ターゲットの SQL データ・タイプの位取りです。例えば、XMLCAST ターゲット・データ・タイプが DECIMAL(9,2) の場合は、DECIMAL_9_2 となります。
- TIME 値に変換される XML 値内には、小数点以降にゼロ以外の数字をもった秒コンポーネントを置くことはできません。派生する XML スキーマ・タイプに使用される名前は、TIME です。

派生した XML スキーマ・タイプ名がメッセージ中に現れるのは、XML 値が、制約事項のいずれかに合致しない場合だけです。このタイプ名はエラー・メッセージの理解に役立ちますが、定義済みのどの XQuery タイプにも対応しません。入力値が、派生した XML スキーマ・タイプ (対応する XQuery ターゲット・タイプ) の基本タイプに準拠しない場合、エラー・メッセージには、そのタイプが代わりに示されることがあります。このような、派生した XML スキーマ・タイプ名のフォーマットは、将来変更される可能性があるので、プログラミング・インターフェースとして使用しないでください。

XQuery キャストでの XML 値の処理の前に、シーケンス中のすべての文書ノードは除去され、除去された文書ノードの直接の子はそれぞれ、そのシーケンス中の項目になります。文書ノードが複数の直接下位ノードをもっていた場合、改訂後のシーケンスの項目数は、元のシーケンスより多くなります。次に、XQuery fn:data 関数を使用して、文書ノードのない XML 値が原子化されます。その結果として生じる原子化シーケンス値は XQuery キャストで使用されます。原子化シーケンス値が空のシーケンスである場合、それ以上の処理を行うことなく、キャストから NULL 値が戻されます。原子化シーケンス値に複数の項目があると、エラーが戻されます (SQLSTATE 10507)。

XMLCAST のターゲット・タイプが SQL データ・タイプの DATE、TIME、または TIMESTAMP である場合、XQuery キャストの結果の XML 値も UTC に調整され、その値の時間帯コンポーネントは除去されます。

対応する XQuery ターゲット・タイプ値から SQL ターゲット・タイプへの変換時には、xs:base64Binary や xs:hexBinary などのバイナリーの XML データ・タイプは、文字フォームから実際のバイナリー・データに変換されます。

INF、-INF、または NaN の xs:double または xs:float 値を SQL データ・タイプ DOUBLE または REAL 値にキャストする (XMLCAST を使用して) と、エラーが戻されます (SQLSTATE 22003)。 -0 の xs:double または xs:float 値は、+0 に変換されます。

ソース・オペランドがユーザー定義特殊タイプでない場合、ターゲット・タイプはユーザー定義特殊タイプであってもかまいません。そのような場合、XMLCAST 仕

様を使用してソース値がユーザー定義特殊タイプ (つまり、ターゲット・タイプ) のソース・タイプにキャストされた後、CAST 仕様を使用してこの値がユーザー定義特殊タイプにキャストされます。

非 Unicode データベースでは、XML 値から XML 以外のターゲット・タイプへのキャストに、内部の UTF-8 形式からデータベース・コード・ページへのコード・ページ変換が含まれます。この変換は、XML 値のコード・ポイントがデータベース・コード・ページに存在しない場合に、置換文字を導入する結果になります。

代入と比較

SQL の基本的な演算は、割り当てと比較です。割り当て演算は、INSERT、UPDATE、FETCH、SELECT INTO、VALUES INTO および SET 遷移変数ステートメントの実行時に行われます。関数の引数も、関数の呼び出し時に割り当てられます。比較演算は、MAX、MIN、DISTINCT、GROUP BY、ORDER BY のような述部およびその他の言語エレメントを組み入れられたステートメントの実行時に行われます。

両方の演算に適用される 1 つの基本的な規則は、関係するオペランドのデータ・タイプは互換でなければならないということです。この互換性規則はセット演算にも適用されます。

割り当て操作の別の基本的な規則は、 NULL 値を入れることができない列や、関連する標識変数がないホスト変数に、 NULL 値を割り当てることができないという規則です。

文字と GRAPHIC データの両方に関与した割り当ておよび比較は、ストリングの 1 つがリテラルであるときにのみサポートされます。

以下は、割り当ておよび比較演算のためのデータ・タイプの互換性を示す互換性マトリックスです。

表 12. 割り当てと比較におけるデータ・タイプの互換性

オペランド	2 進整数	10 進数	浮動小数点	10 進浮動小数点数	文字ストリング	GRAPHIC ストリング	日付	時刻	TIMESTAMP	バイナリー・ストリング	UDT
2 進整数	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[2]
10 進数	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[2]
浮動小数点	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[2]
10 進浮動小数点数	はい	はい	はい	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[2]
文字ストリング	いいえ	いいえ	いいえ	いいえ	はい	はい[5、6]	[1]	[1]	[1]	いいえ[3]	[2]
GRAPHIC ストリング	いいえ	いいえ	いいえ	いいえ	はい[5、6]	はい	[1]	[1]	[1]	いいえ	[2]
日付	いいえ	いいえ	いいえ	いいえ	[1]	[1]	はい	いいえ	いいえ	いいえ	[2]
時刻	いいえ	いいえ	いいえ	いいえ	[1]	[1]	いいえ	はい	いいえ	いいえ	[2]
タイム・スタンプ	いいえ	いいえ	いいえ	いいえ	[1]	[1]	いいえ	いいえ	はい	いいえ	[2]
バイナリー・ストリング	いいえ	いいえ	いいえ	いいえ	なし[3]	いいえ	いいえ	いいえ	いいえ	はい	[2]
UDT	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	[2]	はい

表 12. 割り当てと比較におけるデータ・タイプの互換性 (続き)

オペランド	2 進整数	10 進数	浮動小数点	10 進浮動小数点数	文字ストリング	GRAPHIC ストリング	日付	時刻	TIMESTAMP	バイナリー・ストリング	UDT

[1] DATETIME とストリングの互換性は、割り当てと比較に限定されます。

- DATETIME 値は、ストリング列とストリング変数に割り当てることができます。
- 日付の有効なストリング表記は、DATE 列に割り当てるか、または DATE と比較できます。
- 時刻の有効なストリング表記は、TIME 列に割り当てるか、または TIME と比較できます。
- タイム・スタンプの有効なストリング表記は、 TIMESTAMP 列に割り当てるか、または TIMESTAMP と比較できます。

(GRAPHIC ストリング・サポートは、Unicode データベースの場合にのみ使用可能です。)

[2] ユーザー定義特殊タイプの値は、同じユーザー定義特殊タイプで定義された値とのみ比較できます。一般に、特殊タイプの値とそのソース・データ・タイプとの間では割り当てがサポートされます。ユーザー定義構造化タイプは比較することができません。また、同じ構造化タイプまたはそのスーパータイプのいずれかのオペランドにのみ、割り当てることができます。さらに詳しい情報については、 122 ページの『ユーザー定義タイプの割り当て』を参照してください。

[3] これは、 FOR BIT DATA 属性で定義された文字ストリングもバイナリー・ストリングと互換性がないことを意味します。

[4] 参照タイプの割り当ておよび比較については、 123 ページの『参照タイプの割り当て』 および 129 ページの『参照タイプの比較』 を参照してください。

[5] Unicode データベースの場合にのみサポートされます。

[6] ビット・データと GRAPHIC ストリングは互換性がありません。

数値割り当て

数値割り当てでは、オーバーフローは許可されません。

- 厳密な数値データ・タイプへの割り当てでは、数値の整数部分のいずれかの桁が除去されると、オーバーフローが発生します。必要な場合には、数値の小数部分が切り捨てられます。
- 近似値データ・タイプまたは 10 進浮動小数点数への割り当てでは、数値の整数部分の最上位桁が除去されると、オーバーフローが発生します。浮動小数点数および 10 進浮動小数点数では、数値の整数部分は、浮動小数点数または 10 進浮動小数点数が無制限の精度を持つ 10 進数に変換された場合にその結果となる数値です。必要な場合には、丸めによって数値の最下位桁が除去される場合があります。

  10 進浮動小数点数では、数値の整数部分の切り捨ては許可されておらず、これを行うとエラーになります。

  浮動小数点数では、アンダーフローも許可されていません。アンダーフローは、1 と -1 の間の数値において、ゼロ以外の最上位桁が除去される場合に発生します。 10 進浮動小数点数では、アンダーフローが許可されていて、丸めモードに応じて、ゼロ、または最小の正数または最大の負の数値 (警告付きで表示可能) となります。

  標識変数を持つホスト変数への割り当てでオーバーフローまたはアンダーフローが発生する場合には、エラーの代わりにオーバーフローまたはアンダーフローの警告が返されます。この場合、数値はホスト変数に割り当てられず、標識変数はマイナス 2 に設定されます。

10 進浮動小数点数では、実質上、CURRENT DECFLOAT ROUNDING MODE 特殊レジスターが丸めモードを示します。

整数への割り当て

10 進数、浮動小数点数、または 10 進浮動小数点数が、整数の列または整数変数に割り当てられた場合、その数の小数部分が除去されます。結果として、1 と -1 の間の数値は 0 になります。

10 進数への割り当て

整数を 10 進数の列または変数に割り当てるときは、まず数値が一時的な 10 進数へ変換された後、必要であれば、ターゲットの精度と位取りに変換されます。一時的な 10 進数の精度と位取りは、短精度整数では 5,0、長精度整数では 11,0、 64 ビット整数では 19,0 です。

10 進数が 10 進数の列または変数に割り当てられる場合、その数値は、必要に応じて宛先の精度および位取りに変換されます。必要な数だけ先行ゼロが追加されます。また、10 進数の小数部分では、必要な数の後続ゼロが追加されるか、または必要な数の末尾桁が除去されます。

浮動小数点数を 10 進数の列または変数に割り当てるときは、まず数値が精度 31 の一時的な 10 進数へ変換された後、必要なら、ターゲットの精度と位取りまで切り捨てられます。この変換では、数値は精度 31 の 10 進数に (浮動小数点数算術演算を使って) 丸められます。その結果、1 と -1 の間で、10 進数の列または変数で示すことができる最小の正数未満の数値または最大の負の数値より大きい数値は 0 になります。位取りは、有効数字を消失させることなく数値の整数部分を表現できるような最大可能値になります。

10 進浮動小数点数が 10 進数の列または変数に割り当てられる場合、その数値は、その 10 進数の列または変数の精度および位取りに丸められます。その結果、1 と -1 の間で、10 進数の列または変数で示すことができる最小の正数未満の数値または最大の負の数値より大きい数値は 0 になるか、10 進数の列または変数で示すことができる最小の正の値または最大の負の値に丸められます (丸めモードにより異なる)。

浮動小数点数への割り当て

浮動小数点数は、実数の近似値です。したがって、整数、10 進数、浮動小数点数、または 10 進浮動小数点数が浮動小数点数の列または変数に割り当てられた場合、その結果が元の数値と異なってくる可能性があります。数値は、浮動小数点算術計算を使用して浮動小数点数の列または変数の精度に丸められます。 10 進浮動小数点値は、まずストリング表記に変換され、その後浮動小数点数に変換されます。

10 進浮動小数点数への割り当て

整数を 10 進浮動小数点数の列または変数に割り当てるときは、まず数値が一時的な 10 進数へ変換された後、10 進浮動小数点数に変換されます。一時的な 10 進数の精度と位取りは、短精度整数では 5,0、長精度整数では 11,0、 64 ビット整数では 19,0 です。 BIGINT を DECFLOAT(16) の列または変数に割り当てるときに丸めが行われる場合があります。

10 進数が 10 進浮動小数点数の列または変数に割り当てられるとき、その数値は、ターゲットの精度 (16 または 34) に変換されます。先行ゼロは除去されます。 10 進数の精度と位取り、およびターゲットの精度に応じて、値が丸められる場合があります。

浮動小数点数を 10 進浮動小数点数の列または変数に割り当てるときは、まず数値が一時的な浮動小数点数のストリング表記へ変換されます。数値のストリング表記は次に、10 進浮動小数点数に変換されます。

DECFLOAT(16) の数値を DECFLOAT(34) の列または変数に割り当てるとき、結果の値は DECFLOAT(16) の数値と同一になります。

DECFLOAT(34) の数値を DECFLOAT(16) の列または変数に割り当てるとき、ソースの指数は結果形式において対応する指数に変換されます。DECFLOAT(34) の数値の小数部は、ターゲットの精度に丸められます。

ストリング割り当て

割り当てには以下の 2 つのタイプがあります。

- *ストレージ割り当て*では、値が割り当てられ、有効なデータの切り捨ては望ましくありません (値を列に割り当てた場合など)。
- *検索割り当て*では、値が割り当てられ、切り捨ては許可されます (データをデータベースから検索する場合など)。

ストリング割り当てに関する規則は、割り当てのタイプによって異なります。

ストレージ割り当て

基本的な規則は、ターゲットに割り当てられるストリングの長さが、ターゲットの長さ属性を超えてはならないということです。ストリングの長さがターゲットの長さ属性を超えた場合は、以下の処置が取られることがあります。

- ストリングは、ターゲットの長さ属性に適合するように、(LOB ストリングを除くすべてのストリング・タイプから) 後続ブランクが切り捨てられた上で割り当てられます。
- 以下の場合にはエラー (SQLSTATE 22001) になります。
  - LOB ストリング以外のストリングからブランク以外の文字が切り捨てられるとき
  - LOB ストリングから任意の文字 (またはバイト) が切り捨てられるとき

ストリングが固定長ターゲットに割り当てられる際に、ストリングの長さがターゲットの長さ属性よりも短い場合、ストリングの右端に必要な数の 1 バイト、2 バイト、または UCS-2 のブランクが埋め込まれます。埋め込み文字は、FOR BIT DATA 属性で定義されている列の場合も含めて、常にブランクです。 (UCS-2 は、いくつかの SPACE 文字を異なるプロパティーで定義します。 Unicode データベースの場合、データベース・マネージャーは、常に、 UCS-2 ブランクとして位置 x'0020' にある ASCII SPACE を使用します。 EUC データベースの場合、位置 x'3000' にある IDEOGRAPHIC SPACE は、埋め込み GRAPHIC ストリングに使用されます。)

検索割り当て

ターゲットに割り当てられるストリングの長さは、ターゲットの長さ属性より長くてもかまいません。ストリングがターゲットに割り当てられるときに、ストリングの長さがターゲットの長さ属性より長ければ、ストリングの右側から文字 (またはバイト) が必要な数だけ切り捨てられます。この場合には警告 (SQLSTATE 01004) が戻され、 SQLCA の SQLWARN1 フィールドに値 'W' が割り当てられます。

さらに、標識変数があってその値のソースが LOB でない場合は、その標識変数はストリングの元の長さに設定されます。

文字ストリングが固定長ターゲットに割り当てられる際に、ストリングの長さがターゲットの長さ属性よりも短い場合、ストリングの右端に必要な数の 1 バイト、2 バイト、または UCS-2 のブランクが埋め込まれます。埋め込み文字は、FOR BIT DATA 属性で定義されているストリングの場合も含めて、常にブランクです。(UCS-2 は、いくつかの SPACE 文字を異なるプロパティーで定義します。 Unicode データベースの場合、データベース・マネージャーは、常に、 UCS-2 ブランクとして位置 x'0020' にある ASCII SPACE を使用します。 EUC データベースの場合、位置 x'3000' にある IDEOGRAPHIC SPACE は、埋め込み GRAPHIC ストリングに使用されます。)

C の NUL 終止符ホスト変数の検索割り当ては、 PREP または BIND コマンドに指定されたオプションに基づいて処理されます。

ストリング割り当てに関する変換規則

列またはホスト変数に割り当てられる文字ストリングまたは GRAPHIC ストリングは、必要であれば、まず割り当て先のコード・ページに変換されます。文字変換が必要になるのは、以下の条件がすべて真の場合だけです。

- コード・ページが異なる。
- ストリングが NULL でも空でもない。
- どちらのストリングのコード・ページ値も 0 (FOR BIT DATA) でない。

Unicode データベースの場合、 GRAPHIC 列に文字ストリングを割り当てることができ、文字カラムに GRAPHIC ストリングを割り当てることができます。

文字ストリング割り当てに関する MBCS の考慮事項

1 バイト文字とマルチバイト文字の両方を入れることのできる文字ストリングを割り当てる場合には、いくつかの考慮事項があります。このような考慮事項は、FOR BIT DATA と定義されているものを含めて、すべての文字ストリングに適用されます。

- ブランクの埋め込みは、常に単一バイトのブランク文字 (X'20') を使用して行われます。
- ブランクの切り捨ては、常に単一バイトのブランク文字 (X'20') に基づいて行われます。切り捨てに関しては、2 バイトのブランク文字はその他の文字と同様に扱われます。
- 文字ストリングをホスト変数に割り当てる場合に、割り当て先のホスト変数にソース・ストリング全体を収めるだけの長さがないなら、 MBCS 文字のフラグメ

ント化が発生します。 MBCS 文字がこのようにフラグメント化される場合は、MBCS 文字フラグメントの各バイトがターゲットで単一バイトのブランク文字 (X'20') に設定されます。それ以外のバイトに関してはソースからの移動は行われず、 SQLWARN1 が 'W' に設定されて切り捨ての発生を示します。 MBCS 文字のフラグメント化に関するこの処理は、文字ストリングが FOR BIT DATA と定義されている場合にも同じであることに注意してください。

GRAPHIC ストリング割り当てに関する DBCS の考慮事項

GRAPHIC ストリング割り当ては、文字ストリングに似た方法で処理されます。非 Unicode データベースの場合、GRAPHIC ストリング・データ・タイプが互換であるのは他の GRAPHIC ストリング・データ・タイプとだけであり、数値、文字ストリング、日付/時刻データ・タイプとは互換ではありません。 Unicode データベースの場合、GRAPHIC ストリングのデータ・タイプは、文字ストリングのデータ・タイプと互換性があります。ただし、SELECT INTO または VALUES INTO ステートメントの中では、GRAPHIC ストリングのデータ・タイプと文字ストリングのデータ・タイプを相互に交換可能なものとして使用することはできません。

GRAPHIC ストリング値が GRAPHIC ストリング列に割り当てられる場合、その値の長さがその列の長さを超えてはなりません。

GRAPHIC ストリング値 (「ソース」・ストリング) を固定長 GRAPHIC ストリング・データ・タイプ (「割り当て先」、列またはホスト変数) に割り当てる場合に、ソース・ストリングの長さが割り当て先より短いなら、割り当て先には、ソース・ストリングのコピーの右端に、値の長さが割り当て先の長さに等しくなるために必要な数の 2 バイト・ブランク文字を埋め込んだものが入れられます。

GRAPHIC ストリング値を GRAPHIC ストリングのホスト変数に割り当てる場合に、ソース・ストリングの長さがホスト変数の長さよりも長いなら、ホスト変数には、ソース・ストリングのコピーの右端から、値の長さがホスト変数の長さと等しくなるために必要な数の 2 バイト・ブランク文字を切り捨てたものが入れられます。 (このシナリオでは、切り捨てにおいて 2 バイト文字の二分化を考慮する必要はありません。二分化が発生したなら、それはソース値または割り当て先のホスト変数のどちらかで GRAPHIC ストリング・データ・タイプの定義に異常があるということです。) SQLCA の警告フラグ SQLWARN1 が 'W' に設定されます。標識変数が指定されていれば、標識変数にはソース・ストリングの元の長さ (2 バイト文字の文字数) が入れられます。しかし、DBCLOB の場合は、標識変数に元の長さは入れられません。

C の NUL 終止符ホスト変数 (wchar_t を使って宣言されたもの) の検索割り当ては、 PREP または BIND コマンドに指定されたオプションに基づいて処理されます。

日時割り当て

日時割り当ての基本規則は、DATE、TIME、または TIMESTAMP 値は、一致するデータ・タイプ (DATE、TIME、または TIMESTAMP のいずれか) を持つ列、一致するデータ・タイプを持つ変数、あるいは、固定長か可変長のストリング変数またはストリング列にしか割り当てることができないというものです。
CLOB、DBCLOB または BLOB の変数または列に割り当てることはできません。

日付/時刻値をストリング変数またはストリング列に割り当てるときは、ストリング表記に自動的に変換されます。日付、時刻、タイム・スタンプのどの部分からも先行ゼロが省略されることはありません。割り当て先で必要な長さは、ストリング表記のフォーマットによって異なります。割り当て先の長さが必要よりも長く、割り当て先が固定長ストリングである場合は、割り当て先の右端にブランクが埋め込まれます。割り当て先の長さが必要よりも短い場合は、関係する日付/時刻値のタイプと割り当て先のタイプによって結果が異なります。

ターゲットがホスト変数以外のストリング変数である場合、以下の規則が適用されます。

- **DATE の場合:** 変数の長さが 10 文字未満であればエラーが返されます。
- **TIME の場合:** 変数の長さが 8 文字未満であればエラーが返されます。
- **TIMESTAMP の場合:** ホスト変数が 26 文字未満であればエラーが返されます。

宛先がホスト変数である場合、以下の規則が適用されます。

- **DATE の場合:** ホスト変数の長さが 10 文字未満であればエラーが返されます。
- **TIME の場合:** USA フォーマットでは 8 文字未満の長さのホスト変数を使用できません。その他のフォーマットでは 5 文字未満の長さにすることはできません。

  ISO または JIS フォーマットを使用しホスト変数の長さが 8 文字未満の場合、時刻の 2 番目の部分は結果から省略され、標識変数があれば、その変数に割り当てられます。 SQLCA の SQLWARN1 フィールドに、省略処理を示す値が設定されます。
- **TIMESTAMP の場合:** ホスト変数が 19 文字未満であればエラーが返されます。長さが 19 文字以上 26 文字未満の場合、値のマイクロ秒部分の末尾桁が省略されます。 SQLCA の SQLWARN1 フィールドに、省略処理を示す値が設定されます。

XML の割り当て

XML 割り当ての一般規則として、XML 値だけを XML 列または XML 変数に割り当てることができます。その規則の例外は次のとおりです。

- ***入力 XML ホスト変数を処理する場合:*** これは XML 割り当て規則の特殊ケースです。ホスト変数はストリング値を基にしているためです。 SQL 内で XML への割り当てを行うため、ストリング値が暗黙で解析されて XML 値になります。その際、CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの設定が使用されて、空白を残すか除去するかが決まります。ただし、ホスト変数が XMLVALIDATE 関数の引数である場合は例外で、その場合は不要な空白が常に除去されます。
- ***データ・タイプ XML の入力パラメーター・マーカーにストリングを割り当てる場合:*** 入力パラメーター・マーカーの暗黙的または明示的なデータ・タイプが XML の場合、そのパラメーター・マーカーにバインドする (割り当てる) 値は、文字ストリング変数、GRAPHIC ストリング変数、またはバイナリー・ストリング変数のいずれでもかまいません。その場合、ストリング値が暗黙で解析されて XML 値になります。その際、CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの設定が使用されて、空白を残すか除去するかが決まります。ただし、

パラメーター・マーカーが XMLVALIDATE 関数の引数である場合は例外で、その場合は不要な空白が常に除去されます。

- ***データ変更ステートメント内で XML 列にストリングを直接割り当てる場合:*** データ変更ステートメント内でタイプが XML の列に直接割り当てを行う場合、割り当てられる式も、文字ストリングまたはバイナリー・ストリングのどちらでもかまいません。その場合、`XMLPARSE (DOCUMENT`*`expression`*` STRIP WHITESPACE)` の結果が、ターゲット列に割り当てられます。サポートされるストリングのデータ・タイプは、XMLPARSE 関数用のサポートされている引数で定義されます。このような XML 割り当ての例外があっても、文字ストリングまたはバイナリー・ストリングの値を SQL 変数や、データ・タイプが XML の SQL パラメーターに割り当てることはできないことに注意してください。
- ***取り出し時に XML をストリングに割り当てる場合:*** 組み込み SQL 内で FETCH INTO ステートメントまたは EXECUTE INTO ステートメントを使用して、XML 値をホスト変数に取り出す場合、そのホスト変数のデータ・タイプは、CLOB、DBCLOB、または BLOB のいずれでもかまいません。他のアプリケーション・プログラミング・インターフェース (CLI、JDBC、または .NET など) を使用する場合、該当するアプリケーション・プログラミング・インターフェースでサポートされている文字、GRAPHIC、またはバイナリーのストリング・タイプで XML 値を取り出すことができます。これらのどのケースでも、XML 値は、UTF-8 でエンコードされたストリングに暗黙的にシリアライズ化され、文字または GRAPHIC ストリング変数の場合は、クライアント・コード・ページに変換されます。

文字ストリングまたはバイナリー・ストリングの値を XML ホスト変数に取り出すことはできません。 XML ホスト変数内の値を、文字ストリング・データ・タイプまたはバイナリー・ストリング・データ・タイプの、列、SQL 変数、または SQL パラメーターに割り当てることはできません。

ユーザー定義タイプの割り当て

特殊タイプおよび構造化タイプでは、ホスト変数へ割り当てる場合には、他のすべての割り当てで使用される規則とは異なる規則が適用されます。

特殊タイプ: ホスト変数への割り当ては、特殊タイプのソース・タイプに基づいて行われます。つまり、以下の規則に従います。

- 割り当ての右辺に指定する特殊タイプの値は、その特殊タイプのソース・タイプが割り当ての左辺に指定するホスト変数に割り当て可能な場合にのみ、ホスト変数へ割り当てられます。

割り当ての宛先が、特殊タイプに基づく列である場合、ソース・データ・タイプは、ターゲット・データ・タイプへキャスト可能でなければなりません。

構造化タイプ: ホスト変数に対する割り当ては、ホスト変数の宣言済みタイプに基づきます。つまり、以下の規則に従います。

- 割り当ての右辺に指定する構造化タイプの値は、宣言済みタイプのホスト変数が構造化タイプ、または構造化タイプのスーパータイプである場合にのみ、左辺のホスト変数へ割り当てられます。

割り当ての宛先が、構造化タイプの列である場合、ソース・データ・タイプは、ターゲット・データ・タイプ、またはターゲット・データ・タイプのサブタイプでなければなりません。

配列タイプでは、SQL 変数およびパラメーターへ割り当てる場合には、異なる規則が適用されます。 SQL 変数またはパラメーターへの割り当ての妥当性は、以下の規則にしたがって決定されます。

- 割り当ての右側が SQL 変数またはパラメーター、TRIM_ARRAY 関数の呼び出し、または CAST 式である場合、そのタイプは割り当ての左側の SQL 変数またはパラメーターのタイプと同じでなければなりません。
- 割り当ての右側が配列コンストラクターまたは ARRAY_AGG 関数の呼び出しである場合、それは左側の SQL 変数またはパラメーターのタイプに暗黙的にキャストされます。

例えば、変数 *V* のタイプが MYARRAY であると想定し、次のステートメントがあるとします。

```
SET V = ARRAY[1,2,3];
```

これは、以下と同じ意味になります。

```
SET V = CAST(ARRAY[1,2,3] AS MYARRAY);
```

さらに次のステートメントがあるとします。

```
SELECT ARRAY_AGG(C1) INTO V FROM T
```

これは、以下と同じ意味になります。

```
SELECT CAST(ARRAY_AGG(C1) AS MYARRAY) INTO V FROM T
```

参照タイプの割り当て

ターゲット・タイプ *T* を指定している参照タイプは、ターゲット・タイプ *S* を指定している参照タイプでもある参照タイプ列に割り当てることができます (*S* は *T* のスーパータイプ)。有効範囲が指定されている参照列または変数に割り当てが行われる場合、割り当てられている実際の値が、有効範囲で定義されているターゲット表またはターゲット・ビューに存在することを確認するためのチェックは行われません。

ホスト変数への割り当ては、参照タイプの表示タイプに基づいて行われます。つまり、以下の規則に従います。

- 割り当ての右側に指定する参照タイプの値は、ホスト参照変数の左側にも割り当て可能ですが、それはこの参照タイプの表示タイプがこのホスト変数に割り当て可能な場合だけです。

割り当てのターゲットが列で、その割り当ての右側にホスト変数が指定されている場合、そのホスト変数はそのターゲット列の参照タイプに明示的にキャストされなければなりません。

数値比較

数値は代数的に、つまり符号を考慮して比較されます。例えば、-2 は +1 より小さい値として扱われます。

一方が整数で、もう一方が 10 進数の場合、 10 進数に変換された整数の一時コピーが比較に使用されます。

位取りの異なる 10 進数を比較する場合、比較は、一方の数値の小数部分が、他方の数値の小数部分と同じ桁数になるように、後続ゼロを使って拡張されたその数値の一時コピーを使用して行われます。

一方が浮動小数点数で、他方が整数か 10 進数の場合、この後者の数値を倍精度浮動小数点数に変換したものの一時コピーが比較に使用されます。

2 つの浮動小数点値が等しいのは、正規形のビット構成が同一の場合のみです。

一方が 10 進浮動小数点数で、他方の数値が整数、10 進数、単精度浮動小数点数、または倍精度浮動小数の場合、この後者の数値を 10 進浮動小数点数に変換したものの一時コピーが比較に使用されます。

一方の数値が DECFLOAT(16) で、他方の数値が DECFLOAT(34) の場合、比較される前に DECFLOAT(16) 値は DECFLOAT(34) に変換されます。

10 進浮動小数点データ・タイプは、正のゼロと負のゼロの両方をサポートしています。 正のゼロと負のゼロには異なるバイナリー表記がありますが、= (等しい) 述部は負のゼロと正のゼロの比較で true を戻します。

COMPARE_DECFLOAT および TOTALORDER スカラー関数は、例えば 2.0 <> 2.00 の比較が必要な場合に、バイナリー・レベルで比較を実行するのに使用できます。

10 進浮動小数点データ・タイプは、負の NaN と正の NaN (静止とシグナリング)、および負の無限大と正の無限大の仕様をサポートしています。SQL の観点から見ると、INFINITY = INFINITY、NAN = NAN、SNAN = SNAN、および -0 = 0 です。

特殊値の比較および配列規則は、以下のとおりです。

- (+/-) INFINITY は、同符号の (+/-) INFINITY とのみ等しく比較されます。
- (+/-) NAN は、同符号の (+/-) NAN とのみ等しく比較されます。
- (+/-) SNAN は、同符号の (+/-) SNAN とのみ等しく比較されます。

異なる特殊値間の順序は、次のとおりです。-NAN < -SNAN < -INFINITY < 0 < INFINITY < SNAN < NAN

ストリングの比較

文字ストリングは、データベースの作成時に指定された照合シーケンスに従って比較されます。ただし、FOR BIT DATA 属性の文字ストリングは例外で、そのような文字ストリングは常にビット値に従って比較されます。

長さの異なる文字ストリングを比較する場合、長い方のストリングの長さにあわせて、短い方のストリングの右端にブランクを埋め込んで延長した論理コピーを使用して比較が行われます。この論理的な拡張は、FOR BIT DATA のタグの付いたものも含め、すべての文字ストリングに対して行われます。

文字ストリング (FOR BIT DATA のタグが付けられた文字ストリングを除く) は、データベースの作成時に指定された照合シーケンスに従って比較されます。例えば、データベース・マネージャーによって指定されるデフォルトの照合シーケンスは、同じ文字の小文字と大文字に同じ重みを与えています。データベース・マネージャーは、完全に同一のストリングだけが相互に等しいものとして扱われるようにするために、 2 つの比較を実行します。まず、ストリングがデータベースの照合シーケンスに従って比較されます。ストリングの文字の重みが等しい場合、次の判断基準として、実際のコード・ポイント値に基づいてストリングを比較します。

2 つのストリングは、両方が空であるか、または対応するすべてのバイト数が等しい場合には、等しくなります。どちらかのオペランドが NULL 値の場合の結果は不明です。

基本比較演算子 (=、<>、<、>、<=、および >=) を使用する比較演算では、LOB ストリングはサポートされません。このようなストリングの比較は、 LIKE 述部と POSSTR 関数を使用した比較でサポートされています。

ストリングのうち 4 000 バイト以下の部分は、SUBSTR と VARCHAR のスカラー関数を使用して比較できます。例えば、以下のような列を考えてみます。

```
MY_SHORT_CLOB   CLOB(300)
MY_LONG_VAR     VARCHAR(8000)
```

この場合、以下の演算は有効です。

```
WHERE VARCHAR(MY_SHORT_CLOB) > VARCHAR(SUBSTR(MY_LONG_VAR,1,300))
```

例:

以下の例で、'A'、'Á'、'a'、および 'á' のコード・ポイント値はそれぞれ、X'41'、X'C1'、X'61'、および X'E1' です。

'A'、'Á'、'a'、'á' という文字の重みが 136、139、135、138 である照合シーケンスを考えてみます。このような場合は以下のようになります。

```
'a' < 'A' < 'á' < 'Á'
```

今度は D1、D2、D3、および D4 という 4 つの DBCS 文字を例にとって考えてみます。これらの文字はそれぞれ 0xC141、0xC161、0xE141、および 0xE161 というコード・ポイントを持っています。これらの DBCS 文字が CHAR 列に入っている場合、各文字のバイトが持っている照合重みに従った順序でソートされます。最初の 2 つのバイトの重みは 138 と 139 であるため、 D3 と D4 は D2 と D1 よりも前に来ます。続く 2 つのバイトの重みは 135 と 136 であるため、順序は以下のようになります。

```
D4 < D3 < D2 < D1
```

ただし、比較する値に FOR BIT DATA 属性がある場合や、これらの DBCS 文字が GRAPHIC 列に格納された場合は、照合重みは無視され、これらの文字が持っているコード・ポイントに従って文字が比較されます。以下のようになります。

```
 'A' < 'a' < 'Á' < 'á'
```

DBCS 文字はコード・ポイントの順序でソートされます。以下のようになります。

```
D1 < D2 < D3 < D4
```

次に 'A'、Á'、'a'、'á' という文字が、 74、75、74、および 75 の (ユニークでない) 重みを持つ照合シーケンスを考えてみます。照合重みだけに注目すると (第 1 のパス)、 'a' は 'A' に等しく、'á' は 'Á' に等しいですが、決着を付けるために文字のコード・ポイントを使用すると (第 2 のパス)、以下のようになります。

```
'A' < 'a' < 'Á' < 'á'
```

CHAR 列に入っている DBCS 文字は、最初は重みに従ったバイトの順序 (第 1 パス) でソートされます。それでも決着がつかない場合は、コード・ポイントに従ったバイトの順序 (第 2 パス) でソートされます。最初の 2 つのバイトは重みが同じであるため、コード・ポイント (0xC1 と 0xE1) で決着を付けることになります。結果として、文字 D1 と D2 は文字 D3 と D4 の前にソートされます。続く 2 つのバイトもこれと同じように比較されます。最終的な結果は以下のようになります。

```
D1 < D2 < D3 < D4
```

ここでも、比較する値に FOR BIT DATA 属性がある場合や、これらの DBCS 文字が GRAPHIC 列に格納された場合は、照合重みは無視され、これらの文字が持っているコード・ポイントに従って文字が比較されます。以下のようになります。

```
D1 < D2 < D3 < D4
```

この例では、照合重みが使用されたときと同じ結果が戻されていますが、実際の場面でいつもそのようになるとは限りません。

比較の際の変換規則

2 つのストリングを比較する場合、必要なら、一方のストリングがまずもう一方のストリングのコード化スキームおよびコード・ページに変換されます。

結果の順序付け

結果のソートが必要な場合、 124 ページの『ストリングの比較』で説明されているストリング比較規則に基づいて順序付けが行われます。比較はデータベース・サーバー側で実行されます。クライアント・アプリケーションに結果が戻される時点で、コード・ページ変換が実行されることがあります。後から行われるこのようなコード・ページ変換は、サーバーの決定した結果セットの順序には影響しません。

ストリング比較に関する MBCS の考慮事項

SBCS/MBCS 混合文字ストリングは、データベースの作成時に指定された照合シーケンスに従って比較されます。デフォルト (SYSTEM) 照合シーケンスで作成されたデータベースの場合、 1 バイトの ASCII 文字はすべて正しい順序で保管されますが、 2 バイト文字は必ずしもコード・ポイントの順序になっているとは限りません。 IDENTITY 順序で作成されたデータベースの場合、 2 バイト文字はすべてコード・ポイントの順序で保管され、 1 バイトの ASCII 文字も同様にコード・ポイントの順序で保管されます。 COMPATIBILITY 順序で作成されたデータベースの場合、ほとんどの 2 バイト文字について正しくソートを行い、 ASCII についてもほぼ正しい、中間的な順序が使用されます。これは、DB2 バージョン 2 ではデフォルトの照合表でした。

混合文字ストリングはバイトごとに比較されます。混合ストリング内に現われるマルチバイト文字では通常とは異なる結果になる場合がありますが、これは個々のバイトが別々に扱われるためです。

例:

この例で、'A'、'B'、'a'、および 'b' の 2 バイト文字のコード・ポイント値はそれぞれ、X'8260'、X'8261'、X'8281'、および X'8282' です。

コード・ポイント X'8260'、X'8261'、X'8281'、および X'8282' の重みがそれぞれ 96、65、193、および 194 である照合シーケンスを考えてみます。 この場合は以下のようになります。

```
'B' < 'A' < 'a' < 'b'
```

および

```
'AB' < 'AA' < 'Aa' < 'Ab' < 'aB' < 'aA' < 'aa' < 'ab'
```

GRAPHIC ストリングの比較は、文字ストリングの場合と同じように処理されます。

GRAPHIC ストリングの比較は、DBCLOB を除くすべての GRAPHIC ストリング・データ・タイプの間で有効です。

GRAPHIC ストリングに対しては、データベースの照合シーケンスは使用されません。その代わりに、GRAPHIC ストリングは、常に対応するバイトの数値 (バイナリー値) に基づいて比較されます。

前の例で、リテラルが GRAPHIC ストリングの場合、以下のような結果になります。

```
'A' < 'B' < 'a' < 'b'
```

および

```
'AA' < 'AB' < 'Aa' < 'Ab' < 'aA' < 'aB' < 'aa' < 'ab'
```

長さの異なる GRAPHIC ストリングを比較する場合、短い方のストリングの右端に長い方のストリングの長さになるまで、 2 バイト・ブランク文字を埋め込んだものの論理コピーが比較に使用されます。

2 つの GRAPHIC ストリングの値が等しくなるのは、両方が空であるか、または対応する GRAPHIC がすべて等しい場合です。どちらかのオペランドが NULL 値の場合の結果は不明です。 2 つの値が等しくない場合は、両者の関係は単純なバイナリー・ストリング比較によって決定されます。

この節で説明してきたとおり、バイトに基づくストリングの比較は誤った結果をもたらす場合があります。つまり、文字比較で得られる文字とは異なる結果が生じる場合があります。ここで示した一連の例は、同じ MBCS コード・ページであることを前提にしていますが、実際には、同じ言語を使用していても異なるマルチバイトのコード・ページを使用することがあるので、状況はもっと複雑であるといえます。例えば、日本語 DBCS コード・ページと日本語 EUC コード・ページからのストリングを比較するというような場合が考えられます。

日付/時刻の比較

DATE、TIME、または TIMESTAMP 値は、同じデータ・タイプの別の値か、そのデータ・タイプのストリング表記と比較することができます。すべての比較は日時順に行われます。つまり、0001 年 1 月 1 日からの時間の経過の大きい方が値が大きいということです。

TIME 値と、時刻値のストリング表記とが関係する比較では、常に秒数が組み入れられます。ストリング表記で秒数を省略しているときは、暗黙のうちにゼロ秒が補われます。

TIMESTAMP 値に関する比較は、等しいと見なしてもよいような表示の考慮はしません。日時順に行われます。

例:

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

ユーザー定義タイプの比較

ユーザー定義特殊タイプの値は、完全に同じユーザー定義特殊タイプの値とのみ比較することができます。ユーザー定義特殊タイプは、WITH COMPARISONS 節を使用して定義されていなければなりません。

例:

以下の YOUTH 特殊タイプおよび CAMP_DB2_ROSTER 表を想定します。

```
CREATE TYPE YOUTH AS INTEGER WITH COMPARISONS

CREATE TABLE CAMP_DB2_ROSTER
   ( NAME               VARCHAR(20),
     ATTENDEE_NUMBER    INTEGER NOT NULL,
     AGE                YOUTH,
     HIGH_SCHOOL_LEVEL  YOUTH)
```

以下の比較は有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > HIGH_SCHOOL_LEVEL
```

以下の比較は無効です。

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > ATTENDEE_NUMBER
```

ただし、特殊タイプとそのソース・タイプとの間では、キャストのための関数または CAST 指定を使用することによって、 AGE と ATTENDEE_NUMBER とを比較することができます。以下の比較はすべて有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
   WHERE INTEGER(AGE) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
   WHERE CAST( AGE AS INTEGER) > ATTENDEE_NUMBER

SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > YOUTH(ATTENDEE_NUMBER)

SELECT * FROM CAMP_DB2_ROSTER
   WHERE AGE > CAST(ATTENDEE_NUMBER AS YOUTH)
```

ユーザー定義構造化タイプの値を、他の値と比較することはできません (NULL 述部および TYPE 述部が使えます)。

参照タイプの比較

参照タイプ値を比較できるのは、それらのターゲット・タイプが共通のスーパータイプを持っている場合だけです。共通のスーパータイプのスキーマ名が SQL パスに組み込まれている場合のみ、該当する比較関数が見つかります。比較は参照タイプの表示タイプを使用して行われます。参照の有効範囲は、比較では考慮されません。

非 Unicode データベースでの XML 比較

非 Unicode データベースで実行するとき、XML データと文字または GRAPHIC ストリングの値の比較は、比較される 2 つのデータのセットのうちの 1 つのコード・ページ変換を必要とします。照会の述部として、あるいは文字または GRAPHIC ストリング・データ・タイプを使ったホスト変数として、SQL または XQuery ステートメントで使用される文字またはグラフィックの値は、比較の前にデータベース・コード・ページに変換されます。このデータに含まれている任意の文字が、データベース・コード・ページの一部ではないコード・ポイントを持っている場合、置換文字がその場所に追加され、予期しない照会の結果が生じる可能性があります。

例えば、UTF-8 コード・ページを持つクライアントが、ギリシャ語のエンコード ISO8859-7 で作成されたデータベース・サーバーに接続するのに使用されるとします。 式 $\Sigma_G\Sigma_M$ が、XQuery ステートメントの述部として送信されますが、ここで Σ_G は Unicode ではギリシャ語のシグマの文字 (U+03A3) を表し、Σ_M は Unicode では数学記号のシグマ (U+2211) を表します。 この式は最初にデータベース・コード・ページに変換され、その結果、両方の「Σ」文字はギリシャ語のデータベース・コード・ページにあるシグマのコード・ポイント 0xD3 に対応するように変換されます。このコード・ポイントを Σ_A としましょう。新しく変換された式 $\Sigma_A\Sigma_A$ は、ターゲット XML データと比較するために、もう一度 UTF-8 に変換されます。これら 2 つのコード・ポイントの違いは、述部式をデータベースに渡すのに必要なコード・ページ変換の結果として失われるので、最初の 2 つの異なる値 Σ_G および Σ_M は、式 $\Sigma_G\Sigma_G$ として XML パーサーに渡されます。それで、この式は、XML 文書内の値 $\Sigma_G\Sigma_M$ と比較されたときに、マッチングに失敗します。

コード・ページ変換の問題が原因で予期しない照会結果が発生する可能性を避ける 1 つの方法は、照会式に使用されているすべての文字に一致するコード・ポイントが必ずデータベース・コード・ページにあるようにすることです。一致するコード・ポイントがない文字は、Unicode 文字のエンティティー参照を使用することによって、含めることができます。文字のエンティティー参照は、常にコード・ページ変換をバイパスします。例えば、Σ_M 文字の代わりに文字のエンティティー参照 ࢣ を使用して、データベース・コード・ページに関係なく、正しい Unicode コード・ポイントが比較に確実に使用されるようにします。

結果データ・タイプの規則

結果のデータ・タイプは、演算のオペランドに適用される規則によって決定されます。ここでは、そのような規則について説明します。

これらの規則は以下に適用されます。

- セット演算 (UNION、INTERSECT、および EXCEPT) の全選択における対応する列
- CASE 式の結果式および DECODE スカラー関数
- スカラー関数 COALESCE (また NVL および VALUE) の引数
- スカラー関数 GREATEST、LEAST、MAX、および MIN の引数
- IN 述部の IN リストの式値
- 複数行の VALUES 節の対応する式

これらの規則は、さまざまな演算での LOB ストリングに関するその他の制限にも従って、適用されます。

さまざまなデータ・タイプに関係する規則を以下に示します。一部については、考えられる結果データ・タイプを表に示します。データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっています。

それらの表では、適用される長さまたは精度と位取りも含めて、結果データ・タイプを示します。結果タイプは、オペランドを考慮して決定されます。オペランドの対が複数の場合は、まず最初の対から検討します。それによる結果タイプとその次のオペランドとの組み合わせが検討されて、次の結果タイプが決定される、というようになります。最後の中間結果タイプと最後のオペランドによって、その演算の最終的な結果タイプが決定されます。演算処理は左から右へ行われます。このため、演算が繰り返されるときは、中間結果タイプが重要になります。例えば、以下のような演算を考えてみます。

```
CHAR(2) UNION CHAR(4) UNION VARCHAR(3)
```

最初の対の結果のタイプは CHAR(4) です。この結果の値は常に 4 バイトになります。最終的な結果タイプは VARCHAR(4) です。最初の UNION 演算の結果の値は、常に長さが 4 になります。

文字ストリング

文字ストリング値は、別の文字ストリング値と互換性があります。文字ストリングには、データ・タイプ CHAR、VARCHAR、CLOB が含まれます。

一方のオペランド	他方のオペランド	結果のデータ・タイプ
CHAR(x)	CHAR(y)	CHAR(z)、ただし z = max(x,y)
CHAR(x)	VARCHAR(y)	VARCHAR(z)、ただし z = max(x,y)
VARCHAR(x)	CHAR(y) または VARCHAR(y)	VARCHAR(z)、ただし z = max(x,y)
CLOB(x)	CHAR(y)、VARCHAR(y)、または CLOB(y)	CLOB(z)、ただし z = max(x,y)

結果の文字ストリングのコード・ページは、ストリング変換の規則に基づいて導き出されます。

GRAPHIC ストリング

GRAPHIC ストリング値は、別の GRAPHIC ストリング値と互換性があります。GRAPHIC ストリングには、データ・タイプ GRAPHIC、VARGRAPHIC、DBCLOB が含まれます。

一方のオペランド	他方のオペランド	結果のデータ・タイプ
GRAPHIC(x)	GRAPHIC(y)	GRAPHIC(z)、ただし z = max(x,y)
VARGRAPHIC(x)	GRAPHIC(y) または VARGRAPHIC(y)	VARGRAPHIC(z)、ただし z = max(x,y)
DBCLOB(x)	GRAPHIC(y)、VARGRAPHIC(y)、または DBCLOB(y)	DBCLOB(z)、ただし z = max(x,y)

結果の GRAPHIC ストリングのコード・ページは、ストリング変換の規則に基づいて導き出されます。

Unicode データベース内の文字ストリングおよび GRAPHIC ストリング

Unicode データベースでは、文字ストリング値と GRAPHIC ストリング値の間に互換性があります。

一方のオペランド	他方のオペランド	結果のデータ・タイプ
GRAPHIC(x)	CHAR(y) または GRAPHIC(y)	GRAPHIC(z)、ただし z = max(x,y)
VARGRAPHIC(x)	CHAR(y) または VARCHAR(y)	VARGRAPHIC(z)、ただし z = max(x,y)
VARCHAR(x)	GRAPHIC(y) または VARGRAPHIC	VARGRAPHIC(z)、ただし z = max(x,y)
DBCLOB(x)	CHAR(y)、VARCHAR(y)、または CLOB(y)	DBCLOB(z)、ただし z = max(x,y)
CLOB(x)	GRAPHIC(y) または VARGRAPHIC(y)	DBCLOB(z)、ただし z = max(x,y)

バイナリー・ラージ・オブジェクト (BLOB)

バイナリー・ストリング (BLOB) 値は、別のバイナリー・ストリング (BLOB) 値とのみ互換性があります。BLOB タイプとして扱う必要がある場合、 BLOB スカラー関数を使用して他のタイプからキャストすることができます。結果の BLOB の長さは、すべてのデータ・タイプの中で最大の長さです。

数値

数値タイプは、他の数値タイプ、文字ストリング・データ・タイプと互換性があり、Unicode データベースでは、GRAPHIC ストリング・データ・タイプとも互換性

があります。数値タイプには、SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE、および DECFLOAT が入ります。

一方のオペランド	他方のオペランド	結果のデータ・タイプ
SMALLINT	SMALLINT	SMALLINT
INTEGER	SMALLINT	INTEGER
INTEGER	INTEGER	INTEGER
BIGINT	SMALLINT	BIGINT
BIGINT	INTEGER	BIGINT
BIGINT	BIGINT	BIGINT
DECIMAL(w,x)	SMALLINT	DECIMAL(p,x)、ただし p = x+max(w-x,5) [注 1]
DECIMAL(w,x)	INTEGER	DECIMAL(p,x)、ただし p = x+max(w-x,11) [注 1]
DECIMAL(w,x)	BIGINT	DECIMAL(p,x)、ただし p = x+max(w-x,19) [注 1]
DECIMAL(w,x)	DECIMAL(y,z)	DECIMAL(p,s)、ただし p = max(x,z)+max(w-x,y-z) [注 1]s = max(x,z)
REAL	REAL	REAL
REAL	SMALLINT、INTEGER、BIGINT、または DECIMAL	DOUBLE
DOUBLE	SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、または DOUBLE	DOUBLE
DECFLOAT(n)	SMALLINT、INTEGER、DECIMAL (<=16,s)、REAL、または DOUBLE	DECFLOAT(n)
DECFLOAT(n)	BIGINT	DECFLOAT(34)
DECFLOAT(n)	DECIMAL (>16,s)	DECFLOAT(34)
DECFLOAT(n)	DECFLOAT(m)	DECFLOAT(MAX(n,m))

[1] 精度は 31 以下でなければなりません。

DATE

日付値は、別の日付値、または日付の有効なストリング表記を値とする任意の CHAR または VARCHAR 式と互換性があります。結果のデータ・タイプは DATE です。

TIME

時刻値は、別の時刻値、または時刻の有効なストリング表記を値とする任意の CHAR または VARCHAR 式と互換性があります。結果のデータ・タイプは TIME です。

TIMESTAMP

タイム・スタンプ値は、別のタイム・スタンプ値、またはタイム・スタンプの有効なストリング表記を値とする任意の CHAR または VARCHAR 式と互換性があります。結果のデータ・タイプは TIMESTAMP です。

XML

XML 値は、別の XML 値と互換性があります。結果のデータ・タイプは XML です。

特殊タイプ

ユーザー定義特殊タイプ値は、同じユーザー定義特殊タイプの別の値とのみ互換性があります。結果のデータ・タイプはそのユーザー定義特殊タイプです。

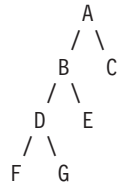
参照タイプ

参照タイプ値は、同じ参照タイプの別の値と互換性がありますが、それは両方のターゲット・タイプが共通のスーパータイプを持っている場合に限ります。結果のデータ・タイプは、共通のスーパータイプをターゲット・タイプとして持っている参照タイプです。すべてのオペランドに同じ有効範囲の表がある場合、結果は有効範囲の表になります。それ以外の場合、結果では効力範囲は指定されません。

構造化タイプ

構造化タイプ値は、同じ構造化タイプの別の値と互換性がありますが、それは両方が共通のスーパータイプを持っている場合に限ります。結果の構造化タイプ列の静的データ・タイプは、いずれかの列の最小限の共通スーパータイプである構造化タイプです。

例えば、以下の構造化タイプ階層について考えてみます。



静的タイプ E と F の構造化タイプは、結果の静的タイプ B と互換性があります。ただし、E および F の最小限の共通スーパータイプです。

結果の NULL 可能属性

INTERSECT と EXCEPT を除き、 2 つのオペランドの両方とも NULL 値が使用できないのでない限り、結果で NULL 値が可能です。

- INTERSECT で、どちらかのオペランドで NULL 値を使えない場合、結果での NULL 値の使用は認められません (論理積が NULL 値になることはありません)。
- EXCEPT では、最初のオペランドで NULL 値を使えない場合、結果での NULL 値の使用は認められません (結果は最初のオペランドの値しか取れないためです)。

ストリング変換の規則

演算の実行に使用されるコード・ページは、その演算のオペランドに適用される規則によって決定されます。ここでは、そのような規則について説明します。

これらの規則は以下に適用されます。

- セット演算 (UNION、INTERSECT、および EXCEPT) の全選択における対応するストリング列
- 連結のオペランド
- 述部のオペランド (LIKE を除く)
- CASE 式の結果式および DECODE スカラー関数
- スカラー関数 COALESCE (また NVL および VALUE) の引数
- スカラー関数 GREATEST、LEAST、MAX、および MIN の引数
- スカラー関数 OVERLAY (および INSERT) の *source-string* および *insert-string* 引数
- IN 述部の IN リストの式値
- 複数行の VALUES 節の対応する式

それぞれの場合で結果コード・ページはバインド時に決定されます。演算の実行時に、ストリングがそのコード・ページで識別されるコード・ページに変換されることがあります。有効な変換がなされていない文字は、置換文字にマップされ、文字セットと SQLWARN10 が SQLCA で 'W' に設定されます。

結果コード・ページは、オペランドのコード・ページによって決定されます。初めの 2 つのオペランドのコード・ページが中間結果コード・ページを決定し、 (該当する場合には) そのコード・ページと次のオペランドのコード・ページが新たな中間結果コード・ページを決定します。以下、同様になります。最後の中間結果コード・ページと最後のオペランドのコード・ページが、最終結果のストリングまたは列のコード・ページを決定します。コード・ページの各ペアについて、次の規則を順番に適用することで結果が決定されます。

- コード・ページが等しい場合、結果はそのコード・ページになります。
- コード・ページが BIT DATA (コード・ページ 0) 結果のコード・ページは BIT DATA になります。
- Unicode データベースでは、一方のコード・ページがもう一方のコード・ページとは異なるコード化スキーム内にデータを表示する場合、結果は、UTF-8 上の UCS-2 (つまり、文字データ・タイプ上の GRAPHIC データ・タイプ) になります。 (非 Unicode データベースでは、異なるコード化スキーム間での変換はサポートされません。)
- ホスト変数であるオペランド (コード・ページは BIT DATA ではない) の場合、結果コード・ページはデータベース・コード・ページになります。そのようなホスト変数からの入力データは、使用前に、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・ページに変換されます。

以下については、必要なら結果のコード・ページへの変換が行われます。

- 連結演算子のオペランド
- スカラー関数 COALESCE (また NVL および VALUE) から選択された引数

- スカラー関数 GREATEST、LEAST、MAX、および MIN の選択された引数
- スカラー関数 OVERLAY (および INSERT) の *source-string* および *insert-string* 引数
- CASE 式の選択された結果式および DECODE スカラー関数
- IN 述部のリストの式
- 複数行の VALUES 節の対応する式
- セット演算に関係した対応する列

文字変換は、次の条件のすべてに該当する場合に必要になります。

- コード・ページが異なる
- いずれのストリングも BIT DATA ではない
- ストリングが NULL でも空でもない

例

例 1: コード・ページ 850 で作成されたデータベースで、以下の条件がある場合は、次のようになります。

式	タイプ	コード・ページ
COL_1	列	850
HV_2	ホスト変数	437

ここで、次のような述部を評価します。

```
COL_1 CONCAT :HV_2
```

ホスト変数データは、使用前に、データベース・コード・ページに変換されるため、 2 つのオペランドの結果コード・ページは 850 になります。

例 2: 上記の例からの情報を使用して、述部を評価すると、

```
COALESCE(COL_1, :HV_2:NULLIND,)
```

結果コード・ページは 850 になります。したがって、 COALESCE スカラー関数の結果コード・ページはコード・ページ 850 になります。

Unicode データベースでのストリング比較

パターン・マッチングは、既存の MBCS データベースの動作と Unicode データベースの動作がいくらか違っている分野の 1 つです。

DB2 Database for Linux, UNIX, and Windowsの MBCS データベースでは、マッチング式に MBCS データが含まれていれば、パターンに SBCS 文字と非 SBCS 文字の両方を組み込める、というのが現在の動作です。パターンの中の特殊文字は、以下のようにして解釈されます。

- SBCS の半角下線文字は、1 つの SBCS 文字を表します。
- 非 SBCS の全角下線文字は、1 つの非 SBCS 文字を表します。
- % (SBCS の半角または非 SBCS の全角) は、0 以上の SBCS または非 SBCS 文字を表します。

Unicode データベースでは、「単一バイト」文字と「非単一バイト」文字の間に実質的な区別がありません。 UTF-8 形式は Unicode 文字の「混合バイト」エンコード方式ですが、 UTF-8 では、SBCS 文字と非 SBCS 文字の間に実質的な区別がありません。 UTF-8 フォーマットでは、文字のバイト数に関係なく、すべての文字が Unicode 文字になります。 Unicode GRAPHIC 列では、半角下線 (U+005F) や半角 % (U+0025) を含め、補足文字以外のすべての文字が 2 バイト幅になります。 Unicode データベースの場合、パターンの中の特殊文字は、以下のようにして解釈されます。

- 文字ストリングでは、半角下線 (X'5F') または全角下線 (X'EFBCBF') が 1 つの Unicode 文字を表します。半角 % (X'25') または全角 % (X'EFBC85') は 0 以上の Unicode 文字を表します。
- GRAPHIC ストリングでは、半角下線 (U+005F) または全角下線 (U+FF3F) が 1 つの Unicode文字を表します。半角 % (U+0025) または全角 % (U+FF05) は 0 以上の Unicode 文字を表します。

注: GRAPHIC 列では、Unicode の補足 GRAPHIC 文字が UCS-2 文字 2 つ分で表されるため、この補足 GRAPHIC 文字と下線を対等にするには、下線が 2 つ必要です。 CHAR 列では、下線 1 つが Unicode の補足文字と等しくなります。

オプションの「エスケープ式」では、下線と % 記号文字の特別な意味を変更するための文字を指定します。この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式

以下の制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。
- CHAR 列の場合、式の結果は 1 文字、つまりちょうど 1 バイト入ったバイナリー・ストリングになります (SQLSTATE 22019)。GRAPHIC 列の場合、式の結果は 1 つの文字になります (SQLSTATE 22019)。

データベース・パーティション互換データ・タイプ

データベース・パーティションの互換性 は、分散キーの対応する列どうしのそれぞれの基本データ・タイプを対象に定義されます。データベース・パーティション互換データ・タイプには、型は異なるものの同じ値を持つ 2 つの変数が、同じデータベース・パーティション関数によって同じ分散マップ索引にマップされるという特性があります。

表 13 は、データベース・パーティションのデータ・タイプの互換性を示しています。

データベース・パーティションの互換性には、次の特性があります。

- DATE、TIME、および TIMESTAMP には内部フォーマットが使用されます。内部フォーマットは相互に互換性がなく、文字またはグラフィック・データ・タイプとの互換性がありません。
- パーティションの互換性は、列の NULL 可能性の影響は受けません。
- パーティションの互換性は、照合の影響を受けます。ロケールに依存する UCA ベースの照合では、完全一致突き合わせが要求されます。ただし、照合の強さ (S) 属性は無視されます。他のすべての照合は、パーティションの互換性を判別する目的においては同等であると見なされます。
- ロケールに依存する UCA ベースの照合以外の照合が使用される場合、FOR BIT DATA で定義される文字の列は、FOR BIT DATA なしの文字の列とのみ互換性があります。
- 互換データ・タイプの NULL 値は同じように取り扱われます。互換性のないデータ・タイプの NULL の場合は異なる結果が生じることがあります。
- UDT の基本データ・タイプは、データベース・パーティションの互換性を分析する場合に使用されます。
- 分散キーの同一値の小数部は、位取りおよび精度が異なっている場合であっても、同一として取り扱われます。
- 文字ストリング (CHAR、VARCHAR、GRAPHIC または VARGRAPHIC) の末尾のブランクは、システムにより提供されるハッシュ関数によって無視されます。
- ロケールに依存する UCA ベースの照合が使用された場合、CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC は互換性のあるデータ・タイプです。他の照合が使用される場合、CHAR と VARCHAR は互換タイプであり、GRAPHIC と VARGRAPHIC は互換タイプですが、CHAR と VARCHAR は GRAPHIC と VARGRAPHIC との互換タイプではありません。長さが異なる CHAR または VARCHAR は、互換データ・タイプです。
- 等しい DECFLOAT 値は、精度が異なっていても同一として取り扱われます。数値的に等しい DECFLOAT 値は、異なる数の有効桁数を持っていても同一として扱われます。

表 13. データベース・パーティションの互換性

オペランド	2 進整数	10 進数	浮動小数点	10 進浮動小数点数	文字ストリング	GRAPHIC ストリング	日付	時刻	TIMESTAMP	特殊タイプ	構造化タイプ
2 進整数	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[1]	いいえ
10 進数	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[1]	いいえ

表 13. データベース・パーティションの互換性 (続き)

オペランド	2 進整数	10 進数	浮動小数点	10 進浮動小数点数	文字ストリング	GRAPHIC ストリング	日付	時刻	TIME STAMP	特殊タイプ	構造化タイプ
浮動小数点	いいえ	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[1]	いいえ
10 進浮動小数点数	いいえ	いいえ	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	[1]	いいえ
文字ストリング[4]	いいえ	いいえ	いいえ	いいえ	はい[2]	2、3	いいえ	いいえ	いいえ	[1]	いいえ
GRAPHIC ストリング[4]	いいえ	いいえ	いいえ	いいえ	2、3	はい[2]	いいえ	いいえ	いいえ	[1]	いいえ
日付	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ	いいえ	[1]	いいえ
時刻	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ	[1]	いいえ
タイム・スタンプ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	[1]	いいえ
特殊タイプ	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	いいえ
構造化タイプ[4]	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ

注:

[1] ユーザー定義特殊タイプ (UDT) の値には、UDT のソース・タイプ、もしくはデータベース・パーティション互換ソース・タイプをもったその他の UDT とのデータベース・パーティション互換性があります。

[2] 照合に互換性のある場合、文字およびグラフィック・ストリング・タイプは互換性があります。

[3] ロケールに依存する UCA ベースの照合が有効である場合、文字およびグラフィック・ストリング・タイプは互換性があります。そうでない場合、これらは互換タイプではありません。

[4] ユーザー定義構造化タイプとデータ・タイプ LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB、および BLOB は、分散キーでサポートされていないので、データベース・パーティション互換性には該当しません。

定数

定数 (リテラル と呼ばれるときもあります) は、値を指定するものです。定数は、ストリング定数と数値定数に分類されます。数値定数はさらに、整数、浮動小数点数、または 10 進数に分類されます。

定数は、すべて NOT NULL の属性を持ちます。

数値定数では、負のゼロ値 (-0) は符号のないゼロ (0) と同じ値です。

ユーザー定義タイプは強力なタイプ指定です。つまり、ユーザー定義タイプはそれ自体のタイプとしか互換性がありません。一方、定数には組み込みタイプがあります。このため、ユーザー定義タイプと定数が関与する演算が実行可能なのは、ユーザー定義タイプがその定数の組み込みタイプにキャストされている場合か、または定数がそのユーザー定義タイプにキャストされている場合のみです。たとえば、128 ページの『ユーザー定義タイプの比較』にある表と特殊タイプを使用する場合、定数 14 との以下の比較が有効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > CAST(14 AS YOUTH)

SELECT * FROM CAMP_DB2_ROSTER
  WHERE CAST(AGE AS INTEGER) > 14
```

以下の比較は無効です。

```
SELECT * FROM CAMP_DB2_ROSTER
  WHERE AGE > 14
```

整数定数

整数定数 は、小数点を除き最大 19 桁の符号付きまたは符号なしの整数を指定します。整数定数の値が長精度整数の範囲内である場合、その整数定数のデータ・タイプは長精度整数 (large integer) です。 整数定数の値が長精度整数の範囲外であるが、64 ビット整数の範囲内にある場合、その整数定数のデータ・タイプは 64 ビット整数 (big integer) です。 64 ビット整数値の範囲外で定義された定数は、10 進定数と見なされます。

長精度整数定数の最小のリテラル表現は -2 147 483 647 であり、整数値の限界である -2 147 483 648 ではありません。同様に、64 ビット整数定数の最小のリテラル表現は、-9 223 372 036 854 775 807 であり、 -9 223 372 036 854 775 808 (64 ビット整数値の限界) ではありません。

例:

```
64     -15     +100     32767     720176      12345678901
```

構文図で 'integer' (整数) という用語は、符号を使用してはならない長精度整数定数を指すために使用されます。

浮動小数点定数

浮動小数点定数 は、E で区切られた 2 つの数値で浮動小数点数を指定します。最初の数値には符号と小数点を指定することができます。 2 番目の数値には符号を指定できますが、小数点を使用することはできません。浮動小数点定数のデータ・タ

イプは倍精度です。定数の値は、最初の数値と、2 番目の数値で指定される 10 の累乗との積であり、浮動小数点数の範囲内になければなりません。定数のバイト数は 30 以下でなければなりません。

例:

```
15E1     2.E5     2.2E-1     +5.E+2
```

10 進定数

10 進定数 は、31 桁以内の数字で構成される符号付きまたは符号なしの数値です。小数点を備えているか、またはバイナリー整数の範囲に収まらないかのどちらかです。これは 10 進数の範囲内になければなりません。精度は桁数の合計数 (前後のゼロを含む)、位取りは小数点の右側の桁数 (後続ゼロを含む) です。

例:

```
25.5     1000.     -15.     +37589.3333333333
```

10 進浮動小数点定数

10 進浮動小数点特殊値 (これは DECFLOAT(34) と解釈される) 以外の 10 進浮動小数点定数はありません。

それらの特殊値は INFINITY、NAN、および SNAN です。 INFINITY は無限大、つまり絶対値が無限に大きい数を表します。 INFINITY の前には、オプションで符号を付けることができます。 INFINITY の代わりに INF を指定できます。 NAN は Not a Number (NaN) を表し、静止 NaN と呼ばれることもあります。これは、警告または例外を発生させない未定義の結果を表す値です。 SNAN はシグナリング NaN (sNaN) を表します。これは、数値演算の中で定義された演算で使用された場合に警告または例外を発生させる未定義の結果を表す値です。 NAN と SNAN の両方の前には、オプションで符号を付けられますが、符号には意味がありません。SNAN は、警告または例外を発生させずに非数値演算で使用できます。例えば、INSERT の VALUES リスト内や、述部内で比較される定数としてなどです。

```
SNAN   -INFINITY
```

すべての非特殊値は、上記に指定された規則に従って、整数、浮動小数点数、または 10 進定数として解釈されます。数値の 10 進浮動小数点値を取得するには、文字ストリング定数を伴う DECFLOAT Cast 関数を使用します。浮動小数点定数を DECFLOAT 関数の引数として使用することは推奨されません。浮動小数点数は正確ではなく、結果の 10 進浮動小数点値は、引数を形成する 10 進数字文字とは異なる可能性があるからです。代わりに、DECFLOAT 関数の引数として文字定数を使用してください。

例えば、DECFLOAT('6.0221415E23', 34) は 10 進浮動小数点値 6.0221415E+23 を戻しますが、DECFLOAT(6.0221415E23, 34) は 10 進浮動小数点値 6.0221415000000003E+23 を戻します。

文字ストリング定数

文字ストリング定数 では、可変長文字ストリングを指定します。文字ストリング定数には、以下の 3 つの形式があります。

- ストリング区切り文字で始まりストリング区切り文字で終わる文字のシーケンス。この場合のストリング区切り文字はアポストロフィ (') です。ストリング区切り文字とストリング区切り文字の間のバイト数は、32672 を超えてはなりません。文字ストリング内で 1 つのストリング区切り文字を表したいときは、ストリング区切り文字を 2 つ連続して使用します。ストリングの中ではない場所でストリング区切り文字を 2 つ連続して使用すると、空ストリングになります。
- X の後に、ストリング区切り文字で始まりストリング区切り文字で終わる文字のシーケンスを記述する形式。この形式の文字ストリング定数のことを *16 進定数* ともいいます。ストリング区切り文字の間にある文字は、偶数個の 16 進数字でなければなりません。ストリング区切り文字の間のブランクは無視されます。 16 進数字の数は、32672 を超えてはなりません。16 進数字は、数字または A から F までのいずれかの文字 (大文字または小文字) です。16 進表記の規則では、1 つの 16 進数字ペアがそれぞれ 1 つの文字に対応します。この形式の文字ストリング定数を使用すれば、キーボード表現のない文字を指定できるようになります。
- U& の後に、ストリング区切り文字で始まりストリング区切り文字で終わる文字のシーケンスを記述する形式。オプションとして、その後に UESCAPE 節を指定することもできます。この形式の文字ストリング定数のことを *Unicode ストリング定数* ともいいます。ストリング区切り文字とストリング区切り文字の間のバイト数は、32672 を超えてはなりません。Unicode ストリング定数は、ステートメント・コンパイル時に、UTF-8 からセクション・コード・ページに変換されます。文字ストリング内で 1 つのストリング区切り文字を表したいときは、ストリング区切り文字を 2 つ連続して使用します。文字ストリング内で 1 つの Unicode エスケープ文字を表したいときは、Unicode エスケープ文字を 2 つ連続して使用します。ただし、これらの文字は、文字定数の長さの計算では、1 つの文字としてカウントされます。ストリングの中ではない場所でストリング区切り文字を 2 つ連続して使用すると、空ストリングになります。UTF-8 の文字は、1 バイトから 4 バイトの範囲にまたがっているので、Unicode ストリング定数の最大長は、実際には 32672 文字よりも少なくなる場合があります。

  文字は、活版印刷文字 (絵文字) でも Unicode コード・ポイントでも表記できます。Unicode 文字のコード・ポイントは、X'000000' から X'10FFFF' までの範囲になります。Unicode 文字をコード・ポイントで表す場合は、Unicode エスケープ文字の後に 4 桁の 16 進数字を記述するか、Unicode エスケープ文字の後に正符号 (+) を入れて、その後に 6 桁の 16 進数字を記述します。デフォルトの Unicode エスケープ文字は、円記号 (¥) ですが、UESCAPE 節で別の文字を指定することもできます。UESCAPE 節では、UESCAPE キーワードの後に、1 つの文字をストリング区切り文字で囲む形で記述します。Unicode エスケープ文字として、正符号 (+)、二重引用符 (")、単一引用符 (')、ブランクは使用できません。また、0 から 9、A から F の文字も、大文字であれ小文字であれ使用できません (SQLSTATE 42604)。例えば、ローマ字 A (大文字) を Unicode コード・ポイントで指定する場合は、¥0041 と ¥+000041 という 2 とおりの方法があります。

定数値は、データベースにバインドされるときに、必ずデータベース・コード・ページに変換されます。それは、データベース・コード・ページのものと見なされます。したがって、定数を FOR BIT DATA 列と結合してその結果が FOR BIT DATA となる式で使用される場合、定数値は使用時にそのデータベース・コード・ページ表記から変換されません。

例:

```
'12/14/1985'    '32'    'DON''T CHANGE'    ''
X'FFFF'    X'46 72 61 6E 6B'
U&'¥0141ód¥017A is a city in Poland'   U&'c:¥¥temp'   U&'@+01D11E' UESCAPE '@'
```

この例の第 2 行の右端のストリングは、VARCHAR パターンの ASCII ストリング 'Frank' に対応しています。最後の行は、'■ód■ is a city in Poland'、'c:¥temp'、音楽のト音記号を表す 1 文字にそれぞれ対応しています。

GRAPHIC ストリング定数

GRAPHIC ストリング定数 は、1 バイトのアポストロフィ (') で始まり、1 バイトのアポストロフィ (') で終わる 2 バイト文字の並びで構成される可変長の GRAPHIC ストリングを指定します。そしてその先頭には、1 バイトの G または N が付けられます。アポストロフィとアポストロフィの間の文字は必ず偶数バイトで、GRAPHIC ストリングの長さは 16336 バイトを超えることはできません。

例:

```
G'double-byte character string'
N'double-byte character string'
```

MBCS 文字の一部としては、アポストロフィ (') を使用しないでください。区切り文字と見なされてしまいます。

Unicode データベースでは、可変長 GRAPHIC ストリングを指定する 16 進 GRAPHIC ストリング定数もサポートされます。 16 進数 GRAPHIC ストリングのフォーマットは、GX の後に、アポストロフィで囲んだ一つながりの文字を続けたものです。アポストロフィの間にある文字は、4 つの 16 進数字の偶数倍でなければなりません。 16 進数字の数は 16336 を超えてはなりません。これを超えると、エラー (SQLSTATE -54002) が戻されます。 16 進 GRAPHIC ストリング定数の形式が正しくない場合には、エラーが戻されます (SQLSTATE 42606)。 4 つの数字から成るグループはそれぞれ、1 つの GRAPHIC 文字に対応します。 Unicode データベースでは、これは、1 つの UCS-2 GRAPHIC 文字になります。

例:

```
GX'FFFF'
```

Unicode データベース内のビット・パターン '1111111111111111' を表します。

```
GX'005200690063006B'
```

Unicode データベース内の ASCII ストリング 'Rick' の VARGRAPHIC パターンを表します。

UCS-2 GRAPHIC ストリング定数

Unicode データベースでは、可変長 UCS-2 GRAPHIC ストリング定数を指定する 16 進 UCS-2 GRAPHIC ストリングがサポートされます。 16 進数 UCS-2 GRAPHIC ストリング定数のフォーマットは、UX の後に、アポストロフィで囲んだ一つながりの文字を続けたものです。アポストロフィの間にある文字は、4 つの 16 進数字の偶数倍でなければなりません。 16 進数字の数は 16336 を超えてはなりません。これを超えると、エラー (SQLSTATE -54002) が戻されます。 16 進

UCS-2 GRAPHIC ストリング定数の形式が正しくない場合には、エラーが戻されます (SQLSTATE 42606)。 4 つの数字から成るグループはそれぞれ、1 つの UCS-2 GRAPHIC 文字に対応します。

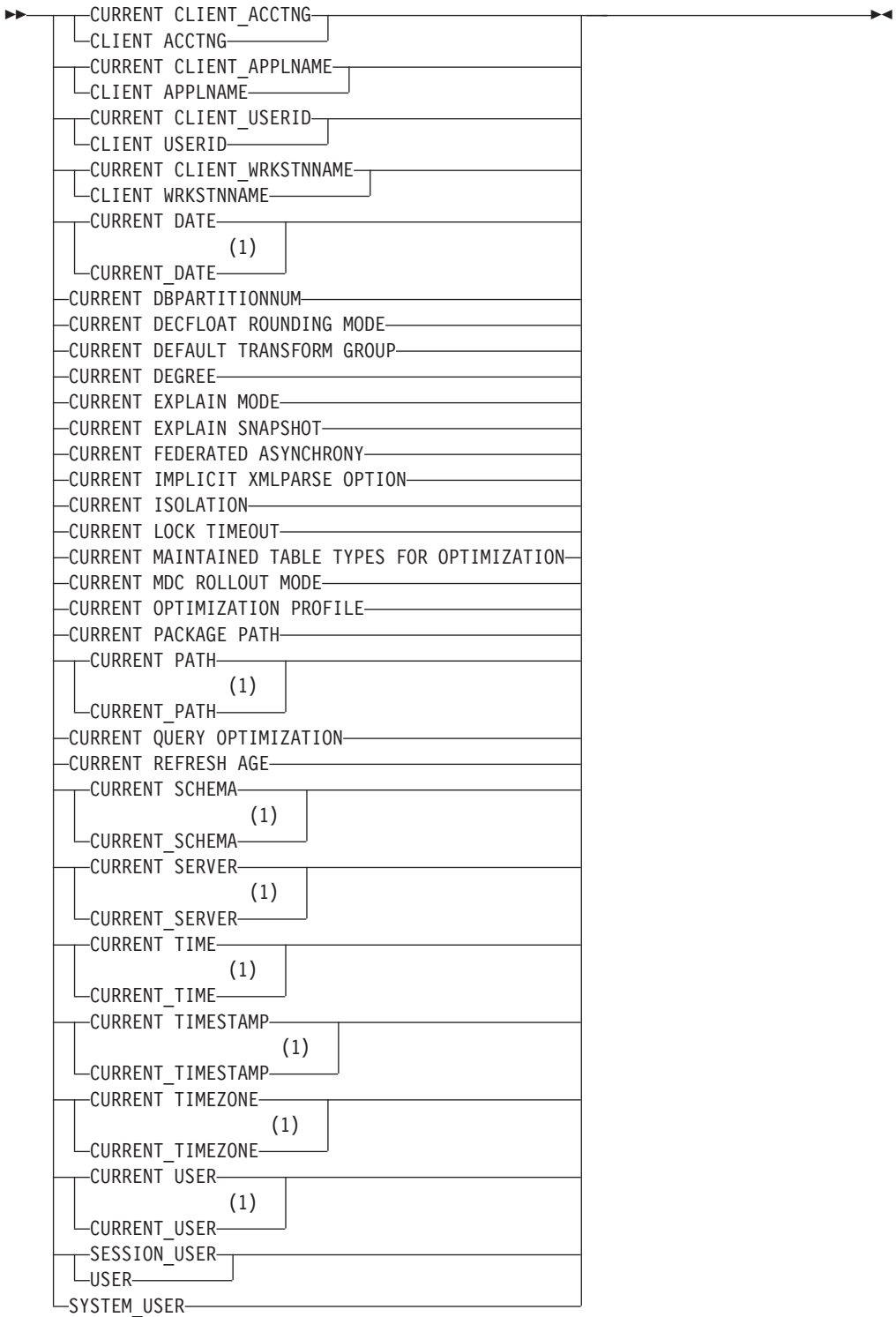
例:

```
UX'0042006F006200620079'
```

ASCII ストリング 'Bobby' の VARGRAPHIC パターンを表します。

特殊レジスター

特殊レジスター は、データベース・マネージャーによってアプリケーション・プロセスに対して定義されるストレージ域です。それは、SQL ステートメントで参照可能な情報を保管するのに使用されます。特殊レジスターの参照は、現行サーバーから提供される値の参照になります。値がストリングの場合、その CCSID は、現行サーバーのデフォルト CCSID になります。特殊レジスターの参照は、次のようにして行うことができます。



注:

1 SQL2003 Core 標準では、下線付きの書式が使用されます。

一部の特殊レジスターは、SET ステートメントを使用して更新できます。以下の表は、どの特殊レジスターを更新できるかを示しています。

表 14. 特殊レジスター

特殊レジスター	更新可能
CURRENT CLIENT_ACCTNG	いいえ
CURRENT CLIENT_APPLNAME	いいえ
CURRENT CLIENT_USERID	いいえ
CURRENT CLIENT_WRKSTNNAME	いいえ
CURRENT DATE	いいえ
CURRENT DBPARTITIONNUM	いいえ
CURRENT DECFLOAT ROUNDING MODE	いいえ
CURRENT DEFAULT TRANSFORM GROUP	はい
CURRENT DEGREE	はい
CURRENT EXPLAIN MODE	はい
CURRENT EXPLAIN SNAPSHOT	はい
CURRENT FEDERATED ASYNCHRONY	はい
CURRENT IMPLICIT XMLPARSE OPTION	はい
CURRENT ISOLATION	はい
CURRENT LOCK TIMEOUT	はい
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION	はい
CURRENT MDC ROLLOUT MODE	はい
CURRENT OPTIMIZATION PROFILE	はい
CURRENT PACKAGE PATH	はい
CURRENT PATH	はい
CURRENT QUERY OPTIMIZATION	はい
CURRENT REFRESH AGE	はい
CURRENT SCHEMA	はい
CURRENT SERVER	いいえ
CURRENT TIME	いいえ
CURRENT TIMESTAMP	いいえ
CURRENT TIMEZONE	いいえ
CURRENT USER	いいえ
SESSION_USER	はい
SYSTEM_USER	いいえ
USER	はい

特殊レジスターがルーチン内で参照されるとき、ルーチン内の特殊レジスターの値はその特殊レジスターが更新可能かどうかによって異なります。更新可能ではない特殊レジスターの場合、値はその特殊レジスターのデフォルト値に設定されます。更新可能な特殊レジスターの場合、初期値はルーチンの起動側から継承されて、ルーチン内の後続の SET ステートメントによって変更できます。

CURRENT CLIENT_ACCTNG

CURRENT CLIENT_ACCTNG (または CLIENT ACCTNG) 特殊レジスターには、この接続用に指定されたクライアント情報からのアカウンティング・ストリングの値が入ります。このレジスターのデータ・タイプは VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (sqleseti) API を使用して、アカウンティング・ストリングの値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: この接続のアカウンティング・ストリングの現行値を入手します。

```
VALUES (CURRENT CLIENT_ACCTNG)
  INTO :ACCT_STRING
```

CURRENT CLIENT_APPLNAME

CURRENT CLIENT_APPLNAME (または CLIENT APPLNAME) 特殊レジスターには、この接続用に指定されたクライアント情報からのアプリケーション名の値が入ります。このレジスターのデータ・タイプは VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (sqleseti) API を使用して、アプリケーション名の値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: この接続に使用されるアプリケーションを使用できる部門を選択します。

```
SELECT DEPT
  FROM DEPT_APPL_MAP
  WHERE APPL_NAME = CURRENT CLIENT_APPLNAME
```

CURRENT CLIENT_USERID

CURRENT CLIENT_USERID (または CLIENT USERID) 特殊レジスターには、この接続用に指定されたクライアント情報からのユーザー ID の値が入ります。このレジスターのデータ・タイプは VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (sqleseti) API を使用して、クライアント・ユーザー ID の値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: 現行のクライアント・ユーザー ID を使用している部門を検出します。

```
SELECT DEPT
  FROM DEPT_USERID_MAP
  WHERE USER_ID = CURRENT CLIENT_USERID
```

CURRENT CLIENT_WRKSTNNAME

CURRENT CLIENT_WRKSTNNAME (または CLIENT WRKSTNNAME) 特殊レジスターには、この接続用に指定されたクライアント情報からのワークステーション名の値が入ります。このレジスターのデータ・タイプは VARCHAR(255) です。このレジスターのデフォルト値は空ストリングです。

Set Client Information (sqleseti) API を使用して、ワークステーション名の値を変更できます。

sqleseti API を使用して指定した値はアプリケーションのコード・ページに入れられ、特殊レジスターの値はデータベースのコード・ページで保管されることに注意してください。 クライアント情報の設定時に使用されるデータ値によっては、特殊レジスターに保管されているデータ値がコード・ページ変換の際に切り捨てられることがあります。

例: この接続で使用されているワークステーション名を入手します。

```
VALUES (CURRENT CLIENT_WRKSTNNAME)
  INTO :WS_NAME
```

CURRENT DATE

CURRENT DATE (または CURRENT_DATE) 特殊レジスターは、アプリケーション・サーバーで SQL ステートメントが実行される時点の、時刻機構の読み取り値にもとづく日付を指定します。この特殊レジスターが単一の SQL ステートメントで何度も使用される場合、または単一のステートメントで CURRENT TIME または CURRENT TIMESTAMP と共に使用される場合、その値はすべて時刻機構の 1 回の読み取りに基づきます。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT DATE の継承はありません。

フェデレーテッド・システムでは、データ・ソースでの使用を目的とした照会で CURRENT DATE を使用できます。この照会が処理されて戻される日付は、フェデレーテッド・サーバーの CURRENT DATE レジスターから取得されたもので、データ・ソースから取得されたものではありません。

例: DB2 CLP から以下のコマンドを実行して、現在日付を取得します。

```
db2 values CURRENT DATE
```

例: 以下の例は、PROJECT 表を使用して、 MA2111 プロジェクト (PROJNO) のプロジェクト終了日付 (PRENDATE) に CURRENT DATE を設定しています。

```
UPDATE PROJECT
  SET PRENDATE = CURRENT DATE
  WHERE PROJNO = 'MA2111'
```

CURRENT DBPARTITIONNUM

CURRENT DBPARTITIONNUM 特殊レジスターは、ステートメントのコーディネーター・ノード番号を識別する INTEGER 値を指定します。アプリケーションから発行されるステートメントの場合は、アプリケーションの接続先のデータベース・パーティションがコーディネーターになります。ルーチンから発行されるステートメントの場合は、ルーチンが呼び出されるデータベース・パーティションがコーディネーターになります。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT DBPARTITIONNUM の継承はありません。

データベース・インスタンスがデータベース・パーティション分割をサポートするように定義されていない場合、CURRENT DBPARTITIONNUM は 0 を戻します。(これはつまり、db2nodes.cfg ファイルが存在しない場合です。パーティション・データベースの場合は、db2nodes.cfg ファイルがあり、そこにデータベース・パーティションの定義が入っている場合です。)

CURRENT DBPARTITIONNUM は、ある一定の条件に該当する場合に限り、CONNECT ステートメントで変更できます。

バージョン 8 以前のバージョンと互換性を持たせるため、DBPARTITIONNUM の部分はキーワード NODE に置き換えられます。

例: 以下の例では、アプリケーションが接続しているデータベース・パーティションの番号をホスト変数 APPL_NODE (整数) に設定しています。

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

CURRENT DECFLOAT ROUNDING MODE

CURRENT DECFLOAT ROUNDING MODE 特殊レジスターは、DECFLOAT 値に使用される丸めモードを指定します。

データ・タイプは VARCHAR(128) です。以下の丸めモードがサポートされています。

- ROUND_CEILING は、値を正の無限大の方向に丸めます。廃棄されたすべての桁がゼロであるか、符号が負の場合、(廃棄された桁の除去以外は) 結果は変わりません。そうでない場合、結果の係数は 1 だけ増分されます。
- ROUND_DOWN は、値を 0 の方向に丸めます (切り捨て)。廃棄された桁は無視されます。
- ROUND_FLOOR は、値を負の無限大の方向に丸めます。廃棄されたすべての桁がゼロであるか、符号が正の場合、(廃棄された桁の除去以外は) 結果は変わりません。そうでない場合、符号は負であり、結果の係数は 1 だけ増分されます。
- ROUND_HALF_EVEN は、値を最も近い値に丸めます。最も近い値がそれぞれ等距離の場合、最終の数字が偶数になるように丸めます。廃棄される数字が、左隣り桁の数の値の 2 分の 1 より大きい場合、結果の係数は 1 だけ増分されます。2 分の 1 より小さい場合、結果の係数は調整されません (つまり、廃棄される桁は無視されます)。そうでない場合、結果の係数は、その右端の数字が偶数の場合は変更されず、右端の数字が奇数の場合は 1 だけ増分されます (偶数にされます)。
- ROUND_HALF_UP は、値を最も近い値に丸めます。最も近い値がそれぞれ等距離の場合、値を切り上げます。廃棄される数字が、左隣り桁の数の値の 2 分の 1 より大きい場合、結果の係数は 1 だけ増分されます。そうでない場合、廃棄される桁は無視されます。

クライアント上の DECFLOAT 丸めモードの値は、SET CURRENT DECFLOAT ROUNDING MODE ステートメントを呼び出すことで、サーバー上のその値との一致を確認できます。ただしこのステートメントは、サーバーの丸めモードを変更するために使用することはできません。 CURRENT DECFLOAT ROUNDING MODE の初期値は、**decflt_rounding** データベース構成パラメーターによって判別され、このデータベース構成パラメーターの値を変更することによってのみ変更できます。

CURRENT DEFAULT TRANSFORM GROUP

CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターは、VARCHAR (18) 値を指定します。ここには、ユーザー定義構造化タイプの値をホスト・プログラムと交換するときに、動的 SQL ステートメントで使用するトランスフォーム・グループの名前を指定します。この特殊レジスターでは、静的 SQL ステートメントで使用する、または外部関数やメソッドを使ったパラメーターと結果の交換で使用するトランスフォーム・グループを指定しません。

その値は SET CURRENT DEFAULT TRANSFORM GROUP ステートメントによって設定することができます。値を設定しない場合、特殊レジスターの初期値は、空ストリングになります (ゼロの長さの VARCHAR)。

動的 SQL ステートメント (つまり、ホスト変数と相互作用するもの) では、値を交換するときに使用するトランスフォーム・グループの名前は、このレジスターに空ストリングが入っていない限り、この特殊レジスターの値と同じになります。レジスターに空ストリングが入っている場合 (SET CURRENT DEFAULT TRANSFORM GROUP ステートメントを使用して、値が設定されていない場合)、トランスフォームのために、DB2_PROGRAM トランスフォーム・グループが使われます。 構造化タイプ・サブジェクト用に DB2_PROGRAM トランスフォーム・グループが定義されていない場合、実行時にエラーが生じます (SQLSTATE 42741)。

例:

デフォルトのトランスフォーム・グループを MYSTRUCT1 に設定します。MYSTRUCT1 トランスフォームで定義される TO SQL および FROM SQL 関数は、ユーザー定義構造化タイプ変数とホスト・プログラムを交換するときに使います。

```
SET CURRENT DEFAULT TRANSFORM GROUP = MYSTRUCT1
```

この特殊レジスターに割り当てられた、デフォルトのトランスフォーム・グループの名前を検索します。

```
VALUES (CURRENT DEFAULT TRANSFORM GROUP)
```

CURRENT DEGREE

CURRENT DEGREE 特殊レジスターは、動的 SQL ステートメントを実行するときの、パーティション内並列処理の度合いを指定します。 (静的 SQL の場合、DEGREE BIND オプションが同じ制御機能として働きます。) このレジスターのデータ・タイプは CHAR(5) です。有効な値は、ANY、または 1 から 32 767 の範囲 (両端の値を含む) の整数のストリング表記です。

SQL ステートメントが動的に準備される時点で、整数として表現される CURRENT DEGREE の値が 1 である場合には、そのステートメントの実行にパーティション内並列処理は使用されません。

SQL ステートメントが動的に準備されるときに、整数として表される CURRENT DEGREE の値が 2 以上 32 767 以下である場合、そのステートメントの実行には、指定された度合いのパーティション内並列処理を伴う場合があります。

SQL ステートメントが動的に準備される時点で、 CURRENT DEGREE の値が ANY である場合、そのステートメントの実行には、データベース・マネージャーによって決定された度合いを用いたパーティション内並列処理を使用できます。

実際の実行時の並列処理の度合いは、以下の低い方になります。

- 最大照会度合 (**max_querydegree**) 構成パラメーターの値
- アプリケーション実行時の多重度
- SQL ステートメントのコンパイル度

intra_parallel のデータベース・マネージャー構成パラメーターが NO に設定される場合、最適化のために CURRENT DEGREE 特殊レジスターの値は無視され、ステートメントはパーティション内並列処理を使用しません。

値は、SET CURRENT DEGREE ステートメントを呼び出すことによって変更できます。

CURRENT DEGREE の初期値は、**dft_degree** データベース構成パラメーターによって判別されます。

CURRENT EXPLAIN MODE

CURRENT EXPLAIN MODE 特殊レジスターには、該当する動的 SQL ステートメントに関連のある Explain 機能の動作を制御するための VARCHAR(254) の値が入れられます。この機能は、Explain 情報を生成し、その情報を Explain 表に挿入します。この情報には、Explain スナップショットは入っていません。使用できる値は、YES、EXPLAIN、NO、REOPT、RECOMMEND INDEXES、および EVALUATE INDEXES です。 (静的 SQL の場合、EXPLAIN BIND オプションは同じ制御機能として働きます。 PREP および BIND コマンドの場合、EXPLAIN オプション値は YES、NO、および ALL です。)

YES Explain 機能を使用可能にし、動的 SQL ステートメントについての Explain 情報をそのステートメントのコンパイル時にキャプチャーします。

EXPLAIN
機能を使用可能にします。ただし、動的ステートメントは実行されません。

NO Explain 機能を使用不可にします。

REOPT
Explain 機能が使用可能になり、動的 (つまり増分バインド) SQL ステートメントに関する Explain 情報がキャプチャーされることになります。ただし、入力変数 (ホスト変数、特殊レジスター、グローバル変数、またはパラメーター・マーカー) の実際の値を使ってこのステートメントが再最適化された場合に限ります。

RECOMMEND INDEXES
各動的照会に一連の索引を推奨します。 ADVISE_INDEX 表に一連の索引を移植します。

EVALUATE INDEXES
推奨されている索引が存在するかのように、動的照会を Explain します。使用される索引は ADVISE_INDEX 表から選出されます。

初期値は NO です。値は、SET CURRENT EXPLAIN MODE ステートメントを呼び出すことによって変更できます。

CURRENT EXPLAIN MODE と CURRENT EXPLAIN SNAPSHOT 特殊レジスター値は、 Explain 機能が呼び出されている場合に相互に作用します。 CURRENT EXPLAIN MODE 特殊レジスター値の方は、 EXPLAIN BIND オプションとも相互に作用します。 RECOMMEND INDEXES と EVALUATE INDEXES を設定できるのは、 CURRENT EXPLAIN MODE レジスターの場合だけです。これらを設定するには、 SET CURRENT EXPLAIN MODE ステートメントを使用します。

例: ホスト変数 EXPL_MODE (VARCHAR(254)) を CURRENT EXPLAIN MODE 特殊レジスターの現在の値に設定します。

```
VALUES CURRENT EXPLAIN MODE
  INTO :EXPL_MODE
```

CURRENT EXPLAIN SNAPSHOT

CURRENT EXPLAIN SNAPSHOT 特殊レジスターには、 Explain スナップショット機能の動作を制御するための CHAR(8) の値が入れられます。この機能は、アクセス・プラン情報、演算子コスト、バインド実行時の統計などに関する情報を圧縮して生成するものです。

次に挙げるステートメントだけがこのレジスターの値として認められます。すなわち、CALL、コンパウンド SQL (動的)、DELETE、INSERT、MERGE、REFRESH、SELECT、SELECT INTO、SET INTEGRITY、UPDATE、VALUES、および VALUES INTO です。使用できる値は、YES、EXPLAIN、NO、および REOPT です。(静的 SQL の場合、EXPLSNAP BIND オプションが同じ制御機能として働きます。 PREP および BIND コマンドの場合、EXPLSNAP オプション値は YES、NO、および ALL です。)

YES Explain スナップショット機能を使用可能にし、動的 SQL ステートメントがコンパイルされるときの、そのステートメントの内部表記のスナップショットを取り出します。

EXPLAIN
Explain スナップショット機能を使用可能にします。ただし、動的ステートメントは実行されません。

NO Explain スナップショット機能を使用不可にします。

REOPT
Explain 機能が使用可能になり、動的 (つまり増分バインド) SQL ステートメントに関する Explain 情報がキャプチャーされることになります。ただし、入力変数 (ホスト変数、特殊レジスター、グローバル変数、またはパラメーター・マーカー) の実際の値を使ってこのステートメントが再最適化された場合に限ります。

初期値は NO です。値は、SET CURRENT EXPLAIN SNAPSHOT ステートメントを呼び出すことによって変更できます。

CURRENT EXPLAIN SNAPSHOT と CURRENT EXPLAIN MODE 特殊レジスター値は、 Explain 機能が呼び出されている場合に相互に作用します。 CURRENT EXPLAIN SNAPSHOT 特殊レジスター値の方は、 EXPLSNAP BIND オプションとも相互に作用します。

例: 以下の例は、ホスト変数 EXPL_SNAP (char(8)) に、 CURRENT EXPLAIN SNAPSHOT 特殊レジスターの現在の値を設定するものです。

```
VALUES CURRENT EXPLAIN SNAPSHOT
  INTO :EXPL_SNAP
```

CURRENT FEDERATED ASYNCHRONY

CURRENT FEDERATED ASYNCHRONY 特殊レジスターは、動的 SQL ステートメントを実行するときの非同期の度合いを指定します。(FEDERATED_ASYNCHRONY バインド・オプションは、静的 SQL に対する同じ制御を提供します。) このレジスターのデータ・タイプは INTEGER です。有効な値は ANY (-1 を表す)、または 0 から 32767 までの整数で、範囲は両端を含みます。 SQL ステートメントが動的に準備される場合、CURRENT FEDERATED ASYNCHRONY の値は以下のようになります。

- 0。この場合、ステートメントの実行は非同期を使用しない。
- 0 より大きく、32767 以下。このステートメントの実行には、指定された度合いを使用する非同期が関係することがあります。
- ANY (-1 を表す)。このステートメントの実行には、データベース・マネージャーにより決定される度合いを使用する非同期が関係することがあります。

CURRENT FEDERATED ASYNCHRONY 特殊レジスターの値は、SET CURRENT FEDERATED ASYNCHRONY ステートメントを起動して変更することができます。

動的ステートメントがコマンド行プロセッサー (CLP) を介して発行される場合、CURRENT FEDERATED ASYNCHRONY 特殊レジスターの初期値は、**federated_async** データベース・マネージャー構成パラメーターで決まります。動的ステートメントが、バインドされるアプリケーションの一部である場合、初期値は FEDERATED_ASYNCHRONY バインド・オプションで決まります。

例: ホスト変数 FEDASYNC (INTEGER) を CURRENT FEDERATED ASYNCHRONY 特殊レジスターの値に設定します。

```
VALUES CURRENT FEDERATED ASYNCHRONY INTO :FEDASYNC
```

CURRENT IMPLICIT XMLPARSE OPTION

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターは、シリアライズされた XML データが、妥当性検査されずに暗黙的に DB2 サーバーにより構文解析される場合に使用される空白文字処理オプションを指定します。暗黙の妥当性検査を行わない構文解析操作は、SQL ステートメントが XML ホスト変数、または XMLVALIDATE 関数の引数ではない暗黙的または明示的な型付き XML パラメーター・マーカーを処理する場合に実行されます。このレジスターのデータ・タイプは VARCHAR(19) です。

CURRENT IMPLICIT XMLPARSE 特殊レジスターの値は、SET CURRENT IMPLICIT XMLPARSE ステートメントを起動して変更することができます。その初期値は 'STRIP WHITESPACE' です。

例:

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターの値を検索して、CURXMLPARSEOPT と呼ばれるホスト変数に入れます。

```
EXEC SQL VALUES (CURRENT IMPLICIT XMLPARSE OPTION) INTO :CURXMLPARSEOPT;
```

CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスターを 'PRESERVE WHITESPACE' に設定します。

```
SET CURRENT IMPLICIT XMLPARSE OPTION = 'PRESERVE WHITESPACE'
```

以下の SQL ステートメントの実行時に、空白文字は保存されます。

```
INSERT INTO T1 (XMLCOL1) VALUES (?)
```

CURRENT ISOLATION

CURRENT ISOLATION 特殊レジスターは、現行のセッション内で発行された動的 SQL ステートメントの分離レベル (他の並行セッションに関する) を識別する CHAR(2) 値を収容します。

使用できる値は次のとおりです。

(ブランク)
未設定。パッケージの分離属性を使用します。

UR 非コミット読み取り

CS カーソル固定

RR 反復可能読み取り

RS 読み取り固定

CURRENT ISOLATION 特殊レジスターの値は、SET CURRENT ISOLATION ステートメントにより変更することができます。

SET CURRENT ISOLATION ステートメントがセッション内で発行されるまでか、または SET CURRENT ISOLATION に対して RESET が指定された後で、CURRENT ISOLATION 特殊レジスターはブランクに設定され、動的 SQL ステートメントには適用されません。使用される分離レベルは、動的 SQL ステートメントを発行したパッケージの分離属性から取られます。 SET CURRENT ISOLATION ステートメントが発行されると、CURRENT ISOLATION 特殊レジスターは、ステートメントを発行したパッケージの設定に関係なく、セッション内でコンパイルされた後続のすべての動的 SQL ステートメントのための分離レベルを提供します。これが有効であるのは、セッションが終了するまでか、または RESET オプションを使用して SET CURRENT ISOLATION ステートメントが発行されるまでです。

例: ホスト変数 ISOLATION_MODE (CHAR(2)) を CURRENT ISOLATION 特殊レジスターに保管されている現在の値に設定します。

```
VALUES CURRENT ISOLATION
  INTO :ISOLATION_MODE
```

CURRENT LOCK TIMEOUT

CURRENT LOCK TIMEOUT 特殊レジスターは、待機の開始から何秒経過したら、ロックをかけられなかったことを示すエラーが戻されるかを指定します。この特殊レジスターは、行、表、索引キー、MDC ブロック、および XML パス (XPath) の各ロックに影響を与えます。このレジスターのデータ・タイプは INTEGER です。

CURRENT LOCK TIMEOUT 特殊レジスターの有効値は、 -1 以上 32767 以下の整数です。この特殊レジスターは、NULL 値に設定することもできます。 -1 の値は、タイムアウトはとられないので、ロックの解放またはデッドロックの検出までアプリケーションは待機することを指定します。 0 の値は、アプリケーションがロックを待機しないことを指定します。ロックをかけられなかった場合、ただちにエラーが戻されます。

CURRENT LOCK TIMEOUT 特殊レジスターの値は、 SET CURRENT LOCK TIMEOUT ステートメントを起動して変更することができます。その初期値はヌルですが、その場合、 **locktimeout** データベース構成パラメーターの現行値がロックの待機時間に使われ、この値が、この特殊レジスターの値として戻されます。

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターは、 VARCHAR(254) の値を指定します。この値は、動的 SQL 照会の処理を最適化する際に考慮される表のタイプを識別するものです。マテリアライズ照会表は、組み込み静的 SQL 照会で使用されることは決してありません。

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION の初期値は SYSTEM です。その値は SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ステートメントによって変更することができます。

CURRENT MDC ROLLOUT MODE

CURRENT MDC ROLLOUT MODE 特殊レジスターは、ロールアウト処理の対象となる DELETE ステートメントのマルチディメンション・クラスタリング (MDC) 表の動作を指定します。

このレジスターのデフォルト値は、DB2_MDC_ROLLOUT レジストリー変数により決定されます。値は、SET CURRENT MDC ROLLOUT MODE ステートメントを呼び出すことによって変更できます。 CURRENT MDC ROLLOUT MODE 特殊レジスターが特定の値に設定された場合、ロールアウトの対象となる後続の DELETE ステートメントの実行動作に影響が出ます。 DELETE ステートメントは、動作を変更するために再コンパイルする必要はありません。

CURRENT OPTIMIZATION PROFILE

CURRENT OPTIMIZATION PROFILE 特殊レジスターは、最適化の目的で動的に準備される DML ステートメントで使用される、最適化プロファイルの修飾名を指定します。

初期値は、NULL 値です。値は、SET CURRENT OPTIMIZATION PROFILE ステートメントを呼び出すことによって変更できます。スキーマ名で修飾されていない最適化プロファイルは、暗黙のうちに CURRENT DEFAULT SCHEMA 特殊レジスターの値で修飾されます。

例 1: 最適化プロファイルを 'JON.SALES' に設定します。

```
SET CURRENT OPTIMIZATION PROFILE = JON.SALES
```

例 2: この接続の最適化プロファイル名の現行値を入手します。

```
VALUES (CURRENT OPTIMIZATION PROFILE) INTO :PROFILE
```

CURRENT PACKAGE PATH

CURRENT PACKAGE PATH 特殊レジスターは、 SQL ステートメントの実行時に必要なパッケージの参照の解決に使用されるパスを特定する VARCHAR(4096) 値を指定します。

この値は、空またはブランクのストリングか、または二重引用符で区切られてさらにコンマで区切られた 1 つ以上のスキーマ名のリストのどちらでもかまいません。ストリング中で二重引用符を使用する場合はすべて、区切り ID の通常の使用法と同様に 2 つの二重引用符で表記する必要があります。区切り文字とコンマは、この特殊レジスターの長さの一部を成します。

この特殊レジスターは、静的と動的の両方のステートメントで使用できます。

ユーザー定義の関数、メソッド、またはプロシージャー内の CURRENT PACKAGE PATH の初期値は、呼び出し側アプリケーションから継承されます。すなわち、CURRENT PACKAGE PATH の初期値は空ストリングです。アプリケーションが SET CURRENT PACKAGE PATH ステートメントを使ってスキーマ・リストを明示的に指定していた場合のみ、その初期値はスキーマのリストになります。

例:

アプリケーションは、複数の SQLJ パッケージ (スキーマ SQLJ1 と SQLJ2 に入っています) と、 1 つの JDBC パッケージ (DB2JAVA に入っています) を使用することになっています。 CURRENT PACKAGE PATH 特殊レジスターを設定して SQLJ1、SQLJ2、および DB2JAVA をこの順序で調べます。

```
SET CURRENT PACKAGE PATH = "SQLJ1", "SQLJ2", "DB2JAVA"
```

ホスト変数 HVPKLIST に、CURRENT PACKAGE PATH 特殊レジスターに現在保管されている値を設定します。

```
VALUES CURRENT PACKAGE PATH INTO :HVPKLIST
```

CURRENT PATH

CURRENT PATH (または CURRENT_PATH) 特殊レジスターは、動的に作成された SQL ステートメントの関数参照やデータ・タイプ参照の解決に使用される SQL パスを識別する、 VARCHAR(2048) 値を指定します。 CURRENT FUNCTION PATH は、CURRENT PATH の同義語です。 CURRENT PATH は、 CALL ステートメントのストアード・プロシージャー参照を解決するのにも使用されます。初期値は、後述するデフォルト値です。静的 SQL の場合は、FUNCPATH BIND オプションで関数およびデータ・タイプの解決のための SQL パスを指定できます。

CURRENT PATH 特殊レジスターには、二重引用符で囲まれ、コンマで区切られた、1 つ以上のスキーマ名のリストが入っています。たとえば、データベース・マネージャーがまず FERMAT スキーマを参照し、次いで XGRAPHIC スキーマ、最後に SYSIBM スキーマを参照するように指定する SQL パスは、 CURRENT PATH 特殊レジスターでは以下のように戻されます。

```
"FERMAT","XGRAPHIC","SYSIBM"
```

デフォルト値は 『SYSIBM』,『SYSFUN』,『SYSPROC』,『SYSIBMADM』,*X* (*X* は USER 特殊レジスターの値) で、それぞれは二重引用符で区切られます。値は、SET CURRENT PATH ステートメントを呼び出すことによって変更できます。SYSIBM スキーマを指定する必要はありません。 SQL パスにスキーマが指定されていなければ、暗黙的にそのスキーマが最初のスキーマとして見なされます。暗黙的に想定されている場合、SYSIBM は 2048 バイトを一切扱いません。

スキーマ名で修飾されないデータ・タイプは、同じ非修飾名のデータ・タイプが入っている SQL パスの最初のスキーマで暗黙的に修飾されます。この規則には、CREATE TYPE (Distinct)、CREATE FUNCTION、COMMENT、および DROP ステートメントの部分で概説されているように例外があります。

例: SYSCAT.ROUTINES カタログ・ビューを使用して、CURRENT PATH 特殊レジスターにスキーマ名が含まれているために、ルーチン名を修飾せずに呼び出せるすべてのユーザー定義ルーチンを検索します。

```
SELECT ROUTINENAME, ROUTINESCHEMA FROM SYSCAT.ROUTINES
  WHERE POSITION (ROUTINESCHEMA, CURRENT PATH, CODEUNITS16) <> 0
```

CURRENT QUERY OPTIMIZATION

CURRENT QUERY OPTIMIZATION 特殊レジスターには、動的 SQL ステートメントのバインド時に、データベース・マネージャーによって行われる照会最適化のクラスを制御する INTEGER 値が入れられます。 QUERYOPT BIND オプションは、静的 SQL ステートメントの照会クラスの最適化を制御します。可能な値の範囲は 0 から 9 です。例えば、照会最適化クラスが 0 (最小の最適化) に設定されている場合、この特殊レジスターの値は 0 です。デフォルト値は、**dft_queryopt** データベース構成パラメーターによって決まります。値は、SET CURRENT QUERY OPTIMIZATION ステートメントを呼び出すことによって変更できます。

例: 以下の例は、SYSCAT.PACKAGES カタログ・ビューを使用して、 CURRENT QUERY OPTIMIZATION 特殊レジスターの現行値と同じ設定でバインドされたすべてのプランを検索しています。

```
SELECT PKGNAME, PKGSCHEMA FROM SYSCAT.PACKAGES
  WHERE QUERYOPT = CURRENT QUERY OPTIMIZATION
```

CURRENT REFRESH AGE

CURRENT REFRESH AGE 特殊レジスターは、データ・タイプが DECIMAL(20,6) のタイム・スタンプ期間値を指定します。これは、タイム・スタンプが作成された、キャッシュ・データ・オブジェクトでの特定のイベント (例えば、システムが保守する REFRESH DEFERRED マテリアライズ照会表で REFRESH TABLE ステートメントを処理するなど) が起きた後、照会の処理の最適化にそのキャッシュ・データ・オブジェクトを使用できる最大期間です。 CURRENT REFRESH AGE の値が 99 999 999 999 999 で、照会の最適化クラスが 5 以上の場合は、動的 SQL 照会の処理を最適化する際に、 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION で指定されたタイプの表が考慮されます。

CURRENT REFRESH AGE の値は、0 または 99 999 999 999 999 でなければなりません。初期値は 0 です。値は、SET CURRENT REFRESH AGE ステートメントを呼び出すことによって変更できます。

CURRENT SCHEMA

CURRENT SCHEMA (または CURRENT_SCHEMA) 特殊レジスターは、動的に作成された SQL ステートメントで可能な場合に、データベース・オブジェクト参照を修飾するのに使用されるスキーマ名を識別する VARCHAR(128) 値を指定します。 DB2 for z/OS との互換性を保つため、CURRENT SQLID (または CURRENT_SQLID) は CURRENT SCHEMA の代わりに指定できます。

CURRENT SCHEMA の初期値は、現行セッション・ユーザーの許可 ID です。値は、SET SCHEMA ステートメントを呼び出すことによって変更できます。

CURRENT SCHEMA の設定は、Explain 表の Explain 機能の選択に影響を与えません。

QUALIFIER BIND オプションは、動的に作成された SQL ステートメントについて可能な場合に、データベース・オブジェクト参照を修飾するのに使用されるスキーマ名を制御します。

例: オブジェクト修飾のスキーマを 'D123' に設定します。

```
SET CURRENT SCHEMA =  'D123'
```

CURRENT SERVER

CURRENT SERVER (または CURRENT_SERVER) 特殊レジスターには、現在のデータベース・サーバー (アプリケーション・サーバーとして言及されることもある) を識別する VARCHAR(18) の値が入れられます。このレジスターにはデータベースの実際の名前 (別名ではない) が入れられます。

CURRENT SERVER は、ある一定の条件に該当する場合に限り、CONNECT ステートメントで変更できます。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT SERVER の継承はありません。

例: 以下の例は、アプリケーションが接続されているデータベース・サーバーの名前をホスト変数 APPL_SERVE (VARCHAR(18)) に設定しています。

```
VALUES CURRENT SERVER INTO :APPL_SERVE
```

CURRENT TIME

CURRENT TIME (または CURRENT_TIME) 特殊レジスターは、アプリケーション・サーバーで SQL ステートメントが実行される時点の時刻機構の読み取り値に基づく時刻を指定します。この特殊レジスターが単一の SQL ステートメントで何度も使用される場合、または単一のステートメントで CURRENT DATE または CURRENT TIMESTAMP と共に使用される場合、その値はすべて時刻機構の 1 回の読み取りに基づく値です。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT TIME の継承はありません。

フェデレーテッド・システムでは、データ・ソースでの使用を目的とした照会で CURRENT TIME を使用できます。この照会が処理されて戻される時刻は、フェデレーテッド・サーバーの CURRENT TIME レジスターから取得されたもので、データ・ソースから取得されたものではありません。

例: DB2 CLP から以下のコマンドを実行して、現在時刻を取得します。

```
db2 values CURRENT TIME
```

例: 以下の例は、CL_SCHED 表を使って、その日のそれ以降に開始される (STARTING) すべてのクラス (CLASS_CODE) を選択するものです。現在のクラスの DAY 列の値は 3 です。

```
SELECT CLASS_CODE FROM CL_SCHED
  WHERE STARTING > CURRENT TIME AND DAY = 3
```

CURRENT TIMESTAMP

CURRENT TIMESTAMP (または CURRENT_TIMESTAMP) 特殊レジスターは、アプリケーション・サーバーで SQL ステートメントが実行される時点の、時刻機構の読み取り値にもとづくタイム・スタンプを指定します。この特殊レジスターが単一の SQL ステートメントで何度も使用される場合、または単一のステートメントで CURRENT DATE または CURRENT TIME と共に使用される場合、その値はすべて時刻機構の 1 回の読み取りに基づく値です。分離 CURRENT TIMESTAMP 特殊レジスターが同じ値に戻ることを要求することは可能です。固有な値が必要な場合は、GENERATE_UNIQUE 機能、シーケンス、または ID 列の使用を考慮してください。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT TIMESTAMP の継承はありません。

フェデレーテッド・システムでは、データ・ソースでの使用を目的とした照会で CURRENT TIMESTAMP を使用できます。この照会が処理されて戻されるタイム・スタンプは、フェデレーテッド・サーバーの CURRENT TIMESTAMP レジスターから取得されたもので、データ・ソースから取得されたものではありません。

例: 以下の例は、1 つの行を IN_TRAY 表に挿入するものです。RECEIVED 列の値は、その行の挿入時点を示すタイム・スタンプでなければなりません。他の 3 つの列の値は、ホスト変数 SRC (char(8))、SUB (char(64))、および TXT (VARCHAR(200)) から取られたものです。

```
INSERT INTO IN_TRAY
  VALUES (CURRENT TIMESTAMP, :SRC, :SUB, :TXT)
```

CURRENT TIMEZONE

CURRENT TIMEZONE (または CURRENT_TIMEZONE) 特殊レジスターには、UTC (協定世界時 (Coordinated Universal Time)。旧 GMT。) とアプリケーション・サーバーのローカル時との差が入れられます。この差は、時刻期間 (最初の 2 桁が時間数、次の 2 桁が分数、最後の 2 桁が秒数である 10 進数) によって表現されます。時間数の数値は -24 と 24 を除く -24 と 24 の間です。ローカル時刻から CURRENT TIMEZONE を減算すると、ローカル時刻を UTC に変換できます。時刻は、SQL ステートメントが実行されるときに、オペレーティング・システムの時刻から計算されます。 (CURRENT TIMEZONE の値は、C のランタイム関数によって決まります。)

CURRENT TIMEZONE 特殊レジスターは、時刻やタイム・スタンプの算術演算など、 DECIMAL(6,0) のデータ・タイプの式が使用されるところならどこででも使用できます。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT TIMEZONE の継承はありません。

例: 以下の例は、RECEIVED 列の UTC タイム・スタンプを使って、 IN_TRAY 表にレコードを挿入します。

```
INSERT INTO IN_TRAY VALUES (
  CURRENT TIMESTAMP - CURRENT TIMEZONE,
  :source,
  :subject,
  :notetext )
```

CURRENT USER

CURRENT USER (または CURRENT_USER) 特殊レジスターは、ステートメントの許可に使用される許可 ID を指定します。静的 SQL ステートメントの場合、値は、パッケージをバインドする際に使用される許可 ID を表します。動的 SQL ステートメントの場合、値は、 DYNAMICRULES(RUN) BIND オプションでバインドされたパッケージの SESSION_USER 特殊レジスターの値と同じになります。このレジスターのデータ・タイプは VARCHAR(128) です。

例: スキーマが CURRENT USER 特殊レジスターの値と一致する表名を選択します。

```
SELECT TABNAME FROM SYSCAT.TABLES
  WHERE TABSCHEMA = CURRENT USER AND TYPE = 'T'
```

このステートメントが静的 SQL ステートメントとして実行される場合は、このステートメントを収めたパッケージのバインド・プログラムと一致するスキーマ名の付いた表が戻されます。このステートメントが動的 SQL ステートメントとして実行される場合は、 SESSION_USER 特殊レジスターの現行の値と一致するスキーマ名の表が戻されます。

SESSION_USER

SESSION_USER 特殊レジスターは、現行セッションに使用される許可 ID を指定します。このレジスターの値は、DYNAMICRULES の実行動作がパッケージに対して有効な場合に、動的 SQL ステートメントの許可検査に使用されます。このレジスターのデータ・タイプは VARCHAR(128) です。

新しい接続での SESSION_USER の初期値は、SYSTEM_USER 特殊レジスターの値と同じです。値は、SET SESSION AUTHORIZATION ステートメントを呼び出すことで変更できます。

SESSION_USER は、USER 特殊レジスターの同義語です。

例: 動的 SQL を使用してどのルーチンを実行できるかを判別します。DYNAMICRULES の実行振る舞いが、ルーチンを呼び出す動的 SQL の発行元パッケージに対して有効であることを想定します。

```
SELECT SCHEMA, SPECIFICNAME FROM SYSCAT.ROUTINEAUTH
  WHERE GRANTEE = SESSION_USER
  AND EXECUTEAUTH IN ('Y', 'G')
```

SYSTEM_USER

SYSTEM_USER 特殊レジスターは、データベースに接続するユーザーの許可 ID を指定します。このレジスターの値は、別の許可 ID を持つユーザーとして接続しない限り、変更できません。このレジスターのデータ・タイプは VARCHAR(128) です。

SET SESSION AUTHORIZATION ステートメントの説明にある、『例』を参照してください。

USER

USER 特殊レジスターは、データベースでアプリケーションが始動するときにデータベース・マネージャーに渡されるランタイム許可 ID を指定します。このレジスターのデータ・タイプは VARCHAR(128) です。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの USER の継承はありません。

例: ユーザー自身が IN_TRAY 表に入れたすべてのメモを、表から選択します。

```
SELECT * FROM IN_TRAY
  WHERE SOURCE = USER
```

グローバル変数

グローバル変数は、SQL ステートメントでアクセスおよび変更できる名前付きメモリー変数です。

グローバル変数を使用すると、SQL ステートメント間でのリレーショナル・データの共用を、そのデータ転送をサポートするアプリケーション・ロジックがなくても実行できます。グローバル変数へのアクセスは、GRANT (グローバル変数特権) および REVOKE (グローバル変数特権) ステートメントを使用して制御できます。

DB2 は、作成されたセッション・グローバル変数をサポートします。セッション・グローバル変数 は特定のセッションに関連付けられており、その内容はそのセッションに固有の値です。作成されたセッション・グローバル変数は、その変数が定義されているデータベースに対して実行されるすべてのアクティブ SQL ステートメントで使用できます。セッション・グローバル変数は複数のセッションに関連付けることができますが、その値はセッションごとに固有になります。作成されたセッション・グローバル変数およびそれらに関連付けられた特権はシステム・カタログで定義されます。

グローバル変数名は、修飾された 2 部構成の名前となります。グローバル変数がスキーマ名を指定せずに参照された場合、ネーム解決のために SQL パスが使用されます。静的 SQL ステートメントに対する SQL パスは、 FUNCPATH BIND オプションを使って指定されます。動的 SQL ステートメントの場合、 SQL パスは CURRENT PATH 特殊レジスターの値です。

パッケージ内の静的 SQL ステートメントの場合、グローバル変数はバインド操作中に解決されます。ビュー、トリガー、および SQL ルーチンでは、データベース・オブジェクトの作成時に解決されます。グローバル変数のいずれかで再度解決が実行される場合、例えば別のスキーマにある名前と同じ名前 (SQL パスにも存在している) で新しいグローバル変数が追加されていると、動作が変わってしまうおそれがあります。これが起きないようにするためにデータベース・マネージャーは、必要であれば常に従来のバインディング・セマンティクスを適用します。これは、最後の明示的なバインドのタイム・スタンプより前に定義された SQL パスのグローバル変数だけが、グローバル変数のネーム解決で考慮されることを意味します。

グローバル変数が 1 つの SQL ステートメント内、または 1 つのトリガー、ビュー、またはルーチン内で参照される場合、そのステートメントまたはオブジェクトのグローバル変数の完全修飾名の従属関係が記録されます。また該当する場合、ステートメントで使用されている許可 ID に、そのグローバル変数に対する READ 特権があるかが検査されます。

グローバル変数は、決定論的である必要がないすべての式の中で参照できます。以下の状況では決定論的式が必要となり、グローバル変数は使用できません。

- チェック制約
- 生成列の定義
- 即時リフレッシュ・マテリアライズ照会表 (MQT)

グローバル変数の値は、SET、SELECT INTO、または VALUES INTO ステートメントを使用して変更できます。その値が CALL ステートメントの OUT または

INOUT パラメーターの引数である場合にも変更できます。

関数

関数 とは、特定の操作に名前、つまり関数名を付けたもので、関数名の後には括弧に囲まれた 1 つ以上のオペランドが続きます。たとえば TIMESTAMP 関数は、入力データ値として DATE および TIME タイプを受け取り、 TIMESTAMP という結果を生成します。 関数には組み込み関数とユーザー定義関数があります。

- *組み込み関数* はデータベース・マネージャーとともに提供されます。これは、単一の結果値を返し、SYSIBM スキーマの一部と識別されます。そのような関数には、列関数 (たとえば AVG)、演算子関数 (たとえば +)、および cast 関数 (たとえば DECIMAL) などがあります。
- *ユーザー定義関数*は、(CREATE FUNCTION ステートメントを使用して) SYSCAT.ROUTINES のデータベースに登録されている関数です。ユーザー定義関数は SYSIBM スキーマの一部ではありません。このような関数の集合の 1 つに、データベース・マネージャーに用意されている SYSFUN という名前のスキーマがあります。

  ユーザー定義関数は、データベース・エンジンそのものにおいて適用できる関数定義 (ユーザーまたは第三者ベンダー作成の) を追加することで、データベース・システムの機能を拡張することができます。データベース関数が拡張されれば、アプリケーションが使用するのと同じ関数をデータベースがエンジン内で活用することができ、それによってアプリケーションとデータベースとの間の相互作用が強化されます。

外部、SQL、およびソース・ユーザー定義関数

ユーザー定義関数は、外部関数、SQL 関数、またはソース関数とすることができます。*外部関数* は、オブジェクト・コード・ライブラリーへの参照と、関数呼び出し時に実行されるそのライブラリー内の関数によってデータベースに対して定義されます。列関数を外部関数にすることはできません。 *SQL 関数* は、SQL RETURN ステートメントだけを使用して、データベースに対して定義されます。スカラー値、行、または表のいずれかを戻すことができます。 SQL 関数を、列関数とすることはできません。*ソース派生関数* は、データベースがすでに認識している別の組み込み関数またはユーザー定義関数への参照によって、データベースに対して定義されます。ソース関数からの派生関数はスカラー関数または列関数です。これらは、ユーザー定義タイプの既存の関数のサポートに有用です。

スカラー、列、行、および表のユーザー定義関数

各ユーザー定義関数は、スカラー関数、列関数、または表関数として類別することもできます。*スカラー関数* は、呼び出されるたびに単一値の応答を戻す関数です。たとえば、組み込み関数 SUBSTR() はスカラー関数です。スカラー UDF は、外部関数とソースからの派生関数のどちらであってもかまいません。

列関数 は、概念上、類似値の集合 (列) を渡され、単一値の応答を戻す関数です。DB2 では、場合によっては*集約関数* と呼ばれることもあります。列関数の例として、組み込み関数 AVG() があります。外部列 UDF を DB2 に対して定義することはできませんが、組み込み列関数のいずれかをソースとして派生する列 UDF を定義することができます。これは、特殊タイプに対して有用です。たとえば、特殊タ

イプ SHOESIZE が基本タイプ INTEGER を使用して定義されている場合、組み込み関数 AVG(INTEGER) をソースとする UDF AVG(SHOESIZE) を定義することができ、これは列関数になります。

行関数 とは、値を一行で戻す関数です。これは、構造化タイプの属性値を行の値に割り当てるトランスフォーム関数としてのみ、使うことができます。行関数は、SQL 関数と定義する必要があります。

表関数 は、その関数を参照する SQL ステートメントに表を戻す関数です。SELECT ステートメントの FROM 節でのみ参照することができます。このような関数を使用して、DB2 データ以外のデータに SQL 言語処理能力を適用することや、このようなデータを DB2 表に変換することができます。たとえば、ファイルを取り出してそれを表に変換することや、WWW からデータをサンプリングしてそれを表にすること、あるいは Lotus Notes® データベースにアクセスして、日付、送信元、メッセージのテキストなどのすべてのメール・メッセージに関する情報を戻すこと、などを行うことができます。このような情報は、データベースの他の表と結合することができます。表関数は、外部関数または SQL 関数と定義できます。(表関数はソース関数であってはなりません。)

関数シグニチャー

関数は、そのスキーマ、関数名、パラメーター数、およびそのパラメーターのデータ・タイプによって識別されます。これは*関数シグニチャー* と呼ばれ、データベース内で固有である必要があります。例: TEST.RISK(INTEGER)。パラメーターの数やパラメーターのデータ・タイプが違っていれば、 1 つのスキーマに同じ名前の関数が複数存在してもかまいません。複数の関数インスタンスのある関数名は、*多重定義* 関数と呼ばれます。ある関数名があるスキーマ内で多重定義される場合、そのスキーマにはその名前で 2 つ以上の関数があるということです。これらの関数は、それぞれ別々のパラメーター・タイプをもっていなければなりません。関数名は SQL パスにおいても多重定義可能です。その場合、そのパス内にその名前の付いた関数が 2 つ以上あることになりますが、必ずしもパラメーター・タイプがそれぞれ異なっている必要はありません。

関数を呼び出すには、後に括弧に入った引数のリストの付いた修飾名 (スキーマ名および関数名) を参照します (使用可能なコンテキストで)。また、スキーマ名を指定せずに関数を呼び出すことも可能であり、その場合は、異なるスキーマの関数のうち、同じパラメーターまたは許容パラメーターをもつ可能な関数を選択できることになります。 このような場合、関数解決に役立つ *SQL パス* が使用されます。SQL パスとは、同じ名前、同じパラメーター数、および受け入れ可能なデータ・タイプを持つ関数を識別するために探索されるスキーマのリストです。静的 SQL ステートメントに対する SQL パスは、 FUNCPATH BIND オプションを使って指定されます。動的 SQL ステートメントの場合、 SQL パスは CURRENT PATH 特殊レジスターの値です。

関数へのアクセスは、EXECUTE 特権を使って制御します。個々の関数または一連の関数を誰が実行できて誰ができないかを指定するには、 GRANT および REVOKE ステートメントを使用します。関数を呼び出すには、EXECUTE 特権 (つまり DBADM 権限) が必要です。関数の定義者には、自動的に EXECUTE 特権が与えられます。外部関数や、基礎となるすべてのオブジェクトでの WITH GRANT オプションをもつ SQL 関数の定義者には、関数に対する EXECUTE 特権をもった

WITH GRANT オプションも与えられます。定義者 (または SYSADM または DBADM) は次にそれを、任意の SQL ステートメントからの関数の呼び出し、任意の DDL ステートメント (たとえば CREATE VIEW、CREATE TRIGGER、または制約の定義時) の参照、またはこの関数をソースとして派生する別の関数の作成を行いたいと考えているユーザーに付与する必要があります。 EXECUTE 特権をユーザーに付与しないと、この関数は、一致性では優れていても関数解決アルゴリズムでの検討対象にはなりません。組み込み関数 (SYSIBM 関数) と SYSFUN 関数は、暗黙で PUBLIC に付与される EXECUTE 特権をもちます。

関数解決

関数呼び出しの後、データベース・マネージャーは、同じ名前をもつ呼び出し可能な関数の中でどれが『最適』かを判別する必要があります。これには、組み込み関数とユーザー定義関数の中から関数を解決する処理が関与します。

引数 とは、呼び出し時に関数に渡される値です。 SQL の中で呼び出されるとき、関数にはゼロ個以上の引数のリストが渡されます。このような引数は、引数のセマンティクスが引数リスト内の位置によって決定されるという意味で定位置と言えます。*パラメーター* は、関数への入力の形式上の定義です。データベースに対して内部的に (組み込み関数) またはユーザーによって (ユーザー定義関数) 関数が定義されるときは、関数のパラメーターが (ゼロ個以上) 指定されます。また、パラメーターの定義の順序によって、パラメーターの位置とセマンティクスが定義されることになります。したがって、どのパラメーターも関数への特定の定位置入力になります。呼び出し時に、引数リスト中のその位置によって、引数は特定のパラメーターに対応します。

データベース・マネージャーは、呼び出しで指定される関数名、その関数に対する EXECUTE 特権、引数の数とデータ・タイプ、SQL パスの中で同じ名前を持つすべての関数、対応するパラメーターのデータ・タイプを使用して、ある関数を選択するかどうかの判断基準とします。メソッドを決定する過程で発生する可能性のある結果について以下に示します。

- 特定の関数が最適であると判断される場合。たとえば、名前が RISK、スキーマが TEST、シグニチャーが以下のように定義された関数を考えてみます。

  ```
  TEST.RISK(INTEGER)
  TEST.RISK(DOUBLE)
  ```

  SQL パスには TEST スキーマが入っており、以下のように関数が参照されたとします (DB は DOUBLE 列)。

  ```
  SELECT ... RISK(DB) ...
  ```

  この場合は、2 番目の RISK が選択されます。

  以下のように関数が参照される場合は (SI は SMALLINT 列)、

  ```
  SELECT ... RISK(SI) ...
  ```

  最初の RISK が選択されます。これは、SMALLINT は INTEGER にプロモート可能であり、優先順位リストでの順位がさらに下になる DOUBLE よりも一致性が高いためです。

構造化タイプである引数について考慮する場合、優先順位リストには、静的タイプの引数のスーパータイプが入っています。最適なのは、構造化タイプ階層の中で、静的タイプの関数引数に最も近いスーパータイプ・パラメーターで定義する関数です。

- 許容できる適合性をもつ関数がないと判断される場合。たとえば、前出の例と同じ 2 つの関数が指定され、以下のように関数が参照されたとします (C は CHAR(5) 列)。

  ```
  SELECT ... RISK(C) ...
  ```

  この場合、引数はどちらの RISK 関数のパラメーターとも整合性がありません。

- SQL パス、および呼び出し時に渡される引数の数とデータ・タイプに基づいて特定の関数が選択される場合。たとえば、名前が RANDOM で、シグニチャーが以下のように定義されている関数を考えてみます。

  ```
  TEST.RANDOM(INTEGER)
  PROD.RANDOM(INTEGER)
  ```

  SQL パスは以下のとおりです。

  ```
  "TEST","PROD"
  ```

  この場合、次の関数参照では、

  ```
  SELECT ... RANDOM(432) ...
  ```

  TEST.RANDOM が選択されます。これは、どちらの RANDOM 関数も同程度に高い一致性を示し (この例の場合は完全一致)、どちらのスキーマも関数パスにありますが、 SQL パスにおいて TEST の方が PROD より先に指定されているためです。

最適の判別

引数のデータ・タイプと、対象とする関数のパラメーターに定義されているデータ・タイプとの比較は、似通った名前の関数の中でどれが「最適」かを決定する基準となります。関数の結果のデータ・タイプまたは関数のタイプ (列、スカラー、または表) は、この決定には関係しないことに注意してください。

関数解決は、以下の手順で行われます。

1. まず、カタログ (SYSCAT.ROUTINES) と組み込み関数から、以下のすべての条件が真となるすべての関数を探します。
   - スキーマ名が指定された呼び出し (修飾子付き参照) の場合、スキーマ名と関数名が呼び出し名に一致する。
   - スキーマ名が指定されていない呼び出し (修飾子なし参照) の場合、関数名が呼び出し名に一致し、 SQL パス中のスキーマの 1 つに一致するスキーマ名がある。
   - 呼び出し側は、関数に対する EXECUTE 特権をもっている。
   - 定義済みパラメーターの数が呼び出しと一致している。
   - 呼び出しの各引数のデータ・タイプが、関数の対応する定義済みパラメーターのデータ・タイプに一致するか、またはそのデータ・タイプに『プロモート可能』である。

2. 次に、関数呼び出しの個々の引数を左から右に検討していきます。引数ごとに、その引数に対して最適な一致ではない関数をすべて除去していきます。ある引数の最適な一致とは、その引数データ・タイプに対応する優先順位リストの中で、そのデータ・タイプのパラメーターをもつ関数が存在するデータ・タイプのうち、最初に記述されているデータ・タイプです。長さ、精度、位取り、および FOR BIT DATA 属性は、この比較では考慮されません。たとえば、DECIMAL(9,1) の引数は DECIMAL(6,5) のパラメーターと完全に一致すると見なされ、 DECFLOAT(34) の引数は DECFLOAT(16) のパラメーターと完全に一致すると見なされ、また VARCHAR(19) の引数は VARCHAR(6) のパラメーターと完全に一致すると見なされます。

   ユーザー定義構造化タイプ引数に最適なのは、それ自身です。次に適しているのは、すぐ上のスーパータイプです。このことは、引数の各スーパータイプに当てはまります。ここで考慮しているのは、静的タイプ (宣言済みタイプ) の構造化タイプ引数であり、動的タイプ (最も特定的なタイプ) ではありません。
3. ステップ 2 の後でも 2 つ以上の関数が候補として残っているときは、残りの関数候補はすべて同じシグニチャーを持っていても、それぞれ異なるスキーマに存在しているはずです。スキーマがそのユーザーの SQL パスで最初に出現する関数が選択されます。
4. ステップ 2 の後で候補となる関数が残らなかった場合は、エラー (SQLSTATE 42884) になります。

組み込み関数の SQL パスに関する考慮事項

組み込み関数は SYSIBM という名前の特殊なスキーマ内に置かれています。SYSFUN および SYSPROC スキーマにはさらに別の使用可能な関数がありますが、それらは組み込み関数とは見なされません。それらはユーザー定義関数として開発されたものであり、処理上の特別な考慮事項がないためです。
SYSIBM、SYSFUN、または SYSPROC スキーマに (あるいは SYS の文字で始まる名前の他のどのスキーマにも)、ユーザーがさらに関数を定義することはできません。

すでに説明したように、関数解決処理の中で、組み込み関数はユーザー定義関数とまったく同じように扱われます。関数解決の観点から見た組み込み関数とユーザー定義関数の違いの 1 つは、組み込み関数は関数解決で常に検討対象とする必要があるということです。したがって、パスから SYSIBM を省いても、 (関数およびデータ・タイプの解決では) SYSIBM がパスの最初のスキーマであることが想定されることになります。

たとえば、ユーザーの SQL パスが以下のように定義されているとします。

```
"SHAREFUN","SYSIBM","SYSFUN"
```

また、引数の数とタイプが SYSIBM.LENGTH と同じである LENGTH 関数が SHAREFUN スキーマに定義されているとします。この場合、このユーザーの SQL ステートメントで、修飾なしで LENGTH を参照すると、SHAREFUN.LENGTH が選択されます。一方、ユーザーの SQL パスが以下のように定義されている場合には、

```
"SHAREFUN","SYSFUN"
```

同じ SHAREFUN.LENGTH 関数が存在していたとしても、このユーザーの SQL ステートメントで修飾せずに LENGTH を参照すると、 SYSIBM.LENGTH が選択されることになります。これは、SYSIBM が暗黙のうちにパス中で最初に出現するためです。

この状況下では、次のようにすれば、問題を未然に最小化することができます。

- ユーザー定義関数には組み込み関数の名前を決して使用しないようにします。
- 何らかの理由で組み込み関数と同じ名前のユーザー定義関数を作成する必要のある場合は、そのような関数への参照には必ず修飾子を付けます。

注: *function-name* という関数の非修飾参照は、スキーマ SYSIBM 内にある組み込み関数 *function-name* を呼び出します。この関数は、スキーマ名で明示的に修飾することができません。

関数解決の例

以下は、正常な関数解決の例を示しています。 (必要なキーワードがすべて示されているわけではないことに注意してください。)

3 つの異なるスキーマに 7 個の ACT 関数があり、以下のように登録されているとします。

```
CREATE FUNCTION AUGUSTUS.ACT (CHAR(5), INT, DOUBLE) SPECIFIC ACT_1 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_2 ...
CREATE FUNCTION AUGUSTUS.ACT (INT, INT, DOUBLE, INT) SPECIFIC ACT_3 ...
CREATE FUNCTION JULIUS.ACT (INT, DOUBLE, DOUBLE) SPECIFIC ACT_4 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DOUBLE) SPECIFIC ACT_5 ...
CREATE FUNCTION JULIUS.ACT (SMALLINT, INT, DOUBLE) SPECIFIC ACT_6 ...
CREATE FUNCTION JULIUS.ACT (INT, INT, DECFLOAT) SPECIFIC ACT_7 ...
CREATE FUNCTION NERO.ACT (INT, INT, DEC(7,2)) SPECIFIC ACT_8 ...
```

以下のように関数が参照されるとします (I1 および I2 は INTEGER 列、 D は DECIMAL 列です)。

```
SELECT ... ACT(I1, I2, D) ...
```

この参照を行うアプリケーションの SQL パスが次のようになっているとします。

```
"JULIUS","AUGUSTUS","CAESAR"
```

アルゴリズムに従っていきます。

- スキーマ NERO が SQL パスに組み込まれていないため、特定の名前 ACT_8 の付いた関数は候補から除かれます。
- パラメーターの数が違っているため、ACT_3 は候補から除かれます。第 1 引数が第 1 パラメーターのデータ・タイプにプロモートできないため、 ACT_1 と ACT_6 はどちらも候補から除かれます。
- この時点で複数の候補が残っているため、次に引数が順に検討されます。
- 最初の引数については、残りのすべての関数 ACT_2、ACT_4、ACT_5、および ACT_7 がその引数タイプと完全に一致します。この検討ではどの関数も検討の対象から除かれないため、次の引数を検討する必要があります。

- 2 番目の引数では、ACT_2、ACT_5、および ACT_7 が完全に一致していますが、 ACT_4 は一致していないため、ACT_4 が検討の対象から除かれます。ACT_2、ACT_5、および ACT_7 の間の何らかの差異を判別するために、さらに次の引数が検討されます。
- 第 3 の最後の引数では、ACT_2、ACT_5、ACT_7 のいずれも、引数のタイプと完全には一致していません。 ACT_2 と ACT_5 の適合度は同程度ですが、ACT_7 は他の 2 つよりも適合度が劣ります。タイプ DOUBLE は DECFLOAT よりも、DECIMAL に近いからです。 ACT_7 は除去されます。
- この時点で、パラメーター・シグニチャーが同じである関数として ACT_2 と ACT_5 の 2 つが残っています。最終的な決定要因は、どちらの関数のスキーマが SQL パスで先に出現するかであり、この基準によって ACT_5 が最終的に選択されます。

関数呼び出し

関数が選択された後も、いくつかの理由でその関数の使用が許可されない場合があります。個々の関数は特定のデータ・タイプの結果を返すように定義されています。この結果のデータ・タイプが、関数が呼び出されるコンテキストと互換性がない場合、エラーが生じます。たとえば、今度は結果のデータ・タイプが異なる STEP という名前の関数が次のように定義されているとします。

```
STEP(SMALLINT) returns CHAR(5)
STEP(DOUBLE) returns INTEGER
```

以下のように関数が参照されると (S は SMALLINT 列)、

```
SELECT ... 3 + STEP(S) ...
```

引数タイプが完全に一致しているため、最初の STEP が選択されます。しかし、結果のタイプが加法演算子の引数として求められる数値タイプではなく CHAR(5) であるため、ステートメントではエラーになります。

この他にこの状態が起きる場合の例は次のとおりです。いずれの例もステートメントのエラーが生じます。

- 関数が FROM 節で参照されたが、関数解決ステップで選択された関数がスカラー関数または列関数であった場合。
- その逆の場合。つまりコンテキストがスカラー関数または列関数を要求し、関数解決が表関数を選択する場合。

関数呼び出しの引数が、選択された関数のパラメーターのデータ・タイプと完全一致でない場合、列への割り当てと同じ規則を適用して、実行時に引数がパラメーターのデータ・タイプに変換されます。これには、引数とパラメーターの間で精度、位取り、または長さが異なる場合も含まれます。

従来のバインディング・セマンティクス

ステートメントの処理時にルーチン、データ・タイプ、およびグローバル変数が解決されて、データベース・マネージャーはこの解決を繰り返さなければならない場合があります。これは、以下の場合に当てはまります。

- パッケージ内の静的 DML ステートメント
- ビュー

- トリガー
- チェック制約
- SQL ルーチン

パッケージ内の静的 DML ステートメントの場合、ルーチン、データ・タイプ、またはグローバル変数の参照はバインド操作時に解決されます。ビュー、トリガー、SQL ルーチン、およびチェック制約の中にあるルーチン、データ・タイプ、およびグローバル変数の参照は、データベース・オブジェクトの作成時に解決されます。

そのようなオブジェクト内のいずれかのルーチン参照に対してルーチン解決をもう一度実行すると、以下の場合はその動作が変わる可能性があります。

- 一致度のより高いシグニチャーをもつ新規のルーチンが追加されたけれども、実際の実行可能ファイルは別の操作で実行される場合。
- 一致度のより高いシグニチャーをもつルーチンに対する実行特権が定義者に付与されたけれども、実際の実行可能ファイルは別の操作で実行される場合。

同様に、これらのオブジェクト内のデータ・タイプ、グローバル変数、またはルーチンのいずれかで二度目の関数解決が実行される場合、他のスキーマにあるのと同じ名前 (SQL パスにも存在している) で新しいデータ・タイプ、グローバル変数、またはルーチンが追加されていると、動作が変わってしまうおそれがあります。これが起きないようにするためにデータベース・マネージャーは、必要であれば常に *従来のバインディング・セマンティクス* を適用します。それによって、ルーチン、データ・タイプ、およびグローバル変数の参照を解決するときは、必ず前のバインド時の解決先と同じ SQL パスと一連のルーチンが使用されることになります。解決時に検討対象となるルーチン、データ・タイプ、およびグローバル変数の作成タイム・スタンプは、ステートメントのバインド時の時刻よりも後にはなりません。(バージョン 6.1 から追加された組み込み関数では、データベースの作成または移行の時刻に基づいた作成タイム・スタンプが押されます。) このようにして、ルーチン、データ・タイプ、およびグローバル変数のうち、ステートメントを最初に処理したときの解決時に考慮されたものだけが考慮されるようにします。したがって、従来のバインディング・セマンティクスが適用される場合、後から作成されて新たに許可されたルーチン、データ・タイプ、およびグローバル変数は考慮されません。

シグニチャー SCHEMA1.BAR(INTEGER) および SCHEMA2.BAR(DOUBLE) をもつ 2 つの関数を擁するデータベースについて考察してみます。 SQL パス内には、SCHEMA1 と SCHEMA2 の 2 つのスキーマ (SQL パス内でのその順序は重要ではありません) があると仮定します。 USER1 には、関数 SCHEMA2.BAR(DOUBLE) に対する EXECUTE 特権が付与されています。 USER1 は、BAR(INT_VAL) を呼び出すビューを作成すると仮定します。これは、関数 SCHEMA2.BAR(DOUBLE) に解決されます。そのビューの作成後に SCHEMA1.BAR(INTEGER) に対する EXECUTE 特権を誰かが USER1 に付与しても、そのビューは常に SCHEMA2.BAR(DOUBLE) を使用します。

パッケージ内の静的 DML の場合、パッケージの再バインドは暗黙的に行うことができますが、 REBIND コマンド (またはこれに対応する API) または BIND コマンド (またはこれに対応する API) を明示的に発行して行うこともできます。暗黙的な再バインドでは、常に保守的バインディング・セマンティクスを適用して、ルーチン、データ・タイプ、およびグローバル変数の解決が実行されます。 REBIND コ

マンドには、従来のバインディング・セマンティクスで解決するか (RESOLVE CONSERVATIVE オプション)、それとも新しいルーチン、データ・タイプ、またはグローバル変数を考慮して解決するか (デフォルト・オプションの RESOLVE ANY) を選択するオプションがあります。

パッケージを暗黙的に再バインドすると、常に同じルーチンが解決されます。一致性のより高いルーチンに対する EXECUTE 特権を付与されていても、そのルーチンは検討の対象にはなりません。パッケージを明示的に再バインドすると、異なるルーチンが選択されることになります。 (ただし、RESOLVE CONSERVATIVE を指定すると、従来のバインディング・セマンティクスに従ってルーチンは解決されます。)

ビュー、トリガー、制約、または SQL ルーチン本体の作成時にルーチンを指定すると、そのルーチンのどのインスタンスが使用されるかは、オブジェクトの作成時のルーチン解決で決まります。オブジェクトの作成が完了した後で EXECUTE 特権が付与されても、オブジェクトが使用する所定のルーチンは変わりません。

シグニチャー SCHEMA1.BAR(INTEGER) および SCHEMA2.BAR(DOUBLE) をもつ 2 つの関数を擁するデータベースについて考察してみます。 USER1 には、関数 SCHEMA2.BAR(DOUBLE) に対する EXECUTE 特権が付与されています。 USER1 は、BAR(INT_VAL) を呼び出すビューを作成すると仮定します。これは、関数 SCHEMA2.BAR(DOUBLE) に解決されます。そのビューの作成後に SCHEMA1.BAR(INTEGER) に対する EXECUTE 特権を誰かが USER1 に付与しても、そのビューは常に SCHEMA2.BAR(DOUBLE) を使用します。

その他のデータベース・オブジェクトでも、これと同じ動作が行われます。たとえば、パッケージを暗黙で再バインドした (おそらく索引のドロップ後に) 場合、その暗黙の再バインドの前でも後でもそのパッケージは同じ所定のルーチンを参照します。ただしパッケージを*明示的に* 再バインドすると、異なるルーチンが選択されることになります。

メソッド

構造化タイプのデータベース・メソッドは、一連の入力データ値と、一連の結果値との関連のことです。ここで、最初の入力値 (またはサブジェクト*引数*) は、メソッドと同じ値になるか、サブジェクト・タイプ (サブジェクト・パラメーター ともいう) のサブタイプになります。たとえば、型が ADDRESS である CITY というメソッドは、型が VARCHAR の入力データ値に渡すことができます。結果は ADDRESS (または ADDRESS のサブタイプ) になります。

メソッドは、ユーザー定義構造化タイプの定義の一環として、暗黙的にあるいは明示的に定義されます。

暗黙的に定義されたメソッドは、構造化タイプごとに作成されます。また、構造化タイプの属性ごとに、*監視用メソッド* が定義されます。監視用メソッドを使うと、アプリケーション側は、該当タイプのインスタンスの属性値を知ることができます。*変更メソッド*も属性ごとに定義されます。これにより、アプリケーション側では、型インスタンスの属性の値を変更することによって型インスタンスを変更できます。上記の CITY メソッドは、型 ADDRESS の変更メソッドの一例です。

明示的に定義したメソッド、すなわち*ユーザー定義メソッド* は、 CREATE TYPE (または ALTER TYPE ADD METHOD) および CREATE METHOD ステートメントを組み合わせて使用して、 SYSCAT.ROUTINES のデータベースに登録されるメソッドです。特定の構造化タイプ用に定義されたメソッドはすべて、その型と同じスキーマで定義されます。

構造化タイプ用のユーザー定義メソッドは、データベース・エンジンの構造化タイプ・インスタンスに適用できる (ユーザーもしくは第三者ベンダーによって提供された) メソッド定義を追加することによって、データベース・システムの機能を拡張します。データベース・メソッドを定義することにより、データベースはアプリケーションが使用するのと同じメソッドをエンジンで活用することができ、アプリケーションとデータベースとの間の相互作用が高まります。

外部および SQL ユーザー定義メソッド

ユーザー定義メソッドは、外部メソッドとするか、 SQL 式に基づくものとすることができます。外部メソッドは、オブジェクト・コード・ライブラリーと、メソッド呼び出し時に実行されるそのライブラリー内の関数によって、データベースに対して定義されます。 SQL 式に基づいたメソッドは、メソッドの呼び出し時に、その SQL 式の結果を戻します。そのようなメソッドでは、すべてが SQL で作成されているので、オブジェクト・コード・ライブラリーが必要ありません。

ユーザー定義メソッドでは、呼び出されるたびに単一値の応答を戻すことができます。この値は、構造化タイプとすることができます。また、メソッドを (SELF AS RESULT を使用して) *型保持* として定義し、メソッドの戻される型として、動的型のサブジェクト引数を戻すようにすることが可能です。暗黙的に定義した変更メソッドは、型保持されます。

メソッド・シグニチャー

メソッドは、そのサブジェクト・タイプ、メソッド名、パラメーター数、およびそのパラメーターのデータ・タイプによって識別されます。これはメソッド・シグニチャー と呼ばれ、データベース内でユニークである必要があります。

以下の場合に、同じ名前を付けられた構造化タイプのメソッドが、複数存在する可能性があります。

- パラメーターの数やパラメーターのデータ・タイプが違う場合。または
- メソッドが同じメソッド階層の一部である場合 (つまり、メソッド同士がオーバーライド関係にあるか、同じ元のメソッドをオーバーライドしている場合)。または
- (最初のパラメーターとして、サブジェクト・タイプ、またはそのサブタイプかスーパータイプを使用した) 同じ関数シグニチャーが存在しない場合。

複数のメソッド・インスタンスを持つメソッド名のことを、*多重定義メソッド* といいます。あるメソッド名があるタイプ内で多重定義される場合、そのタイプには、その名前で 2 つ以上のメソッドがあるということです (それぞれのタイプにはすべて、異なるパラメーター・タイプがあります)。メソッド名はサブジェクト・タイプ階層においても多重定義可能です。その場合、そのタイプ階層にその名前のメソッドが 2 つ以上ありますが、それぞれのメソッドには異なるパラメーター・タイプがなければなりません。

メソッドを呼び出すときには (許可されているコンテキストで) 構造化タイプ・インスタンス (サブジェクト引数) への参照と、二重ドット演算子の両方が先頭に記されているメソッド名を参照します。 その後に、括弧で囲まれた引数のリストが続きます。実際に呼び出されるメソッドは、次の項で説明されているメソッド解決プロセスにより、静的タイプのサブジェクト・タイプに基づいて決定されます。関数呼び出しを使い、WITH FUNCTION ACCESS で定義されているメソッドを呼び出すこともできます。その場合、関数解決のための通常の規則が適用されます。

関数解決の結果が WITH FUNCTION ACCESS で定義されているメソッドであれば、メソッド呼び出しのその後のステップはすべて処理されます。

メソッドへのアクセスは EXECUTE 特権を通して制御されます。 GRANT および REVOKE ステートメントは、特定のメソッドまたはメソッドのセットを誰が実行できて誰ができないのかを指定するために使用します。メソッドを呼び出すには、EXECUTE 特権 (または DBADM 権限) が必要です。メソッドの定義者には、自動的に EXECUTE 特権が付与されます。すべての基礎オブジェクトに対する WITH GRANT オプションをもつ外部メソッドまたは SQL メソッドの定義者にはまた、メソッドに対する EXECUTE 特権付きの WITH GRANT オプションが GRANT されます。定義者 (または SYSADM または DBADM) は、これを、任意の SQL ステートメントからメソッドを呼び出したり、任意の DDL ステートメント (CREATE VIEW、CREATE TRIGGER、または制約を定義するときなど) でメソッドを参照したりするユーザーに付与します。ユーザーに EXECUTE 特権が付与されていない場合、そのメソッドは、たとえ一致の度合いが高くても、メソッド解決アルゴリズムによって考慮されません。

メソッド解決

メソッドの呼び出しの後、データベース・マネージャーは、同じ名前をもつ呼び出し可能なメソッドの中でどれが「最適」かを判別する必要があります。メソッド解決時には、関数 (組み込みまたはユーザー定義) は考慮されません。

*引数*とは、呼び出し時にメソッドに渡される値です。 SQL の中で呼び出されるとき、メソッドには (特定構造化タイプの) サブジェクト引数、およびゼロ個以上の引数のリストが渡されます。このような引数は、引数のセマンティクスが引数リスト内の位置によって決定されるという意味で定位置と言えます。パラメーター は、メソッドへの入力の形式上の定義です。メソッドがデータベースに対して暗黙的に (特定タイプ用にシステム生成される)、またはユーザーによって (ユーザー定義メソッド) 定義されるとき、メソッドのパラメーターが (最初のパラメーターとしてのサブジェクト・パラメーター付きで) 指定されます。パラメーターの定義の順序がパラメーターの位置、およびその結果としてパラメーターのセマンティクスを定義することになります。したがって、どのパラメーターもメソッドの特定の定位置入力です。呼び出し時に、引数リスト中のその位置によって、引数は特定のパラメーターに対応します。

データベース・マネージャーは、呼び出しで指定されるメソッド名、メソッドに対する EXECUTE 特権、引数の数とデータ・タイプ、サブジェクト引数の静的タイプ (およびそのスーパータイプ) 用に同じ名前をもつすべてのメソッド、対応するパラメーターのデータ・タイプを使用して、あるメソッドを選択するかどうかの判断基準とします。メソッドを決定する過程で発生する可能性のある結果について以下に示します。

- 特定のメソッドが最適であると判断される場合。たとえば、シグニチャーが以下のように定義されたタイプ SITE の RISK というメソッドを考えてみます。

  ```
  PROXIMITY(INTEGER) FOR SITE
  PROXIMITY(DOUBLE) FOR SITE
  ```

  続くメソッドの呼び出しは次のようになります (ここで、ST は SITE 列、DB は DOUBLE 列です)。

  ```
  SELECT ST..PROXIMITY(DB) ...
  ```

  この場合は、2 番目の PROXIMITY が選択されます。

  以下のようにメソッドが呼び出される場合は (SI は SMALLINT 列)、

  ```
  SELECT ST..PROXIMITY(SI) ...
  ```

  最初の PROXIMITY が選択されます。これは、SMALLINT は INTEGER にプロモート可能であり、優先順位リストでの順位がさらに下になる DOUBLE よりも一致性が高いためです。

  構造化タイプである引数について考慮する場合、優先順位リストには、静的タイプの引数のスーパータイプが入っています。最適なのは、構造化タイプ階層の中で、静的タイプの関数引数に最も近いスーパータイプ・パラメーターで定義する関数です。
- 許容できる適合性をもつメソッドがないと判断される場合。たとえば、前出の例と同じ 2 つの関数が指定され、以下のように関数が参照されたとします (C は CHAR(5) 列)。

```
SELECT ST..PROXIMITY(C) ...
```

この場合、引数はどちらの PROXIMITY 関数のパラメーターとも整合性がありません。

- タイプ階層のメソッド、および呼び出し時に渡される引数の数とデータ・タイプに基づいて特定のメソッドが選択される場合。たとえば、シグニチャーが以下のように定義された、タイプ SITE および DRILLSITE (SITE のサブタイプ) の RISK というメソッドを考えてみます。

  ```
  RISK(INTEGER) FOR DRILLSITE
  RISK(DOUBLE) FOR SITE
  ```

  続くメソッドの呼び出しは次のようになります (ここで、DRST は DRILLSITE 列、DB は DOUBLE 列です)。

  ```
  SELECT DRST..RISK(DB) ...
  ```

  DRILLSITE は SITE へプロモートできるので、この場合は、2 番目の RISK が選択されます。

  以下のようにメソッドが参照される場合は (SI は SMALLINT 列)、

  ```
  SELECT DRST..RISK(SI) ...
  ```

  最初の RISK が選択されます。これは、 SMALLINT は INTEGER にプロモート可能 (優先順位リストでの順位が DOUBLE よりも近い) であり、 DRILLSITE は、SITE よりも一致性が高いスーパータイプであるためです。

  同じタイプ階層内のメソッドで同じシグニチャーを使い、サブジェクト・パラメーター以外のパラメーターで利用することはできません。

最適の判別

引数のデータ・タイプと、対象とするメソッドのパラメーターに定義されているデータ・タイプとの比較は、似通った名前のメソッドのグループ中でどれが『最適』かを決定する基準となります。考慮しているメソッドの結果のデータ・タイプは、この決定には関係しないことに注意してください。

メソッド解決は、以下の手順で実行されます。

1. まず、カタログ (SYSCAT.ROUTINES) から、以下のすべての条件が真となるすべてのメソッドを探します。
   - メソッド名が呼び出し名と同じで、サブジェクト・パラメーターが、サブジェクト引数の静的タイプと同じであるか、そのスーパータイプである。
   - 呼び出し側がメソッドに対する EXECUTE 特権を持っている。
   - 定義済みパラメーターの数が呼び出しと一致している。
   - 呼び出しの各引数のデータ・タイプが、メソッドの対応する定義済みパラメーターのデータ・タイプに一致するか、またはそのデータ・タイプに「プロモート可能」である。
2. 次に、メソッド呼び出しの個々の引数を左から右に検討していきます。左端の引数 (すなわち最初の引数) は、暗黙的な SELF パラメーターです。例えば、タイプ ADDRESS_T に定義したメソッドには、暗黙的な最初のパラメーターとしてタイプ ADDRESS_T があります。引数ごとに、その引数に対して最適な一致で

はない関数をすべて除去していきます。ある引数の最適な一致とは、その引数データ・タイプに対応する優先順位リストの中で、そのデータ・タイプのパラメーターをもつ関数が存在するデータ・タイプのうち、最初に記述されているデータ・タイプです。長さ、精度、位取り、および FOR BIT DATA 属性は、この比較では考慮されません。例えば、DECIMAL(9,1) の引数は DECIMAL(6,5) のパラメーターと完全に一致すると見なされ、 DECFLOAT(34) の引数は DECFLOAT(16) のパラメーターと完全に一致すると見なされ、また VARCHAR(19) の引数は VARCHAR(6) のパラメーターと完全に一致すると見なされます。

ユーザー定義構造化タイプ引数に最適なのは、それ自身です。次に適しているのは、すぐ上のスーパータイプです。このことは、引数の各スーパータイプに当てはまります。ここで考慮しているのは、静的タイプ (宣言済みタイプ) の構造化タイプ引数であり、動的タイプ (最も特定的なタイプ) ではありません。

3. ほとんどの場合、ステップ 2 の実行後に候補メソッドが 1 つ残ります。このメソッドを選択します。
4. ステップ 2 の後で候補となるメソッドが残らなかった場合は、エラー (SQLSTATE 42884) になります。

メソッド解決の例

以下は、正常なメソッド解決の例を示しています。

GOVERNOR の階層内で定義された 3 つの構造化タイプには、 HEADOFSTATE のサブタイプとしての EMPEROR のサブタイプとして、 7 つの FOO メソッドがあります。それぞれ、以下のシグニチャーで登録されています。

```
CREATE METHOD FOO (CHAR(5), INT, DOUBLE)  FOR HEADOFSTATE SPECIFIC FOO_1 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR HEADOFSTATE SPECIFIC FOO_2 ...
CREATE METHOD FOO (INT, INT, DOUBLE, INT) FOR HEADOFSTATE SPECIFIC FOO_3 ...
CREATE METHOD FOO (INT, DOUBLE, DOUBLE)   FOR EMPEROR     SPECIFIC FOO_4 ...
CREATE METHOD FOO (INT, INT, DOUBLE)      FOR EMPEROR     SPECIFIC FOO_5 ...
CREATE METHOD FOO (SMALLINT, INT, DOUBLE) FOR EMPEROR     SPECIFIC FOO_6 ...
CREATE METHOD FOO (INT, INT, DEC(7,2))    FOR GOVERNOR    SPECIFIC FOO_7 ...
```

以下のようにメソッドが参照されるとします (I1 および I2 は INTEGER 列、 D は DECIMAL 列、そして E は EMPEROR 列です)。

```
SELECT E..FOO(I1, I2, D) ...
```

アルゴリズムに従っていきます。

- タイプ GOVERNOR は EMPEROR のサブタイプなので (スーパータイプではない)、 FOO_7 は候補から除かれます。
- パラメーターの数が違っているため、FOO_3 は候補から除かれます。
- 第 1 引数 (サブジェクト引数ではない) が第 1 パラメーターのデータ・タイプにプロモートできないため、 FOO_1 と FOO_6 はどちらも候補から除かれます。この時点で複数の候補が残っているため、次に引数が順に検討されます。
- サブジェクト引数の場合、FOO_2 はスーパータイプですが、 FOO_4 と FOO_5 はサブジェクト引数に一致しています。
- 最初の引数については、残りのメソッド FOO_4 および FOO_5 がその引数タイプと完全に一致します。この検討ではどのメソッドも検討の対象から除かれないため、次の引数を検討する必要があります。

- 2 番目の引数では、FOO_5 が完全に一致しているのに対し、 FOO_4 は一致していないため、FOO_4 が検討の対象から除かれます。これにより、メソッド FOO_5 が選ばれます。

メソッドの呼び出し

メソッドが選択された後も、いくつかの理由でそのメソッドの使用が許可されない場合があります。

個々のメソッドは特定のデータ・タイプの結果を戻すように定義されています。この結果のデータ・タイプが、メソッドが呼び出されるコンテキストと互換性がない場合、エラーが生じます。例えば、STEP というメソッドが定義されていて、結果としてそれぞれが別々のデータ・タイプを持っているとします。

```
STEP(SMALLINT) FOR TYPEA RETURNS CHAR(5)
STEP(DOUBLE) FOR TYPEA RETURNS INTEGER
```

以下のようにメソッドが参照されると (S は SMALLINT 列で TA は TYPEA の列)、

```
SELECT 3 + TA..STEP(S) ...
```

引数タイプが完全に一致しているため、最初の STEP が選択されます。しかし、結果のタイプが加算演算子の引数として求められる数値タイプではなく CHAR(5) であるため、ステートメントではエラーになります。

選択されたメソッドから、「メソッドの動的ディスパッチング」で記述されたアルゴリズムを使用して、コンパイル時にディスパッチ可能なメソッドのセットが構築されます。正確にどのメソッドが呼び出されるかは、「メソッドの動的ディスパッチング」で記述されます。

選択したメソッドがタイプ保持メソッドである場合、以下の点に注意してください。

- 関数解決に続く静的結果タイプは、メソッド呼び出しのサブジェクト引数の静的タイプと同じです。
- メソッドを呼び出すときの動的結果タイプは、メソッド呼び出しのサブジェクト引数の動的タイプと同じです。

これは、タイプ保持メソッド定義で指定した結果タイプのサブタイプになる可能性がありますが、逆に、メソッドの処理時に実際に戻される動的タイプのスーパータイプになる場合もあります。

メソッド呼び出しの引数が、選択されたメソッドのパラメーターのデータ・タイプと完全一致でない場合、列への割り当てと同じ規則を適用して、実行時に引数がパラメーターのデータ・タイプに変換されます。これには、引数とパラメーターの間で精度、位取り、または長さが異なる場合も含まれます。ただし、引数の動的タイプが、パラメーターの静的タイプのサブタイプである場合を除きます。

メソッドの動的ディスパッチング

メソッドは機能を提供し、あるタイプのデータをカプセル化します。メソッドはあるタイプに対して定義され、常にこのタイプと関連付けることができます。メソッドのパラメーターの 1 つは暗黙的な SELF パラメーターです。 SELF パラメータ

ーは、メソッドが宣言されているタイプのパラメーターです。 DML ステートメントでメソッドが呼び出されるときに SELF 引数として渡される引数は、「サブジェクト」と呼ばれます。

メソッドがメソッド解決 (192 ページの『メソッド解決』を参照) によって選択されているか、あるいはメソッドが DDL ステートメントで指定されている場合、そのメソッドは「最も具体的な適用可能許可メソッド」として認識されます。サブジェクトが構造化タイプの場合、そのメソッドは 1 つ以上のオーバーライド・メソッドを持つ可能性があります。 DB2 は、実行時のサブジェクトの動的タイプ (最も具体的なタイプ) に基づいて、これらのどのメソッドを呼び出すかを決定しなければなりません。この決定は「最も具体的なディスパッチ可能メソッドの決定」と呼ばれます。このプロセスについて以下で説明します。

1. 最も具体的な適用可能許可メソッドを備えたメソッド階層で元のメソッドを見つけます。これは、*ルート・メソッド* と呼ばれます。
2. ディスパッチ可能メソッドのセットを作成します。これには以下が関与します。
   - 最も具体的な適用可能許可メソッド。
   - 最も具体的な適用可能許可メソッドをオーバーライドする任意のメソッド。これは、この呼び出しのサブジェクトのサブタイプであるタイプに対して定義されます。
3. 以下のように、最も具体的なディスパッチ可能メソッドを決定します。
   a. 一連のディスパッチ可能メソッドのエレメントで、サブジェクトの動的タイプか、そのいずれかのスーパータイプのメソッドである任意のメソッドを開始します。これは、初期の最も具体的なディスパッチ可能メソッドになります。
   b. 一連のディスパッチ可能メソッドのエレメントに関して上記を繰り返します。各メソッドについて、そのメソッドが、最も具体的なディスパッチ可能メソッドが定義されているタイプの適切なサブタイプのいずれかに対して定義されている場合、および、そのメソッドが、サブジェクトの最も具体的なタイプのスーパータイプのいずれかに対して定義されている場合は、そのメソッドを最も具体的なディスパッチ可能メソッドとして、ステップ 2 を繰り返します。
4. 最も具体的なディスパッチ可能メソッドを呼び出します。

例:

3 つのタイプ「Person」、「Employee」、および「Manager」があります。「Person」に対して定義された、人物の収入を計算する元のメソッド「income」があります。人物はデフォルトで無職 (子供や退職者など) です。したがって、タイプ「Person」に対する「income」は、常にゼロを戻します。タイプ「Employee」およびタイプ「Manager」の場合、収入を計算するには別のアルゴリズムを適用する必要があります。そのため、「Employee」および「Manager」では、「Person」に対するメソッド「income」がオーバーライドされます。

表を作成してデータを追加するには以下のようにします。

```
CREATE TABLE aTable (id integer, personColumn Person);
  INSERT INTO aTable VALUES (0, Person()), (1, Employee()), (2, Manager());
```

$40000 以上の収入を得ている人物をリストします。

```
SELECT id, person, name
  FROM aTable
  WHERE person..income() >= 40000;
```

タイプ「Person」に対するメソッド「income」は、メソッド解決によって、最も具体的な適用可能許可メソッドとして選択されています。

1. ルート・メソッドは「Person」に対する「income」自体です。
2. 上記のアルゴリズムの 2 番目のステップが実行され、一連のディスパッチ可能メソッドが構成されます。
   - タイプ「Person」に対するメソッド「income」は、最も具体的な適用可能許可メソッドなので、これは組み込まれます。
   - タイプ「Employee」に対するメソッド「income」とタイプ「Manager」に対するメソッド「income」の両メソッドは、ルート・メソッドをオーバーライドするもので、「Employee」と「Manager」は「Person」のサブタイプなので、これらのメソッドは組み込まれます。

   したがって、ディスパッチ可能メソッドのセットは、 {「Person」に対する「income」、「Employee」に対する「income」、「Manager」に対する「income」} です。
3. 最も具体的なディスパッチ可能メソッドを決定します。
   - 最も具体的なタイプが「Person」であるサブジェクトの場合:
     a. 初期の最も具体的なディスパッチ可能メソッドを、タイプ「Person」に対する「income」にします。
     b. ディスパッチ可能メソッドのセットには、「Person」の適切なサブタイプとサブジェクトの最も具体的なタイプのスーパータイプに対して定義された他のメソッドがないので、「Person」に対する「income」が最も具体的なディスパッチ可能メソッドになります。
   - 最も具体的なタイプが「Employee」であるサブジェクトの場合:
     a. 初期の最も具体的なディスパッチ可能メソッドを、タイプ「Person」に対する「income」にします。
     b. 一連のディスパッチ可能メソッドごとに繰り返します。タイプ「Employee」に対するメソッド「income」は、「Person」の適切なサブタイプとサブジェクト (注: タイプはこの独自のスーパータイプとサブタイプである) の最も具体的なタイプのスーパータイプに対して定義されているので、「Employee」に対するメソッド「income」が、最も具体的なディスパッチ可能メソッドにより合致するメソッドになります。タイプ「Employee」に対するメソッド「income」を最も適切なディスパッチ可能メソッドとしてこのステップを繰り返します。
     c. ディスパッチ可能メソッドのセットには、「Employee」の適切なサブタイプおよびサブジェクトの最も具体的なタイプのスーパータイプに対して定義された他のメソッドがないので、「Employee」に対するメソッド「income」が最も具体的なディスパッチ可能メソッドになります。
   - 最も具体的なタイプが「Manager」であるサブジェクトの場合:
     a. 初期の最も具体的なディスパッチ可能メソッドを、タイプ「Person」に対する「income」にします。

b. 一連のディスパッチ可能メソッドごとに繰り返します。タイプ「Manager」に対するメソッド「income」は、「Person」の適切なサブタイプとサブジェクト (注: タイプはこの独自のスーパータイプとサブタイプである) の最も具体的なタイプのスーパータイプに対して定義されているので、「Manager」に対するメソッド「income」が、最も具体的なディスパッチ可能メソッドにより合致するメソッドになります。タイプ「Manager」に対するメソッド「income」を最も適切なディスパッチ可能メソッドとしてこのステップを繰り返します。

c. ディスパッチ可能メソッドのセットには、「Manager」の適切なサブタイプおよびサブジェクトの最も具体的なタイプのスーパータイプに対して定義された他のメソッドがないので、「Manager」に対するメソッド「income」が最も具体的なディスパッチ可能メソッドになります。

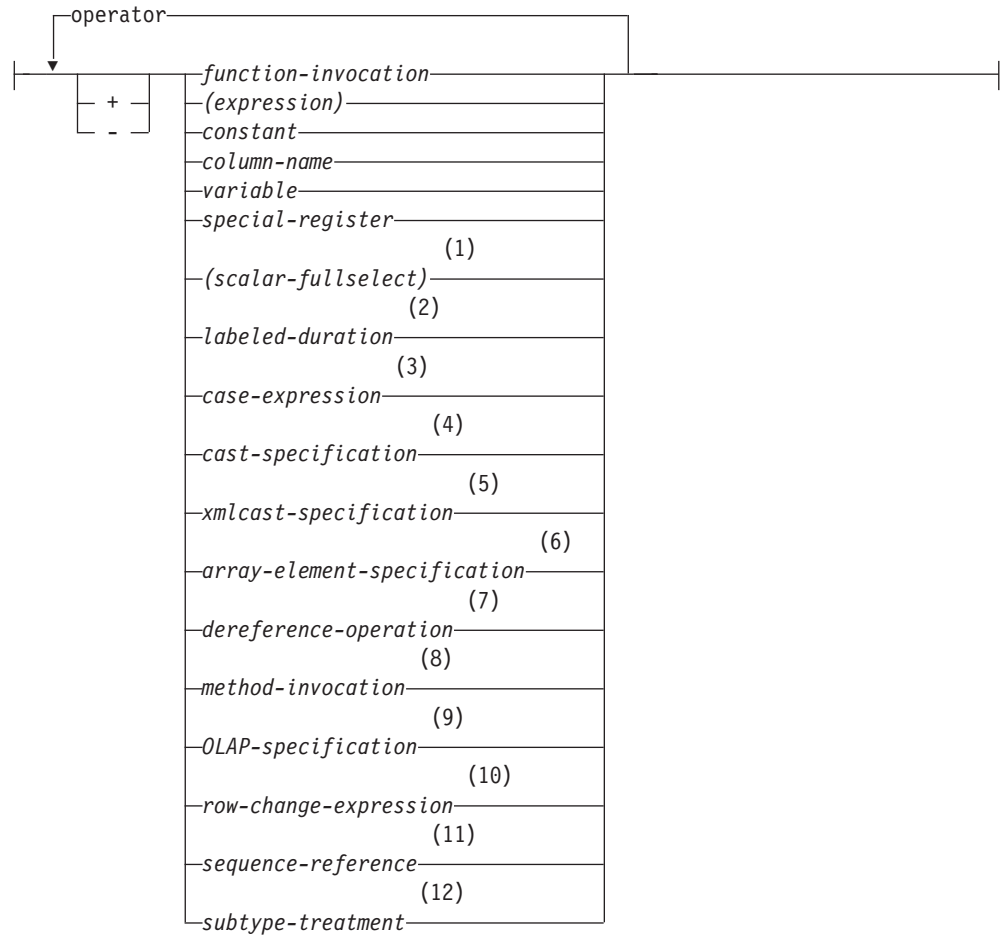
4. 最も具体的なディスパッチ可能メソッドを呼び出します。

式

式は値を指定します。これは、定数や列名のみで構成される簡単な値にすることも、もっと複雑な値にすることも可能です。同じような複雑な式を繰り返し使用する場合は、共通の式をカプセル化するための SQL 関数を検討することもできます。

Unicode データベースでは、文字ストリングまたは GRAPHIC ストリングを受け入れる式は、変換をサポートされている任意のストリング・タイプを受け入れます。

expression:



operator:

注:

1 詳しくは、 209 ページの『スカラー全選択』を参照してください。

2 詳しくは、 210 ページの『期間』を参照してください。

3 詳しくは、 215 ページの『CASE 式』を参照してください。

4 詳しくは、 218 ページの『CAST 指定』を参照してください。

5 詳しくは、 223 ページの『XMLCAST 指定』を参照してください。

6 詳しくは、 225 ページの『ARRAY エレメント仕様』を参照してください。

7 詳しくは、 226 ページの『間接参照操作』を参照してください。

8 詳しくは、 237 ページの『メソッドの呼び出し』を参照してください。

9 詳しくは、 228 ページの『OLAP 仕様』を参照してください。

10 詳しくは、 244 ページの『ROW CHANGE 式』を参照してください。

11 詳しくは、 240 ページの『シーケンス参照』を参照してください。

12 詳しくは、 239 ページの『サブタイプの扱い』を参照してください。

13 CONCAT の同義語として || を使用できます。

演算子がない式

演算子を使用しない式では、指定した値が式の結果になります。

例:

```
SALARY:SALARY'SALARY'MAX(SALARY)
```

連結演算子がある式

連結演算子 (CONCAT) は、2 つのストリング・オペランドを連結して、 1 つのストリング式 にします。

連結するオペランドは、互換性のあるストリングでなければなりません。 FOR BIT DATA と定義されている文字ストリングも含め、バイナリー・ストリングを文字ストリングと連結することはできません (SQLSTATE 42884)。

Unicode データベースでは、文字ストリング・オペランドと GRAPHIC ストリング・オペランドの両方がかかわる連結の場合、まず文字オペランドが GRAPHIC オペランドに変換されます。非 Unicode データベースでは、文字と GRAPHIC の両方のオペランドが連結にかかわることはないことに注意してください。

いずれかのオペランドが NULL 値になる可能性がある場合は、結果も NULL 値になる可能性があり、いずれかが NULL 値なら結果は NULL 値になります。そうでない場合、結果は第 1 オペランド・ストリングの後に第 2 オペランド・ストリングが続いた形式となります。連結時に混合データが不正に形成されても、それに対する検査は行われません。

結果ストリングの長さは、オペランドの長さの合計になります。

結果のデータ・タイプと長さ属性は、以下の表に示すように、オペランドのデータ・タイプと長さ属性によって決まります。

表 15. 連結するオペランドのデータ・タイプと長さ

オペランド	連結後の長さ属性	結果
CHAR(A) CHAR(B)	<255	CHAR(A+B)
CHAR(A) CHAR(B)	>254	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
CHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
CHAR(A) LONG VARCHAR	-	LONG VARCHAR
VARCHAR(A) VARCHAR(B)	<4001	VARCHAR(A+B)
VARCHAR(A) VARCHAR(B)	>4000	LONG VARCHAR
VARCHAR(A) LONG VARCHAR	-	LONG VARCHAR
LONG VARCHAR LONG VARCHAR	-	LONG VARCHAR
CLOB(A) CHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) VARCHAR(B)	-	CLOB(MIN(A+B, 2G))
CLOB(A) LONG VARCHAR	-	CLOB(MIN(A+32K, 2G))
CLOB(A) CLOB(B)	-	CLOB(MIN(A+B, 2G))
GRAPHIC(A) GRAPHIC(B)	<128	GRAPHIC(A+B)
GRAPHIC(A) GRAPHIC(B)	>127	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
GRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
GRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
VARGRAPHIC(A) VARGRAPHIC(B)	<2001	VARGRAPHIC(A+B)
VARGRAPHIC(A) VARGRAPHIC(B)	>2000	LONG VARGRAPHIC
VARGRAPHIC(A) LONG VARGRAPHIC	-	LONG VARGRAPHIC
LONG VARGRAPHIC LONG VARGRAPHIC	-	LONG VARGRAPHIC
DBCLOB(A) GRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) VARGRAPHIC(B)	-	DBCLOB(MIN(A+B, 1G))
DBCLOB(A) LONG VARGRAPHIC	-	DBCLOB(MIN(A+16K, 1G))
DBCLOB(A) DBCLOB(B)	-	DBCLOB(MIN(A+B, 1G))
BLOB(A) BLOB(B)	-	BLOB(MIN(A+B, 2G))

旧バージョンとの互換性を保つために、 LONG データ・タイプに関連した結果は LOB データ・タイプに自動的にエスカレーションされないことに注意してください。例えば、CHAR(200) の値と、完全に文字の詰まった LONG VARCHAR の値とを連結した場合、 CLOB データ・タイプへプロモートされるのではなくエラーになります。

結果のコード・ページは派生コード・ページと見なされ、そのオペランドのコード・ページによって決定されます。

一方のオペランドはパラメーター・マーカーにすることができます。パラメーター・マーカーが使用されている場合、そのオペランドのデータ・タイプと長さ属性

は、パラメーター・マーカーでないオペランドと同じであると見なされます。ネストした連結の場合、これらの属性を決定できるように演算の順序を考慮する必要があります。

注: データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

例 1: FIRSTNME が Pierre で LASTNAME が Fermat である場合、以下のようになります。

```
FIRSTNME CONCAT ' ' CONCAT LASTNAME
```

Pierre Fermat の値が戻されます。

例 2: 以下を条件とします。

- COLA は、 'AA' の値を持つ VARCHAR(5) と定義されている。
- :host_var は、長さが 5 で値が 'BB ' である文字ホスト変数と定義されている。
- COLC は、値が 'CC' の CHAR(5) と定義されている。
- COLD は、値が 'DDDDD' の CHAR(5) と定義されている。

COLA **CONCAT** :host_var **CONCAT** COLC **CONCAT** COLD の値は、'AABB CC DDDDD' です。

データ・タイプが VARCHAR で、長さ属性は 17、結果コード・ページはデータベース・コード・ページとなります。

例 3: 以下を条件とします。

- COLA は、CHAR(10) と定義する。
- COLB は、VARCHAR(5) と定義する。

次の式の中のパラメーター・マーカーは、

```
COLA CONCAT COLB CONCAT ?
```

VARCHAR(15) と見なされます。これは、COLA CONCAT COLB が最初に評価され、その結果が 2 番目の CONCAT 演算の第 1 オペランドとなるためです。

ユーザー定義タイプ

ユーザー定義タイプは、ストリング・タイプのソース・データ・タイプがある特殊タイプであっても、連結演算子は使用できません。 連結するためには、そのソースとしての CONCAT 演算子を使った関数を作成する必要があります。例えば、TITLE と TITLE_DESCRIPTION という特殊タイプがあり、どちらも VARCHAR(25) データ・タイプである場合は、以下に示すユーザー定義関数 ATTACH でそれらを連結することができます。

```
CREATE FUNCTION ATTACH (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

別の方法として、新規のデータ・タイプを追加するユーザー定義関数を使用し、連結演算子を多重定義することもできます。

```
CREATE FUNCTION CONCAT (TITLE, TITLE_DESCRIPTION)
  RETURNS VARCHAR(50) SOURCE CONCAT (VARCHAR(), VARCHAR())
```

算術演算子がある式

算術演算子が使用されている場合、式の結果は、演算子をオペランドの値に適用して導かれた値となります。

いずれかのオペランドが NULL 値になる可能性がある場合、またはデータベースが DFT_SQLMATHWARN を yes に設定して構成されている場合、結果も NULL 値になる可能性があります。

どちらか一方のオペランドが NULL 値ならば、式の結果は NULL 値になります。

算術演算子は、符号付き数値タイプと日時タイプに適用できます (211 ページの『SQL における日付/時刻の算術演算』を参照)。例えば、USER+2 は無効です。ソース関数については、符号付き数値タイプであるソース・タイプを持つ特殊タイプ上の算術演算子に定義できます。

接頭演算子、+ (単項加算) はそのオペランドを変更しません。接頭演算子、- (単項減算) は、ゼロ以外の非 10 進浮動小数点オペランドの符号を逆にします。接頭演算子、- (単項減算) は、ゼロおよび特殊値を含め、すべての 10 進浮動小数点オペランドの符号を逆にします。その特殊値とはすなわち、シグナリング NaN と非シグナリング NaN、および正と負の無限大です。 A のデータ・タイプが短精度整数である場合、-A のデータ・タイプは長精度整数になります。接頭演算子の後に続くトークンの先頭の文字は、正または負の符号であってはなりません。

挿入演算子 +、-、*、および / はそれぞれ、加算、減算、乗算、および除算を指定します。除算の第 2 オペランドの値はゼロにすることはできません。これらの演算子は関数としても扱われます。したがって、式 *"+"(a,b)* は、式 *a+b* の演算子の機能と同じ意味になります。

算術演算エラー

ゼロ除算や数値のオーバーフローなどの算術演算エラーが、非 10 進浮動小数点式の処理の過程で生じると、エラー (SQLSTATE 22003 または 22012) が戻されます。 10 進浮動小数点式の場合は、算術計算条件の性質によって異なる警告 (SQLSTATE 0168C、0168D、0168E、または 0168F) が戻されます。

データベースは、非 10 進浮動小数点式で算術演算エラーが生じた場合に NULL 値を戻すように構成することが可能で (DFT_SQLMATHWARN を yes に設定して)、照会は警告 (SQLSTATE 01519 または 01564) を戻して、その SQL ステートメントの処理を続けます。

10 進浮動小数点式の場合は DFT_SQLMATHWARN に効果はありません。算術計算条件は適切な値を戻し (おそらく 10 進浮動小数点特殊値)、照会は警告を戻して (SQLSTATE 0168C、0168D、0168E、または 0168F)、その SQL ステートメントの処理を続けます。戻される特殊値には、正と負の無限大および Not a Number が含まれます。 1 つ以上の 10 進浮動小数点数を含む算術式は、式の 1 つ以上の引数が NULL でない限り、結果が NULL 値になることはありません。

算術計算エラーが NULL 値として扱われる場合、SQL ステートメントの結果に影響があります。以下は、このような影響の例を示しています。

- 列関数の引数の式で算術演算エラーが起きると、その列関数の結果を判別する際に行が無視されます。算術演算エラーがオーバーフローである場合、結果の値に大きな影響を与える場合があります。
- WHERE 節の述部の式で算術演算エラーが起きると、結果に行が入っていない場合があります。
- チェック制約の述部の式で算術演算エラーが起きても、制約には誤りがないため更新または挿入は続行されます。

このようなタイプの影響が受け入れられない場合、算術演算エラーを処理するのに必要な他の処置を行って、受け入れ可能な結果を生成する必要があります。例えば次のような処置です。

- ゼロによる除算の有無を検査するために CASE 式を追加して、このような状態に対応する必要な値を設定する。
- NULL 値を処理する述部を追加する (NULL 可能でない列のチェック制約は次のようになります)。

```
check (c1*c2 is not null and c1*c2>5000)
```

(これにより、オーバーフローの制約に違反する場合があります。)

2 つの整数オペランド

算術演算子のオペランドが両方とも整数の場合、その演算はバイナリー数で実行され、いずれかの (または両方の) オペランドが 64 ビット整数 (big integer) でない限り、その結果は*長精度整数 (large integer)* になります。いずれかの (または両方の) オペランドが 64 ビット整数である場合は、結果は 64 ビット整数になります。除算の剰余は失われます。整数算術演算 (単項減算符号を含む) の結果は、結果タイプの範囲内でなければなりません。

整数と 10 進数オペランド

一方のオペランドが整数で、もう一方のオペランドが 10 進数の場合、その演算は、精度 *p* および位取り 0 の 10 進数に変換されたその整数の一時コピーを使用して、 10 進数で行われます。 *p* は、64 ビット整数 (big integer) の場合 19 であり、長精度整数 (large integer) の場合 11 であり、短精度整数 (small integer) の場合 5 です。

2 つの 10 進数オペランド

オペランドが両方とも 10 進数の場合、その演算は 10 進数で行われます。 10 進数の算術演算の結果は 10 進数であり、その結果の精度と位取りは、演算の種類およびオペランドの精度と位取りによって異なります。演算が加算または減算で、オペランドの位取りが同じでない場合は、オペランドの一方の一時コピーを使用して演算が行われます。短い方のオペランドの小数部分が、長い方のオペランドと同じ桁数になるように、短い方のオペランドのコピーに後続ゼロを加えて拡張されます。

10 進数演算の結果は、精度が 31 以下でなければなりません。 10 進数の加算、減算、および乗算の結果は、精度が 31 を超える一時結果から求められることがあります。一時結果の精度が 31 を超えない場合、最終結果は一時結果と同じです。

SQL での 10 進数演算

以下の公式により、SQL における 10 進数演算の結果の精度および位取りが決まります。記号 *p* と *s* は第 1 オペランドの精度と位取りを表し、記号 *p'* と *s'* は第 2 オペランドの精度と位取りを表します。

加算および減算

精度は min(31,max(*p-s,p'-s'*) +max(*s,s'*)+1) になります。加算および減算の結果の位取りは max(*s,s'*) です。

乗算

乗算結果の精度は (31,*p* +*p'*)、位取りは min(31,*s* +*s'*) です。

除算

除算結果の精度は 31 です。位取りは *31-p+s-s'*です。位取りは負であってはなりません。

注: min_dec_div_3 データベース構成パラメーターは、除法に関係する 10 進算術演算の位取りを変更します。パラメーター値を NO に設定した場合、位取りは *31-p+s-s'* として計算されます。パラメーターを YES に設定した場合、位取りは MAX(3, *31-p+ s-s'*) として計算されます。これにより、10 進数の除算の結果は常に、少なくとも 3 桁になります (精度は常に 31 です)。

浮動小数点オペランド

算術演算子のいずれかのオペランドが浮動小数点数であるものの、10 進浮動小数点数ではない場合、演算は浮動小数点数で実行されます。オペランドは、必要に応じて、まず倍精度浮動小数点数に変換されます。したがって、式のエレメントのいずれかが浮動小数点数の場合、その式の結果は倍精度浮動小数点数になります。

浮動小数点数と整数の演算は、倍精度浮動小数点に変換した整数の一時コピーを使って実行されます。浮動小数点数と 10 進数の演算は、倍精度浮動小数点に変換した 10 進数の一時コピーを使って実行されます。浮動小数点数と 10 進数に関与した演算は、倍精度浮動小数点に変換した 10 進数の一時コピーを使って実行されます。浮動小数点数演算の結果は、浮動小数点数の範囲内でなければなりません。

浮動小数点数オペランドは実数の近似表現であるため、浮動小数点数オペランド (または関数の引数) が処理される順序が結果に多少影響する場合があります。オペランドが処理される順序がオプティマイザーによって暗黙的に変更される可能性があるため (例えば、使用する並列処理の度合いや、使用するアクセス・プランをオプティマイザーが決定する場合があります)、アプリケーションで浮動小数点数オペランドを使用する場合、SQL ステートメントが実行される度に毎回結果が厳密に同一であると期待すべきではありません。

10 進浮動小数点数オペランド

算術演算子のいずれかのオペランドが 10 進浮動小数点数の場合、その演算は 10 進浮動小数点数で実行されます。

整数と 10 進浮動小数点数オペランド
一方のオペランドが短精度整数または長精度整数で、他方のオペランドが DECFLOAT(*n*) の数値である場合、演算は DECFLOAT(*n*) で実行されます。その際、DECFLOAT(*n*) の数値に変換された整数の一時コピーが使用されます。一方のオペランドが 64 ビット整数 (big integer) で、他方のオペランドが 10 進浮動小数点数である場合には、64 ビット整数の一時コピーが DECFLOAT(34) の数値に変換されます。そして、2 つの 10 進浮動小数点数オペランドについての規則が適用されます。

10 進数と 10 進浮動小数点数オペランド
一方のオペランドが 10 進数で、他方のオペランドが 10 進浮動小数点数である場合、演算は 10 進浮動小数点数で実行されます。その際、10 進数の精度に基づいて 10 進浮動小数点数に変換された 10 進数の一時コピーが使用されます。 10 進数の精度が 17 より小さい場合、10 進数は DECFLOAT(16) の数値に変換されます。それ以外の場合、10 進数は DECFLOAT(34) の数値に変換されます。そして、2 つの 10 進浮動小数点数オペランドについての規則が適用されます。

浮動小数点数と 10 進浮動小数点数オペランド
一方のオペランドが浮動小数点数 (REAL または DOUBLE) で、他方のオペランドが DECFLOAT(*n*) の数値である場合、演算は 10 進浮動小数点数で実行されます。その際、DECFLOAT(*n*) の数値に変換された浮動小数点数の一時コピーが使用されます。

2 つの 10 進浮動小数点数オペランド
オペランドが両方とも DECFLOAT(*n*) である場合、演算は DECFLOAT(*n*) で実行されます。一方のオペランドが DECFLOAT(16) で、他方のオペランドが DECFLOAT(34) である場合、演算は DECFLOAT(34) で実行されます。

10 進浮動小数点数のための一般算術演算規則

以下の一般的な規則が、10 進浮動小数点データ・タイプでのすべての算術演算に適用されます。

- 有限数ではどの演算も、係数について整数算術計算を可能な限り使用して、正確な計算結果が算出されるかのように実行されます。

  理論上正確な結果の係数が、精度を反映する桁数 (16 または 34) 以下である場合は、変更なく、それが結果に使用されます (アンダーフローまたはオーバーフロー条件が起きない限り)。係数の桁数が精度を反映する桁数を上回る場合には、その精度を反映するちょうどの桁数 (16 または 34) に丸められて、取り除かれた桁数分だけ指数が増やされます。

  CURRENT DECFLOAT ROUNDING MODE 特殊レジスターは、丸めモードを決定します。

調整された結果の指数の値が E_{min} よりも小さい場合は、算出された係数と指数が結果を形成します。ただし、指数の値が E_{tiny} よりも小さい場合は例外です。その場合は指数が E_{tiny} に設定され、係数は指数の調整に一致するように (おそらくゼロに) 丸められて、符号は変わりません。この丸めによって不正確な結果になる場合は、アンダーフロー例外条件が戻されます。

調整された結果の指数の値が E_{max} よりも大きい場合は、オーバーフロー例外条件が戻されます。この場合、結果はオーバーフロー例外条件として定義され、無限大になる可能性があります。それは理論上の結果と同じ符号を持ちます。

- 特殊値の無限大を使用する算術計算は、通常の規則に従います。ここで、負の無限大はすべての有限数よりも小さく、正の無限大はすべての有限数よりも大きくなります。こうした規則のもとでは、無限大の結果は常に正確です。無限大のある種の使用は、無効な演算条件を戻します。以下のリストは、無効な演算条件が生じる可能性のある演算を示しています。そのような演算の結果は、一方のオペランドが無限大で、他方のオペランドが NaN または sNaN でない場合に、NaN になります。
  - 加算または減算演算中に +infinity を -infinity に加算する
  - 0 に +infinity または -infinity を乗算する
  - +infinity または -infinity のいずれかを +infinity または -infinity のいずれかで除算する
  - QUANTIZE 関数のいずれかの引数が +infinity または -infinity である
  - POWER 関数の 2 番目の引数が +infinity または -infinity である
  - 算術演算のオペランドとして、シグナリング NaN が使用されている

  以下の規則が算術演算および NaN 値に適用されます。
  - NaN (静止またはシグナリング) オペランドを持つすべての算術演算の結果は NaN になります。結果の符号は、シグナリング NaN である最初のオペランドからコピーされます。どちらのオペランドもシグナリングでない場合は、NaN である最初のオペランドからコピーされます。結果が NaN であるときはいつでも、結果の符号はコピーされたオペランドだけによって決まります。
  - 乗算または除算演算の結果の符号が負になるのは、2 つのオペランドの符号が異なっていて、どちらも NaN でない場合だけです。
  - 加算または減算演算の結果の符号が負になるのは、結果がゼロより小さく、どちらのオペランドも NaN でない場合だけです。例外として、以下のケースでは、結果が負の 0 になります。
    - 結果がゼロに丸められ、丸められる前の値が負符号を持っていた
    - -0 が 0 に加算された
    - 0 が -0 から減算された
    - 反対の符号を持つオペランドが加算され、あるいは同じ符号を持つオペランドが減算された結果として、係数が 0 になり、丸めモードが ROUND_FLOOR である
    - オペランドが乗算または除算された結果として係数が 0 になり、2 つのオペランドの符号が異なっている
    - POWER 関数の第 1 引数が -0 で、第 2 引数が正の奇数である
    - CEIL、FLOOR、または SQRT 関数の引数が -0 である

- ROUND または TRUNCATE 関数の第 1 引数が -0 である

以下の例は、オペランドとしての特殊 10 進浮動小数点値を示しています。

```
INFINITY + 1          = INFINITY
INFINITY + INFINITY   = INFINITY
INFINITY + -INFINITY  = NAN         -- 警告
NAN + 1               = NAN
NAN + INFINITY        = NAN
1 - INFINITY          = -INFINITY
INFINITY - INFINITY   = NAN         -- 警告
-INFINITY - -INFINITY = NAN         -- 警告
-0.0 - 0.0E1          = -0.0
-1.0 * 0.0E1          = -0.0
1.0E1 / 0             = INFINITY    -- 警告
-1.0E5 / 0.0          = -INFINITY   -- 警告
1.0E5 / -0            = -INFINITY   -- 警告
INFINITY / -INFINITY  = NAN         -- 警告
INFINITY / 0          = INFINITY
-INFINITY / 0         = -INFINITY
-INFINITY / -0        = INFINITY
```

オペランドとしてのユーザー定義タイプ

ユーザー定義タイプは、そのソース・データ・タイプが数値であっても算術演算子には使用できません。 算術演算を実行するには、そのソースとしての算術演算子を使用する関数を作成する必要があります。例えば、INCOME と EXPENSES という特殊タイプがあり、どちらも DECIMAL(8,2) データ・タイプである場合は、以下に示すユーザー定義関数 REVENUE を使って一方からもう一方を減算することができます。

```
CREATE FUNCTION REVENUE (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

別の方法として、新規のデータ・タイプを減算するユーザー定義関数を使って - (マイナス) 演算子を多重定義することも可能です。

```
CREATE FUNCTION "-" (INCOME, EXPENSES)
  RETURNS DECIMAL(8,2) SOURCE "-" (DECIMAL, DECIMAL)
```

演算の優先順位

括弧の中の式および間接参照操作が、最初に左から右へと評価されます。 (括弧は、subselect ステートメントや、検索条件、関数でも使用される点に注意してください。ただし、SQL ステートメント内でセクションを任意にグループ分けするのに使用することはできません。) 評価の順序が括弧で指定されていない場合は、まず接頭演算子が乗算および除算に先立って行われ、次に乗算と除算が加算および減算に先立って行われます。同じ優先順位の演算子は左から右に行われます。

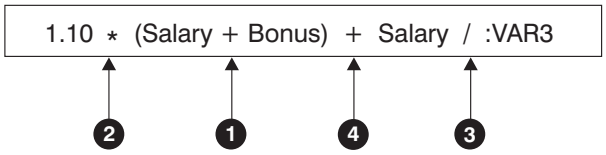


図 12. 演算の優先順位

スカラー全選択

式でサポートされるスカラー全選択 は、括弧で囲まれる全選択であり、1 つの列値で構成される 1 つの行を戻します。全選択が行を戻さない場合、式の結果は NULL 値になります。選択リスト・エレメントが単なる列名か間接参照の式である場合、その列の名前に基づいて結果列の名前が付けられます。スカラー全選択に必要な許可は、SQL 照会に必要な許可と同じです。

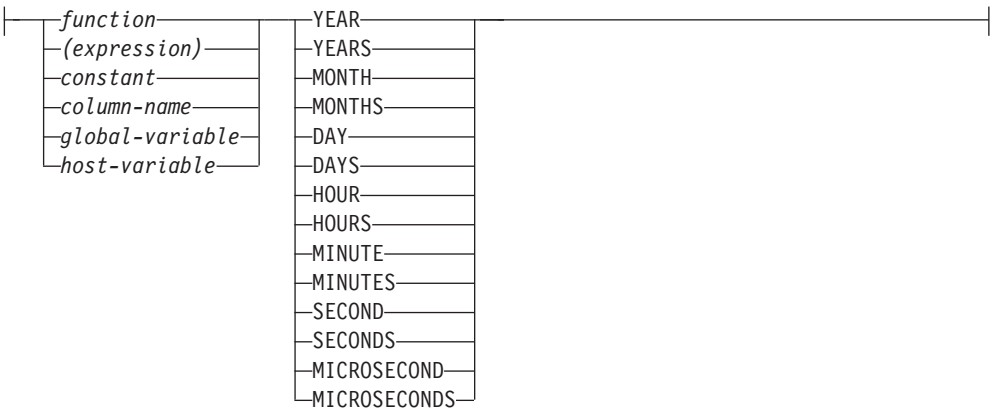
日付/時刻演算と期間

日付/時刻の値は、増分、減分、および減算を行うことができます。このような演算には、期間と呼ばれる 10 進数を使用する場合もあります。後続のいくつかのセクションでは、期間のタイプや日付/時刻算術計算の規則の詳細について説明しています。

期間

期間 とは、時間のインターバルを表す数値です。期間には以下の 4 つのタイプがあります。

labeled-duration:



ラベル付き期間 (labeled-duration) は、特定の時間単位を表すもので、数値 (式の結果でも可) の後に 7 つの期間キーワード
YEARS、MONTHS、DAYS、HOURS、MINUTES、SECONDS、または MICROSECONDS のうちの 1 つを付けたものです。 (これらのキーワードの単数形 YEAR、MONTH、DAY、HOUR、MINUTE、SECOND、および MICROSECOND も可能です。) 指定した値は、DECIMAL(15,0) の数値へ割り当てられる場合と同様に変換されます。ラベル付き期間は、算術演算子の 1 つのオペランドとしてのみ使用でき、このときの他方のオペランドは DATE、TIME、または TIMESTAMP です。したがって、式 HIREDATE + 2 MONTHS + 14 DAYS は有効ですが、式 HIREDATE + (2 MONTHS + 14 DAYS) は有効ではありません。どちらの式でも 2 MONTHS と 14 DAYS がラベル付き期間です。

日付期間 は、DECIMAL(8,0) の数値として表現される年数、月数、および日数を表します。正しく解釈されるには、この数値は *yyyymmdd* というフォーマットにする必要があります (*yyyy* は年数、*mm* は月数、*dd* は日数を表します)。 (このフォーマットの期間は、DECIMAL データ・タイプを示します。) 式 HIREDATE - BRTHDATE のように、ある日付値から別の日付値を減算した結果が日付期間です。

時刻期間 は、DECIMAL(6,0) の数値として表現される時間数、分数、および秒数を表します。正しく解釈されるには、この数値は *hhmmss.* というフォーマットにする

必要があります (*hh* は時間数、*mm* は分数、*ss* は秒数を表します)。 (このフォーマットの期間は、DECIMAL データ・タイプを示します。) ある時刻値から別の時刻値を減算した結果が時刻期間です。

タイム・スタンプ期間 は、DECIMAL(20,6) の数値として表現され、年数、月数、日数、時間数、分数、秒数、およびマイクロ秒数を表します。正しく解釈されるには、この数値を *yyyymmddhhmmss.nnnnnn* というフォーマットにする必要があります (*yyyy*、*mm*、*dd*、*hh*、*mm*、*ss*、および *nnnnnn* はそれぞれ、年数、月数、日数、時間数、分数、秒数、およびマイクロ秒数を表します)。あるタイム・スタンプ値から別のタイム・スタンプ値を減算した結果が、タイム・スタンプ期間です。

SQL における日付/時刻の算術演算

日付/時刻値に関して実行できる算術演算は加算と減算だけです。日付/時刻値が加算のオペランドである場合、他方のオペランドは期間でなければなりません。日付/時刻の値を使う加算演算子を使用するときには、次のような特有の規則があります。

- 一方のオペランドが日付である場合、もう一方のオペランドは日付期間、または YEARS、MONTHS、DAYS のラベル付き期間であることが必要です。
- 一方のオペランドが時刻である場合、もう一方のオペランドは時刻期間、または HOURS、MINUTES、SECONDS のラベル付き期間であることが必要です。
- 一方のオペランドがタイム・スタンプである場合、もう一方のオペランドは期間でなければなりません。この場合、期間のどのタイプでも有効です。
- 加算演算子のどちらのオペランドにも、パラメーター・マーカーは使用できません。

日付/時刻の値に減算演算子を使用する際の規則は、加算演算子の場合とは異なります。これは、日付/時刻の値を期間から引くことができないため、さらに 2 つの日付/時刻の値を差し引くことと期間を日付/時刻の値から差し引くこととは異なるためです。日付/時刻の値を使う減算演算子を使用するときには、次のような特有の規則があります。

- 第 1 オペランドが日付の場合、第 2 オペランドは日付、日付期間、日付のストリング表記、または YEARS、MONTHS、DAYS のラベル付き期間であることが必要です。
- 第 2 オペランドが日付の場合、第 1 オペランドは、日付または日付のストリング表記であることが必要です。
- 第 1 オペランドが時刻の場合、第 2 オペランドは、時刻、時刻期間、時刻のストリング表記、または HOURS、MINUTES、SECONDS のラベル付き期間であることが必要です。
- 第 2 オペランドが時刻の場合、第 1 オペランドは、時刻または時刻のストリング表記であることが必要です。
- 第 1 オペランドがタイム・スタンプの場合、第 2 オペランドは、タイム・スタンプまたはタイム・スタンプのストリング表記、または期間であることが必要です。
- 第 2 オペランドがタイム・スタンプの場合、第 1 オペランドは、タイム・スタンプまたはタイム・スタンプのストリング表記であることが必要です。
- 減算演算子のどちらのオペランドにも、パラメーター・マーカーは使用できません。

日付の算術演算

日付は、減算、増分、および減分を行うことができます。

- ある日付 (DATE2) を別の日付 (DATE1) から減算した結果は、これら 2 つの日付の間の年数、月数、日数を示す日付期間です。結果のデータ・タイプは DECIMAL(8,0) です。 DATE1 が DATE2 以上の場合、DATE1 から DATE2 が減算されます。これに対し、DATE1 が DATE2 より小さい場合は、DATE2 から DATE1 が減算され、結果の符号が負になります。演算 RESULT = DATE1 - DATE2 の実行ステップを、以下に順に示します。

  ```
  If DAY(DATE2) <= DAY(DATE1)
  then DAY(RESULT) = DAY(DATE1) - DAY(DATE2).

  If DAY(DATE2) > DAY(DATE1)
  then DAY(RESULT) = N + DAY(DATE1) - DAY(DATE2)
  where N = the last day of MONTH(DATE2).
  MONTH(DATE2) is then incremented by 1.

  If MONTH(DATE2) <= MONTH(DATE1)
  then MONTH(RESULT) = MONTH(DATE1) - MONTH(DATE2).

  If MONTH(DATE2) > MONTH(DATE1)
  then MONTH(RESULT) = 12 + MONTH(DATE1) - MONTH(DATE2).
  YEAR(DATE2) is then incremented by 1.

  YEAR(RESULT) = YEAR(DATE1) - YEAR(DATE2).
  ```

  例えば、DATE('3/15/2000') - '12/31/1999' の結果は 00000215 になります。 (すなわち、0 年 2 カ月 15 日の期間です。)
- 日付に期間を加算したり、日付から期間を減算したりすると、結果自体は日付となります。 (この演算では、月はカレンダーのページに相当します。つまり、日付に月を加算することは、その日付のページから順にカレンダーをめくっていくようなものです。) 結果は、0001 年 1 月 1 日以後 9999 年 12 月 31 日以前の日付となる必要があります。

  年の期間を加算または減算する場合、影響を受けるのは日付の年の部分だけです。月も日も変更されませんが、その結果がうるう年でない年の 2 月 29 日となった場合は別です。その場合は日が 28 に変更され、SQLCA の警告標識が日付調整の発生を示すように設定されます。

  同様に、月の期間を加算または減算する場合、影響を受けるのは月の部分だけです。ただし、必要に応じて年の部分にも影響が及びます。日付の日の部分は変更されませんが、結果が無効な場合 (例えば 9 月 31 日など) は別です。その場合は日がその月の最後の日に設定され、SQLCA の警告標識が日付調整の発生を示すように設定されます。

  日の期間を加算または減算すると、日付の中の日の部分は当然影響を受けますが、月および年も影響を受ける可能性があります。

  日付期間も、正負にかかわらず、日付に対して加減算が行えます。ラベル付き期間の場合と同じように、結果は有効な日付となり、月末の調整が必要になれば SQLCA の警告標識が設定されます。

正の日付期間が日付に加算されるとき、または負の日付期間が日付から減算されるときは、日付は、指定した年数、月数、日数の順で増分されます。したがって、X が正の DECIMAL(8,0) の数値であるとき、DATE1 + X は以下の式と同等です。

```
DATE1 + YEAR(X) YEARS + MONTH(X) MONTHS + DAY(X) DAYS.
```

正の日付期間を日付から減算するとき、または負の日付期間を日付に加算するとき、日付は、指定した日数、月数、年数の順で減分されます。したがって、X が正の DECIMAL(8,0) の数値であるとき、DATE1 - X は以下の式と同等です。

```
DATE1 - DAY(X) DAYS - MONTH(X) MONTHS - YEAR(X) YEARS.
```

期間を日付に加算するとき、特定の日付に 1 カ月を加算すると、1 カ月後の同じ日付になります。ただし、1 カ月後にその日付が存在しない場合は扱いが異なります。その場合、日付は 1 カ月後の最後の日に設定されます。例えば、1 月 28 日に 1 カ月を加えると 2 月 28 日になります。 1 月 29、30、または 31 日に 1 カ月を加えると通常の年では 2 月 28 日、うるう年では 2 月 29 日になります。

注: 特定の日付に 1 カ月以上の月数を加算し、その結果から同じ月数を減算した場合、最終的な日付が元の日付と同じになるとは限りません。

時刻の算術演算

時刻は、減算、増分、または減分を行うことができます。

- ある時刻 (TIME2) を別の時刻 (TIME1) から減算した結果は、それら 2 つの時刻の間の時間数、分数、秒数を示す時刻期間です。結果のデータ・タイプは DECIMAL(6,0) です。

  TIME1 が TIME2 より大か等しい場合、TIME1 から TIME2 が減算されます。

  これに対し、TIME1 が TIME2 より小さい場合は、TIME2 から TIME1 が減算され、結果の符号が負になります。演算 RESULT = TIME1 - TIME2 の実行ステップを、以下に順に示します。

  ```
  If SECOND(TIME2) <= SECOND(TIME1)
  then SECOND(RESULT) = SECOND(TIME1) - SECOND(TIME2).

  If SECOND(TIME2) > SECOND(TIME1)
  then SECOND(RESULT) = 60 + SECOND(TIME1) - SECOND(TIME2).
  MINUTE(TIME2) is then incremented by 1.

  If MINUTE(TIME2) <= MINUTE(TIME1)
  then MINUTE(RESULT) = MINUTE(TIME1) - MINUTE(TIME2).

  If MINUTE(TIME1) > MINUTE(TIME1)
  then MINUTE(RESULT) = 60 + MINUTE(TIME1) - MINUTE(TIME2).
  HOUR(TIME2) is then incremented by 1.

  HOUR(RESULT) = HOUR(TIME1) - HOUR(TIME2).
  ```

  例えば、TIME('11:02:26') - '00:32:56' の結果は 102930 になります。 (10 時間 29 分、30 秒の期間です。)
- 時刻に期間を加算したり、時刻から期間を減算したりすると、結果自体は時刻となります。時間数のオーバーフローやアンダーフローは捨てられ、これにより常に結果が時刻となります。時間数で指定する期間を加算または減算する場合、影響を受けるのは時間数の部分だけです。分数と秒数は変更されません。

同様に、分数で指定する期間を加算または減算する場合、影響を受けるのは分の部分だけです。ただし、必要に応じて時間数の部分にも影響が及びます。時刻の秒の部分は変更されません。

秒の期間を加算または減算すると、時刻の中の秒の部分は当然影響を受けますが、分および時も影響を受ける可能性があります。

時刻期間も、正負にかかわらず、時刻との加減算を行えます。結果は、指定した時間数、分数、秒数の順に増分または減分された時刻となります。 TIME1 + X (「X」は DECIMAL(6,0)) は次の式と同等です。

```
TIME1 + HOUR(X) HOURS + MINUTE(X) MINUTES + SECOND(X) SECONDS
```

注: 時刻 '24:00:00' は有効な値として受け付けられますが、時刻の加減算の結果として戻されることはありません。これは、期間オペランドがゼロであっても同じです (例えば、時刻 ('24:00:00')±0 秒 ='00:00:00' となります)。

タイム・スタンプの算術演算

タイム・スタンプは、減算、増分、または減分を行うことができます。

- あるタイム・スタンプ (TS2) を別のタイム・スタンプ (TS1) から減算した結果は、それら 2 つのタイム・スタンプの間の年数、月数、日数、時間数、分数、秒数、およびマイクロ秒数を示すタイム・スタンプ期間です。結果のデータ・タイプは DECIMAL(20,6) です。

  TS1 が TS2 以上の場合、TS1 から TS2 が減算されます。これに対し、TS1 が TS2 より小さい場合は、TS2 から TS1 が減算され、結果の符号が負になります。演算 RESULT = TS1 - TS2 の実行ステップを、以下に順に示します。

  ```
  If MICROSECOND(TS2) <= MICROSECOND(TS1)
  then MICROSECOND(RESULT) = MICROSECOND(TS1) -
  MICROSECOND(TS2).

  If MICROSECOND(TS2) > MICROSECOND(TS1)
  then MICROSECOND(RESULT) = 1000000 +
  MICROSECOND(TS1) - MICROSECOND(TS2)
  and SECOND(TS2) is incremented by 1.
  ```

  タイム・スタンプの秒および分の部分は、時刻の減算規則で指定されたように減算されます。

  ```
  If HOUR(TS2) <= HOUR(TS1)
  then HOUR(RESULT) = HOUR(TS1) - HOUR(TS2).

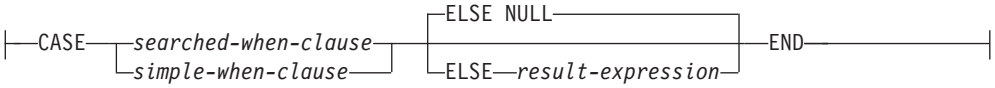
  If HOUR(TS2) > HOUR(TS1)
  then HOUR(RESULT) = 24 + HOUR(TS1) - HOUR(TS2)
  and DAY(TS2) is incremented by 1.
  ```

  タイム・スタンプの日付の部分は、日付の減算規則での説明と同じようにして減算されます。

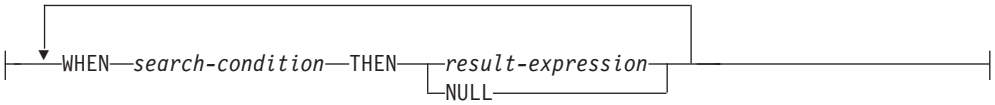
- タイム・スタンプに期間を加算したり、タイム・スタンプから期間を減算したりすると、結果自体はタイム・スタンプとなります。日付と時刻の算術演算はすでに説明したとおりに実行されますが、時間数のオーバーフローとアンダーフローは結果の日付の部分に繰り上げまたは繰り下げられ、有効な日付の範囲内に収められます。マイクロ秒のオーバーフローは秒に繰り上げられます。

CASE 式

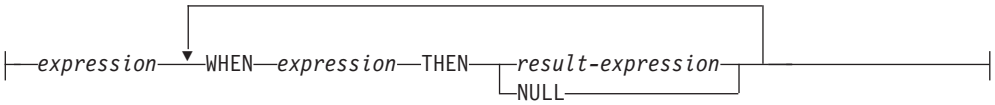
case-expression:



searched-when-clause:



simple-when-clause:



CASE 式は、1 つ以上の条件の評価に基づいて式を選択するためのものです。一般に、CASE 式の値は、評価が「真」である最初の (左端の) ケースの後に来る *result-expression* (結果式) の値になります。評価が「真」であるケースがなく、ELSE キーワードが指定されている場合、結果は ELSE の *result-expression* (結果式) または NULL になります。評価が「真」であるケースがなく、ELSE キーワードが指定されていない場合、結果は NULL になります。あるケースの評価が「不明」の場合 (NULL のため)、そのケースは「真」ではなく、したがって評価が「偽」であるケースと同じように扱われます。

CASE 式が VALUES 節、IN 述部、GROUP BY 節、または ORDER BY 節中にある場合、 searched-when-clause の *search-condition* は、比較述部、全選択を使用する IN 述部、または EXISTS 述部 (SQLSTATE 42625) にすることはできません。

simple-when-clause (単純 WHEN 節) を使用する場合は、最初の *WHEN* キーワードの前の *expression* (式) の値が、その WHEN キーワードの後にある *expression* の値と等しいかどうかが検査されます。このため、最初の *WHEN* キーワードの前の *expression* は、 WHEN キーワードの後に来るそれぞれの *expression* のデータ・タイプと互換である必要があります。 *simple-when-clause* の中の最初の *WHEN* キーワードの前にある *expression* で、可変の関数または外部処理を伴う関数を使用することはできません (SQLSTATE 42845)。

result-expression (結果式) は、 THEN または ELSE キーワードの後に指定する式です。 CASE 式では、少なくとも 1 つの *result-expression* を指定する必要があります (すべてのケースに NULL を指定することはできません) (SQLSTATE 42625)。すべての結果式のデータ・タイプは互換でなければなりません (SQLSTATE 42804)。

例

- 以下の例では、部門番号の先頭文字が組織内の部を示すものとし、 CASE 式を使用して、各社員が属する部の正式名称を取り出します。

```
SELECT EMPNO, LASTNAME,
  CASE SUBSTR(WORKDEPT,1,1)
    WHEN 'A' THEN 'Administration'
    WHEN 'B' THEN 'Human Resources'
    WHEN 'C' THEN 'Accounting'
    WHEN 'D' THEN 'Design'
    WHEN 'E' THEN 'Operations'
  END
  FROM EMPLOYEE;
```

- 就学年数は、学歴レベルを示す目的で EMPLOYEE 表で使用されています。CASE 式を使用して、これらを分類し、学歴レベルを示します。

```
SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
  CASE
    WHEN EDLEVEL < 15 THEN 'SECONDARY'
    WHEN EDLEVEL < 19 THEN 'COLLEGE'
    ELSE 'POST GRADUATE'
  END
  FROM EMPLOYEE
```

- CASE ステートメントの別の有効な使い方として、ゼロ除算によるエラーを防止することができます。例えば以下のコードは、収入のすべてではないが 25% より多くを歩合で得ている社員を検索しています。

```
SELECT EMPNO, WORKDEPT, SALARY+COMM FROM EMPLOYEE
WHERE (CASE WHEN SALARY=0 THEN NULL
  ELSE COMM/SALARY
  END) > 0.25;
```

- 以下の 2 つの CASE 式は同じものです。

```
SELECT LASTNAME,
  CASE
  WHEN LASTNAME = 'Haas' THEN 'President'
  ...

SELECT LASTNAME,
  CASE LASTNAME
  WHEN 'Haas' THEN 'President'
  ...
```

CASE の機能の一部を処理する目的で、スカラー関数の NULLIF と COALESCE が特別に用意されています。 表 16 に、 CASE を使用した場合とそれらの関数を使用した場合とで同等の式を示します。

表 16. 同等の CASE 式

式	同等の式

CASE
  WHEN e1=e2 THEN NULL
  ELSE e1
END
	NULLIF(e1,e2)

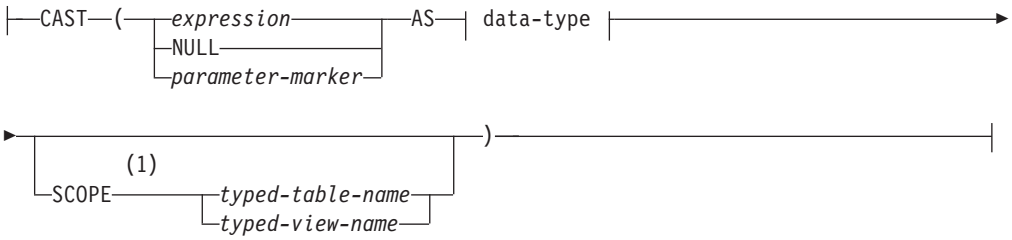
CASE
  WHEN e1 IS NOT NULL THEN e1
  ELSE e2
END
	COALESCE(e1,e2)

表 16. 同等の CASE 式 (続き)

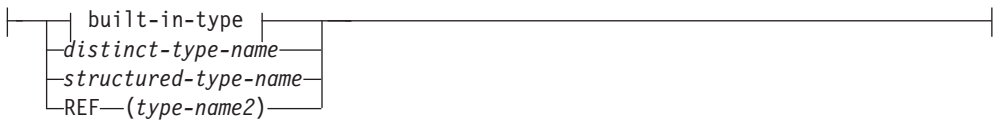
式	同等の式
CASE WHEN e1 IS NOT NULL THEN e1 ELSE COALESCE(e2,...,eN) END	COALESCE(e1,e2,...,eN)
CASE WHEN c1=var1 OR (c1 IS NULL AND var1 IS NULL) THEN 'a' WHEN c1=var2 OR (c1 IS NULL AND var2 IS NULL) THEN 'b' ELSE NULL END	DECODE(c1,var1, 'a', var2, 'b')

CAST 指定

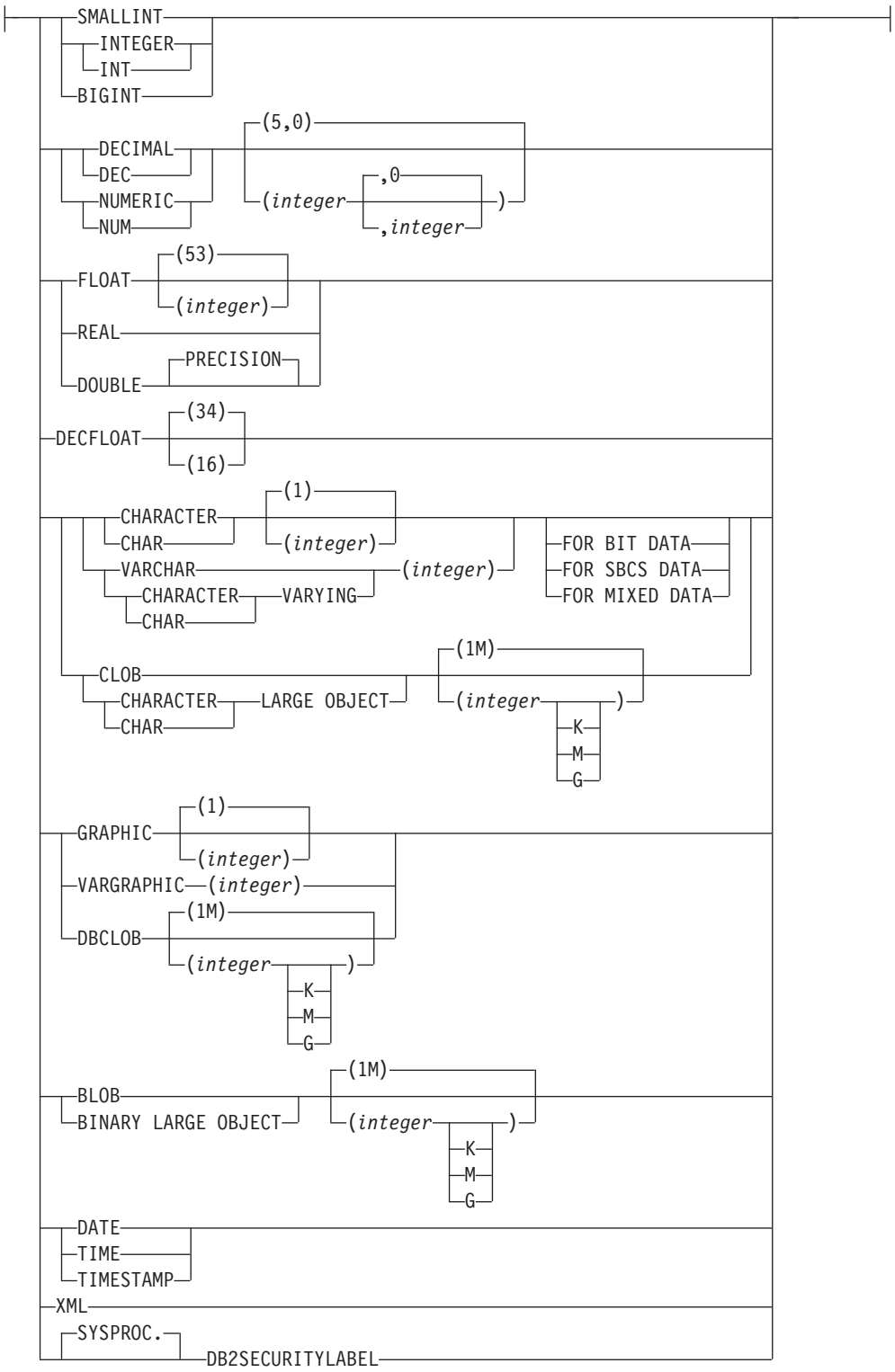
cast-specification:



data-type:



built-in-type:



注:

1 SCOPE 節が適用されるのは、REF データ・タイプのみです。

CAST 指定は、*data-type* によって指定されたタイプにキャストされたキャスト・オペランド (第 1 オペランド) を戻します。キャストがサポートされていない場合、エラー (SQLSTATE 42846) が戻されます。

expression
キャスト・オペランドが式 (パラメーター・マーカーまたは NULL ではなく) である場合、結果は、指定されたターゲット *data-type* に変換された引数値です。

文字ストリング (CLOB 以外) を長さの異なる文字ストリングにキャストするとき、後続ブランク以外の文字が切り捨てられると、警告 (SQLSTATE 01004) が戻されます。 GRAPHIC ストリング (DBCLOB 以外) を長さの異なる GRAPHIC ストリングにキャストするとき、後続ブランク以外の文字が切り捨てられると、警告 (SQLSTATE 01004) が戻されます。キャスト・オペランドが BLOB、CLOB、および DBCLOB の場合、何らかの文字が切り捨てられると警告が発行されます。

配列をキャストするとき、ターゲット・データ・タイプはユーザー定義の配列データ・タイプでなければなりません (SQLSTATE 42821)。配列の要素のデータ・タイプは、ターゲット配列データ・タイプの要素のデータ・タイプと同じでなければなりません (SQLSTATE 42846)。配列のカーディナリティーは、ターゲット配列データ・タイプの最大カーディナリティー以下でなければなりません (SQLSTATE 2202F)。

NULL
キャスト・オペランドがキーワード NULL である場合、結果は、指定された *data-type* の NULL 値です。

parameter-marker
パラメーター・マーカー (疑問符で指定されるもの) は通常は式として見なされますが、ここでは特別な意味をもつため別個に説明します。キャスト・オペランドが *parameter-marker* である場合、指定された *data-type* は、指定されたデータ・タイプに置き換えが割り当て可能である (ストリングの記憶割り当てを使用して) ことを示す合意であると見なされます。このようなパラメーター・マーカーは、*型付きパラメーター・マーカー* と見なされます。型付きパラメーター・マーカーは、関数解決、選択リストの DESCRIBE、または列割り当てを行う目的で、他の型付き値と同じように扱われます。

data-type
既存のデータ・タイプの名前。このタイプ名が修飾されていない場合は、SQL パスを使用してデータ・タイプ解決が行われます。 *data-type* を指定するとき、長さや精度と位取りのような属性が関連付けられているデータ・タイプは、これらの属性を組み込む必要があります。 (指定されていない場合、CHAR は長さ 1 にデフォルト解釈され、DECIMAL は精度 5 および位取り 0 にデフォルト解釈され、DECFLOAT は精度 34 にデフォルト解釈されます。) FOR SBCS DATA 節または FOR MIXED DATA 節 (データベースがグラフィック・データ・タイプをサポートするかどうかによって、 1 つだけがサポートされる) を使用して、FOR BIT DATA ストリングをデータベース・コード・ページにキャストできます。サポートされるデータ・タイプに関する制限は、指定したキャスト・オペランドに基づいて適用されます。

- キャスト・オペランドが式 の場合にサポートされるターゲット・データ・タイプは、キャスト・オペランドのデータ・タイプ (ソース・データ・タイプ) によって異なります。
- キャスト・オペランドがキーワード NULL の場合、既存のどのデータ・タイプでも指定できます。

- キャスト・オペランドがパラメーター・マーカーの場合、ターゲット・データ・タイプは、既存の任意のデータ・タイプとすることができます。データ・タイプがユーザー定義特殊タイプの場合、パラメーター・マーカーを使用するアプリケーションは、そのユーザー定義特殊タイプのソース・データ・タイプを使用します。データ・タイプがユーザー定義構造化タイプの場合、パラメーター・マーカーを使用するアプリケーションは、そのユーザー定義構造化タイプの TO SQL トランスフォーム関数の入力パラメーター・タイプを使用します。

SCOPE

データ・タイプが参照タイプの場合、有効範囲は参照のターゲット表またはターゲット・ビューを識別するように定義することができます。

typed-table-name

型付き表の名前。表名はすでに指定されていなければなりません (SQLSTATE 42704)。キャストは *data-type* REF(*S*) にするものでなければなりません。ここでの *S* は *typed-table-name* (SQLSTATE 428DM) のタイプを表しています。

typed-view-name

型付きビューの名前。そのビューは存在しているか、あるいはビュー定義の一部としてキャストを組み込むように作成されているビューと同じ名前でなければなりません (SQLSTATE 42704)。キャストは *data-type* REF(*S*) にするものでなければなりません。ここでの *S* は *typed-view-name* (SQLSTATE 428DM) のタイプを表しています。

数値データを文字データにキャストする場合、結果のデータ・タイプは固定長文字ストリングです。文字データを数値データにキャストする場合、結果のデータ・タイプは指定した数値のタイプによって異なります。例えば整数へのキャストの場合、結果のデータ・タイプは長精度整数になります。

例

- アプリケーションが、EMPLOYEE 表の SALARY (decimal(9,2) と定義) の整数部だけを使用するとします。社員番号や SALARY の整数値を備えた、以下のような照会が考えられます。

```
SELECT EMPNO, CAST(SALARY AS INTEGER) FROM EMPLOYEE
```

- SMALLINT に基づいて定義された T_AGE という名前の特殊タイプがあり、PERSONNEL 表に AGE 列を作成するために使用されるとします。さらに INTEGER に基づいて定義された R_YEAR という名前の特殊タイプがあり、PERSONNEL 表に RETIRE_YEAR 列を作成するために使用されるとします。以下のような更新ステートメントが考えられます。

```
UPDATE PERSONNEL SET RETIRE_YEAR =?
  WHERE AGE = CAST( ? AS T_AGE)
```

第 1 パラメーターは、データ・タイプ R_YEAR のタイプなしパラメーター・マーカーです。一方、アプリケーションはこのパラメーター・マーカーの整数部を使用します。この場合、これは割り当てなので、明示的な CAST 指定をする必要はありません。

2 番目のパラメーター・マーカーは、特殊タイプ T_AGE としてキャストされる型付きパラメーター・マーカーです。これにより、比較は互換データ・タイプと

の間でなければならない、という要件が満たされます。アプリケーションは、ソース・データ・タイプ (SMALLINT) を使用してこのパラメーター・マーカーを処理します。

このステートメントの正常な処理では、SQL パスには、2 つの特殊タイプを定義した 1 つ以上のスキーマのスキーマ名が入っていることを前提としています。

- アプリケーションは、例えばオーディオ・ストリームのような一つながりの値を提供しますが、その値は SQL ステートメントで使用される前にコード・ページの変換を経由してはなりません。アプリケーションは、次のような CAST を使用することができます。

  ```
  CAST( ? AS VARCHAR(10000) FOR BIT DATA)
  ```

- 配列タイプおよび表が以下のように作成されたと想定します。

  ```
  CREATE TYPE PHONELIST AS DECIMAL(10, 0) ARRAY[5]

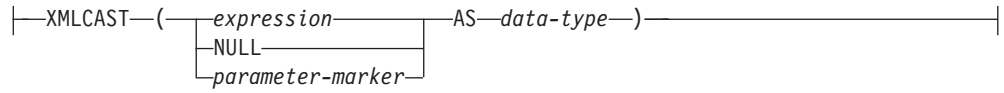
  CREATE TABLE EMP_PHONES
    (ID          INTEGER,
     PHONENUMBER DECIMAL(10,0) )
  ```

  以下のプロシージャーは、 ID が 1775 の従業員の電話番号を含む配列を戻します。この従業員に 5 より多い電話番号がある場合、エラーが戻されます (SQLSTATE 2202F)。

  ```
  CREATE PROCEDURE GET_PHONES(OUT EPHONES PHONELIST)
  BEGIN
    SELECT CAST(ARRAY_AGG(PHONENUMBER) AS PHONELIST)
    INTO EPHONES
    FROM EMP_PHONES
    WHERE ID = 1775;
  END
  ```

XMLCAST 指定

xmlcast-specification:



XMLCAST 指定は、データ・タイプによって指定されたタイプにキャストされたキャスト・オペランド (第 1 オペランド) を戻します。 XMLCAST は、非 XML データ・タイプと XML データ・タイプ間の変換を含む、XML 値に関するキャストをサポートします。キャストがサポートされていない場合、エラー (SQLSTATE 22003) が戻されます。

expression
キャスト・オペランドが式 (パラメーター・マーカーまたは NULL ではなく) である場合、結果は、指定されたターゲット・データ・タイプに変換された引数値です。式またはターゲット・データ・タイプは、XML データ・タイプでなければなりません (SQLSTATE 42846)。

NULL
キャスト・オペランドがキーワード NULL である場合、ターゲット・データ・タイプは XML データ・タイプでなければなりません (SQLSTATE 42846)。結果はヌル XML 値になります。

parameter-marker
キャスト・オペランドがパラメーター・マーカーである場合、ターゲット・データ・タイプは XML でなければなりません (SQLSTATE 42846) パラメーター・マーカー (疑問符で指定されるもの) は通常は式と見なされますが、ここでは特別な意味をもつため別個に説明します。キャスト・オペランドがパラメーター・マーカーである場合、指定されたデータ・タイプは、指定された (XML) データ・タイプに置き換えが割り当て可能である (記憶割り当てを使用して) ことを示す合意であると見なされます。このようなパラメーター・マーカーは型付きパラメーター・マーカーと見なされ、関数解決、選択リストの記述操作、または列割り当ての目的で他の型付き値と同様に扱われます。

data-type
既存の SQL データ・タイプの名前。この名前が修飾されていない場合は、SQL パスを使用してデータ・タイプ解決が行われます。データ・タイプに長さや精度と位取りのような属性が関連付けられている場合、*data-type* の値を指定する際に、これらの属性を組み込む必要があります。指定されていない場合、CHAR は長さ 1 にデフォルト解釈され、DECIMAL は精度 5 および位取り 0 にデフォルト解釈されます。サポートされるデータ・タイプに関する制限は、指定したキャスト・オペランドに基づいて適用されます。

- キャスト・オペランドが式の場合にサポートされるターゲット・データ・タイプは、キャスト・オペランドのデータ・タイプ (ソース・データ・タイプ) によって異なります。
- キャスト・オペランドがキーワード NULL である場合、ターゲット・データ・タイプは XML でなければなりません。

- キャスト・オペランドがパラメーター・マーカーである場合、ターゲット・データ・タイプは XML でなければなりません。

注:

1. **非 Unicode データベースでのサポート**: XMLCAST が XML 値を SQL データ・タイプに変換するのに使用されるとき、コード・ページ変換が実行されます。 キャスト式のエンコードが UTF-8 からデータベース・コード・ページに変換されます。 データベース・コード・ページには存在しない元の式の文字は、この変換の結果として、置換文字に置き換えられます。
2. **複数のデータベース・パーティションを持つデータベースでのサポート**: XMLCAST 指定は、単一データベース・パーティションを使用するデータベースでのみサポートされます (SQLSTATE 42997)。

例

- ヌル XML 値を作成する。

```
XMLCAST(NULL AS XML)
```

- XMLQUERY 式から抽出した値を INTEGER に変換する。

```
XMLCAST(XMLQUERY('$m/PRODUCT/QUANTITY'
  PASSING BY REF xmlcol AS "m" RETURNING SEQUENCE) AS INTEGER)
```

- XMLQUERY 式から抽出した値を可変長文字ストリングに変換する。

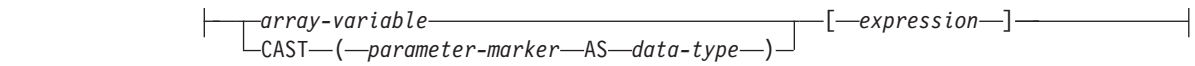
```
XMLCAST(XMLQUERY('$m/PRODUCT/ADD-TIMESTAMP'
  PASSING BY REF xmlcol AS "m" RETURNING SEQUENCE) AS VARCHAR(30))
```

- SQL スカラー副照会から抽出した値を XML 値に変換する。

```
XMLCAST((SELECT quantity FROM product AS p
  WHERE p.id = 1077) AS XML)
```

ARRAY エレメント仕様

array-element-specification:



ARRAY エレメント仕様は、*expression* で指定された配列からエレメントを戻します。*expression* のいずれかの引数が NULL である場合、NULL 値が戻されます。

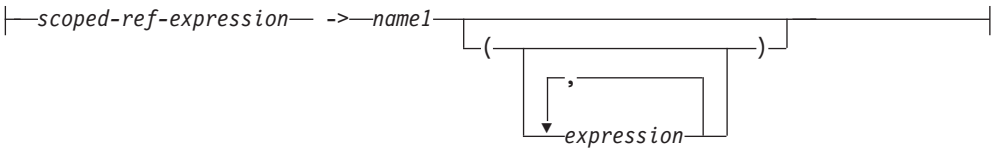
array-variable
タイプ ARRAY の変数またはパラメーターを SQL プロシージャーに指定します (SQLSTATE 428H0)。

CAST (*parameter-marker* **AS** *data-type***)**
(疑問符 (?) 文字として指定される) パラメーター・マーカーは、通常、式とみなされますが、この場合、これはユーザー定義の配列データ・タイプに明示的にキャストされる必要があります。

[*expression***]**
配列から抽出されるエレメントの副索引を指定します。副索引は、位取りがゼロの厳密な数値タイプでなければなりません (SQLSTATE 428H1)。その値は、1 と、配列のカーディナリティーの間でなければなりません (SQLSTATE 2202E)。

間接参照操作

dereference-operation:



有効範囲を指定した参照式の有効範囲は、ターゲット 表またはビューという表またはビューになります。効力範囲を指定した参照式は、*ターゲット行* を識別します。*ターゲット行* は、ターゲット表またはビューの (あるいは、副表かサブビューのいずれかの) 行です。この行のオブジェクト ID (OID) 列値は、参照式と一致しています。間接参照操作を使い、ターゲット行の列にアクセスしたり、そのターゲット行をメソッドのサブジェクトとして使用してメソッドを呼び出すことができます。間接参照操作の結果は、常に NULL になり得ます。 間接参照操作は、他のすべての操作よりも優先されます。

scoped-ref-expression
有効範囲を持っている参照タイプである式 (SQLSTATE 428DT)。式がホスト変数、パラメーター・マーカー、またはほかの有効範囲がない参照タイプ値の場合、 SCOPE 節での CAST 指定で有効範囲の参照を指定する必要があります。

name1
修飾なしの ID を指定します。

name1 の後に括弧がなく、 *name1* がターゲット・タイプの属性名と一致している場合、間接参照操作の値は、ターゲット行の名前付き列の値になります。その場合、列のデータ・タイプ (NULL 可能) の値により、間接参照操作の結果タイプが決まります。オブジェクト ID が参照式と一致するターゲット行がない場合、間接参照操作の結果は NULL になります。間接参照操作が選択リストで使用され、式の一部分としては組み込まれていない場合、 *name1* が結果の列名になります。

name1 の後に括弧があるか、*name1* がターゲット・タイプの属性名と一致しない場合、間接参照操作はメソッド呼び出しとして扱われます。呼び出されるメソッドの名前は *name1* です。メソッドのサブジェクトは、ターゲット行であり、その構造化タイプのインスタンスと見なされます。オブジェクト ID が参照式と一致するターゲット行がない場合、メソッドのサブジェクトは、ターゲット・タイプの NULL 値になります。括弧内に式があれば、それはメソッド呼び出しの残りのパラメーターを指定するものです。メソッド呼び出しの解決には、通常の処理が使われます。選択したメソッドの結果タイプ (NULL 可能) の値により、間接参照操作の結果タイプが決まります。

間接参照操作を使用するステートメントの許可 ID は、 *scoped-ref-expression* のターゲット表での SELECT 特権を持っていなければなりません (SQLSTATE 42501)。

間接参照操作では、データベース内の値を変更できません。間接参照操作を使用して変更メソッドを呼び出す場合、その変更メソッドはターゲット行のコピーを変更してコピーを戻しますが、データベースは未変更のままです。

例

- DEPTREF という列がある EMPLOYEE 表 (属性 DEPTNAME を持っているタイプに基づく型付き表を効力範囲とする参照タイプ) があるとします。表 EMPLOYEE の DEPTREF の値は、 DEPTREF 列のターゲット表にある OID 列値と対応していなければなりません。

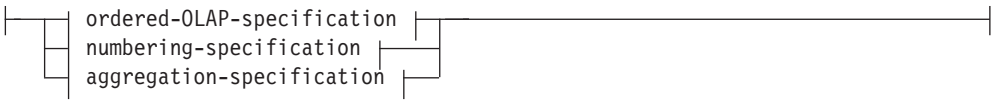
```
SELECT EMPNO, DEPTREF->DEPTNAME
  FROM EMPLOYEE
```

- 前の例と同じ表を使用し、間接参照操作を使って BUDGET というメソッドを呼び出します。その際に、ターゲット行をサブジェクト・パラメーターとして、そして '1997' を追加パラメーターとして指定します。

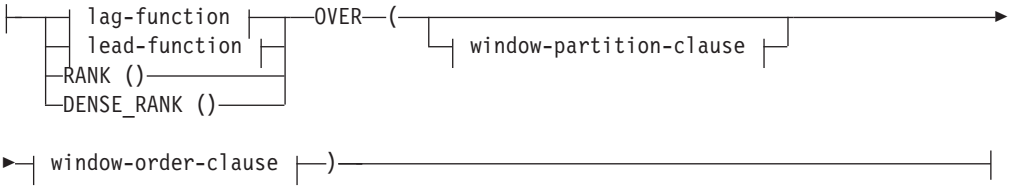
```
SELECT EMPNO, DEPTREF->BUDGET('1997') AS DEPTBUDGET97
  FROM EMPLOYEE
```

OLAP 仕様

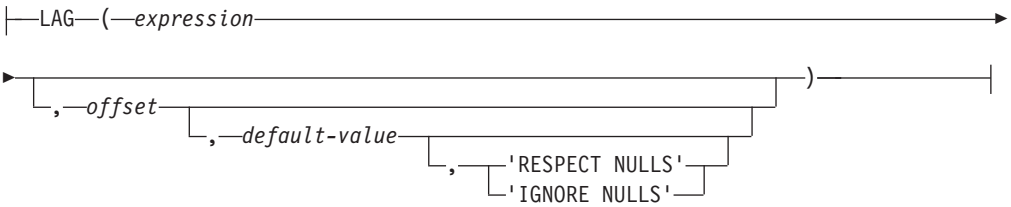
OLAP-specification:



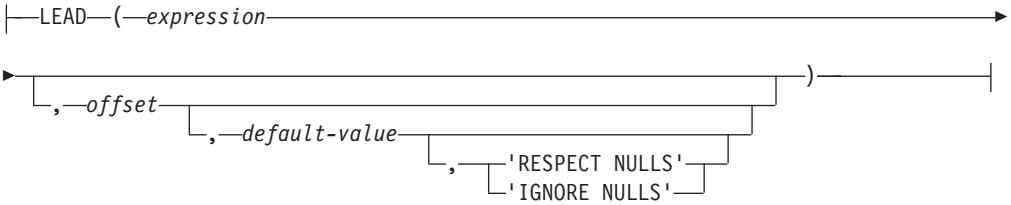
ordered-OLAP-specification:



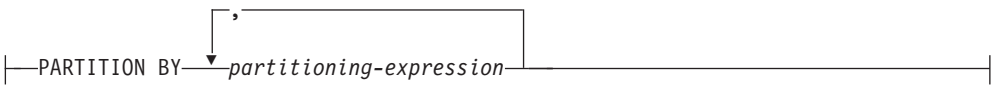
lag-function:



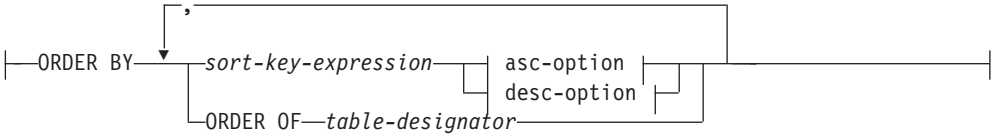
lead-function:



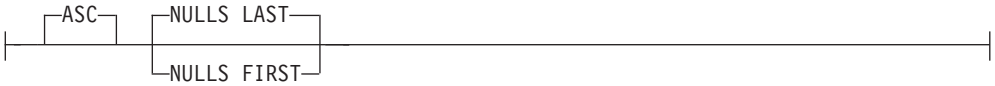
window-partition-clause:



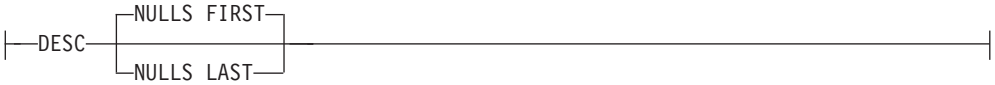
window-order-clause:



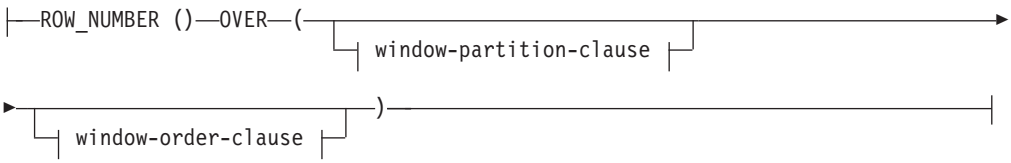
asc-option:



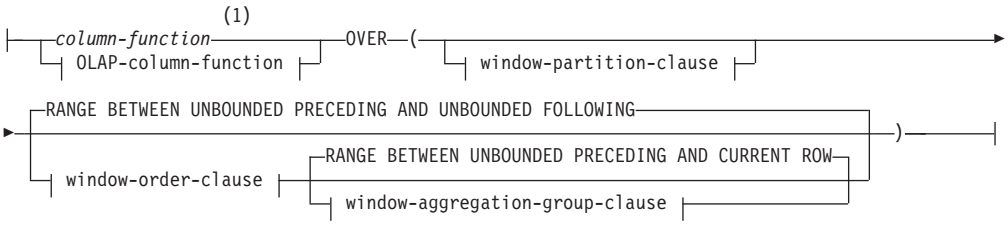
desc-option:



numbering-specification:



aggregation-specification:

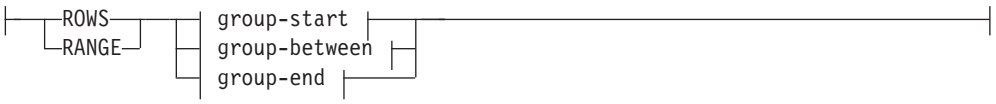


OLAP-column-function:

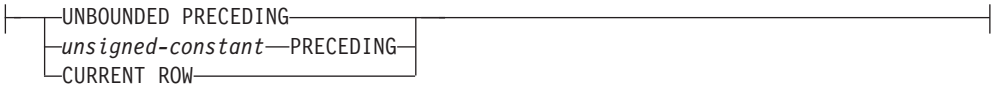
first-value-function:

last-value-function:

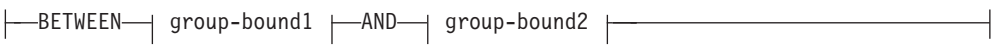
window-aggregation-group-clause:



group-start:



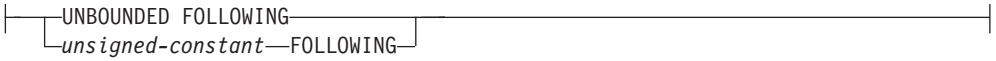
group-between:



group-bound1:

group-bound2:

group-end:



注:

1 ARRAY_AGG は、 *aggregation-specification* 内で列関数としてサポートされていません (SQLSTATE 42887)。

OLAP (On-Line Analytical Processing) 関数には、照会の結果の中で、ランキング、行番号、および既存の列関数情報をスカラー値で戻す機能があります。 OLAP 関数は、select-list の式、または select-statement の ORDER BY 節に組み込むことができます (SQLSTATE 42903)。 OLAP 関数は、XMLQUERY または XMLEXISTS 式への引数内で使用することはできません (SQLSTATE 42903)。 OLAP 関数を列関数の引数として使うことはできません (SQLSTATE 42607)。 OLAP 関数を適用したときの照会の結果は、その OLAP 関数の入った最も内側の副選択の結果表です。

OLAP 関数を指定するときには、関数を適用する行を定義したり、その順序を定義する枠が指定されます。列関数とともに使用すると、該当する行をさらに詳細化し

て、現在行との相対関係で、その前後の行範囲または行数として扱うことができます。例えば、月単位のパーティションでは、直前の四半期の平均を計算することができます。

ランキング関数は、枠内の行の序数ランクを計算します。それぞれの枠内での順序がはっきりしていない行は、同位に割り当てられます。ランキングの結果については、重複する値の結果の数値にギャップがあってもなくても定義できます。

RANK を指定すると、該当行に先行する行数に 1 を足した数で、行のランクが定義されます。したがって、順序がはっきりしていない行が 2 行以上あると、通しランク番号には、1 つ以上のギャップができます。

DENSE_RANK (または DENSERANK) を指定すると、順序の明確な先行行数に 1 を足して行のランクが定義されます。したがって、通しランク番号にはギャップはありません。

ROW_NUMBER (または ROWNUMBER) 関数は、最初の行を 1 行目とする順序付けで定義された枠内の行の通し番号を計算します。枠内で ORDER BY 節を指定していない場合、 (SELECT ステートメントの ORDER BY 節に基づくのではなく) 副選択で戻されたとおりに、任意の順番で行に行番号が割り当てられます。

FETCH FIRST *n* ROWS ONLY 節が ROW_NUMBER 関数と共に使用される場合、行番号が順序どおりに表示されないことがあります。 FETCH FIRST 節は、結果セット (ROW_NUMBER 割り当てを含む) が生成された後に適用されます。そのため、行番号の順序が結果セットの順序と同じでない場合、割り当てられた番号の一部がシーケンスから欠落することがあります。

RANK、DENSE_RANK、または ROW_NUMBER の結果のデータ・タイプは BIGINT です。結果が NULL 値になることはありません。

LAG 関数は、現在の行から *offset* 行前にある行の式の値を戻します。 *offset* は、正の整数定数でなければなりません (SQLSTATE 42815)。0 の *offset* 値は、現在行を意味します。window-partition-clause が指定されている場合、 *offset* とは現在のパーティションに含まれる、現在の行から *offset* 行前のことです。 *offset* が指定されていない場合、値 1 が使用されます。 *default-value* (これは式であることが可能) が指定されている場合、オフセットが現在のパーティションの有効範囲を超える場合にその値が戻されます。指定されていない場合は、NULL 値が戻されます。 'IGNORE NULLS' が指定されている場合、行の式値が NULL 値であるすべての行は計算に算入されません。 'IGNORE NULLS' が指定されていて、すべての行が NULL の場合、 *default-value* (または *default-value* が指定されていない場合には NULL 値) が戻されます。

LEAD 関数は、現在の行から *offset* 行後にある行の式の値を戻します。 *offset* は、正の整数定数でなければなりません (SQLSTATE 42815)。0 の *offset* 値は、現在行を意味します。window-partition-clause が指定されている場合、 *offset* とは現在のパーティションに含まれる、現在の行から *offset* 行後のことです。 *offset* が指定されていない場合、値 1 が使用されます。 *default-value* (これは式であることが可能) が指定されている場合、オフセットが現在のパーティションの有効範囲を超える場合にその値が戻されます。指定されていない場合は、NULL 値が戻されます。 'IGNORE NULLS' が指定されている場合、行の式値が NULL 値であるすべての行

は計算に算入されません。 'IGNORE NULLS' が指定されていて、すべての行が NULL の場合、 *default-value* (または *default-value* が指定されていない場合には NULL 値) が戻されます。

FIRST_VALUE 関数は、OLAP ウィンドウ内の最初の行の式値を戻します。 'IGNORE NULLS' が指定されている場合、行の式値が NULL 値であるすべての行は計算に算入されません。 'IGNORE NULLS' が指定されていて、OLAP ウィンドウ内のすべての値が NULL の場合、 FIRST_VALUE は NULL 値を戻します。

LAST_VALUE 関数は、OLAP ウィンドウ内の最後の行の式の値を戻します。 'IGNORE NULLS' が指定されている場合、行の式値が NULL 値であるすべての行は計算に算入されません。 'IGNORE NULLS' が指定されていて、OLAP ウィンドウ内のすべての値が NULL の場合、 LAST_VALUE は NULL 値を戻します。

FIRST_VALUE、LAG、LAST_VALUE、および LEAD の結果のデータ・タイプは、式のデータ・タイプとなります。結果は NULL 値の場合もあります。

PARTITION BY (*partitioning-expression***,...)**
関数を適用するときのパーティションを定義します。 *partitioning-expression* は、結果セットのパーティションを定義するときに使う式です。 *partitioning-expression* で参照されている各 *column-name* は、OLAP 関数副選択ステートメントの結果セット列をはっきり参照するものでなければなりません (SQLSTATE 42702 または 42703)。 *partitioning-expression* には、スカラー全選択、XMLQUERY または XMLEXISTS 式 (SQLSTATE 42822)、決定論的でないか外部アクションを持つ関数または照会 (SQLSTATE 42845) を含めることはできません。

window-order-clause

ORDER BY (*sort-key-expression***,...)**
OLAP 関数の値、または window-aggregation-group-clause の ROW 値の意味を決める、パーティション内の行の順序を定義します (照会結果セットの順序を定義するものではありません)。

sort-key-expression
枠のパーティション内の行の順序を定義するのに使う式。 *sort-key-expression* で参照されている各 column-name は、OLAP 関数を含む副選択の結果セットの列をはっきり参照するものでなければなりません (SQLSTATE 42702 または 42703)。 *sort-key-expression* には、スカラー全選択、XMLQUERY または XMLEXISTS 式 (SQLSTATE 42822)、決定論的でないか外部アクションを持つ関数または照会 (SQLSTATE 42845) を含めることはできません。この節は、RANK および DENSE_RANK 関数 (SQLSTATE 42601) で必要になります。

ASC
sort-key-expression の値を昇順に使用します。

DESC
sort-key-expression の値を降順に使用します。

NULLS FIRST
ウィンドウ配列において、ソート順序は、すべての非 NULL 値の*前*に NULL 値が置かれます。

NULLS LAST
ウィンドウ配列において、ソート順序は、すべての非 NULL 値の*後に* NULL 値が置かれます。

ORDER OF *table-designator*
表指定子 で使用されているのと同じ順序付けを、副選択の結果表にも適用することを指定します。この節を指定する副選択の FROM 節内には、*表指定子* に一致する表参照がなければなりません (SQLSTATE 42703)。指定された *表指定子* に対応する副選択 (または全選択) には、データに依存する ORDER BY 節が入っていなければなりません (SQLSTATE 428FI)。適用される順序は、ネストされた副選択 (または全選択) 内の ORDER BY 節の列が外部副選択 (または全選択) に入っていた場合、およびそれらの列が ORDER OF 節の代わりに指定された場合と同じです。

window-aggregation-group-clause
行 R の集約グループは、(R のパーティションの行の順序付け内の) R に関連して定義されている行のセットです。その節は集約グループを指定します。この節が指定されない場合で、window-order-clause も指定されなければ、集約グループはウィンドウ・パーティションのすべての行から構成されます。 このデフォルトは、RANGE (示したように) または ROWS を明示的に使用して指定できます。

window-order-clause が指定された場合、デフォルトの動作は window-aggregation-group-clause が指定されていない場合は異なります。ウィンドウ集約グループは、window-order-clause により定義されたウィンドウ・パーティションのウィンドウ順序内で、R に先行するまたは R のピアである R のパーティションのすべての行で構成されます。

ROWS
集約グループがカウント行によって定義されることを示します。

RANGE
集約グループがソート・キーからのオフセットによって定義されることを示します。

group-start
集約グループの開始点を指定します。集約グループの終了は current row です。 group-start 節の仕様は、 "BETWEEN group-start AND CURRENT ROW" 形式の group-between 節と同じです。

group-between
ROWS または RANGE に基づいて、集約グループの開始および終了を指定します。

group-end
集約グループの終了点を指定します。集約グループの開始は current row です。 group-end 節の仕様は、 "BETWEEN CURRENT ROW AND group-end" 形式の group-between 節と同じです。

UNBOUNDED PRECEDING
current row の前のパーティション全体を組み込みます。これは、ROWS または RANGE のいずれかと一緒に指定できます。 window-order-clause 内の複数の sort-key-expressions と一緒に指定することもできます。

UNBOUNDED FOLLOWING
current row に続くパーティション全体を組み込みます。これは、ROWS または RANGE のいずれかと一緒に指定できます。 window-order-clause 内の複数の sort-key-expressions と一緒に指定することもできます。

CURRENT ROW
current row に基づいて、集約グループの開始および終了を指定します。 ROWS が指定された場合、current row が集約グループ境界です。 RANGE が指定された場合、集約グループ境界には、 current row と同じ値を *sort-key-expressions* として持つ行のセットが組み込まれます。 *group-bound1* で *value* FOLLOWING が指定されている場合、この節を *group-bound2* で指定することはできません。

value **PRECEDING**
current row の前の行の範囲または行数のいずれかを指定します。 ROWS が指定された場合、*value* は行数を示す正の整数です。 RANGE が指定された場合、*value* のデータ・タイプは、 window-order-clause の sort-key-expression のタイプと互換性がなければなりません。 sort-key-expression は 1 つのみで、 sort-key-expression のデータ・タイプは減算を許可しなければなりません。 *group-bound1* が CURRENT ROW または *value* FOLLOWING の場合、この節を *group-bound2* で指定することはできません。

value **FOLLOWING**
current row の後の行の範囲または行数のいずれかを指定します。 ROWS が指定された場合、*value* は行数を示す正の整数です。 RANGE が指定された場合、*value* のデータ・タイプは、 window-order-clause の sort-key-expression のタイプと互換性がなければなりません。 sort-key-expression は 1 つのみで、 sort-key-expression のデータ・タイプは加算を許可しなければなりません。

例

- 給与合計 (給与 + ボーナス) が $30,000 を超えている従業員のランキングを、それぞれの給与合計に基づいて、姓の順に表示します。

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
  FROM EMPLOYEE WHERE SALARY+BONUS > 30000
  ORDER BY LASTNAME
```

  結果をランキング順に並べる場合、ORDER BY LASTNAME を以下のように置き換えます。

```
ORDER BY RANK_SALARY
```

  または

```
ORDER BY RANK() OVER (ORDER BY SALARY+BONUS DESC)
```

- それぞれの給与合計の平均に基づいて部門をランク付けします。

```
SELECT WORKDEPT, AVG(SALARY+BONUS) AS AVG_TOTAL_SALARY,
    RANK() OVER (ORDER BY AVG(SALARY+BONUS) DESC) AS RANK_AVG_SAL
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  ORDER BY RANK_AVG_SAL
```

- それぞれの学歴に基づいて部門内で従業員をランク付けします。部門内で同じランクの従業員が複数いた場合は、次のランキング値を増やさないようにします。

```
SELECT WORKDEPT, EMPNO, LASTNAME, FIRSTNME, EDLEVEL,
    DENSE_RANK() OVER
      (PARTITION BY WORKDEPT ORDER BY EDLEVEL DESC) AS RANK_EDLEVEL
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- 照会の結果に行番号を示します。

```
SELECT ROW_NUMBER() OVER (ORDER BY WORKDEPT, LASTNAME) AS NUMBER,
    LASTNAME, SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, LASTNAME
```

- 収入の多い上位 5 人をリストします。

```
SELECT EMPNO, LASTNAME, FIRSTNME, TOTAL_SALARY, RANK_SALARY
  FROM (SELECT EMPNO, LASTNAME, FIRSTNME, SALARY+BONUS AS TOTAL_SALARY,
    RANK() OVER (ORDER BY SALARY+BONUS DESC) AS RANK_SALARY
    FROM EMPLOYEE) AS RANKED_EMPLOYEE
  WHERE RANK_SALARY < 6
  ORDER BY RANK_SALARY
```

  ランクを WHERE 節で使うために、事前にそのランキングも含めた結果をまず計算するのに、ネストされた表の式が使われていることに注意してください。共通表式も使われています。

- 部門ごとに、従業員の給与をリストして、各人の給与がその部門で次に給与の高い従業員と比較してどれほど少ないかを示します。

```
SELECT EMPNO, WORKDEPT, LASTNAME, FIRSTNME, JOB, SALARY,
    LEAD(SALARY, 1) OVER (PARTITION BY WORKDEPT
      ORDER BY SALARY) - SALARY AS DELTA_SALARY
  FROM EMPLOYEE
  ORDER BY WORKDEPT, SALARY
```

- 従業員の給与を、同じタイプの仕事のために最初に雇用された従業員との相対関係で計算します。

```
SELECT JOB, HIREDATE, EMPNO, LASTNAME, FIRSTNME, SALARY,
    FIRST_VALUE(SALARY) OVER (PARTITION BY JOB
      ORDER BY HIREDATE) AS FIRST_SALARY,
    SALARY - FIRST_VALUE(SALARY) OVER (PARTITION BY JOB
      ORDER BY HIREDATE) AS DELTA_SALARY
  FROM EMPLOYEE
  ORDER BY JOB, HIREDATE
```

- 2006 年 1 月中の株式 XYZ の平均終値を計算します。指定の日に株式が取り引きされなかった場合、DAILYSTOCKDATA 表でのその終値は NULL 値になります。株式が取り引きされなかった日に対して NULL 値を戻す代わりに、COALESCE 関数および LAG 関数を使用して、株式が取り引きされた直近の日付での終値を戻すようにします。直前の非 NULL の終値を検索することを、2006 年 1 月から 1 カ月前に制限します。

```
WITH V1(SYMBOL, TRADINGDATE, CLOSEPRICE) AS
(
SELECT SYMBOL, TRADINGDATE,
  COALESCE(CLOSEPRICE,
    LAG(CLOSEPRICE,
      1,
      CAST(NULL AS DECIMAL(8,2)),
      'IGNORE NULLS')
    OVER (PARTITION BY SYMBOL
    ORDER BY TRADINGDATE)
)
```

```
FROM DAILYSTOCKDATA
WHERE SYMBOL = 'XYZ' AND
  TRADINGDATE BETWEEN '2005-12-01' AND '2006-01-31'
)
SELECT SYMBOL, AVG(CLOSEPRICE) AS AVG
  FROM V1
  WHERE TRADINGDATE BETWEEN '2006-01-01' AND '2006-01-31'
  GROUP BY SYMBOL
```

- 2005 年中の株式 ABC および XYZ の 30 日移動平均を計算します。

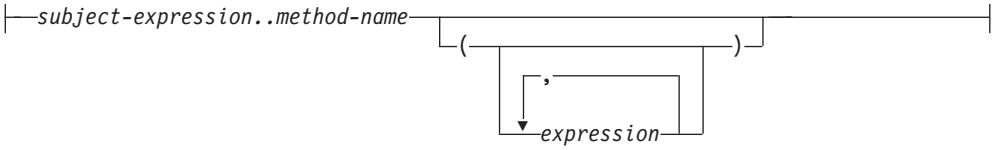
```
WITH V1(SYMBOL, TRADINGDATE, MOVINGAVG30DAY) AS
(
SELECT SYMBOL, TRADINGDATE,
  AVG(CLOSEPRICE) OVER (PARTITION BY SYMBOL
    ORDER BY TRADINGDATE
  ROWS BETWEEN 29 PRECEDING AND CURRENT ROW)
FROM DAILYSTOCKDATA
WHERE SYMBOL IN ('ABC', 'XYZ')
  AND TRADINGDATE BETWEEN DATE('2005-01-01') - 2 MONTHS
    AND '2005-12-31'
)
SELECT SYMBOL, TRADINGDATE, MOVINGAVG30DAY
  FROM V1
  WHERE TRADINGDATE BETWEEN '2005-01-01' AND '2005-12-31'
  ORDER BY SYMBOL, TRADINGDATE
```

- カーソル位置を定義する式を使用して、その位置よりも 50 行前のスライディング・ウィンドウを照会します。

```
SELECT DATE, FIRST_VALUE(CLOSEPRICE + 100) OVER
    (PARTITION BY SYMBOL
    ORDER BY DATE
    ROWS BETWEEN 50 PRECEDING AND 1 PRECEDING) AS FV
  FROM DAILYSTOCKDATA
  ORDER BY DATE
```

メソッドの呼び出し

method-invocation:



システム生成による監視および変更メソッドの両方、さらにユーザー定義メソッドも、二重ドット演算子を使って呼び出されます。

subject-expression
ユーザー定義構造化タイプである静的結果タイプを持つ式。

method-name
修飾なしのメソッド名。 *subject-expression* の静的タイプまたはそのスーパータイプのいずれかに、指定した名前を持つメソッドが入っている必要があります。

(*expression***,...)**
括弧内に *method-name* の引数を指定します。引数がないことを示すときには、括弧内を空にしておくことができます。特定のメソッドを解決するときに、*subject-expression* の静的タイプに基づき、 *method-name* と、指定した引数の式のデータ・タイプを使用します。

メソッド呼び出しに使う二重ドット演算子は、優先順位が高い順に左から右へ列挙される挿入演算子です。たとえば、以下の 2 つの式は同じことを意味します。

```
a..b..c + x..y..z
```

および

```
((a..b)..c) + ((x..y)..z)
```

メソッドにサブジェクト以外のパラメーターがない場合、括弧があってもなくても呼び出すことができます。たとえば、以下の 2 つの式は同じことを意味します。

```
point1..x
point1..x()
```

メソッド呼び出しの NULL サブジェクトは、次のように扱われます。

- システム生成の変更メソッドが NULL サブジェクトで呼び出される場合、エラーになります (SQLSTATE 2202D)。
- システム生成の変更メソッド以外のメソッドが NULL サブジェクトで呼び出される場合、そのメソッドは実行されず、結果は NULL になります。この規則は、SELF AS RESULT を指定したユーザー定義メソッドにも当てはまります。

データベース・オブジェクト (パッケージ、ビュー、またはトリガーなど) を作成する場合、それぞれのメソッド呼び出しのための最適な方法を見つけられます。

注: 定義した WITH FUNCTION ACCESS タイプのメソッドは、通常の関数表記を使用して呼び出すこともできます。関数解決では、候補となる関数として、すべての関数だけでなく、関数アクセスのあるメソッドも考慮します。ただし、メソッド呼び出しを使用して関数を呼び出すことはできません。メソッド解決では、候補と

なるメソッドとして、すべてのメソッドを考慮しますが、関数については考慮しません。適切な関数またはメソッドの解決に失敗すると、エラーになります (SQLSTATE 42884)。

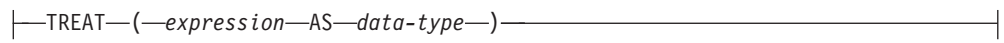
例

- 二重ドット演算子を使用して、AREA というメソッドを呼び出します。構造化タイプ CIRCLE の列 CIRCLE_COL をもった RINGS という表が存在するとします。また、CIRCLE タイプのために、メソッド AREA が、 AREA() RETURNS DOUBLE としてあらかじめ定義されているとします。

```
SELECT CIRCLE_COL..AREA() FROM RINGS
```

サブタイプの扱い

subtype-treatment:



subtype-treatment は、構造化タイプの式を、そのサブタイプのいずれかへキャストするときに使用します。 *expression* の静的タイプは、ユーザー定義構造化タイプでなければなりません。このタイプは、*data-type* と同じタイプであるか、またはそのスーパータイプでなければなりません。 *data-type* のタイプ名が修飾されていない場合は、 SQL パスを使用してタイプ参照を解決します。 subtype-treatment の結果の静的タイプは *data-type* であり、 subtype-treatment の値は expression の値になります。実行時に、expression の動的タイプが *data-type* ではないか、 *data-type* のサブタイプでない場合、エラーが戻されます (SQLSTATE 0D000)。

例

- 列 CIRCLE_COL のすべての列オブジェクト・インスタンスに、動的タイプ COLOREDCIRCLE があることを、アプリケーション側が認識している場合、次の照会を使って、そのようなオブジェクト上でメソッド RGB を呼び出します。構造化タイプ CIRCLE の列 CIRCLE_COL をもった RINGS という表が存在するとします。また、COLOREDCIRCLE は CIRCLE のサブタイプであり、COLOREDCIRCLE のために、メソッド RGB が RGB() RETURNS DOUBLE としてあらかじめ定義されているとします。

  ```
  SELECT TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
    FROM RINGS
  ```

  実行時に、動的タイプ CIRCLE のインスタンスが存在する場合、エラーになります (SQLSTATE 0D000)。このエラーは、次に示すように、CASE 式の中で TYPE 述部を使うことで避けることができます。

  ```
  SELECT (CASE
    WHEN CIRCLE_COL IS OF (COLOREDCIRCLE)
      THEN TREAT (CIRCLE_COL AS COLOREDCIRCLE)..RGB()
      ELSE NULL
    END)
    FROM RINGS
  ```

シーケンス参照

sequence-reference:

nextval-expression:

prevval-expression:

NEXT VALUE FOR *sequence-name*
NEXT VALUE 式は、 *sequence-name* で指定されたシーケンスの次の値を生成して返します。

PREVIOUS VALUE FOR *sequence-name*
PREVIOUS VALUE 式は、現行アプリケーション・プロセス内の直前のステートメントに指定されたシーケンスについて最後に生成された値を返します。この値は、シーケンスの名前が指定されている PREVIOUS VALUE 式を使用して、繰り返し参照することができます。単一ステートメント内に同じシーケンスを指定している PREVIOUS VALUE 式のインスタンスが複数存在する可能性があり、それらはすべて同じ値を返します。パーティション・データベース環境では、最も新しく生成された値が PREVIOUS VALUE 式によって返されない場合があります。

同じシーケンス名が指定されている NEXT VALUE 式が既に現行アプリケーション・プロセスで、現在または前のトランザクションで参照されている場合のみ、 PREVIOUS VALUE 式を使用できます (SQLSTATE 51035)。

注

- NEXT VALUE 式がシーケンスの名前を指定していれば、そのシーケンスの新しい値が生成されます。ただし、照会の中に同じシーケンス名を指定している NEXT VALUE 式のインスタンスが複数ある場合、シーケンスのカウンターは結果の行ごとに 1 つずつ増えていき、 NEXT VALUE のすべてのインスタンスが結果の行に同じ値を戻します。
- 以下に示すように、同じシーケンス番号は、先頭の行の NEXT VALUE 式 (これはシーケンス値を生成します) およびその他の行の PREVIOUS VALUE 式 (PREVIOUS VALUE のインスタンスは現在のセッションで最後に生成されたシーケンス値を参照します) を使用してシーケンス番号を参照することによって、 2 つの異なる表内のユニーク・キー値として使用することができます。

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- NEXT VALUE 式と PREVIOUS VALUE 式は、以下の位置に指定できます。

  - select-statement または SELECT INTO ステートメント (ステートメントに DISTINCT キーワード、 GROUP BY 節、ORDER BY 節、UNION キーワード、INTERSECT キーワード、または EXCEPT キーワードが入っていなければ、select-clause 内)
  - INSERT ステートメント (VALUES 節内)
  - INSERT ステートメント (全選択の select-clause 内)
  - UPDATE ステートメント (SET 節内 (検索条件付き、または位置指定 UPDATE ステートメントのいずれか)、ただし NEXT VALUE は SET 節にある式の全選択の select-clause 内に指定できない)
  - SET 変数ステートメント (式の全選択の select-clause 内を除きます。トリガー内に NEXT VALUE 式を指定することができますが、PREVIOUS VALUE 式は指定できません)。
  - VALUES INTO ステートメント (式の全選択の select-clause 内)
  - CREATE PROCEDURE ステートメント (SQL プロシージャーのルーチン本体内)
  - トリガー・アクション内の CREATE TRIGGER ステートメント (NEXT VALUE 式は指定できるが、 PREVIOUS VALUE 式は指定できない)
- NEXT VALUE 式と PREVIOUS VALUE 式は、以下の位置には指定できません。
  - 完全外部結合の結合条件
  - CREATE TABLE または ALTER TABLE ステートメント内の列の DEFAULT 値
  - CREATE TABLE または ALTER TABLE ステートメント内の生成列定義
  - CREATE TABLE または ALTER TABLE ステートメント内のサマリー表定義
  - CHECK 制約の条件
  - CREATE TRIGGER ステートメント (NEXT VALUE 式は指定できるが、PREVIOUS VALUE 式は指定できない)
  - CREATE VIEW ステートメント
  - CREATE METHOD ステートメント
  - CREATE FUNCTION ステートメント
  - XMLQUERY、XMLEXISTS、または XMLTABLE 式の引数リスト
- また、以下の位置に NEXT VALUE 式を指定することはできません (SQLSTATE 428F9)。
  - CASE 式
  - 総計関数のパラメーター・リスト
  - それ以前に明示的に許可されていない場合、コンテキスト内の副照会
  - 外部 SELECT に DISTINCT 演算子を備えた SELECT ステートメント
  - 結合の結合条件
  - 外部 SELECT に GROUP BY 節を備えた SELECT ステートメント
  - 外部 SELECT が UNION、INTERSECT、または EXCEPT セット演算子を使用して他の SELECT ステートメントと組み合わされている場合の SELECT ステートメント

  - ネストされた表の式
  - 表関数のパラメーター・リスト
  - 最外部の SELECT ステートメントか、DELETE または UPDATE ステートメントの WHERE 節
  - 最外部の SELECT ステートメントの ORDER BY 節
  - UPDATE ステートメントの SET 節にある式の全選択の select-clause
  - SQL ルーチンにおける IF、 WHILE、 DO...UNTIL、または CASE ステートメント
- シーケンスについて値が生成されると、その値を再使用できなくなるため、次に値が要求されたときに新しい値が生成されます。 NEXT VALUE 式が組み込まれているステートメントが失敗した場合やロールバックされた場合でも、これが当てはまります。

  列の VALUES リストにある NEXT VALUE 式が INSERT ステートメントに組み込まれており、 INSERT 実行中のある時点でエラー (次のシーケンス値を生成しているときの問題、あるいは別の列の値に問題があると考えられる) が起こった場合、挿入は失敗し (SQLSTATE 23505)、シーケンスについて生成した値は再使用できないものと見なされます。場合によっては、同じ INSERT ステートメントを再発行することによって、正しく動作します。

  例えば、NEXT VALUE が使用されていた列のユニーク索引が存在する結果としてエラーが起こり、既に生成されているシーケンス値がその索引に存在するとします。シーケンスについて生成される次の値は、索引には存在しない値になることが考えられるため、後続の INSERT が正しく動作します。
- シーケンスの値の生成において、そのシーケンスが最大値 (または降順シーケンスの最小値) に達し、循環が許可されていない場合、エラーが起こります (SQLSTATE 23522)。この場合、ユーザーはシーケンスを ALTER して許容値の範囲を拡張、またはシーケンスの循環を可能にでき、あるいは値の範囲がより大きな、異なるデータ・タイプを持つ新しいシーケンスを DROP および CREATE することができます。

  たとえば、シーケンスがデータ・タイプ SMALLINT で定義されていて、その結果、そのシーケンスが割り当て可能な値を使い果たしてしまうことがあります。シーケンスを新しい定義で DROP および再作成して、そのシーケンスを INTEGER として再定義しなければならない場合があります。
- カーソルの SELECT ステートメント内の NEXT VALUE に対する参照は、結果表の行について生成される値を参照します。データベースから取り出される行ごとに NEXT VALUE 式のシーケンス値が生成されます。クライアントでブロッキングを行うと、サーバーで FETCH ステートメントの処理の前に値が生成されることがあります。この状況は、結果表の行がブロッキングされている場合に生じることがあります。クライアント・アプリケーションが、データベースでマテリアライズされている行をすべて明示的に FETCH しないと、 (マテリアライズされている行のうち戻されなかったものの) シーケンス値が生成されません。
- カーソルの SELECT ステートメント内の PREVIOUS VALUE に対する参照は、そのカーソルをオープンする前に、生成された指定シーケンスの値を参照します。しかしながら、カーソルをクローズすると、後続するステートメント内の、PREVIOUS VALUE によって戻される指定シーケンスの値に影響が生じることが

あります。このことは、カーソルを再オープンした同じステートメントの場合でも生じることがあります。カーソルの SELECT ステートメントに入っている NEXT VALUE に対する参照中のシーケンス名が同じである場合はこのようになります。

- ***互換性***
  - 以前のバージョンの DB2 との互換性:
    - NEXTVAL と PREVVAL は、NEXT VALUE と PREVIOUS VALUE の代わりに指定できます。
  - IBM IDS との互換性:
    - NEXT VALUE FOR *sequence-name* の代わりに *sequence-name*.NEXTVAL を指定できます。
    - PREVIOUS VALUE FOR *sequence-name* の代わりに *sequence-name*.CURRVAL を指定できます。

例

"order" という表があり、 "order_seq" という以下のようなシーケンスが作成されると想定します。

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

以下は、NEXT VALUE 式で "order_seq" シーケンス番号を生成する方法例を示しています。

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

または

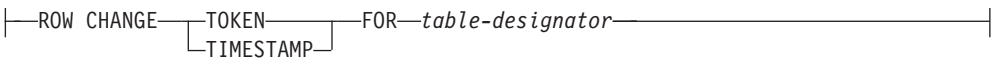
```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

または

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

ROW CHANGE 式

row-change-expression:



ROW CHANGE 式は、行に対する最終変更を表すトークンまたはタイム・スタンプを戻します。

TOKEN
行の変更シーケンス内での相対点を表す BIGINT 値を戻すことを指定します。行が変更されていない場合、結果は初期値が挿入されたときを表すトークンとなります。結果は NULL 値の場合もあります。 ROW CHANGE TOKEN は、決定論的ではありません。

TIMESTAMP
行が最後に変更されたときを表す TIMESTAMP を戻すことを指定します。行が変更されていない場合、結果は初期値が挿入された時刻となります。結果は NULL 値の場合もあります。 ROW CHANGE TIMESTAMP は、決定論的ではありません。

FOR *table-designator*
式が参照される表を識別します。 *table-designator* は、基本表、ビュー、またはネストされた表の式を一意的に識別する必要があります (SQLSTATE 42867)。 *table-designator* がビューまたはネストされた表の式を識別する場合、ビューまたはネストされた表の式の基本表の ROW CHANGE 式は TOKEN または TIMESTAMP を戻します。ビューまたはネストされた表の式は、外部副選択に 1 つだけの基本表を含んでいる必要があります (SQLSTATE 42867)。 *table-designator* は削除可能でなければなりません (SQLSTATE 42703)。削除可能なビューについて詳しくは、『CREATE VIEW』の『注』セクションを参照してください。ROW CHANGE TIMESTAMP 式の表指定子は、解決して行変更のタイム・スタンプ列を含む基本表になる必要があります (SQLSTATE 55068)。

注

- オプティミスティック・ロックを使用するアプリケーションは、 ROW CHANGE TOKEN 式によって戻される値を RID_BIT スカラー関数と共に使用できます。

例

- 部門 20 の従業員の EMPLOYEE 表にある各行への最新の変更に対応するタイム・スタンプ値を戻します。 EMPLOYEE 表が変更されて、ROW CHANGE TIMESTAMP 節で定義された列を含むようになったと想定します。

```
SELECT ROW CHANGE TIMESTAMP FOR EMPLOYEE
  FROM EMPLOYEE WHERE DEPTNO = 20
```

- 従業員番号 3500 に対応する、行の変更シーケンス内での相対点を表す BIGINT 値を戻します。また、オプティミスティック・ロック DELETE シナリオで使用される RID_BIT スカラー関数値も戻します。 WITH UR オプションを指定して、最新の ROW CHANGE TOKEN 値を取得します。

```
SELECT ROW CHANGE TOKEN FOR EMPLOYEE, RID_BIT (EMPLOYEE)
  FROM EMPLOYEE WHERE EMPNO = '3500' WITH UR
```

上記のステートメントは、行変更のタイム・スタンプ列が EMPLOYEE 表にあってもなくても成功します。以下の探索済み DELETE ステートメントは、ROW CHANGE TOKEN および RID_BIT 値で指定された行を、 2 つのパラメーター・マーカー値が上記のステートメントから取得した値に設定されたと想定して、上記の SELECT ステートメントから削除します。

```
DELETE FROM EMPLOYEE E
  WHERE RID_BIT (E) = ? AND ROW CHANGE TOKEN FOR E = ?
```

述部

述部 とは、特定の行またはグループに対して「真」、「偽」、または「不明」の条件を指定するものです。

以下の規則は、すべてのタイプの述部に適用されます。

- 述部の中で指定される値は、すべて互換でなければなりません。
- 基本、比較、IN、または BETWEEN 述部の式の結果が、長さ属性が 4000 を超える文字ストリング、長さ属性が 2000 を超える GRAPHIC ストリング、任意のサイズの LOB ストリングになってはなりません。
- ホスト変数の値は、NULL 値にすることができます (つまり、変数が負の標識変数を持つことがあります)。
- LIKE を除き、2 つ以上のオペランドを伴う述部のオペランドのコード・ページ変換は、ストリング変換の規則に従って行われます。
- 構造化タイプ値の使用は、NULL 述部と TYPE 述部に限定されています。
- Unicode データベースでは、文字または GRAPHIC ストリングを受け入れるすべての述部は、変換をサポートされている任意のストリング・タイプを受け入れます。

全選択は、SELECT ステートメントの 1 つの形式で、述部で使用されるとき、*副照会*とも呼ばれます。

照会の述部の処理

述部とは、比較演算を表す、あるいは暗黙のうちに示す検索条件の 1 つの要素です。述部は、その述部が評価プロセスにおいていつ、どのように用いられるかによって 4 つのカテゴリーに分類できます。そのカテゴリーは以下に示すとおりです (パフォーマンスの高いものから順に配列しています)。

- 範囲区切り述部は、索引スキャンの範囲を限定するために使用される述部です。これらは、索引検索のための開始キーまたは停止キーの値を指定します。これらの述部は、索引マネージャーによって評価されます。
- 索引検索引数述部は、検索の範囲を限定するためには使用されず、選択された場合に索引から評価されます。これは、この述部に関係する列が索引キーの一部であるためです。これらの述部も、索引マネージャーによって評価されます。
- データ検索引数述部は、索引マネージャーでは評価できない述部ですが、データ管理サービス (DMS) によって評価できます。通常、これらの述部には、基本表の個々の行へのアクセスが必要となります。必要に応じて DMS は、述部を評価するために必要な列を取り出し、さらに SELECT リストの列を満たすもののうち索引からは取得できなかったものを取り出します。
- 残余述部は、基本表の単純アクセスの域を越えた入出力が必要となる述部です。残余述部の例には、定量化された副照会 (ANY、ALL、SOME、または IN を含む副照会) を使用する述部や、表とは別に保管されている LONG VARCHAR またはラージ・オブジェクト (LOB) のデータを読み取る述部が含まれます。これらの述部はリレーショナル・データ・サービス (RDS) によって評価され、述部の 4 つのカテゴリーの中で最もコストがかかります。

次の表では、さまざまな述部の例と、使用されるコンテキストに基づくそれらのタイプを示しています。

注: この例では、 (c1, c2, c3) に複数列の昇順の索引が存在し、それが必要に応じて述部の評価に使用されることを想定しています。索引内のいずれかの列が降順である場合は、範囲区切り述部の開始キーと停止キーが入れ替わる可能性があります。

表 17. 各種の照会の述部の処理

述部	列 c1	列 c2	列 c3	コメント
c1 = 1 and c2 = 2 and c3 = 3	範囲区切り (開始-停止)	範囲区切り (開始-停止)	範囲区切り (開始-停止)	索引のすべての列についての等号述部を、開始-停止キーとして適用できます。
c1 = 1 and c2 = 2 and c3 ≥ 3	範囲区切り (開始-停止)	範囲区切り (開始-停止)	範囲区切り (開始)	列 c1 と c2 は等号述部によってバインドされ、c3 についての述部のみが開始キーとして適用されます。
c1 ≥1 and c2 = 2	範囲区切り (開始)	範囲区切り (開始-停止)	該当なし	先行列 c1 は ≥ 述部を持つので、開始キーとして使用できます。後続の列 c2 は等号述部を持つので、これも開始-停止キーとして適用できます。

表 17. 各種の照会の述部の処理 (続き)

述部	列 c1	列 c2	列 c3	コメント
c1 = 1 and c3 = 3	範囲区切り (開始-停止)	該当なし	索引検索引数述部	c2 についての述部がないので、c3 についての述部は開始-停止キーとして使用できません。しかし、索引検索引数述部として適用できます。
c1 = 1 and c2 > 2 and c3 = 3	範囲区切り (開始-停止)	範囲区切り (開始)	索引検索引数述部	前の列が > 述部を持っているので、c3 についての述部は開始-停止述部として適用できません。仮にこれが ≥ であったなら、これは開始-停止キーとして使用できます。
c1 = 1 and c2 ≤ 2 and c4 = 4	範囲区切り (開始-停止)	範囲区切り (停止)	データ検索引数述部	この例では、c2 についての述部が ≤ 述部です。 これは停止キーとして使用できます。c4 についての述部はこの索引には適用できず、FETCH の間にデータ検索引数述部として適用されます。
c2 = 2 and UDF_with_external_action (c4)	該当なし	索引検索引数述部	残余	先行列 c1 は述部を持たないので、c2 の述部は索引検索引数述部として適用でき、この場合は索引全体がスキャンされます。外部アクションを含むユーザー定義関数が関係する述部は、残余述部として適用されます。
c1 = 1 or c2 = 2	索引検索引数述部	索引検索引数述部	該当なし	OR があるため、この複数列索引は開始-停止キーとして使用できません。仮に 2 つの索引があって、一方の先行列が c1、もう一方の先行列が c2 であり、なおかつ DB2 オプティマイザーが「索引の ORing」プランを選択していたとすれば、そのように使用することもできます。 しかし、この例では、2 つの述部は索引検索引数述部として扱われます。
c1 < 5 and (c2 = 2 or c3 = 3)	範囲区切り (停止)	索引検索引数述部	索引検索引数述部	この例では、先行列 c1 を利用し、停止キーによって索引スキャンによる述部の使用を停止しています。c2 と c3 についての OR 述部は、索引検索引数述部として適用されます。

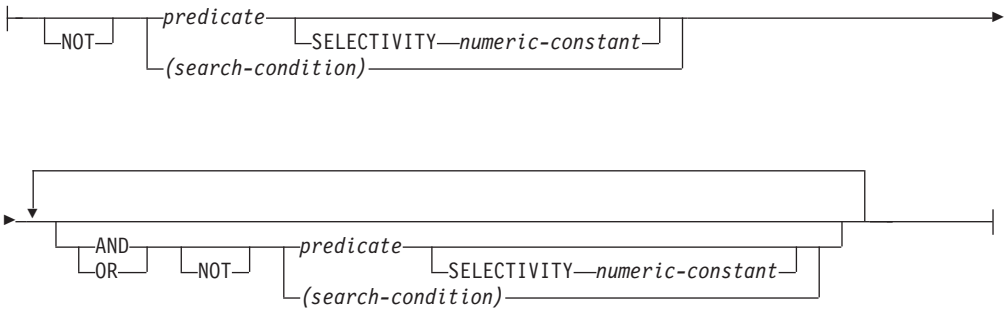
DB2 オプティマイザーは、多数の複雑なユーザー作成述部を、以下の表で示すような、よりパフォーマンスの良い照会にトランスフォームするための照会書き直し機構を採用しています。

表 18. 照会書き直しの述部

元の述部または照会	最適化された述部	コメント
c1 between 5 and 10	c1 ≥ 5 and c1 ≤ 10	BETWEEN 述部は、それと同等の範囲区切り述部に書き直されて、ユーザーが範囲区切り述部を指定した場合と同様に内部で使用できるようになります。
c1 not between 5 and 10	c1 < 5 or c1 > 10	OR 述部がある場合は、 DB2 オプティマイザーが索引の ORing のプランを選択していない限り、開始-停止キーは使用できません。
SELECT * **FROM** t1 **WHERE EXISTS** (**SELECT** c1 **FROM** t2 **WHERE** t1.c1 = t2.c1)	**SELECT** t1.* **FROM** t1 EOJOIN t2 **WHERE** t1.c1= t2.c1	副照会は結合にトランスフォームされる可能性があります。
SELECT * **FROM** t1 **WHERE** t1.c1 **IN** (**SELECT** c1 **FROM** t2)	**SELECT** t1* **FROM** t1 EOJOIN t2 **WHERE** t1.c1= t2.c1	これは、上記の EXISTS 述部のトランスフォーメーションの例と類似しています。
c1 like 'abc%'	c1 ≥ 'abc X X X ' and c1 ≤ 'abc Y Y Y'	索引の先行列が c1 の場合、DB2 はこれらの述部を生成して、範囲区切りの開始-停止述部として適用できるようにします。 この例で、文字 X と Y は、最低と最高の照合文字を表しています。
c1 like 'abc%def'	c1 ≥ 'abc X X X ' and c1 ≤ 'abc Y Y Y' and c1 like 'abc%def'	これは前の例と類似していますが、さらに元の述部を索引検索引数述部として適用する必要があるという点が異なります。そうすることにより、「def」の文字が正しく検出されるようにします。

検索条件

search-condition:



search-condition (検索条件) は、特定の行について「真」、「偽」、または「不明」となる条件を指定します。

検索条件の結果は、指定した各述部の結果に、指定した*論理演算子* (AND、OR、NOT) を適用することによって求められます。論理演算子の指定がない場合、検索条件の結果は指定された述部の結果になります。

AND と OR は、表 19 で定義されています。表中の P と Q は任意の述部です。

表 19. AND と OR の真理値表

P	Q	P AND Q	P OR Q
真	真	真	真
真	偽	偽	真
真	不明	不明	真
偽	真	偽	真
偽	偽	偽	偽
偽	不明	偽	不明
不明	真	不明	真
不明	偽	偽	不明
不明	不明	不明	不明

NOT(true) は偽、NOT(false) は真、NOT(unknown) は不明です。

括弧の中の検索条件が最初に評価されます。評価の順序を括弧によって指定していない場合、 NOT が AND の前に適用され、AND が OR の前に適用されます。同じ優先順位の演算子が評価される順序は、検索条件の最適化を図るために定義されていません。

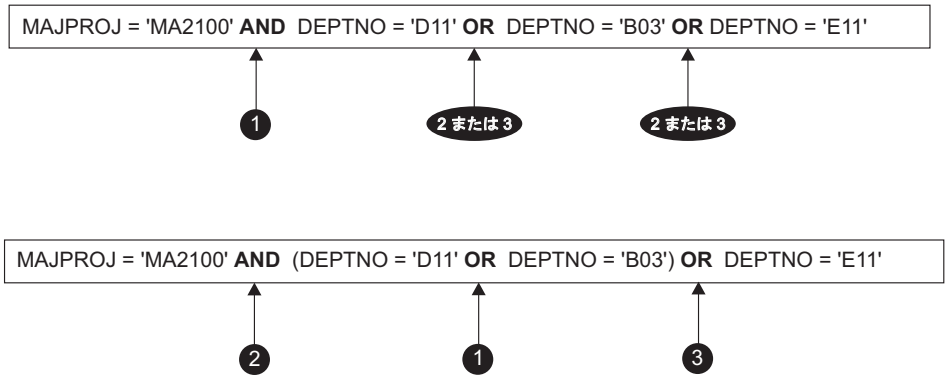


図 13. 検索条件の評価順序

SELECTIVITY *value*

SELECTIVITY 節は、述部に指定する選択の予想パーセントを、DB2 に知らせるときに使用します。 SELECTIVITY を指定できるのは、述部がユーザー定義述部である場合だけです。

ユーザー定義述部とは、述部が指定されているコンテキストの中で、ユーザー定義関数呼び出しで構成される述部のことです。これは、CREATE FUNCTION の PREDICATES 節で指定した述部と一致します。例えば、PREDICATES WHEN=1... で関数 foo が定義される場合、 SELECTIVITY を次のように使うことができます。

```
SELECT *
  FROM STORES
  WHERE foo(parm,parm) = 1 SELECTIVITY 0.004
```

この SELECTIVITY の値は、 0 から 1 の範囲の数値リテラル値でなければなりません (SQLSTATE 42615)。 SELECTIVITY を指定しない場合、デフォルト値は 0.01 になります (つまり、ユーザー定義述部は、表内にあるすべての行の 1% を除いて、すべての行をフィルターで除外することになります)。 SYSSTAT.ROUTINES ビュー内の SELECTIVITY 列を更新すれば、どの関数の SELECTIVITY デフォルトでも変更することができます。ユーザー定義述部以外の述部に SELECTIVITY 節を指定すると、エラーが戻されます (SQLSTATE 428E5)。

ユーザー定義関数 (UDF) はユーザー定義述部として使うことができるので、以下の場合、索引を利用するときにも使える可能性があります。

- CREATE FUNCTION ステートメントに述部の指定がある場合
- WHERE 節で UDF が呼び出されていて、述部を指定したときの指定方法で (文法的に) 比較される場合
- 「否定」(NOT 演算子) がない場合

例

次の照会では、WHERE 節に within UDF が指定されていて、 3 つの条件がすべて満たされているので、ユーザー定義述部であると見なされます。

```
SELECT *
  FROM customers
  WHERE within(location, :sanJose) = 1 SELECTIVITY  0.2
```

ただし、次の照会に within を指定しても、「否定」が入っているため、索引を利用できません。これは、ユーザー定義述部とは見なされません。

```
SELECT *
  FROM customers
  WHERE NOT(within(location, :sanJose) = 1) SELECTIVITY  0.3
```

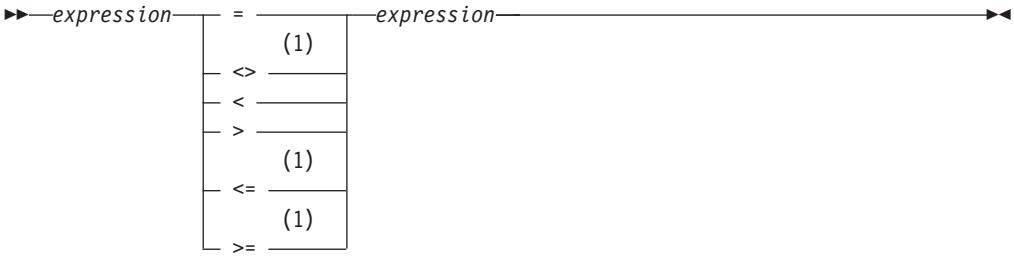
次の例では、相互が特定の距離内にいる顧客と店を識別します。特定の店から別の店の距離は、顧客が居住している都市の半径の範囲に基づいて計算されます。

```
SELECT *
  FROM customers, stores
  WHERE distance(customers.loc, stores.loc) <
    CityRadius(stores.loc) SELECTIVITY 0.02
```

上記の照会では、WHERE 節の述部は、ユーザー定義述部であると見なされます。CityRadius による結果は、範囲を生成する関数に対する検索引数として使われます。

しかし、CityRadius によって生成される結果は、範囲を生成する関数として使われるため、上記のユーザー定義述部では、 stores.loc 列で定義した索引の拡張機能を十分に利用することができません。したがって、UDF は customers.loc 列で定義した索引のみを利用します。

基本述部



注:

1 基本述部および比較述部では、^=、^<、^>、!=、!<、および !> の形式の比較演算子もサポートされています。コード・ページ 437、819、および 850 では、¬=、¬<、および ¬> の形式もサポートされています。このような製品固有の比較演算子の形式は、このような演算子を使用する既存の SQL ステートメントをサポートすることのみを目的としており、新たに SQL ステートメントを書く場合には使用しないようお勧めします。

基本述部 は 2 つの値を比較します。

一方のオペランドの値が NULL 値の場合、述部の結果は不明です。それ以外の場合の結果は、真または偽のいずれかになります。

値 *x* および *y* について、次のような関係が成り立ちます。

述部　次の場合にのみ「真」になります。

x **=** *y*　*x* は *y* に等しい

x **<>** *y*　*x* は *y* に等しくない

x **<** *y*　*x* は *y* より小さい

x **>** *y*　*x* は *y* より大きい

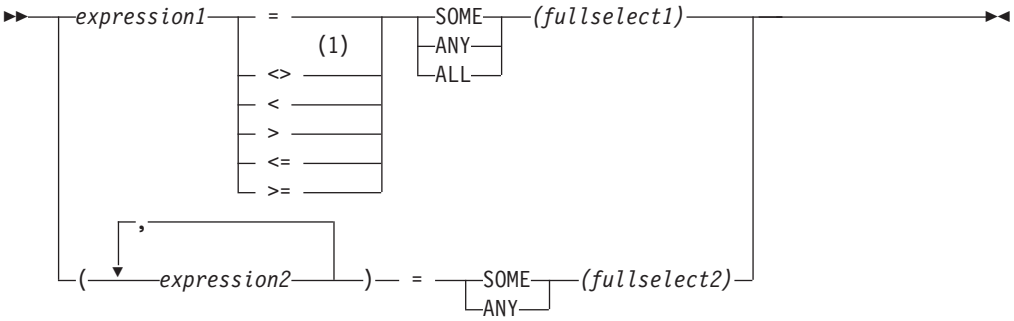
x **>=** *y*　*x* は *y* より大きいか等しい

x **<=** *y*　*x* は *y* より小さいか等しい

例:

```
EMPNO='528671'
SALARY < 20000
PRSTAFF <> :VAR1
SALARY > (SELECT AVG(SALARY) FROM EMPLOYEE)
```

比較述部



注:

1 基本述部および比較述部では、^=、^<、^>、!=、!<、および !> の形式の比較演算子もサポートされています。コード・ページ 437、819、および 850 では、¬=、¬<、および ¬> の形式もサポートされています。このような製品固有の比較演算子の形式は、このような演算子を使用する既存の SQL ステートメントをサポートすることのみを目的としており、新たに SQL ステートメントを書く場合には使用しないようお勧めします。

比較述部 は、 1 つの値もしくは複数の値と、値の集合との間で比較を行います。

全選択は、述部演算子の左側に指定されている式の数と同じ数の列を識別しなければなりません (SQLSTATE 428C4)。全選択は、任意の行数を戻すことができます。

ALL を指定した場合、

- 全選択が値をまったく戻さない場合、または指定したリレーションシップが、全選択によって戻される値のすべてに対して「真」である場合、述部の結果は「真」となります。
- 指定したリレーションシップが、全選択によって戻される値の少なくとも 1 つに対して「偽」である場合、述部の結果は「偽」となります。
- 指定したリレーションシップが、全選択によって戻されるどの値に対しても「偽」でなく、少なくとも 1 つの比較が NULL 値のために「不明」である場合、述部の結果は「不明」となります。

SOME または ANY を指定した場合、

- 指定したリレーションシップが、全選択によって戻される少なくとも 1 つの行の各値に対して「真」である場合、述部の結果は「真」になります。
- 全選択が行を戻さない場合、または指定したリレーションシップが、全選択によって戻されるすべての行の少なくとも 1 つの値に対して「偽」である場合、述部の結果は「偽」になります。
- 指定したリレーションシップがどの行に対しても「真」でなく、少なくとも 1 つの比較が NULL 値のために「不明」である場合、述部の結果は「不明」になります。

例: 以降の例を参照する場合、次の表を使用してください。

TBL*AB*:

COL*A*	COL*B*
1	12
2	12
3	13
4	14
-	-

TBL*XY*:

COL*X*	COL*Y*
2	22
3	23

図 14. 比較述部の表の例

例 1

```
SELECT COLA FROM TBLAB
   WHERE COLA = ANY(SELECT COLX FROM TBLXY)
```

結果は 2、3 です。副選択は (2,3) を戻します。行 2 と 3 の COLA は、それらの値の少なくとも 1 つに等しくなっています。

例 2

```
SELECT COLA FROM TBLAB
   WHERE COLA > ANY(SELECT COLX FROM TBLXY)
```

結果は 3、4 です。副選択は (2,3) を戻します。行 3 と 4 の COLA は、それらの値の少なくとも 1 つより大きくなっています。

例 3

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY)
```

結果は 4 です。副選択は (2,3) を戻します。それらの値の両方より大きいものは、行 4 の COLA しかありません。

例 4

```
SELECT COLA FROM TBLAB
   WHERE COLA > ALL(SELECT COLX FROM TBLXY
                    WHERE COLX<0)
```

結果は 1、2、3、4、NULL 値です。副選択は値を戻しません。したがって、述部は TBLAB のすべての行に対して真です。

例 5

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB+10) = SOME (SELECT COLX, COLY FROM TBLXY)
```

副選択は TBLXY からすべての項目を戻します。述部は副選択に対して真であるため、結果は次のようになります。

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

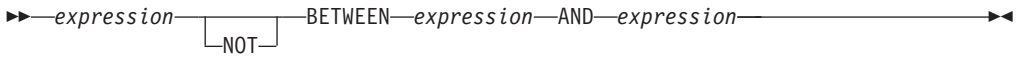
例 6

```
SELECT * FROM TBLAB
   WHERE (COLA,COLB) = ANY (SELECT COLX,COLY-10 FROM TBLXY)
```

副選択は TBLXY から COLX および COLY-10 を戻します。述部は副選択に対して真であるため、結果は次のようになります。

```
COLA        COLB
----------- -----------
          2          12
          3          13
```

BETWEEN 述部



BETWEEN 述部は、ある値を値の範囲と比較します。

次の BETWEEN 述部は、

```
value1 BETWEEN value2 AND value3
```

次の検索条件と同等です。

```
value1 >= value2 AND value1 <= value3
```

次の BETWEEN 述部は、

```
value1 NOT BETWEEN value2 AND value3
```

次の検索条件と同等です。

```
NOT(value1 BETWEEN value2 AND value3); that is,
value1 < value2 OR value1 > value3.
```

第 1 オペランド (expression) 内で、可変の関数または外部処理を伴う関数を使用することはできません (SQLSTATE 426804)。

日付/時刻値と日付/時刻値のストリング表記が混在している場合、すべての値は日付/時刻オペランドのデータ・タイプに変換されます。

例

例 1

```
EMPLOYEE.SALARY BETWEEN 20000 AND 40000
```

結果は $20,000.00 と $40,000.00 の間のすべての給与となります。

例 2

```
SALARY NOT BETWEEN 20000 + :HV1 AND 40000
```

:HV1 が 5000 であるとすると、結果は $25,000.00 より低いか $40,000.00 より高いすべての給与となります。

EXISTS 述部

EXISTS 述部は、特定の行の存在を調べるためのものです。

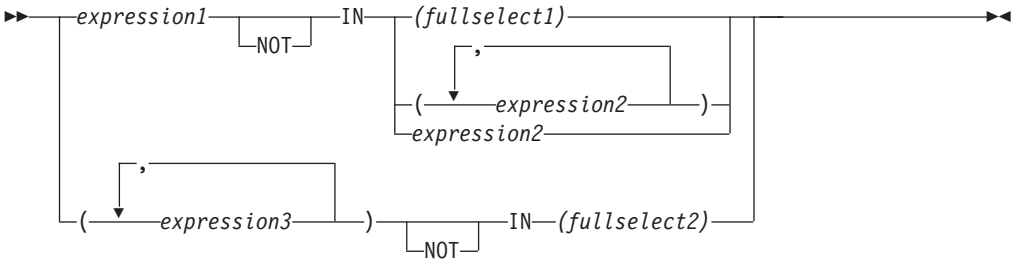
fullselect (全選択) には、必要な数の列を指定できます。

- fullselect に指定された行数がゼロでない場合にのみ、結果が「真」になります。
- 指定された行数がゼロの場合にのみ、結果が「偽」になります。
- 結果が「不明」になることはありません。

例

```
EXISTS (SELECT * FROM TEMPL WHERE SALARY < 10000)
```

IN 述部



IN 述部は、1 つの値または複数の値を値の集合と比較します。

全選択は、 IN キーワードの左側に指定されている式の数と同じ数の列を識別しなければなりません (SQLSTATE 428C4)。全選択は、任意の行数を戻すことができます。

- 次の形式の IN 述部があるとします。

  ```
  expression IN expression
  ```

  上記は、以下の形式の基本述部と同等です。

  ```
  expression = expression
  ```

- 次の形式の IN 述部があるとします。

  ```
  expression IN (fullselect)
  ```

  上記は、以下の形式の比較述部と同等です。

  ```
  expression = ANY (fullselect)
  ```

- 次の形式の IN 述部があるとします。

  ```
  expression NOT IN (fullselect)
  ```

  上記は、以下の形式の比較述部と同等です。

  ```
  expression <> ALL (fullselect)
  ```

- 次の形式の IN 述部があるとします。

  ```
  expression IN (expressiona, expressionb, ..., expressionk)
  ```

  これは、以下と同じ意味になります。

  ```
  expression = ANY (fullselect)
  ```

  この fullselect は、values 節形式では次のようになります。

  ```
  VALUES (expressiona), (expressionb), ..., (expressionk)
  ```

- 次の形式の IN 述部があるとします。

  ```
  (expressiona, expressionb,..., expressionk) IN (fullselect)
  ```

  上記は、以下の形式の比較述部と同等です。

  ```
  (expressiona, expressionb,..., expressionk) = ANY (fullselect)
  ```

IN 述部の *expression1* および *expression2* の値、または *fullselect1* の列には、互換性が必要です。 IN 述部の *expression3* の各値、およびそれに対応する *fullselect2*

の列にも互換性が必要です。結果データ・タイプの規則を使って、比較で使用される結果の属性を判別することができます。

IN 述部の式の値 (全選択の対応する列を含めて) のコード・ページが異なっていても構いません。変換が必要な場合にコード・ページを判別するには、まず IN リストに対してストリング変換の規則を適用し、次に第 2 オペランドとして IN リストの派生コード・ページを使って同じ規則を述部に適用します。

例

例 1: DEPTNO 列で評価の対象となる行の値に D01、B01、または C01 が入っている場合、以下は真であると評価されます。

```
DEPTNO IN ('D01', 'B01', 'C01')
```

例 2: 左側の EMPNO (従業員番号) が部門 E11 の従業員の EMPNO と一致する場合のみ、以下は真であると評価されます。

```
EMPNO IN (SELECT EMPNO FROM EMPLOYEE WHERE WORKDEPT = 'E11')
```

例 3: 以下の情報に基づき、COL_1 列の行の特定の値が、リスト内のいずれかの値と一致する場合、この例は真であると評価されます。

表 20. IN 述部の例

式	タイプ	コード・ページ
COL_1	列	850
HV_2	ホスト変数	437
HV_3	ホスト変数	437
CON_1	定数	850

ここで、次のような述部を評価します。

```
COL_1 IN (:HV_2, :HV_3, CON_4)
```

この場合、ストリング変換の規則に基づいて、2 個のホスト変数がコード・ページ 850 に変換されます。

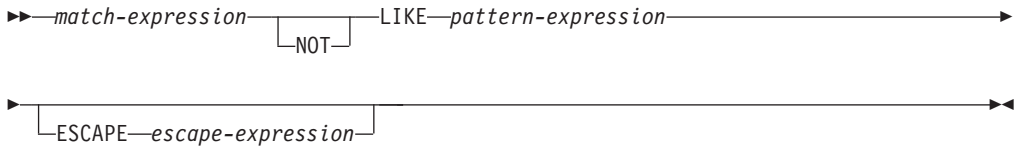
例 4: EMENDATE に指定された年 (プロジェクトの従業員の活動が終了した日付) が、リストに指定された値のいずれか (現在の年または過去 2 年) と一致する場合、以下は真と評価されます。

```
YEAR(EMENDATE) IN (YEAR(CURRENT DATE),
                   YEAR(CURRENT DATE - 1 YEAR),
                   YEAR(CURRENT DATE - 2 YEARS))
```

例 5: 左側の ID と DEPT の両方が、 ORG 表の任意の行の MANAGER と DEPTNUMB にそれぞれ一致する場合、以下は真と評価されます。

```
(ID, DEPT) IN (SELECT MANAGER, DEPTNUMB FROM ORG)
```

LIKE 述部



LIKE 述部は、ある一定のパターンをもったストリングを探索するものです。パターンは、特殊な意味のある下線とパーセント記号を使ったストリングによって指定されます。パターンでは後続ブランクもパターンの一部です。

引数のうちのいずれかの値が NULL 値である場合、LIKE 述部の結果は不明になります。

match-expression (一致式)、 *pattern-expression* (パターン式)、および *escape-expression* (エスケープ式) の値は、互換性のある文字ストリング式です。サポートされる文字ストリング式のタイプは、引数ごとに少々異なります。式の有効なタイプは、各引数の説明で示します。式の有効なタイプは、各引数の説明の中に示します。

どの式も特殊タイプを生成するものであってはなりません。ただし、特殊タイプをそのソース・タイプへキャストする関数は可能です。

match-expression
特定の文字パターンに適合するかどうか調べる対象のストリングを指定する式。

この式は、以下によって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数 (ロケーター変数またはファイル参照変数を含む)
- スカラー関数
- ラージ・オブジェクトのロケーター
- 列名
- 上記のいずれかを連結する式

pattern-expression
一致すべき基準となるストリングを指定する式。

この式は、以下によって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式
- SQL プロシージャー・パラメーター

次のような制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。
- *pattern-expression* の実際の長さは、32672 バイトを超えてはなりません。

以下に、無効なストリング式またはストリングの例を示します。

- SQL ユーザー定義関数パラメーター
- トリガー遷移変数
- 動的コンパウンド・ステートメント内のローカル変数

LIKE 述部について**簡単に説明**すると、これは *match-expression* の値のための適合基準を指定するために使用されるパターンです。これには以下の規則があります。

- 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。
- その他の文字は、その文字自身を表します。

pattern-expression に下線またはパーセント文字を使用する必要がある場合は、パターンの中で下線文字またはパーセント文字の前に置く文字を *escape-expression* で指定します。

LIKE 述部について**厳密に説明**すると、以下のようになります。この説明では、*escape-expression* の使用については述べません。それについては後述します。

- *m* が *match-expression* の値を、 *p* が *pattern-expression* の値を表すとします。ストリング *p* は、一連の最小数のサブストリング指定子として解釈され、*p* の各文字は正確に 1 つのサブストリング指定子の一部となります。サブストリング指定子とは、下線、パーセント記号、または下線およびパーセント記号以外の任意の空でない一連の文字です。

  m または *p* が NULL 値の場合は、述部の結果が不明になります。それ以外の場合の結果は、真か偽のどちらかになります。 *m* と *p* の両方が空ストリングの場合、または以下のようにして *m* をサブストリングにパーティション化したものが存在する場合、結果は真になります。

  – *m* のサブストリングがゼロ個以上の連続する文字の並びで、 *m* の各文字が正確に 1 つのサブストリングの一部である。
  – *n* 番目のサブストリング指定子が下線の場合、 *m* の *n* 番目のサブストリング指定子は任意の 1 文字である。
  – *n* 番目のサブストリング指定子がパーセント記号の場合、 *m* の *n* 番目のサブストリング指定子は 0 個以上の文字の並びである。
  – *n* 番目のサブストリング指定子が下線でもパーセント記号でもない場合、*m* の *n* 番目のサブストリングは、対応するサブストリング指定子と等しく、同じ長さである。
  – *m* のサブストリングの数は、サブストリング指定子の数と同じである。

  したがって、*p* が空ストリングで、 *m* が空ストリングでない場合、結果は偽になります。同様に、*m* が空ストリングで、 *p* が空ストリングでない (パーセント記号だけから成るストリングを除く) 場合、結果は偽になります。

述部 *m* NOT LIKE *p* は、検索条件 NOT (*m* LIKE *p*) と同等です。

escape-expression が指定されている場合、直後にエスケープ文字、下線文字、またはパーセント記号文字が続くのでない限り、 *pattern-expression* の中に、*escape-expression* で指定されるエスケープ文字が入っていてはなりません (SQLSTATE 22025)。

match-expression が MBCS データベースの文字ストリングの場合、それには混合データを収容することができます。この場合は、パターン内で SBCS 文字と非 SBCS 文字の両方を使用することができます。非 Unicode データベースの場合、パターンの中の特殊文字は、以下のようにして解釈されます。

- SBCS の半角下線文字は、1 つの SBCS 文字を表します。
- 非 SBCS の全角下線文字は、1 つの非 SBCS 文字を表します。
- SBCS の半角または非 SBCS の全角パーセント記号文字は、0 以上の SBCS または非 SBCS 文字を表します。

Unicode データベースでは、「単一バイト」文字と「非単一バイト」文字の間に実質的な区別はありません。 UTF-8 形式は Unicode 文字の「混合バイト」エンコード方式ですが、 UTF-8 では、SBCS 文字と非 SBCS 文字の間に実質的な区別がありません。 UTF-8 フォーマットでは、文字のバイト数に関係なく、すべての文字が Unicode 文字になります。

Unicode GRAPHIC 列では、半角下線文字 (U&'¥005F') や半角パーセント記号文字 (U&'¥0025') を含め、補足文字以外のすべての文字が 2 バイト幅になります。 Unicode データベースの場合、パターンの中の特殊文字は次のように解釈されます。

- 文字ストリングでは、半角下線文字 (X'5F') または全角下線文字 (X'EFBCBF') が 1 つの Unicode 文字を表し、半角パーセント記号文字 (X'25') または全角パーセント記号文字 (X'EFBC85') が 0 以上の Unicode 文字を表します。
- GRAPHIC ストリングでは、半角下線文字 (U&'¥005F') または全角下線文字 (U&'¥FF3F') が 1 つの Unicode 文字を表し、半角パーセント記号文字 (U&'¥0025') または全角パーセント記号文字 (U&'¥FF05') が 0 以上の Unicode 文字を表します。
- ロケールに依存する UCA ベースの照合が有効である場合に、特殊文字として認識されるためには、下線文字および % 記号文字の後に、スペースなしで表記する記号 (発音区別符号) を続けてはなりません。例えば、パターン U&'%¥0300' (% 記号に、スペースなしで表記する抑音符号が続いたもの) は、抑音符号の付いた文字が続くゼロ以上の Unicode 文字の検索ではなく、 の検索として解釈されます。

1 つの Unicode 補足文字は、Unicode グラフィック列で、2 ポイントのグラフィック・コードとして保管されます。Unicode グラフィック列内の Unicode の補足文字を一致させるためには、データベースが、ロケールに依存する UCA ベースの照合を使用する場合は下線 1 つを、その他の場合は下線 2 つを使用します。 Unicode 文字列内の Unicode の補足文字に一致させるためには、すべての照合に下線 1 つを使用します。ロケールに依存する UCA ベースの照合をデータベースが使用する場合、スペースなしで結合される文字が 1 つ以上後続する基本文字と一致させるためには、下線 1 つを使用します。その他の場合は、スペースなしで表記する文字に基本文字を加えた数の下線を使用します。

escape-expression

これはオプションの引数であり、 *pattern-expression* 内での下線 (_) 文字とパーセント (%) 文字の持つ特別な意味を変更するために使用する文字を指定する式です。これにより、実際にパーセントや下線文字の入った値との一致を調べるために LIKE 述部を使うことができます。

この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式

以下の制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。
- 文字の列の場合、式の結果は 1 文字、つまりちょうど 1 バイト入ったバイナリー・ストリングになります (SQLSTATE 22019)。
- GRAPHIC 列の場合、式の結果は 1 つの文字になります (SQLSTATE 22019)。
- 式の結果をスペースなしの結合文字シーケンス (U&'¥0301'、結合揚音アクセント、など) にすることはできません。

パターン・ストリングにエスケープ文字が入っている場合、下線、パーセント記号、またはエスケープ文字は、それ自体のリテラル・オカレンスを表すことができます。これは、その文字の前に奇数個の連続したエスケープ文字がある場合です。そうでない場合は当てはまりません。

パターンの中で、連続するエスケープ文字の並びは以下のように扱われます。

- S がそのような並びであり、エスケープ文字のより長い連続の一部分ではないものとします。また、S が合計 n 個の文字を収めているものとします。このとき、S に適用される規則は、n の値により以下のように異なります。
  - n が奇数の場合、S の後には下線またはパーセント記号がなければなりません (SQLSTATE 22025)。 S とその後に続く文字は、エスケープ文字の (n-1)/2 個のリテラル・オカレンスの後に下線記号またはパーセント記号のリテラル・オカレンスが続くことを表します。
  - n が偶数の場合、S はエスケープ文字の n/2 個のリテラル・オカレンスを表します。 n が奇数の場合とは異なり、S でパターンが終了する場合もあります。 S でパターンが終了しない場合、S の後にはどんな文字が続いても構いません (ただし、 S はエスケープ文字のより長い連続の一部分ではないという前提に違反するため、当然エスケープ文字は除外されます)。 S の後に下線記号またはパーセント記号が続く場合、その文字には特別な意味があります。

以下は、エスケープ文字 (この場合は、円記号 (¥)) が連続して出現する場合にどうなるかを示しています。

パターン・ストリング
実際のパターン

¥% パーセント記号

¥¥% 1 つの円記号の後にゼロ個以上の任意の文字が続く

¥¥¥% 1 つの円記号の後に 1 つのパーセント記号が続く

比較で使用されるコード・ページは、 *match-expression* の値のコード・ページに基づいて決定されます。

- *match-expression* の値が変換されることはありません。
- *pattern-expression* のコード・ページが、 *match-expression* のコード・ページと異なる場合、 *pattern-expression* の値が *match-expression* のコード・ページに変換されます。ただし、どちらかのオペランドが FOR BIT DATA で定義されている場合は除きます (その場合は変換されません)。
- *escape-expression* のコード・ページが、 *match-expression* のコード・ページと異なる場合、 *escape-expression* の値が *match-expression* のコード・ページに変換されます。ただし、どちらかのオペランドが FOR BIT DATA で定義されている場合は除きます (その場合は変換されません)。

注

- 末尾ブランクの数は、 *match-expression* と *pattern-expression* のどちらでも重要です。ストリング同士が同じ長さでない場合に、短いほうのストリングにブランクが埋め込まれることはありません。例えば、`'PADDED ' LIKE 'PADDED'` という式は、一致していないことになります。
- LIKE 述部でパラメーター・マーカーをパターンとして指定した場合に、そのパラメーター・マーカーを固定長文字のホスト変数に置き換える場合は、正しい長さのホスト変数値を指定しなければなりません。正しい長さを指定しないと、所定の結果が SELECT から戻されません。

  例えば、ホスト変数を CHAR(10) と定義した場合に、そのホスト変数に値 WYSE% を割り当てると、割り当ての際にそのホスト変数はブランクで埋め込まれます。以下のパターンが使用されます。

  ```
  'WYSE%     '
  ```

  データベース・マネージャーは、 WYSE で始まって 5 つのブランク・スペースで終わるすべての値を検索します。 WYSE で始まる値だけを検索するつもりの場合は、ホスト変数に値 WSYE%%%%% を割り当てます。
- パターン・マッチングには、データベースの照合が使用されます。ただし、どちらかのオペランドが FOR BIT DATA で定義されている場合は除きます。その場合は、バイナリー比較によってパターン・マッチングが行われます。

例

- PROJECT 表の PROJNAME 列で 'SYSTEMS' というストリングを探索します。

  ```
  SELECT PROJNAME FROM PROJECT
    WHERE PROJECT.PROJNAME LIKE '%SYSTEMS%'
  ```

- EMPLOYEE 表の FIRSTNME 列で、先頭の文字が 'J' で、長さがちょうど 2 文字のストリングを探索します。

```
SELECT FIRSTNME FROM EMPLOYEE
  WHERE EMPLOYEE.FIRSTNME LIKE 'J_'
```

- EMPLOYEE 表の FIRSTNME 列で、先頭の文字が 'J' で、任意の長さのストリングを探索します。

```
SELECT FIRSTNME FROM EMPLOYEE
  WHERE EMPLOYEE.FIRSTNME LIKE 'J%'
```

- CORP_SERVERS 表で、LA_SERVERS 列のストリングのうち、 CURRENT SERVER 特殊レジスターの値と一致するものを探索します。

```
SELECT LA_SERVERS FROM CORP_SERVERS
  WHERE CORP_SERVERS.LA_SERVERS LIKE CURRENT SERVER
```

- 表 T の列 A 中の文字シーケンス '_¥' で始まるすべてのストリングを検索します。

```
SELECT A FROM T
  WHERE T.A LIKE '¥_¥¥%' ESCAPE '¥'
```

- 一致およびパターンのデータ・タイプ (どちらも BLOB) と互換である 1 バイトのエスケープ文字を得るには、次のように BLOB スカラー関数を使用します。

```
SELECT COLBLOB FROM TABLET
  WHERE COLBLOB LIKE :pattern_var ESCAPE BLOB(X'0E')
```

- 大/小文字を区別しない照合 UCA500R1_LEN_S1 で定義された Unicode データベース内で、'Bill' で始まるすべての名前を検索します。

```
SELECT NAME FROM CUSTDATA WHERE NAME LIKE 'Bill%'
```

これにより、'Bill Smith'、'billy simon'、および 'BILL JONES' という名前が戻されます。

NULL 述部

NULL 述部は、NULL 値かどうかを検査するものです。

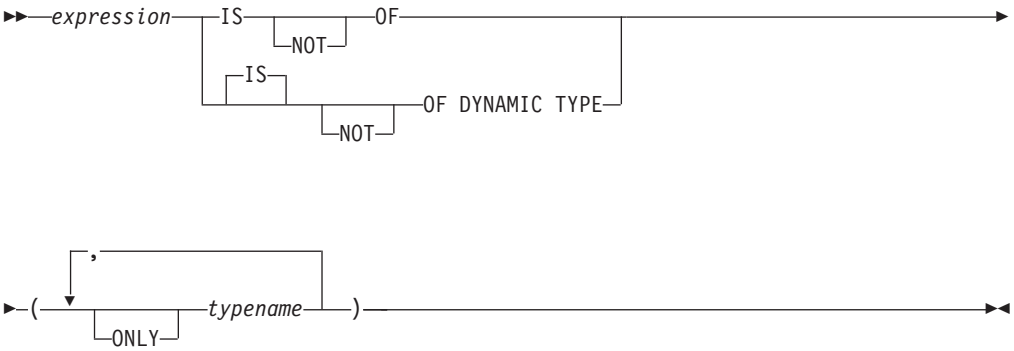
NULL 述部の結果が不明になることはありません。式の値が NULL 値の場合、結果は真になります。値が NULL 値でない場合、結果は偽になります。 NOT が指定されている場合、結果は逆になります。

例

```
PHONENO IS NULL

SALARY IS NOT NULL
```

TYPE 述部



TYPE 述部 は、式のタイプと 1 つまたは複数のユーザー定義構造化タイプとを比較します。

参照タイプの間接参照に関与した式の動的タイプは、ターゲットの型付き表またはビューにある参照される行の実際のタイプです。これは、その参照に関与した式のターゲット・タイプ (式の静的タイプと呼ばれる) とは異なる場合があります。

expression の値が NULL の場合、述部の結果は不明です。 *expression* の動的タイプが *typename* で指定された構造化タイプの 1 つのサブタイプの場合、述部の結果は「真」になり、そうでない場合は「偽」になります。 ONLY のあとに *typename* がある場合、その型の適切なサブタイプは考慮されません。

typename が修飾されていない場合、SQL パスを使用して解決されます。 *typename* は、 *expression* の静的タイプのタイプ階層にあるユーザー定義タイプを識別しなければなりません (SQLSTATE 428DU)。

DEREF 関数は、参照タイプの値に関与した式が TYPE 述部にある場合はいつでも、使用されなければなりません。 *expression* がこの形式の場合の静的タイプは、参照のターゲット・タイプです。

構文上の IS OF と OF DYNAMIC TYPE は、TYPE 述部では同じ働きをします。同様に、IS NOT OF と NOT OF DYNAMIC TYPE も TYPE 述部では同じ働きをします。

例

ある表階層には、タイプ EMP のルート表 EMPLOYEE と、タイプ MGR の副表 MANAGER があります。別の表 ACTIVITIES は、 REF(EMP) SCOPE EMPLOYEE として定義されている WHO_RESPONSIBLE という列を備えています。WHO_RESPONSIBLE と対応する行が管理者の場合に、結果が「真」となるタイプ述部を以下に示します。

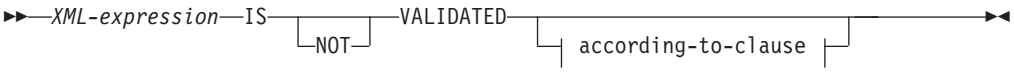
```
DEREF (WHO_RESPONSIBLE) IS OF (MGR)
```

表にタイプ EMP の列 EMPLOYEE が入っている場合、EMPLOYEE には、タイプ EMP の値だけではなく、MGR のようなサブタイプの値を使用することができます。次のような述部は、

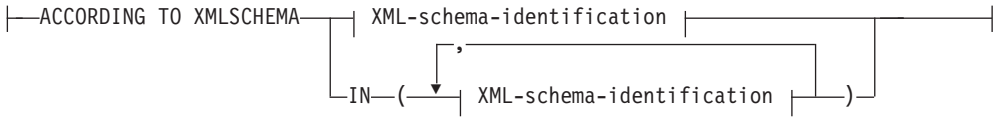
```
EMPL IS OF (MGR)
```

EMPL が NULL ではなく、実際に管理職である場合に、「真」を戻します。

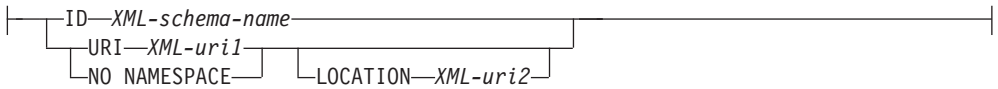
VALIDATED 述部



according-to-clause:



XML-schema-identification:



VALIDATED 述部は、*XML-expression* によって指定された値が、XMLVALIDATE 関数を使用して妥当性検査されたかどうかを検査します。 指定された値が NULL の場合、妥当性検査の制約の結果は不明です。そうでない場合、妥当性検査の制約の結果は、true または false のいずれかになります。指定する値はタイプ XML でなければなりません。

ACCORDING TO XMLSCHEMA 節が指定されていない場合、妥当性検査に使用される XML スキーマは妥当性検査の制約の結果に影響しません。

説明

XML-expression
検査される XML 値を指定します。ここで、*XML-expression* は、XML 文書、XML コンテンツ、XML ノードのシーケンス、XML *column-name*、または XML *correlation-name* で構成できます。

XML *column-name* が指定されている場合、述部は、指定された列名に関連する XML 文書が妥当性検査されたかどうかを評価します。

トリガーの一部としてのタイプ XML の相関名の指定についての情報は、『CREATE TRIGGER』を参照してください。

IS VALIDATED または IS NOT VALIDATED
XML-expression のオペランドに必要な妥当性検査の状態を指定します。

true として評価するように IS VALIDATED を指定する制約では、オペランドは妥当性検査されている必要があります。 オプションの ACCORDING TO XMLSCHEMA 節に 1 つまたはいくつかの XML スキーマが含まれている場合、オペランドは識別された XML スキーマの 1 つを使用して妥当性検査されている必要があります。

false として評価するように IS NOT VALIDATED を指定する制約の場合、オペランドは妥当性検査された状態でなければなりません。オプションの ACCORDING TO XMLSCHEMA 節に 1 つまたはいくつかの XML スキーマが

含まれている場合、オペランドは識別された XML スキーマの 1 つを使用して妥当性検査されている必要があります。

according-to-clause
オペランドが妥当性検査されていなければならないか、またはされていてはならないかを 1 つまたはいくつかの XML スキーマに対して指定します。 XML スキーマ・リポジトリー に事前に登録された XML スキーマのみが指定されます。

ACCORDING TO XMLSCHEMA

ID *XML-schema-name*
XML スキーマの SQL ID を指定します。この名前 (暗黙的または明示的 SQL スキーマ修飾子を含む) は、現行のサーバーで XML スキーマ・リポジトリー内の既存の XML スキーマを固有に識別しなければなりません。暗黙的または明示的に指定した SQL スキーマにこの名前の XML スキーマが存在しない場合は、エラー (SQLSTATE 42704) が戻されます。

URI *XML-uri1*
XML スキーマのターゲット・ネーム・スペース URI を指定します。 *XML-uri1* の値は、URI を空でない文字ストリング定数として指定します。 URI は、登録済み XML スキーマのターゲット・ネーム・スペースでなければならず (SQLSTATE 4274A)、LOCATION 節を指定しない場合は、登録済み XML スキーマを固有に識別する必要があります (SQLSTATE 4274B)。

NO NAMESPACE
XML スキーマはターゲット・ネーム・スペースを持たないことを指定します。ターゲット・ネーム・スペース URI は、明示的なターゲット・ネーム・スペース URI として指定できない空の文字ストリングと同等です。

LOCATION *XML-uri2*
XML スキーマの XML スキーマ・ロケーション URI を指定します。 *XML-uri2* の値は、URI を空でない文字ストリング定数として指定します。 XML スキーマ・ロケーション URI は、ターゲット・ネーム・スペース URI と結合されて登録済み XML スキーマを識別する必要があり (SQLSTATE 4274A)、登録済みのそのような XML スキーマは 1 つだけでなければなりません (SQLSTATE 4274B)。

例

例 1: 表 T1 に列 XMLCOL が定義されているとします。何らかの XML スキーマで妥当性検査が行われた XML 値のみ取り出します。

```
SELECT XMLCOL FROM T1
  WHERE XMLCOL IS VALIDATED
```

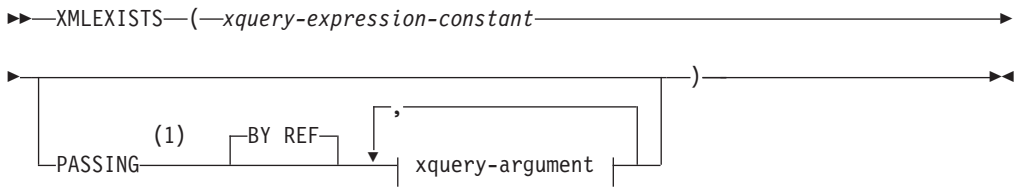
例 2: 表 T1 に列 XMLCOL が定義されているとします。妥当性検査が行われていない値は挿入も更新もできないという規則を施行します。

```
ALTER TABLE T1 ADD CONSTRAINT CK_VALIDATED
  CHECK (XMLCOL IS VALIDATED)
```

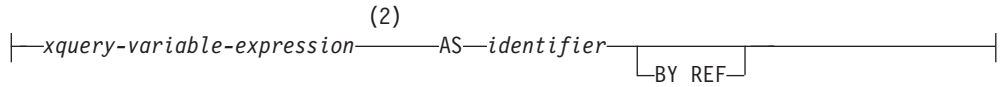
例 3: XML スキーマ URI http://www.posample.org で妥当性検査された XML 列 XMLCOL のある 表 T1 から、これらの行のみを選択するとします。

```
SELECT XMLCOL FROM T1
  WHERE XMLCOL IS VALIDATED
   ACCORDING TO XMLSCHEMA URI
   'http://www.posample.org'
```

XMLEXISTS 述部



xquery-argument:



注:

1 データ・タイプを DECFLOAT にすることはできません。

2 式のデータ・タイプを DECFLOAT にすることはできません。

XMLEXISTS 述部は、XQuery 式が 1 つ以上の項目のシーケンスを戻すかどうかをテストします。

xquery-expression-constant
XQuery 式として解釈される SQL 文字ストリング定数を指定します。定数ストリングは、データベース・コード・ページまたはセクション・コード・ページに変換されることなく、UTF-8 に直接変換されます。 XQuery 式は、オプション・セットの入力 XML 値を使用して実行し、XMLEXISTS 述部の結果を決定するためにテストされる出力シーケンスを戻します。 *xquery-expression-constant* の値は、空ストリングまたはブランク文字のストリングにすることはできません (SQLSTATE 10505)。

PASSING
入力値、およびそれらの値を *xquery-expression-constant* で指定された XQuery 式に渡す方法を指定します。デフォルトでは、関数が呼び出された有効範囲内にあるすべての固有の列名が、列の名前を変数名として使用して XQuery 式に暗黙的に渡されます。指定の xquery-argument 内の *identifier* が有効範囲内の列名と一致する場合、明示的な xquery-argument はその暗黙的な列をオーバーライドして XQuery 式に渡されます。

BY REF
デフォルトの受け渡しメカニズムを、データ・タイプ XML の任意の *xquery-variable-expression* の参照によると指定します。 XML 値を参照で渡す場合、XQuery の評価は、入力ノード・ツリーがあればそれを使用します。その場合は指定された入力式から直接、元のノードの ID および文書順序を含めすべてのプロパティーを保持したまま使用します。 2 つの引数が同じ XML 値を渡す場合、その 2 つの入力引数の間に含まれている何らかのノードに関係するノード ID 比較および文書順序比較は、同じ XML ノード・ツリー内のノードを参照する場合があります。

この節は、非 XML 値の受け渡しには影響を与えません。非 XML 値は、XML へのキャスト中に値の新規コピーを作成します。

xquery-argument

xquery-expression-constant により指定された XQuery 式に渡される引数を指定します。引数は、値およびその値が渡される方法を指定します。引数には、評価される SQL 式が組み込まれます。

- 結果の値は、XML 型である場合、*input-xml-value* になります。 NULL の XML 値は、XML の空シーケンスに変換されます。
- 結果の値は、XML 型でない場合、XML データ・タイプにキャスト可能でなければなりません。 NULL 値は、XML の空シーケンスに変換されます。変換される値は、*input-xml-value* になります。

xquery-expression-constant が評価されるとき、XQuery 変数は *input-xml-value* と等しい値、および AS 節により指定された名前で示されます。

xquery-variable-expression

実行中に *xquery-expression-constant* により指定された XQuery 式が使用できる値を持つ SQL 式を指定します。式には、シーケンス参照 (SQLSTATE 428F9) または OLAP 関数 (SQLSTATE 42903) を含めることはできません。式のデータ・タイプを DECFLOAT にすることはできません。

AS *identifier*

xquery-variable-expression により生成された値が、*xquery-expression-constant* に XQuery 変数として渡されることを指定します。変数名は *identifier* になります。 XQuery 言語の変数名に先行する先頭のドル記号 ($) は、*identifier* には含められません。 identifier は有効な XQuery 変数名でなければならず、XML NCName に制限されます。 identifier は、長さが 128 バイトを超えてはなりません。同じ PASSING 節内の 2 つの引数が同じ identifier を使用することはできません (SQLSTATE 42711)。

BY REF

XML 入力値が参照により渡されるように指示します。 XML 値を参照で渡す場合、XQuery の評価は、入力ノード・ツリーがあればそれを使用します。その場合は指定された入力式から直接、元のノードの ID および文書順序を含めすべてのプロパティーを保持したまま使用します。2 つの引数が同じ XML 値を渡す場合、その 2 つの入力引数の間に含まれている何らかのノードに関係するノード ID 比較および文書順序比較は、同じ XML ノード・ツリー内のノードを参照する場合があります。 BY REF が *xquery-variable-expression* に続いて指定されない場合、XML 引数は、PASSING キーワードに続く構文により提供されるデフォルトの受け渡しメカニズムによって渡されます。このオプションは、非 XML 値に指定することはできません。非 XML 値が渡される場合、値は XML に変換されます。このプロセスによりコピーが作成されます。

注

XMLEXISTS 述部は、以下のものにすることはできません。

- JOIN 演算子または MERGE ステートメントと関連した ON 節の一部 (SQLSTATE 42972)

- CREATE INDEX EXTENSION ステートメントの GENERATE KEY USING または RANGE THROUGH 節の一部 (SQLSTATE 428E3)
- CREATE FUNCTION (外部スカラー) ステートメント内の FILTER USING 節の一部、または CREATE INDEX EXTENSION ステートメント内の FILTER USING 節の一部 (SQLSTATE 428E4)
- チェック制約の一部、または列生成式の一部 (SQLSTATE 42621)
- group-by 節の一部 (SQLSTATE 42822)
- 列関数の引数の一部 (SQLSTATE 42607)

副照会に関係する XMLEXISTS 述部は、副照会を制限するステートメントにより制限されることがあります。

XMLEXISTS 述部は、単一のデータベース・パーティションを持つデータベースでのみ使用できます (SQLSTATE 42997)。

例

```
SELECT c.cid FROM customer c
  WHERE XMLEXISTS('$d/*:customerinfo/*:addr[ *:city = "Aurora" ]'
    PASSING info AS "d")
```

第 3 章 関数

関数の概要

関数 とは、関数名の後に 1 対の括弧で囲んだ引数の指定を伴う演算です (引数がない場合もあります)。

組み込み関数 は、データベース・マネージャーに提供されている関数です。この関数は、1 つの結果値を戻し、SYSIBM スキーマの一部として識別されます。組み込み関数には、列関数 (AVG など)、演算子関数 (+ など)、 cast 関数 (DECIMAL など)、およびその他の関数 (SUBSTR など) などがあります。

ユーザー定義関数 は、(CREATE FUNCTION ステートメントを使用して) SYSCAT.ROUTINES のデータベースに登録されます。ユーザー定義関数は SYSIBM スキーマの一部ではありません。このような関数のセットは、1 つは SYSFUN という名前のスキーマで、もう 1 つは SYSPROC というスキーマで、データベース・マネージャーに付属しています。

関数は、集約 (列) 関数、スカラー関数、行関数、および表関数に分類されます。

- *集約関数* の引数は、互いに似通った値の集合です。集約関数は、単一の値 (あるいは NULL) を戻しますが、式を使用できる場所であればどこでも SQL ステートメント内に指定できます。
- *スカラー関数* の引数は、それぞれ型や意味が異なる可能性のある個々のスカラー値です。スカラー関数は、単一の値 (あるいは NULL) を戻しますが、式を使用できる場所であればどこでも SQL ステートメント内に指定できます。
- *行関数* の引数は、構造化タイプです。行関数は、組み込みデータ・タイプの行を戻しますが、構造化タイプの変換関数としてだけ指定できます。
- *表関数* の引数は、それぞれ型や意味が異なる可能性のある個々のスカラー値です。表関数は、SQL ステートメントへ表を戻し、SELECT ステートメントの FROM 節にのみ指定することができます。

関数の完全修飾名は、関数名とスキーマの組み合わせからなっています。スキーマ、関数名、および入力パラメーターの組み合わせが*関数シグニチャー* を構成します。

入力パラメーターのタイプは、特定の組み込みデータ・タイプとして指定される場合と、*any-numeric-type* (任意の数値タイプの意) のような汎用変数を使用して指定される場合があります。特定のデータ・タイプが指定される場合は、指定されたデータ・タイプに対してのみ完全一致が成立します。一方、汎用変数が使用される場合は、その変数に関連付けられているどのデータ・タイプに対しても完全一致が成立します。

関数シグニチャーの 1 つをソースにして別のスキーマでユーザー定義関数を作成することにより、追加の関数も使用できます。アプリケーションで外部関数を作成することも可能です。

© Copyright IBM Corp. 1993, 2010

サポートされている関数および管理 SQL ルーチンとビュー

表 21 は、サポートされている関数の詳細を要約しています。関数の完全修飾名は、関数名とスキーマの組み合わせからなっています。「入力」列には、関数呼び出し時の各引数の所定のデータ・タイプを示します。関数の多くには、さまざまな入力パラメーターが備えられており、異なるデータ・タイプまたは異なる数の引数を使用することができます。スキーマ、関数名、および入力パラメーターの組み合わせが関数シグニチャーを構成します。「戻り値のタイプ」列には、関数から戻される値として使用可能なデータ・タイプが示されます。管理 SQL ルーチンおよびビューの詳細は、入力および戻り情報の参照情報を参照してください。

型別に分類されたサポート対象の組み込み関数のリストは、以下の表を参照してください。

- 集約関数 (299 ページの表 22)
- CAST スカラー関数 (300 ページの表 23)
- パーティション化スカラー関数 (301 ページの表 24)
- DATETIME スカラー関数 (301 ページの表 25)
- 数値スカラー関数 (302 ページの表 26)
- セキュリティー・スカラー関数 (303 ページの表 27)
- XML 関数 (303 ページの表 28)
- ストリング・スカラー関数 (304 ページの表 29)
- その他のスカラー関数 (306 ページの表 30)

サポートされている管理 SQL ルーチンおよびビューを機能別に分類したリストは、「*管理ルーチンおよびビュー*」の『サポートされる管理 SQL ルーチンおよび管理ビュー』を参照してください。このようなルーチンおよびビューは、次のようなグループに分かれます。

- アクティビティー・モニター管理 SQL ルーチン
- ADMIN_CMD ストアード・プロシージャーおよび関連する管理 SQL ルーチン
- 構成管理 SQL ルーチンおよびビュー
- 環境管理ビュー
- ヘルス・スナップショット管理 SQL ルーチン
- MQSeries® 管理 SQL ルーチン
- セキュリティー管理 SQL ルーチンおよびビュー
- スナップショット管理 SQL ルーチンおよびビュー
- SQL プロシージャー管理 SQL ルーチン
- 段階的な再配分管理 SQL ルーチン
- ストレージ管理ツール管理 SQL ルーチン
- その他の管理 SQL ルーチンおよびビュー

表 21. サポートされている関数

関数名	スキーマ	入力	戻り値のタイプ	説明
335 ページの『ABS または ABSVAL』	SYSIBM	組み込み数値データ・タイプを返す式。	引数と同じデータ・タイプおよび長さ	このスカラー関数は、引数の絶対値を戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
335 ページの『ABS または ABSVAL』	SYSFUN	• SMALLINT • INTEGER • BIGINT • DOUBLE	• SMALLINT • INTEGER • BIGINT • DOUBLE	このスカラー関数は、引数の絶対値を戻します。
336 ページの『ACOS』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のアークコサイン (逆余弦) の角度を戻します (ラジアン単位)。
309 ページの『ARRAY_AGG』	SYSIBM	• *any-type*[8] • *any-type*[8], *any-type*	• 配列 • 配列	集約関数は、エレメントのセットを配列に集約します。
337 ページの『ASCII』	SYSFUN	• CHAR • VARCHAR(4000) • CLOB(1M)	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数の左端の文字の ASCII コード値を整数として戻します。
338 ページの『ASIN』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のアークサイン (逆正弦) の値をラジアン単位で戻します。
339 ページの『ATAN』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。
341 ページの『ATANH』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数の双曲線アークタンジェント (逆正接) の値を戻します。引数はラジアン単位の角度です。
340 ページの『ATAN2』	SYSIBM	• DOUBLE, DOUBLE	• DOUBLE	このスカラー関数は、最初と 2 番目の引数によってそれぞれ指定された *x* 座標と *y* 座標のアークタンジェント (逆正接) の値をラジアン単位の角度として戻します。
311 ページの『AVG』	SYSIBM	• *numeric-type*[4]	• *numeric-type*[1]	この集約関数は、一連の数値の平均値を戻します。
342 ページの『BIGINT』	SYSIBM	• *numeric-type* • VARCHAR	• BIGINT • BIGINT	このスカラー関数は、数値または文字ストリングを 64 ビットで表した整数を、整数定数の形で戻します。
344 ページの『BITAND、BITANDNOT、BITOR、BITXOR、および BITNOT』	SYSIBM	• これらの関数の詳細説明を参照してください。	• これらの関数の詳細説明を参照してください。	これらのビット単位関数は、入力引数の整数値の「2 の補数」表記で処理し、結果を入力引数のデータ・タイプに基づくデータ・タイプで対応する基数 10 (10 進数) の整数値として返します。
346 ページの『BLOB』	SYSIBM	• *string-type* • *string-type*, INTEGER	• BLOB • BLOB	このスカラー関数は、オプションの長さを指定してソース・タイプから BLOB にキャストします。
347 ページの『CARDINALITY』	SYSIBM	• 配列	• BIGINT	このスカラー関数は、配列のエレメント数を戻します。
348 ページの『CEILING』	SYSIBM	• SMALLINT • INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	• SMALLINT • INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	このスカラー関数は、引数より大きいか等しい最小の整数を戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
349 ページの『CHAR』	SYSIBM	• *character-type* • *character-type*, INTEGER • *datetime-type* • *datetime-type*, *keyword*[2] • SMALLINT • INTEGER • BIGINT • DECIMAL • DECIMAL、VARCHAR • DECFLOAT(*n*)	• CHAR • CHAR(*integer*) • CHAR • CHAR • CHAR(6) • CHAR(11) • CHAR(20) • CHAR(2+*precision*) • CHAR(2+*precision*) • CHAR(42)	このスカラー関数は、ソース・タイプのストリング表記を戻します。
349 ページの『CHAR』	SYSFUN	• DOUBLE	• CHAR(24)	このスカラー関数は、浮動小数点数の文字ストリング表記を戻します。
354 ページの『CHARACTER_LENGTH』	SYSIBM	• *string-type*, *string-unit*	• INTEGER	このスカラー関数は、指定された *string-unit* で式の長さを戻します。
356 ページの『CHR』	SYSFUN	• INTEGER	• CHAR(1)	このスカラー関数は、引数で指定された ASCII コード値をもつ文字を戻します。引数の値は 0 から 255 の範囲でなければなりません。そうでない場合、戻り値は NULL 値になります。
357 ページの『CLOB』	SYSIBM	• *character-type* • *character-type*, INTEGER	• CLOB • CLOB	このスカラー関数は、オプションの長さを指定してソース・タイプから CLOB にキャストします。
358 ページの『COALESCE』 [3]	SYSIBM	• *any-type*, *any-union-compatible-type*,...	• *any-type*	このスカラー関数は、一連の引数のうち、NULL 以外の最初の引数を戻します。
359 ページの『COLLATION_KEY_BIT』	SYSIBM	• CHAR、照合を指定するストリング • VARCHAR、照合を指定するストリング • GRAPHIC、照合を指定するストリング • VARGRAPHIC、照合を指定するストリング • CHAR、照合を指定するストリング、INTEGER • VARCHAR、照合を指定するストリング、INTEGER • GRAPHIC、照合を指定するストリング、INTEGER • VARGRAPHIC、照合を指定するストリング、INTEGER	• VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA	このスカラー関数は、指定された *collation-name* 内に指定された *string-expression* の照合キーを表すストリングを返します。
361 ページの『COMPARE_DECFLOAT』	SYSIBM	• DECFLOAT(*n*), DECFLOAT(*n*)	• INTEGER	このスカラー関数は、2 つの引数が等しいか順不同であるか、あるいは一方の引数が他方より大きいかどうかを示す SMALLINT 値を戻します。
363 ページの『CONCAT』	SYSIBM	• *string-type*, *compatible-string-type*	• *max-string-type*	このスカラー関数は、2 つのストリング引数を連結して戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
313 ページの『CORRELATION』	SYSIBM	• *numeric-type*, *numeric-type* • *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*), *numeric-type* • *numeric-type*, DECFLOAT(*n*)	• DOUBLE • DOUBLE • DECFLOAT(34) • DECFLOAT(34)	この集約関数は、一連の数値ペアの相関係数を戻します。
364 ページの『COS』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のコサイン (余弦) を戻します。引数はラジアン単位の角度です。
365 ページの『COSH』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数の双曲線コサイン (余弦) を戻します。引数はラジアン単位の角度です。
366 ページの『COT』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のコタンジェント (余接) を戻します。引数はラジアン単位の角度です。
314 ページの『COUNT』	SYSIBM	• *any-builtin-type*[4]	• INTEGER	この集約関数は、一連の行または値の中の行または値の数を戻します。
315 ページの『COUNT_BIG』	SYSIBM	• *any-builtin-type*[4]	• DECIMAL(31,0)	この集約関数は、一連の行または値の中の行または値の数を戻します。その結果は整数の最大値より大きい場合があります。
317 ページの『COVARIANCE』	SYSIBM	• *numeric-type*, *numeric-type* • *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*), *numeric-type* • *numeric-type*, DECFLOAT(*n*)	• DOUBLE • DOUBLE • DECFLOAT(34) • DECFLOAT(34)	この集約関数は、一連の数値ペアの共分散を戻します。
367 ページの『DATAPARTITIONNUM』	SYSIBM	• *any-type*	• INTEGER	このスカラー関数は、行が置かれているデータ・パーティションのシーケンス番号 (SYSDATAPARTITIONS.SEQNO) を戻します。引数は表内の任意の列名です。
368 ページの『DATE』	SYSIBM	• DATE • TIMESTAMP • DOUBLE • VARCHAR • DECFLOAT(*n*)	• DATE • DATE • DATE • DATE • DATE	このスカラー関数は、単一の入力値から日付を戻します。
369 ページの『DAY』	SYSIBM	• VARCHAR • DATE • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、値の日の部分を戻します。
370 ページの『DAYNAME』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• VARCHAR(100) • VARCHAR(100) • VARCHAR(100)	このスカラー関数は、db2start が発行された時点のロケールに基づいて、引数の日の部分の曜日名から成る大文字小文字混合文字ストリング (たとえば、 Friday) を戻します。
371 ページの『DAYOFWEEK』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数内の曜日を 1 から 7 の範囲の整数値で戻します。1 は日曜日を表します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
372 ページの『DAYOFWEEK_ISO』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数内の曜日を 1 から 7 の範囲の整数値で戻します。1 は月曜日を表します。
373 ページの『DAYOFYEAR』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数内の年間通算日を、1 から 366 の範囲の整数値で戻します。
374 ページの『DAYS』	SYSIBM	• VARCHAR • TIMESTAMP • DATE	• INTEGER • INTEGER • INTEGER	このスカラー関数は、日付の整数表記を戻します。
375 ページの『DBCLOB』	SYSIBM	• *graphic-type* • *graphic-type*, INTEGER	• DBCLOB • DBCLOB	このスカラー関数は、ソース・タイプからオプションの長さを指定して DBCLOB にキャストします。
376 ページの『DBPARTITIONNUM』 [3]	SYSIBM	• *any-type*	• INTEGER	このスカラー関数は、行のデータベース・パーティション番号を戻します。引数は表内の任意の列名です。
378 ページの『DECFLOAT』	SYSIBM	• *numeric-type* • *numeric-type*, INTEGER	• DECFLOAT(34) • DECFLOAT(*n*)。ここで、*n* はタイプ INTEGER の (オプションの) 2 番目の引数で、16 または 34 です。	このスカラー関数は、オプションの精度を指定した 10 進浮動小数点表記の数値を戻します。
378 ページの『DECFLOAT』	SYSIBM	• VARCHAR • VARCHAR, INTEGER • VARCHAR, INTEGER, VARCHAR	• DECFLOAT(34) • DECFLOAT(*n*)。ここで、*n* はタイプ INTEGER の (オプションの) 2 番目の引数で、16 または 34 です。 • DECFLOAT(*n*)。ここで、*n* はタイプ INTEGER の (オプションの) 2 番目の引数で、16 または 34 です。	このスカラー関数は、オプションの精度および小数点文字を指定した 10 進浮動小数点表記の数値を戻します。
380 ページの『DECIMAL』	SYSIBM	• *numeric-type* • *numeric-type*, INTEGER • *numeric-type* INTEGER, INTEGER	• DECIMAL • DECIMAL • DECIMAL	このスカラー関数は、オプションの精度と位取りを指定した 10 進表記の数値を戻します。
380 ページの『DECIMAL』	SYSIBM	• VARCHAR • VARCHAR, INTEGER • VARCHAR, INTEGER, INTEGER • VARCHAR, INTEGER, INTEGER, VARCHAR	• DECIMAL • DECIMAL • DECIMAL • DECIMAL	このスカラー関数は、オプションの精度、位取り、および小数点文字を指定した 10 進数表記の文字ストリングを戻します。
384 ページの『DECODE』	SYSIBM	• *any-type*	• *any-type*	このスカラー関数は、それぞれ指定された *expression2* を *expression1* と比較します。*expression1* が *expression2* と等しいか、または *expression1* と *expression2* の両方が NULL の場合は、その次の *result-expresssion* の値が返されます。*expression2* が *expression1* に一致しない場合、*else-expression* の値が返されます。これら以外の場合は、NULL 値が返されます。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
386 ページの『DECRYPT_BIN および DECRYPT_CHAR』	SYSIBM	• VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA, VARCHAR	• VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA	このスカラー関数は、パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
386 ページの『DECRYPT_BIN および DECRYPT_CHAR』	SYSIBM	• VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA, VARCHAR	• VARCHAR • VARCHAR	このスカラー関数は、パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
388 ページの『DEGREES』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	このスカラー関数は、ラジアン単位の引数から変換した度数を戻します。
388 ページの『DEGREES』	SYSFUN	• DOUBLE	• DOUBLE	このスカラー関数は、ラジアン単位の引数から変換した度数を戻します。
389 ページの『DEREF』	SYSIBM	• 有効範囲を定義された REF(*any-structured-type*)	• *any-structured-type* (入力ターゲット・タイプと同じ)	このスカラー関数は、参照タイプ引数のターゲット・タイプのインスタンスを戻します。
390 ページの『DIFFERENCE』	SYSFUN	• VARCHAR(4000), VARCHAR(4000)	• INTEGER	このスカラー関数は、SOUNDEX 関数によって判別された 2 つの引数ストリングの語の音の相違を戻します。 4 の値は、それらのストリングが同じ音であることを意味します。
391 ページの『DIGITS』	SYSIBM	• DECIMAL	• CHAR	このスカラー関数は、数値の文字ストリング表記を戻します。
392 ページの『DOUBLE』	SYSIBM	• *numeric-type*	• DOUBLE	このスカラー関数は、数値の浮動小数点表記を戻します。
392 ページの『DOUBLE』	SYSFUN	• VARCHAR	• DOUBLE	このスカラー関数は、数値の文字ストリング表記に対応する浮動小数点数を戻します。*引数* に先行ブランクや後続ブランクがあっても、それは無視されます。
EMPTY_BLOB、EMPTY_CLOB、および EMPTY_DBCLOB スカラー関数	SYSIBM	• 引数なし	• BLOB	EMPTY_BLOB スカラー関数は長さがゼロの値を返します。
EMPTY_BLOB、EMPTY_CLOB、および EMPTY_DBCLOB スカラー関数	SYSIBM	• 引数なし	• CLOB	EMPTY_CLOB スカラー関数は長さがゼロの値を返します。
EMPTY_BLOB、EMPTY_CLOB、および EMPTY_DBCLOB スカラー関数	SYSIBM	• 引数なし	• DBCLOB	EMPTY_DBCLOB スカラー関数は長さがゼロの値を返します。
394 ページの『ENCRYPT』	SYSIBM	• VARCHAR • VARCHAR, VARCHAR • VARCHAR, VARCHAR, VARCHAR	• VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA • VARCHAR FOR BIT DATA	このスカラー関数は、データ・ストリング式の暗号化の結果である値を返します。
396 ページの『EVENT_MON_STATE』	SYSIBM	• VARCHAR	• INTEGER	このスカラー関数は、特定のイベント・モニターの作動状態を戻します。
397 ページの『EXP』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(34)	このスカラー関数は、引数の指数関数を戻します。
397 ページの『EXP』	SYSFUN	• DOUBLE	• DOUBLE	このスカラー関数は、引数の指数関数を戻します。
398 ページの『FLOAT』	SYSIBM			このスカラー関数は、DOUBLE と同じです。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
399 ページの『FLOOR』	SYSIBM	• SMALLINT • INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	• SMALLINT • INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	このスカラー関数は、引数より小か等しい最大の整数を戻します。
400 ページの『GENERATE_UNIQUE』	SYSIBM	• 引数なし	• CHAR(13) FOR BIT DATA	このスカラー関数は、同じ関数の他の実行に照らし合わせてユニークなビット・データ文字ストリングを戻します。
402 ページの『GETHINT』	SYSIBM	• VARCHAR または CLOB	• VARCHAR	このスカラー関数は、パスワードのヒントが検出された場合にそれを戻します。
403 ページの『GRAPHIC』	SYSIBM	• *graphic-type* • *graphic-type*, INTEGER	• GRAPHIC • GRAPHIC	このスカラー関数は、オプションの長さを指定してソース・タイプから GRAPHIC にキャストします。
405 ページの『GREATEST』	SYSIBM	• *any-builtin-type*[9]	• 入力タイプと同じ	このスカラー関数は、一連の値の中の最大値を戻します。
318 ページの『GROUPING』	SYSIBM	• *any-type*	• SMALLINT	この集約関数は、グループ化集合によって生成された小計行を示すために、グループ化集合およびスーパー・グループで使用されます。戻される値は 0 または 1 です。1 の値は、戻された行の引数の値は NULL 値であり、行がグループ化集合用に生成されたことを意味します。生成されたこの行は、グループ化集合の小計を示します。
406 ページの『HASHEDVALUE』 [3]	SYSIBM	• *any-type*	• INTEGER	このスカラー関数は、行の分散マップ索引 (0 から 4095) を戻します。引数は表内の列名です。
408 ページの『HEX』	SYSIBM	• *any-builtin-type*	• VARCHAR	このスカラー関数は、値の 16 進表記を戻します。
410 ページの『HOUR』	SYSIBM	• VARCHAR • TIME • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、値の時間の部分を戻します。
411 ページの『IDENTITY_VAL_LOCAL』	SYSIBM		• DECIMAL	このスカラー関数は、ID 列に一番後で割り当てられた値を戻します。
415 ページの『INSERT』	SYSIBM	• この関数の詳細説明を参照してください。	• この関数の詳細説明を参照してください。	このスカラー関数は、指定された source-string 内の start から始めて、指定されたコード単位の length が削除され、insert-string が挿入されたストリングを返します。
415 ページの『INSERT』	SYSFUN	• VARCHAR(4000), INTEGER, INTEGER, VARCHAR(4000) • CLOB(1M), INTEGER, INTEGER, CLOB(1M) • BLOB(1M), INTEGER, INTEGER, BLOB(1M)	• VARCHAR(4000) • CLOB(1M) • BLOB(1M)	このスカラー関数は、*argument2* から始まる *argument3* バイトを *argument1* から削除し、 *argument2* から始まる *argument1* に *argument4* を挿入したストリングを戻します。
419 ページの『INTEGER』	SYSIBM	• *numeric-type* • VARCHAR	• INTEGER • INTEGER	このスカラー関数は、数値の整数表記を戻します。
421 ページの『JULIAN_DAY』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、紀元前 4712 年 1 月 1 日 (ユリウス暦の起点) からの経過日数を表す整数値を *argument* に指定された日付値に戻します。
422 ページの『LCASE』	SYSIBM			このスカラー関数は、LOWER と同じです。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
423 ページの『LCASE (ロケール依存)』	SYSIBM			このスカラー関数は、LOWER と同じです。
424 ページの『LEAST』	SYSIBM	• *any-builtin-type*[9]	• 入力タイプと同じ	このスカラー関数は、一連の値の中の最小値を戻します。
425 ページの『LEFT』	SYSIBM	• CHAR または VARCHAR、INTEGER • CLOB、INTEGER • GRAPHIC または VARGRAPHIC、INTEGER • DBCLOB, INTEGER	• VARCHAR • CLOB • VARGRAPHIC • DBCLOB	スカラー関数は、string-expression の左端にある長さ length (指定されたストリング単位) のストリングを戻します。string-expression が文字ストリングである場合、結果は文字ストリングです。string-expression が GRAPHIC ストリングである場合、結果は GRAPHIC ストリングです。
425 ページの『LEFT』	SYSFUN	• VARCHAR(4000), INTEGER • CLOB(1M), INTEGER • BLOB(1M), INTEGER	• VARCHAR(4000) • CLOB(1M) • BLOB(1M)	このスカラー関数は、*argument1* の左端の *argument2* バイトからなるストリングを戻します。
429 ページの『LENGTH』	SYSIBM	• *any-builtin-type* • *any-builtin-type*, *string-unit*	• INTEGER • INTEGER	このスカラー関数は、オペランドの長さをバイト数で戻します (長さを 2 バイト文字で戻す 2 バイト・ストリング型を除きます)。
439 ページの『LOWER』	SYSIBM	• CHAR • VARCHAR	• CHAR • VARCHAR	このスカラー関数は、すべての文字が小文字に変換されたストリングを戻します。
431 ページの『LN』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	このスカラー関数は、引数の自然対数値を戻します (LOG と同じ)。
431 ページの『LN』	SYSFUN	• DOUBLE	• DOUBLE	このスカラー関数は、引数の自然対数値を戻します (LOG と同じ)。
432 ページの『LOCATE スカラー関数』	SYSIBM	• *string-type*, *compatible-string-type* • *string-type*, *compatible-string-type*, INTEGER • *string-type*, *compatible-string-type*, *string-unit* • *string-type*, *compatible-string-type*, INTEGER, *string-unit*	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、*argument2* の中の *argument1* の最初のオカレンスの開始位置を戻します。オプションの INTEGER 引数を指定すると、その引数は *argument2* の中で探索を開始する文字位置を示します。*argument1* が *argument2* 内にない場合、値 0 が戻されます。
436 ページの『LOG10』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	このスカラー関数は、引数の 10 を底とする対数を戻します。
436 ページの『LOG10』	SYSFUN	• DOUBLE	• DOUBLE	このスカラー関数は、引数の 10 を底とする対数を戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
440 ページの『LOWER (ロケール依存)』	SYSIBM	• CHAR、ロケールを指定するストリング • VARCHAR、ロケールを指定するストリング • GRAPHIC、ロケールを指定するストリング • VARGRAPHIC、ロケールを指定するストリング • CHAR、ロケールを指定するストリング、INTEGER • VARCHAR、ロケールを指定するストリング、INTEGER • GRAPHIC、ロケールを指定するストリング、INTEGER • VARGRAPHIC、ロケールを指定するストリング、INTEGER	• VARCHAR • VARCHAR • VARGRAPHIC • VARGRAPHIC • VARCHAR • VARCHAR • VARGRAPHIC • VARGRAPHIC	このスカラー関数は、指定されたロケールに関連付けられた Unicode 規格の規則を使用して、すべての文字が小文字に変換されたストリングを戻します。
442 ページの『LTRIM』	SYSIBM	• CHAR • VARCHAR • GRAPHIC • VARGRAPHIC	• VARCHAR • VARCHAR • VARGRAPHIC • VARGRAPHIC	このスカラー関数は、先行ブランクを除去した引数の文字を戻します。
320 ページの『MAX』	SYSIBM	• *any-builtin-type*[5]	• 入力タイプと同じ	この集約関数は、一連の値の中の最大値を戻します。
443 ページの『MAX』	SYSIBM	• *any-builtin-type*[9]	• 入力タイプと同じ	このスカラー関数は、一連の値の中の最大値を戻します。
444 ページの『MAX_CARDINALITY』	SYSIBM	• 配列	• BIGINT	このスカラー関数は、配列に含めることができるエレメントの最大数を戻します。
445 ページの『MICROSECOND』	SYSIBM	• VARCHAR • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER	このスカラー関数は、値のマイクロ秒 (時間単位) の部分を戻します。
446 ページの『MIDNIGHT_SECONDS』	SYSFUN	• VARCHAR(26) • TIME • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、午前 0 時から *argument* に指定された時刻値までの秒数を表す 0 から 86 400 の範囲の整数値を戻します。
322 ページの『MIN』	SYSIBM	• *any-builtin-type*[5]	• 入力タイプと同じ	この集約関数は、一連の値の中の最小値を戻します。
447 ページの『MIN』	SYSIBM	• *any-builtin-type*[9]	• 入力タイプと同じ	このスカラー関数は、一連の値の中の最小値を戻します。
448 ページの『MINUTE』	SYSIBM	• VARCHAR • TIME • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、値の分の部分を戻します。
449 ページの『MOD』	SYSFUN	• SMALLINT, SMALLINT • INTEGER, INTEGER • BIGINT, BIGINT	• SMALLINT • INTEGER • BIGINT	このスカラー関数は、*>argument1* を *argument2* で除算した剰余 (モジュラス) を戻します。結果は、*argument1* が負の場合にのみ負になります。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
450 ページの『MONTH』	SYSIBM	• VARCHAR • DATE • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、値の月の部分を戻します。
451 ページの『MONTHNAME』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• VARCHAR(100) • VARCHAR(100) • VARCHAR(100)	このスカラー関数は、データベース開始時のロケールに基づいて、日付またはタイム・スタンプである引数の月の部分の月名 (January など) から成る大文字小文字混合文字ストリングを戻します。
452 ページの『MULTIPLY_ALT』	SYSIBM	• *exact-numeric-type*, *exact-numeric-type*	• DECIMAL	このスカラー関数は、2 つの引数の積を 10 進数で返します。この関数は、引数の精度の合計が 31 より大きい場合に便利です。
454 ページの『NORMALIZE_DECFLOAT』	SYSIBM	• DECFLOAT(*n*)	• DECFLOAT(*n*)	このスカラー関数は、最もシンプルな形式に設定された引数の結果である 10 進浮動小数点値を戻します。
455 ページの『NULLIF』 [3]	SYSIBM	• *any-type*[5], *any-comparable-type*[5]	• *any-type*	このスカラー関数は、引数が等しい場合は NULL 値、等しくない場合は最初の引数を戻します。
456 ページの『NVL』	SYSIBM	• *any-type*, *any-union-compatible-type*,...	• *any-type*	このスカラー関数は、NULL 値以外の最初の引数を戻します。
457 ページの『OCTET_LENGTH』	SYSIBM	• *string-type*	• INTEGER	このスカラー関数は、OCTETS (バイト数) で式の長さを戻します。
458 ページの『OVERLAY』	SYSIBM	• この関数の詳細説明を参照してください。	• この関数の詳細説明を参照してください。	このスカラー関数は、指定された *source-string* 内の *start* から始めて、指定されたコード単位の *length* が削除され、*insert-string* が挿入されたストリングを返します。
462 ページの『PARAMETER』	SYSIBM	• INTEGER	• この関数の詳細説明を参照してください。	スカラー関数は、db2-fn:sqlquery 関数の呼び出しの一部として XQuery によって値が動的に提供される SQL ステートメント内の位置を表します。
463 ページの『POSITION スカラー関数』	SYSIBM	• *string-type*, *string-type*, *string-unit*	• INTEGER	このスカラー関数は、*argument1* の中の *argument2* の開始位置を戻します。
466 ページの『POSSTR スカラー関数』	SYSIBM	• *string-type*, *compatible-string-type*	• INTEGER	このスカラー関数は、あるストリングが他のストリングに収容されている位置を戻します。
468 ページの『POWER』	SYSIBM	• INTEGER, INTEGER • BIGINT, BIGINT • DOUBLE, INTEGER • DOUBLE, DOUBLE • DECFLOAT(*n*)	• INTEGER • BIGINT • DOUBLE • DOUBLE • DECFLOAT(34)	このスカラー関数は、*argument1* の *argument2* 乗の値を戻します。
468 ページの『POWER』	SYSFUN	• INTEGER, INTEGER • BIGINT, BIGINT • DOUBLE, INTEGER • DOUBLE, DOUBLE	• INTEGER • BIGINT • DOUBLE • DOUBLE	このスカラー関数は、*argument1* の *argument2* 乗の値を戻します。
469 ページの『QUANTIZE』	SYSIBM	• *numeric-type*, *numeric-type*	• DECFLOAT(*n*)。ここで *n* は両方の引数が DECFLOAT(16) の場合 16 で、そうでない場合は、DECFLOAT(34) です。	このスカラー関数は、値と符号が最初の引数と等しく、その指数が 2 番目の引数の指数と等しい 10 進浮動小数点数を返します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
471 ページの『QUARTER』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数に指定された日付が属する四半期を示す 1 から 4 の範囲の整数値を戻します。
472 ページの『RADIANS』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	このスカラー関数は、度単位の引数をラジアン単位の角度に変換して戻します。
472 ページの『RADIANS』	SYSFUN	• DOUBLE	• DOUBLE	このスカラー関数は、度単位の引数をラジアン単位の角度に変換して戻します。
473 ページの『RAISE_ERROR』 [3]	SYSIBM	• VARCHAR, VARCHAR	• *any-type*[6]	このスカラー関数は、SQLCA にエラーを発生させます。戻される sqlstate は *argument1* で指定します。 2 番目の引数には、戻されるテキストが入れられます。
475 ページの『RAND』	SYSFUN	• 引数なし • INTEGER	• DOUBLE • DOUBLE	このスカラー関数は、引数をオプションのシード値として使用して、 0 から 1 のランダムな浮動小数点数値を戻します。
476 ページの『REAL』	SYSIBM	• *numeric-type*	• REAL	このスカラー関数は、数値の単精度浮動小数点表記を戻します。
477 ページの『REC2XML』	SYSIBM	• DECIMAL, VARCHAR, VARCHAR, *any-type*[7]	• VARCHAR	このスカラー関数は、XML タグでフォーマット設定されて列名と列データを収めたストリングを戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_AVGX 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_AVGY 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• INTEGER • DECFLOAT(*n*)	REGR_COUNT 集約関数は、回帰直線をフィッティングするために使用される NULL ではない数字のペアの数を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_INTERCEPT または REGR_ICPT 集約関数は、回帰直線の y 切片を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_R2 集約関数は、回帰を決定する係数を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_SLOPE 集約関数は、直線の傾きを戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_SXX 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_SXY 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	SYSIBM	• *non-decfloat-numeric-type*, *non-decfloat-numeric-type* • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	REGR_SYY 集約関数は、診断統計の計算に使用される数量を戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
482 ページの『REPEAT』	SYSFUN	• VARCHAR(4000), INTEGER • CLOB(1M), INTEGER • BLOB(1M), INTEGER	• VARCHAR(4000) • CLOB(1M) • BLOB(1M)	このスカラー関数は、*argument2* 回繰り返された *argument1* から成る文字ストリングを戻します。
483 ページの『REPLACE』	SYSFUN	• VARCHAR(4000), VARCHAR(4000), VARCHAR(4000) • CLOB(1M), CLOB(1M), CLOB(1M) • BLOB(1M), BLOB(1M), BLOB(1M)	• VARCHAR(4000) • CLOB(1M) • BLOB(1M)	このスカラー関数は、 *argument1* 内に存在する *argument2* をすべて *argument3* に置き換えます。
485 ページの『RID_BIT および RID』	SYSIBM	• 表指定子	• VARCHAR(16) FOR BIT DATA	RID_BIT スカラー関数は、行の行 ID (RID) を文字ストリング形式で戻します。RID_BIT 関数は、RID 関数よりも優先されます。
485 ページの『RID_BIT および RID』	SYSIBM	• 表指定子	• BIGINT	RID スカラー関数は、行の RID を長精度整数形式で返します。この関数は、パーティション・データベース環境ではサポートされません。
487 ページの『RIGHT』	SYSIBM	• CHAR または VARCHAR、INTEGER • CLOB、INTEGER • GRAPHIC または VARGRAPHIC、INTEGER • DBCLOB、INTEGER	• VARCHAR • CLOB • VARGRAPHIC • DBCLOB	スカラー関数は、string-expression の右端にある長さ length (指定されたストリング単位) のストリングを戻します。 string-expression が文字ストリングである場合、結果は文字ストリングです。 string-expression が GRAPHIC ストリングである場合、結果は GRAPHIC ストリングです。
487 ページの『RIGHT』	SYSFUN	• VARCHAR(4000), INTEGER • CLOB(1M), INTEGER • BLOB(1M), INTEGER	• VARCHAR(4000) • CLOB(1M) • BLOB(1M)	このスカラー関数は、*argument1* の右端の *argument2* バイトからなるストリングを戻します。
491 ページの『ROUND』	SYSIBM	• INTEGER, INTEGER • BIGINT, INTEGER • DOUBLE, INTEGER • DECFLOAT(*n*), INTEGER	• INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	このスカラー関数は、最初の引数を小数点以下 *argument2* 桁目で丸めて戻します。*argument2* が負の場合、 *argument1* は小数点の左側の *argument2* 桁数の絶対値に丸められます。
493 ページの『RTRIM』	SYSIBM	• CHAR • VARCHAR • GRAPHIC • VARGRAPHIC	• VARCHAR • VARCHAR • VARGRAPHIC • VARGRAPHIC	このスカラー関数は、末尾ブランクを除去した引数の文字を戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
494 ページの『SECLABEL』	SYSIBM	• CHAR、セキュリティー・ラベル・ストリング・フォーマットのストリング • VARCHAR、セキュリティー・ラベル・ストリング・フォーマットのストリング • GRAPHIC、セキュリティー・ラベル・ストリング・フォーマットのストリング • VARGRAPHIC、セキュリティー・ラベル・ストリング・フォーマットのストリング	• DB2SECURITYLABEL	このスカラー関数は、名前の付いていないセキュリティー・ラベルを戻します。
495 ページの『SECLABEL_BY_NAME』	SYSIBM	• CHAR, DB2SECURITYLABEL • VARCHAR, DB2SECURITYLABEL • GRAPHIC, DB2SECURITYLABEL • VARGRAPHIC, DB2SECURITYLABEL	• DB2SECURITYLABEL	このスカラー関数は、特定のセキュリティー・ラベルを戻します。
496 ページの『SECLABEL_TO_CHAR』	SYSIBM	• CHAR, DB2SECURITYLABEL • VARCHAR, DB2SECURITYLABEL • GRAPHIC, DB2SECURITYLABEL • VARGRAPHIC, DB2SECURITYLABEL	• セキュリティー・ラベル・ストリング・フォーマットのストリング	このスカラー関数は、セキュリティー・ラベルを受け入れ、セキュリティー・ラベル内のすべてのエレメントを入れたストリングを戻します。
498 ページの『SECOND』	SYSIBM	• VARCHAR • TIME • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、値の秒 (時間単位) の部分を戻します。
499 ページの『SIGN』	SYSIBM	• SMALLINT • INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	• SMALLINT • INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	このスカラー関数は、引数の符号の標識を戻します。引数が負の場合は、-1 が戻されます。引数がゼロの場合は、0 が戻されます。引数が正の場合には、1 が戻されます。
499 ページの『SIGN』	SYSFUN	• SMALLINT • INTEGER • BIGINT • DOUBLE	• SMALLINT • INTEGER • BIGINT • DOUBLE	このスカラー関数は、引数の符号の標識を戻します。引数が負の場合は、-1 が戻されます。引数がゼロの場合は、0 が戻されます。引数が正の場合には、1 が戻されます。
500 ページの『SIN』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のサイン (正弦) を戻します。引数はラジアン単位の角度です。
501 ページの『SINH』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数の双曲線サイン (正弦) を戻します。引数はラジアン単位の角度です。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
502 ページの『SMALLINT』	SYSIBM	• *numeric-type* • VARCHAR	• SMALLINT • SMALLINT	このスカラー関数は、数値の短精度整数表記を戻します。
503 ページの『SOUNDEX』	SYSFUN	• VARCHAR(4000)	• CHAR(4)	このスカラー関数は、引数内の語の音を表す 4 文字コードを戻します。この結果を、他のストリングの音と比較することができます。
504 ページの『SPACE』	SYSFUN	• INTEGER	• VARCHAR(4000)	このスカラー関数は、*argument1* 個のブランクから成る文字ストリングを戻します。
505 ページの『SQRT』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	このスカラー関数は、引数の平方根を戻します。
505 ページの『SQRT』	SYSFUN	• DOUBLE	• DOUBLE	このスカラー関数は、引数の平方根を戻します。
326 ページの『STDDEV』	SYSIBM	• DOUBLE • DECFLOAT(*n*)	• DOUBLE • DECFLOAT(*n*)	この集約関数は、一連の数値の標準偏差を戻します。
506 ページの『STRIP』	SYSIBM	• CHAR • VARCHAR • GRAPHIC • VARGRAPHIC • CHAR, CHAR • VARCHAR, CHAR • GRAPHIC, CHAR • VARGRAPHIC, CHAR	• CHAR • VARCHAR • GRAPHIC • VARGRAPHIC • CHAR • VARCHAR • GRAPHIC • VARGRAPHIC	このスカラー関数は、先行ブランクまたは末尾ブランク、または指定されたその他の先行文字または末尾の文字を、ストリング式から除去します。
507 ページの『SUBSTR』	SYSIBM	• *string-type*, INTEGER • *string-type*, INTEGER, INTEGER	• *string-type* • *string-type*	このスカラー関数は、*argument2* から始めてストリング *argument1* のサブストリングを戻します。このサブストリングは、*argument3* バイトの長さです。 *argument3* を指定しないと、ストリングの残りが字数と想定されます。
510 ページの『SUBSTRING』	SYSIBM	• *string-type*, INTEGER, *string-unit* • *string-type*, INTEGER, INTEGER, *string-unit*	• *string-type* • *string-type*	このスカラー関数は、*argument2* から始めてストリング *argument1* のサブストリングを戻します。このサブストリングは、*argument3* 字数の長さです。 2 番目の INTEGER 引数を指定しないと、ストリングの残りが字数と想定されます。
327 ページの『SUM』	SYSIBM	• *numeric-type*[4]	• *max-numeric-type*[1]	この集約関数は、一連の数値の和を戻します。
513 ページの『TABLE_NAME』	SYSIBM	• VARCHAR • VARCHAR, VARCHAR	• VARCHAR(128) • VARCHAR(128)	このスカラー関数は、*argument1* に指定したオブジェクト名と、 *argument2* に指定したオプションのスキーマ名に基づく表またはビューの非修飾名を戻します。戻された値は、別名の解決に使用されます。
514 ページの『TABLE_SCHEMA』	SYSIBM	• VARCHAR • VARCHAR, VARCHAR	• VARCHAR(128) • VARCHAR(128)	このスカラー関数は、*argument1* 内のオブジェクト名と、 *argument2* 内のオプションのスキーマ名で指示された 2 つの部分からなる表名またはビュー名のスキーマ名部分を戻します。戻された値は、別名の解決に使用されます。
516 ページの『TAN』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数のタンジェント (正接) を戻します。引数はラジアン単位の角度です。
517 ページの『TANH』	SYSIBM	• DOUBLE	• DOUBLE	このスカラー関数は、引数の双曲線タンジェント (正接) を戻します。引数はラジアン単位の角度です。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
518 ページの『TIME』	SYSIBM	• TIME • TIMESTAMP • VARCHAR	• TIME • TIME • TIME	このスカラー関数は、値から時刻を戻します。
519 ページの『TIMESTAMP』	SYSIBM	• TIMESTAMP • VARCHAR • VARCHAR, VARCHAR • VARCHAR, TIME • DATE, VARCHAR • DATE, TIME	• TIMESTAMP • TIMESTAMP • TIMESTAMP • TIMESTAMP • TIMESTAMP • TIMESTAMP	このスカラー関数は、1 つの値または値のペアからタイム・スタンプを戻します。
521 ページの『TIMESTAMP_FORMAT』	SYSIBM	• VARCHAR, VARCHAR	• TIMESTAMP	このスカラー関数は、フォーマット・テンプレート (*argument2*) を使って解釈された文字ストリング (*argument1*) からタイム・スタンプを戻します。
525 ページの『TIMESTAMP_ISO』	SYSFUN	• DATE • TIME • TIMESTAMP • VARCHAR(26)	• TIMESTAMP • TIMESTAMP • TIMESTAMP • TIMESTAMP	このスカラー関数は、日付、時刻、またはタイム・スタンプの引数に基づいてタイム・スタンプ値を戻します。引数が日付の場合は、時間エレメントのすべてにゼロが入れられます。引数が時刻の場合、日付エレメントには CURRENT DATE の値、時刻の小数エレメントにはゼロが入れられます。
526 ページの『TIMESTAMPDIFF』	SYSFUN	• INTEGER, CHAR(22)	• INTEGER	このスカラー関数は、2 つのタイム・スタンプの差に基づいて、タイプ *argument1* の推定インターバル数を戻します。 2 番目の引数は、2 つのタイム・スタンプ・タイプの減算を行い、その結果を CHAR に変換した結果です。有効なインターバル・タイプは、以下のとおりです。 **1** 秒の小数部 **2** 秒 **4** 分 **8** 時間 **16** 日 **32** 週 **64** 月 **128** 四半期 **256** 年
528 ページの『TO_CHAR』	SYSIBM	• VARCHAR_FORMAT と同じ。	• VARCHAR_FORMAT と同じ。	このスカラー関数は、タイム・スタンプの文字表記を戻します。
529 ページの『TO_DATE』	SYSIBM	• TIMESTAMP_FORMAT と同じ。	• TIMESTAMP_FORMAT と同じ。	このスカラー関数は、文字ストリングからタイム・スタンプを戻します。
530 ページの『TOTALORDER』	SYSIBM	• DECFLOAT(*n*), DECFLOAT(*n*)	• SMALLINT	このスカラー関数は、2 つの引数の比較の順序を示す -1、0、または 1 の SMALLINT 値を戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
532 ページの『TRANSLATE スカラー関数』	SYSIBM	• CHAR • VARCHAR • CHAR, VARCHAR, VARCHAR • VARCHAR, VARCHAR, VARCHAR • CHAR, VARCHAR, VARCHAR, VARCHAR • VARCHAR, VARCHAR, VARCHAR, VARCHAR • GRAPHIC, VARGRAPHIC, VARGRAPHIC • VARGRAPHIC, VARGRAPHIC, VARGRAPHIC • GRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC • VARGRAPHIC, VARGRAPHIC, VARGRAPHIC, VARGRAPHIC	• CHAR • VARCHAR • CHAR • VARCHAR • CHAR • VARCHAR • GRAPHIC • VARGRAPHIC • GRAPHIC • VARGRAPHIC	このスカラー関数は、1 つ以上の文字を他の文字に変換したストリングを戻します。
535 ページの『TRIM』	SYSIBM	• CHAR • VARCHAR • GRAPHIC • VARGRAPHIC • CHAR, CHAR • CHAR、VARCHAR • CHAR, GRAPHIC • CHAR, VARGRAPHIC	• CHAR • VARCHAR • GRAPHIC • VARGRAPHIC • CHAR • VARCHAR • GRAPHIC • VARGRAPHIC	このスカラー関数は、先行ブランクまたは末尾ブランク、または指定されたその他の先行文字または末尾の文字を、ストリング式から除去します。
537 ページの『TRUNCATE』	SYSIBM	• INTEGER, INTEGER • BIGINT, INTEGER • DOUBLE, INTEGER • DECFLOAT(*n*), INTEGER	• INTEGER • BIGINT • DOUBLE • DECFLOAT(*n*)	このスカラー関数は、*argument1* を小数点以下 *argument2* 桁目で切り捨てて戻します。 *argument2* が負の場合、 *argument1* は、小数点の左側の *argument2* 桁数の絶対値に切り捨てられます。
539 ページの『TYPE_ID』 [3]	SYSIBM	• *any-structured-type*	• INTEGER	このスカラー関数は、引数の動的データ・タイプの内部データ・タイプ ID を戻します。この関数の結果はデータベース間で移動できないことに注意してください。
540 ページの『TYPE_NAME』 [3]	SYSIBM	• *any-structured-type*	• VARCHAR(18)	このスカラー関数は、引数の動的データ・タイプの非修飾名を戻します。
541 ページの『TYPE_SCHEMA』 [3]	SYSIBM	• *any-structured-type*	• VARCHAR(128)	このスカラー関数は、動的タイプの引数のスキーマ名を戻します。
542 ページの『UCASE』	SYSIBM			このスカラー関数は、UPPER と同じです。
543 ページの『UCASE (ロケール依存)』	SYSIBM			このスカラー関数は、UPPER と同じです。
544 ページの『UPPER』	SYSFUN	• VARCHAR	• VARCHAR	このスカラー関数は、すべての文字が大文字に変換されたストリングを戻します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
544 ページの『UPPER』	SYSIBM	• CHAR • VARCHAR	• CHAR • VARCHAR	このスカラー関数は、すべての文字が大文字に変換されたストリングを戻します。
545 ページの『UPPER (ロケール依存)』	SYSIBM	• CHAR、ロケールを指定するストリング • VARCHAR、ロケールを指定するストリング • GRAPHIC、ロケールを指定するストリング • VARGRAPHIC、ロケールを指定するストリング • CHAR、ロケールを指定するストリング、INTEGER • VARCHAR、ロケールを指定するストリング、INTEGER • GRAPHIC、ロケールを指定するストリング、INTEGER • VARGRAPHIC、ロケールを指定するストリング、INTEGER	• VARCHAR • VARCHAR • VARGRAPHIC • VARGRAPHIC • VARCHAR • VARCHAR • VARGRAPHIC • VARGRAPHIC	このスカラー関数は、指定されたロケールに関連付けられた Unicode 規格の規則を使用して、すべての文字が大文字に変換されたストリングを戻します。
547 ページの『VALUE』 [3]	SYSIBM			このスカラー関数は、COALESCE と同じです。
548 ページの『VARCHAR』	SYSIBM	• *character-type* • *character-type*, INTEGER • *datetime-type*	• VARCHAR • VARCHAR • VARCHAR	このスカラー関数は、最初の引数の VARCHAR 表記を戻します。 2 番目の引数を指定した場合、その値は結果の長さを指定します。
551 ページの『VARCHAR_FORMAT』	SYSIBM	• TIMESTAMP, VARCHAR • VARCHAR, VARCHAR	• VARCHAR • VARCHAR	このスカラー関数は、フォーマット・テンプレート (*argument2*) どおりにフォーマット設定されたタイム・スタンプ (*argument1*) の文字表現を戻します。
555 ページの『VARGRAPHIC』	SYSIBM	• *graphic-type* • *graphic-type*, INTEGER • VARCHAR	• VARGRAPHIC • VARGRAPHIC • VARGRAPHIC	このスカラー関数は、最初の引数の VARGRAPHIC 表記を戻します。 2 番目の引数を指定した場合、その値は結果の長さを指定します。
328 ページの『VARIANCE』	SYSIBM	• DOUBLE • DECFLOAT(*n*), INTEGER	• DOUBLE • DECFLOAT(*n*)	この集約関数は、一連の数値の差異を戻します。
557 ページの『WEEK』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数の年間通算週番号を、1 から 54 の範囲の整数値で戻します。
558 ページの『WEEK_ISO』	SYSFUN	• VARCHAR(26) • DATE • TIMESTAMP	• INTEGER • INTEGER • INTEGER	このスカラー関数は、引数の年間通算週番号を、1 から 53 の範囲の整数値で戻します。週の最初の日は月曜日です。第 1 週は、年の第 1 週目で木曜日が入っています。
329 ページの『XMLAGG』	SYSIBM	• XML	• XML	この集約関数は、XML 値のセットの中の NULL 以外の値ごとに 1 つずつ項目を収めた、XML シーケンスを戻します。
559 ページの『XMLATTRIBUTES』	SYSIBM	• 任意の SQL 式 (構造化タイプを除く)。ただし、スカラー全選択をその中に入れることはできません。	• XML	このスカラー関数は、引数から XML 属性を作成します。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
561 ページの『XMLCOMMENT』	SYSIBM	• *character-type*	• XML	このスカラー関数は、単一の XQuery コメント・ノード (内容は入力引数) を持った XML 値を戻します。
562 ページの『XMLCONCAT』	SYSIBM	• XML	• XML	このスカラー関数は、変更可能な数の XML 入力引数の連結が入ったシーケンスを戻します。
564 ページの『XMLDOCUMENT』	SYSIBM	• XML	• XML	このスカラー関数は、ゼロ個以上の下位ノードを持った単一の XQuery 文書ノードを持つ XML 値を戻します。
566 ページの『XMLELEMENT』	SYSIBM	• XML	• XML	このスカラー関数は、XML エレメント・ノードである XML 値を戻します。
574 ページの『XMLFOREST』	SYSIBM	• 任意の SQL 式 (構造化タイプを除く)。ただし、スカラー全選択をその中に入れることはできません。	• XML	このスカラー関数は、XML エレメント・ノードのシーケンスである XML 値を戻します。
331 ページの『XMLGROUP』	SYSIBM	• 任意の SQL 式 (構造化タイプを除く)。	• XML	この集約関数は、XQuery 文書ノードを 1 つ持つ XML 値を戻します。これには最上位エレメント・ノードが 1 つ含まれています。
578 ページの『XMLNAMESPACES』	SYSIBM	• *character-type*	• XML	このスカラー関数は、引数からネーム・スペース宣言を作成します。
580 ページの『XMLPARSE』	SYSIBM	• *character-type* • BLOB	• XML • XML	このスカラー関数は、引数を XML 文書として解析し、XML 値を戻します。
583 ページの『XMLPI』	SYSIBM	• *character-type*	• XML	このスカラー関数は、単一の XQuery 処理命令ノードを持った XML 値を戻します。
585 ページの『XMLQUERY』	SYSIBM	• *character-type*	• XML	このスカラー関数は、おそらく、指定された入力引数を XQuery 変数として使用して、XQuery 式の評価から XML 値を戻します。
588 ページの『XMLROW』	SYSIBM	• 任意の SQL 式 (構造化タイプを除く)。	• XML	このスカラー関数は、XQuery 文書ノードを 1 つ持つ XML 値を戻します。これには最上位エレメント・ノードが 1 つ含まれています。
590 ページの『XMLSERIALIZE』	SYSIBM	• *character-type* • BLOB	• XML • XML	このスカラー関数は、シリアライズ化した XML 値を、引数から生成した指定のデータ・タイプで戻します。
605 ページの『XMLTABLE』	SYSIBM	• この関数の詳細説明を参照してください。	• 入力引数で指定された列に基づいた表。	この表関数は、おそらく、指定された入力引数を XQuery 変数として使用して、XQuery 式の評価から表を戻します。行 XQuery 式の結果シーケンス内の各シーケンス項目は、結果表の行を表しています。
592 ページの『XMLTEXT』	SYSIBM	• *character-type*	• XML	このスカラー関数は、単一の XQuery テキスト・ノード (内容は入力引数) を持った XML 値を戻します。
594 ページの『XMLVALIDATE』	SYSIBM	• XML	• XML	このスカラー関数は、デフォルト値およびタイプのアノテーションを含め、XML スキーマの妥当性検査から得られた情報を加えた入力 XML 値のコピーを戻します。
599 ページの『XMLXSROBJECTID』	SYSIBM	• XML	• BIGINT	このスカラー関数は、引数に指定された XML 文書の妥当性検査で使用された XML スキーマの XSR オブジェクト ID を戻します。
600 ページの『XSLTRANSFORM』	SYSIBM	• XML	• XML	このスカラー関数は、XML データを他の形式に変換します。これには、特定の XML スキーマに準拠する XML 文書を別のスキーマに準拠する文書に変換することも含まれます。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
604 ページの『YEAR』	SYSIBM	• VARCHAR • DATE • TIMESTAMP • DECIMAL	• INTEGER • INTEGER • INTEGER • INTEGER	このスカラー関数は、値の年の部分を戻します。
"+"	SYSIBM	• *numeric-type*, *numeric-type*	• *max-numeric-type*	2 つの数値オペランドを加算します。いずれかのオペランドが特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。
"+"	SYSIBM	• *numeric-type*	• *numeric-type*	単項加算演算子。
"+"	SYSIBM	• DATE, DECIMAL(8,0) • TIME, DECIMAL(6,0) • TIMESTAMP, DECIMAL(20,6) • DECIMAL(8,0), DATE • DECIMAL(6,0), TIME • DECIMAL(20,6), TIMESTAMP • *datetime-type*, DOUBLE, *labeled-duration-code* • *datetime-type*, DECFLOAT, *labeled-duration-code*	• DATE • TIME • TIMESTAMP • DATE • TIME • TIMESTAMP • *datetime-type* • *datetime-type*	日付/時刻加算演算子
"-"	SYSIBM	• *numeric-type*, *numeric-type*	• *max-numeric-type*	2 つの数値オペランドを減算します。いずれかのオペランドが特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。
"-"	SYSIBM	• *numeric-type*	• *numeric-type*[1]	単項減算演算子。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
"-"	SYSIBM	• DATE, DATE • TIME, TIME • TIMESTAMP, TIMESTAMP • DATE, VARCHAR • TIME, VARCHAR • TIMESTAMP, VARCHAR • VARCHAR, DATE • VARCHAR, TIME • VARCHAR, TIMESTAMP • DATE, DECIMAL(8,0) • TIME, DECIMAL(6,0) • TIMESTAMP, DECIMAL(20,6) • *datetime-type*, DOUBLE, *labeled-duration-code* • *datetime-type*, DECFLOAT, *labeled-duration-code*	• DECIMAL(8,0) • DECIMAL(6,0) • DECIMAL(20,6) • DECIMAL(8,0) • DECIMAL(6,0) • DECIMAL(20,6) • DECIMAL(8,0) • DECIMAL(6,0) • DECIMAL(20,6) • DATE • TIME • TIMESTAMP • *datetime-type* • *datetime-type*	日付/時刻減算演算子。
"*"	SYSIBM	• *numeric-type*, *numeric-type*	• *max-numeric-type*	2 つの数値オペランドを乗算します。いずれかのオペランドが特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。
"/"	SYSIBM	• *numeric-type*, *numeric-type*	• *max-numeric-type*	2 つの数値オペランドを除算します。いずれかのオペランドが特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。
"‖"	SYSIBM			CONCAT と同じ。

注

- 長さで修飾されていないストリング・データ・タイプへの参照は、最大長のデータ・タイプをサポートするものと見なされます。
- 精度と位取りを指定していない DECIMAL データ・タイプへの参照は、サポートされているすべての精度と位取りが可能であると見なされます。
- スカラー関数 LONG_VARCHAR と LONG_VARGRAPHIC は、引き続きサポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明

凡例

any-builtin-type
特殊タイプ以外の任意のデータ・タイプ。

any-type データベースに定義されている任意の型。

any-structured-type
データベースに定義されているユーザー定義の構造化タイプ。

any-comparable-type
115 ページの『代入と比較』 の定義どおりに、他の引数タイプと比較可能なタイプ。

any-union-compatible-type
130 ページの『結果データ・タイプの規則』 の定義どおりに、他の引数タイプと互換性のあるタイプ。

character-type
文字ストリング・タイプ CHAR、VARCHAR、CLOB のいずれか。

compatible-string-type
他の引数と同じグループのストリング・タイプ (例えば、引数の一方が *character-type* であれば、もう一方の引数も *character-type* でなければなりません)。

datetime-type
日付/時刻タイプ DATE、TIME、TIMESTAMP のいずれか。

exact-numeric-type
厳密な数タイプ SMALLINT、INTEGER、BIGINT、DECIMAL のいずれか。

graphic-type
2 バイト文字ストリング・タイプ GRAPHIC、VARGRAPHIC、DBCLOB のいずれか。

labeled-duration-code
これは、タイプとしては SMALLINT です。関数が、プラスまたはマイナス演算子の挿入形式を使用して呼び出される場合は、『式』で定義されているラベル付き期間を使用することができます。プラスまたはマイナスの演算子文字を名前として使用しないソース関数の場合、関数を呼び出すときに labeled-duration-code 引数に次の値を使用する必要があります。

1 YEAR または YEARS

2 MONTH または MONTHS

3 DAY または DAYS

4 HOUR または HOURS

5 MINUTE または MINUTES

6 SECOND または SECONDS

7 MICROSECOND または MICROSECONDS

LOB-type ラージ・オブジェクト・タイプ BLOB、CLOB、DBCLOB のいずれか。

max-numeric-type
引数の最大数値タイプ。最大値は右端の *numeric-type* と定義されます。

max-string-type
引数の最大ストリング・タイプ。最大値は右端の *character-type* または *graphic-type* と定義されます。引数が BLOB の場合、*max-string-type* は BLOB です。

non-decfloat-numeric-type
10 進浮動小数点を除く、数値タイプ SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE のいずれか。

numeric-type
数値タイプ SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE、DECFLOAT のいずれか。

string-unit ストリングの長さを判別するときに使用する単位を指定します。OCTETS、CODEUNITS16 または CODEUNITS32 のいずれでもかまいません。

string-type *character-type*、 *graphic-type*、または BLOB のいずれか。

表 21. サポートされている関数 (続き)

関数名	スキーマ	入力	戻り値のタイプ	説明
表の脚注				
[1]	入力パラメーターが SMALLINT の場合、結果タイプは INTEGER です。入力パラメーターが REAL の場合、結果タイプは DOUBLE です。			
[2]	使用できる keyword (キーワード) は、ISO、USA、EUR、JIS、および LOCAL です。この関数シグニチャーは、ソース関数からの派生関数としてはサポートされていません。			
[3]	この関数は、ソース関数としては使用できません。			
[4]	第 1 パラメーターの前にキーワード ALL または DISTINCT を使用できます。 DISTINCT を指定した場合、LOB-type またはユーザー定義構造化タイプは使用できません。			
[5]	LOB-type またはユーザー定義構造化タイプは使用できません。			
[6]	RAISE_ERROR によって戻されるタイプは、それを使うコンテキストによって異なります。特定のタイプにキャストしない場合、RAISE_ERROR は、CASE 式内でのその呼び出しに適したタイプを戻します。			
[7]	*graphic-type* または *LOB-type* は使用できません。			
[8]	REFERENCE、SYSPROC.DB2SECURITYLABEL、XML、またはユーザー定義タイプの使用はサポートされません。			
[9]	ARRAY、LOB-type、XML、以上のタイプのいずれかに基づく特殊タイプ、または構造化タイプの使用はサポートされません。			

表 22. 集約関数

関数	説明
309 ページの『ARRAY_AGG』	エレメントのセットを配列に集約します。
311 ページの『AVG』	一連の数値の平均を戻します。
313 ページの『CORRELATION』	一連の数値の相関係数を戻します。
314 ページの『COUNT』	一連の行または値の中の行数または値の数を戻します。
315 ページの『COUNT_BIG』	一連の行または値の中の行数または値の数を戻します。その結果は整数の最大値より大きい場合があります。
317 ページの『COVARIANCE』	一連の数値ペアの共分散を戻します。
318 ページの『GROUPING』	グループ化集合によって生成された小計行を示すために、グループ化集合およびスーパー・グループで使用されます。戻される値は 0 または 1 です。1 の値は、戻された行の引数の値は NULL 値であり、行がグループ化集合用に生成されたことを意味します。生成されたこの行は、グループ化集合の小計を示します。
320 ページの『MAX』	一連の値の最大値を戻します。
322 ページの『MIN』	一連の値の最小値を戻します。
323 ページの『回帰関数』	REGR_AVGX 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	REGR_AVGY 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	REGR_COUNT 集約関数は、回帰直線をフィッティングするために使用される NULL ではない数字のペアの数を戻します。
323 ページの『回帰関数』	REGR_INTERCEPT または REGR_ICPT 集約関数は、回帰直線の y 切片を戻します。

表 22. 集約関数 (続き)

関数	説明
323 ページの『回帰関数』	REGR_R2 集約関数は、回帰を決定する係数を戻します。
323 ページの『回帰関数』	REGR_SLOPE 集約関数は、直線の傾きを戻します。
323 ページの『回帰関数』	REGR_SXX 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	REGR_SXY 集約関数は、診断統計の計算に使用される数量を戻します。
323 ページの『回帰関数』	REGR_SYY 集約関数は、診断統計の計算に使用される数量を戻します。
326 ページの『STDDEV』	一連の数値の標準偏差を戻します。
327 ページの『SUM』	一連の数値の和を戻します。
328 ページの『VARIANCE』	一連の数値の差異を戻します。

表 23. CAST スカラー関数

関数	説明
342 ページの『BIGINT』	数値または文字ストリングを 64 ビットで表した整数を、整数定数の形で戻します。
346 ページの『BLOB』	任意の型のストリングの BLOB 表記を戻します。
349 ページの『CHAR』	値の文字表現を戻します。
357 ページの『CLOB』	値の CLOB 表現を戻します。
368 ページの『DATE』	値から日付を戻します。
375 ページの『DBCLOB』	ストリングの DBCLOB 表現を戻します。
378 ページの『DECFLOAT』	数値の 10 進浮動小数点表記を戻します。
380 ページの『DECIMAL』	数値の DECIMAL 表現を戻します。
392 ページの『DOUBLE』	数値の倍精度表現を戻します。
EMPTY_BLOB、EMPTY_CLOB、および EMPTY_DBCLOB スカラー関数	関連付けられたデータ・タイプの、長さがゼロの値を戻します。
398 ページの『FLOAT』	数値の浮動小数点表現を戻します。
403 ページの『GRAPHIC』	ストリングの GRAPHIC 表現を戻します。
419 ページの『INTEGER』	数値の整数表記を戻します。
476 ページの『REAL』	数値の実数表現を戻します。
502 ページの『SMALLINT』	数値の SMALLINT 表現を戻します。
518 ページの『TIME』	値から時刻を戻します。
519 ページの『TIMESTAMP』	値または値のペアからタイム・スタンプを戻します。
548 ページの『VARCHAR』	値の VARCHAR 表現を戻します。
555 ページの『VARGRAPHIC』	値の VARGRAPHIC 表現を戻します。

表 24. パーティション化スカラー関数

関数	説明
367 ページの『DATAPARTITIONNUM』	行が置かれているデータ・パーティションのシーケンス番号 (SYSDATAPARTITIONS.SEQNO) を戻します。引数は表内の任意の列名です。
376 ページの『DBPARTITIONNUM』	行のデータベース・パーティション番号を戻します。引数は表内の任意の列名です。
406 ページの『HASHEDVALUE』	行の分散マップ索引 (0 から 4095) を戻します。引数は表内の列名です。

表 25. DATETIME スカラー関数

関数	説明
369 ページの『DAY』	値の日の部分を戻します。
370 ページの『DAYNAME』	db2start が出された時点のロケールに基づいて、引数の日の部分の曜日名から成る大文字小文字混合文字ストリング (例えば、 Friday) を戻します。
371 ページの『DAYOFWEEK』	値から曜日を戻します。ただし 1 は Sunday、7 は Saturday です。
372 ページの『DAYOFWEEK_ISO』	値から曜日を戻します。ただし 1 は Monday、7 は Sunday です。
373 ページの『DAYOFYEAR』	値から、年頭からの通算日数を戻します。
374 ページの『DAYS』	日付の整数表記を戻します。
410 ページの『HOUR』	値の時間の部分を戻します。
421 ページの『JULIAN_DAY』	紀元前 4712 年 1 月 1 日からの経過日数を表す整数値を、引数に指定された日付値に戻します。
445 ページの『MICROSECOND』	値のマイクロ秒の部分を戻します。
446 ページの『MIDNIGHT_SECONDS』	午前 0 時から指定した時刻値までの秒数を表す整数値を戻します。
448 ページの『MINUTE』	値の分の部分を戻します。
450 ページの『MONTH』	値の月の部分を戻します。
451 ページの『MONTHNAME』	データベース開始時のロケールに基づいて、日付またはタイム・スタンプである引数の月の部分の月名 (January など) などから成る大文字小文字混合文字ストリングを戻します。
471 ページの『QUARTER』	日付が属する四半期を表す整数を戻します。
498 ページの『SECOND』	値の秒の部分を戻します。
521 ページの『TIMESTAMP_FORMAT』	フォーマット・テンプレート (*argument2*) を使って解釈された文字ストリング (*argument1*) からタイム・スタンプを戻します。
525 ページの『TIMESTAMP_ISO』	日付、時刻、またはタイム・スタンプの引数に基づいてタイム・スタンプ値を戻します。引数が日付の場合は、時間エレメントのすべてにゼロが入れられます。引数が時刻の場合、日付エレメントには CURRENT DATE の値、時刻の小数エレメントにはゼロが入れられます。

表 25. DATETIME スカラー関数 (続き)

関数	説明
526 ページの『TIMESTAMPDIFF』	2 つのタイム・スタンプの差に基づいて、タイプ *argument1* の推定インターバル数を戻します。 2 番目の引数は、2 つのタイム・スタンプ・タイプの減算を行い、その結果を CHAR に変換した結果です。
528 ページの『TO_CHAR』	タイム・スタンプの文字表現を戻します。
529 ページの『TO_DATE』	文字ストリングからタイム・スタンプを戻します。
551 ページの『VARCHAR_FORMAT』	フォーマット・テンプレート (*argument2*) どおりにフォーマット設定されたタイム・スタンプ (*argument1*) の文字表現を戻します。
557 ページの『WEEK』	値から、年頭からの通算週を戻します。ただしその週は、日曜日から始まります。
558 ページの『WEEK_ISO』	値から、年頭からの通算週を戻します。ただしその週は、月曜日から始まります。
604 ページの『YEAR』	値の年の部分を戻します。

表 26. 数値スカラー関数

関数	説明
335 ページの『ABS または ABSVAL』	数値の絶対値を戻します。
336 ページの『ACOS』	数値のラジアン単位のアークコサイン (逆余弦) を戻します。
338 ページの『ASIN』	数値のラジアン単位のアークサイン (逆正弦) を戻します。
339 ページの『ATAN』	数値のラジアン単位のアークタンジェント (逆正接) を戻します。
341 ページの『ATANH』	数値のラジアン単位の双曲線アークタンジェント (逆正接) を戻します。
340 ページの『ATAN2』	x 座標および y 座標のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。
348 ページの『CEILING』	数値よりも大きいかまたは等しい最小整数値を戻します。
361 ページの『COMPARE_DECFLOAT』	2 つの引数が等しいか順不同であるか、あるいは一方の引数が他方より大きいかどうかを示す SMALLINT 値を戻します。
364 ページの『COS』	数値のコサイン (余弦) を戻します。
365 ページの『COSH』	数値の双曲線コサイン (余弦) を戻します。
366 ページの『COT』	引数に対するコタンジェント (余接) の値を戻します。引数はラジアン単位の角度です。
388 ページの『DEGREES』	角度の度数を戻します。
391 ページの『DIGITS』	数値の絶対値の文字ストリング表現を戻します。
397 ページの『EXP』	引数で指定された累乗の自然対数 (e) の底である値を戻します。
399 ページの『FLOOR』	数値より小さいか等しい最大整数値を戻します。
431 ページの『LN』	数値の自然対数値を戻します。
436 ページの『LOG10』	数値の常用対数 (底 10) を戻します。

表 26. 数値スカラー関数 (続き)

関数	説明
449 ページの『MOD』	最初の引数を 2 番目の引数で割った剰余を戻します。
452 ページの『MULTIPLY_ALT』	2 つの引数の積を 10 進数として返します。この関数は、引数の精度の合計が 31 より大きい場合に便利です。
454 ページの『NORMALIZE_DECFLOAT』	最もシンプルな形式に設定された引数の結果である 10 進浮動小数点値を戻します。
468 ページの『POWER』	最初の引数を 2 番目の引数に累乗した結果を戻します。
469 ページの『QUANTIZE』	値と符号が最初の引数と等しく、その指数が 2 番目の引数の指数と等しい 10 進浮動小数点数を返します。
472 ページの『RADIANS』	度単位で表された引数のラジアン数を戻します。
475 ページの『RAND』	乱数を戻します。
491 ページの『ROUND』	指定された小数点以下の桁数に丸めた数値を戻します。
499 ページの『SIGN』	数値の符号を戻します。
500 ページの『SIN』	数値のサイン (正弦) を戻します。
501 ページの『SINH』	数値の双曲線サイン (正弦) を戻します。
505 ページの『SQRT』	数値の平方根を戻します。
516 ページの『TAN』	数値のタンジェント (正接) を戻します。
517 ページの『TANH』	数値の双曲線タンジェント (正接) を戻します。
530 ページの『TOTALORDER』	2 つの引数の比較の順序を示す -1、0、または 1 の SMALLINT 値を戻します。
537 ページの『TRUNCATE』	指定された小数点以下の桁数に切り捨てられた数値を戻します。

表 27. セキュリティー・スカラー関数

関数	説明
494 ページの『SECLABEL』	名前の付いていないセキュリティー・ラベルを戻します。
495 ページの『SECLABEL_BY_NAME』	特定のセキュリティー・ラベルを戻します。
496 ページの『SECLABEL_TO_CHAR』	セキュリティー・ラベルを受け入れ、セキュリティー・ラベル内のすべてのエレメントを入れたストリングを戻します。

表 28. XML 関数

関数	説明
462 ページの『PARAMETER』	db2-fn:sqlquery 関数の呼び出しの一部として XQuery によって値が動的に提供される SQL ステートメント内の位置を表します。
329 ページの『XMLAGG』	XML 値のセットの中の NULL 以外の値ごとに 1 つずつ項目を収めた、XML シーケンスを戻します。
559 ページの『XMLATTRIBUTES』	引数から XML 属性を作成します。
561 ページの『XMLCOMMENT』	単一の XQuery コメント・ノード (内容は入力引数) を持った XML 値を戻します。

表 28. XML 関数 (続き)

関数	説明
562 ページの『XMLCONCAT』	変更可能な数の XML 入力引数の連結が入ったシーケンスを戻します。
564 ページの『XMLDOCUMENT』	ゼロ個以上の下位ノードを持った単一の XQuery 文書ノードを持つ XML 値を戻します。
566 ページの『XMLELEMENT』	XML エレメント・ノードである XML 値を戻します。
574 ページの『XMLFOREST』	XML エレメント・ノードのシーケンスである XML 値を戻します。
331 ページの『XMLGROUP』	XQuery 文書ノードを 1 つ持つ XML 値を戻します。これには最上位エレメント・ノードが 1 つ含まれています。
578 ページの『XMLNAMESPACES』	引数からネーム・スペース宣言を作成します。
580 ページの『XMLPARSE』	引数を XML 文書として解析し、XML 値を戻します。
583 ページの『XMLPI』	単一の XQuery 処理命令ノードを持った XML 値を戻します。
585 ページの『XMLQUERY』	おそらく、指定された入力引数を XQuery 変数として使用して、XQuery 式の評価から XML 値を戻します。
588 ページの『XMLROW』	XQuery 文書ノードを 1 つ持つ XML 値を戻します。これには最上位エレメント・ノードが 1 つ含まれています。
590 ページの『XMLSERIALIZE』	シリアライズ化した XML 値を、引数から生成した指定のデータ・タイプで戻します。
605 ページの『XMLTABLE』	おそらく、指定された入力引数を XQuery 変数として使用して、XQuery 式の評価から表を戻します。行 XQuery 式の結果シーケンス内の各シーケンス項目は、結果表の行を表しています。
592 ページの『XMLTEXT』	単一の XQuery テキスト・ノード (内容は入力引数) を持った XML 値を戻します。
594 ページの『XMLVALIDATE』	デフォルト値およびタイプのアノテーションを含め、XML スキーマの妥当性検査から得られた情報を加えた入力 XML 値のコピーを戻します。
599 ページの『XMLXSROBJECTID』	引数に指定された XML 文書の妥当性検査で使用された XML スキーマの XSR オブジェクト ID を戻します。
600 ページの『XSLTRANSFORM』	XML データを他の形式に変換します。これには、1 つの XML スキーマに準拠する XML 文書を別のスキーマに準拠する文書に変換することも含まれます。

表 29. ストリング・スカラー関数

関数	説明
337 ページの『ASCII』	引数の左端の文字の ASCII コード値を整数として戻します。
354 ページの『CHARACTER_LENGTH』	指定された *string-unit* で式の長さを戻します。
356 ページの『CHR』	引数で指定される ASCII コード値の文字を戻します。

表 29. ストリング・スカラー関数 (続き)

関数	説明
359 ページの『COLLATION_KEY_BIT』	指定された *collation-name* 内に指定された *string-expression* の照合キーを表す VARCHAR FOR BIT DATA ストリングを返します。
363 ページの『CONCAT』	2 つのストリングを連結したストリングを戻します。
386 ページの『DECRYPT_BIN および DECRYPT_CHAR』	パスワード・ストリングを使用した暗号化データの暗号化解除の結果である値を返します。
390 ページの『DIFFERENCE』	SOUNDEX 関数で判別された 2 つの引数ストリングの語の音の差を戻します。 4 の値は、それらのストリングが同じ音であることを意味します。
394 ページの『ENCRYPT』	データ・ストリング式の暗号化の結果である値を返します。
400 ページの『GENERATE_UNIQUE』	同じ関数の他の実行とは異なるユニークなビット・データ文字ストリングを戻します。
402 ページの『GETHINT』	パスワードのヒントが検出された場合にそれを戻します。
415 ページの『INSERT』	*argument2* から始まる *argument3* バイトを *argument1* から削除し、 *argument2* から始まる *argument1* に *argument4* を挿入したストリングを戻します。
422 ページの『LCASE』	すべての SBCS 文字を小文字に変換したストリングを戻します。
423 ページの『LCASE (ロケール依存)』	指定されたロケールに関連付けられた Unicode 規格の規則を使用して、すべての文字が小文字に変換されたストリングを戻します。
440 ページの『LOWER (ロケール依存)』	指定されたロケールに関連付けられた Unicode 規格の規則を使用して、すべての文字が小文字に変換されたストリングを戻します。
425 ページの『LEFT』	ストリングから左端の文字を戻します。
432 ページの『LOCATE スカラー関数』	別のストリング内にあるストリングの開始位置を戻します。
439 ページの『LOWER』	すべての文字を小文字に変換したストリングを戻します。
442 ページの『LTRIM』	ストリング式の先頭にあるブランクを除去します。
457 ページの『OCTET_LENGTH』	OCTETS (バイト数) で式の長さを戻します。
458 ページの『OVERLAY』	指定された *source-string* 内の *start* から始めて、指定されたコード単位の *length* が削除され、*insert-string* が挿入されたストリングを返します。
463 ページの『POSITION スカラー関数』	*argument1* の中の *argument2* の開始位置を戻します。
466 ページの『POSSTR スカラー関数』	別のストリング内にあるストリングの開始位置を戻します。
482 ページの『REPEAT』	*argument2* 回繰り返された *argument1* から成る文字ストリングを戻します。
483 ページの『REPLACE』	*argument1* 内に存在する *argument2* のすべての出現箇所を *argument3* に置き換えます。
487 ページの『RIGHT』	ストリングから右端の文字を戻します。

表 29. ストリング・スカラー関数 (続き)

関数	説明
493 ページの『RTRIM』	ストリング式の末尾にあるブランクを除去します。
503 ページの『SOUNDEX』	引数内の語の音を示す 4 文字コードを戻します。この結果を、他のストリングの音と比較することができます。
504 ページの『SPACE』	指定数のブランクから成る文字ストリングを戻します。
506 ページの『STRIP』	先行ブランクまたは末尾ブランク、または指定されたその他の先行文字または末尾の文字を、ストリング式から除去します。
507 ページの『SUBSTR』	ストリングのサブストリングを戻します。
510 ページの『SUBSTRING』	ストリングのサブストリングを戻します。
532 ページの『TRANSLATE スカラー関数』	1 つまたは複数の文字を他の文字に変換したストリングを戻します。
535 ページの『TRIM』	先行ブランクまたは末尾ブランク、または指定されたその他の先行文字または末尾の文字を、ストリング式から除去します。
542 ページの『UCASE』	UCASE 関数は、最初の引数 (*char-string-exp*) だけが指定されるという点を除けば、 TRANSLATE 関数と同じです。
543 ページの『UCASE (ロケール依存)』	指定されたロケールに関連付けられた Unicode 規格の規則を使用して、すべての文字が大文字に変換されたストリングを戻します。
544 ページの『UPPER』	すべての文字を大文字に変換したストリングを戻します。
545 ページの『UPPER (ロケール依存)』	指定されたロケールに関連付けられた Unicode 規格の規則を使用して、すべての文字が大文字に変換されたストリングを戻します。

表 30. その他のスカラー関数

関数	説明
344 ページの『BITAND、BITANDNOT、BITOR、BITXOR、および BITNOT』	これらのビット単位関数は、入力引数の整数値の「2 の補数」表記で処理し、結果を入力引数のデータ・タイプに基づくデータ・タイプで対応する基数 10 (10 進数) の整数値として返します。
347 ページの『CARDINALITY』	配列のエレメント数を示すタイプ BIGINT の値を戻します
358 ページの『COALESCE』	NULL 以外の最初の引数を戻します。
384 ページの『DECODE』	それぞれ指定された *expression2* を *expression1* と比較します。 *expression1* が *expression2* と等しいか、または *expression1* と *expression2* の両方が NULL の場合は、その次の *result-expresssion* の値が返されます。 *expression2* が *expression1* に一致しない場合、*else-expression* の値が返されます。これら以外の場合は、NULL 値が返されます。
389 ページの『DEREF』	参照タイプ引数のターゲット・タイプのインスタンスを戻します。
396 ページの『EVENT_MON_STATE』	特定のイベント・モニターの作動状態を戻します。
405 ページの『GREATEST』	一連の値の最大値を戻します。

表 30. その他のスカラー関数 (続き)

関数	説明
408 ページの『HEX』	値の 16 進数表現を戻します。
411 ページの『IDENTITY_VAL_LOCAL』	ID 列に割り当てられた最新の値を戻します。
424 ページの『LEAST』	一連の値の最小値を戻します。
429 ページの『LENGTH』	値の長さを戻します。
443 ページの『MAX』	一連の値の最大値を戻します。
444 ページの『MAX_CARDINALITY』	配列に入れることができるエレメントの最大数を示すタイプ BIGINT の値を戻します。
447 ページの『MIN』	一連の値の最小値を戻します。
455 ページの『NULLIF』	引数が等しい場合は NULL 値を戻し、それ以外の場合には最初の引数の値を戻します。
456 ページの『NVL』	NULL 以外の最初の引数を戻します。
473 ページの『RAISE_ERROR』	SQLCA にエラーを発生させます。戻される sqlstate は *argument1* で指定します。 2 番目の引数には、戻されるテキストが入れられます。
477 ページの『REC2XML』	XML タグでフォーマット設定されて列名と列データを収めたストリングを戻します。
485 ページの『RID_BIT および RID』	RID_BIT スカラー関数は、行の行 ID (RID) を文字ストリング形式で戻します。 RID スカラー関数は、行の RID を長精度整数形式で返します。 RID 関数は、パーティション・データベース環境ではサポートされません。RID_BIT 関数は、RID 関数よりも優先されます。
513 ページの『TABLE_NAME』	*argument1* に指定したオブジェクト名と、 *argument2* に指定したオプションのスキーマ名に基づく表またはビューの非修飾名を戻します。戻された値は、別名の解決に使用されます。
514 ページの『TABLE_SCHEMA』	*argument1* 内のオブジェクト名と、 *argument2* 内のオプションのスキーマ名で指示された 2 つの部分からなる表名またはビュー名のスキーマ名部分を戻します。戻された値は、別名の解決に使用されます。
539 ページの『TYPE_ID』	引数の動的データ・タイプの内部データ・タイプ ID を戻します。この関数の結果はデータベース間で移植できません。
540 ページの『TYPE_NAME』	引数の動的データ・タイプの非修飾名を戻します。
541 ページの『TYPE_SCHEMA』	動的データ・タイプの引数のスキーマ名を戻します。
547 ページの『VALUE』	NULL 以外の最初の引数を戻します。

集約関数

集約関数の引数は、1 つの式から派生する一連の値の集合です。式に列を組み込むことはできますが、スカラー全選択、他の列関数、または XMLQUERY や XMLEXISTS の式 を組み込むことはできません (SQLSTATE 42607)。その集合の有効範囲は、グループ、または中間結果表です。

GROUP BY 節が照会内に指定されている場合に、 FROM、WHERE、GROUP BY および HAVING 節の中間結果が空のセットであると、集約関数は適用されません。つまり、照会の結果は空のセットとなり、 SQLCODE は +100 に設定され、SQLSTATE は '02000' に設定されます。

GROUP BY 節が照会の中に指定されておらず、FROM、WHERE、および HAVING の節の中間結果が空のセットの場合、集約関数はその空のセットに適用されます。

例えば、次の SELECT ステートメントの結果は、部門 D01 の社員に対して重複しない JOBCODE 値の数となります。

```
SELECT COUNT(DISTINCT JOBCODE)
  FROM CORPDATA.EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

キーワード DISTINCT は、関数の引数ではなく、関数が適用される以前に実行される演算の指定と見なされます。 DISTINCT を指定すると、重複する値は除去されます。数値的に等しい 10 進浮動小数点値に関して DISTINCT 節を解釈するとき、値の中の有効数字桁数は考慮されません。例えば、10 進浮動小数点数 123.00 が、10 進浮動小数点数 123 と区別されることはありません。照会から戻される数の表記は、検出された表記のいずれかになります (例えば 123.00 か 123 のいずれか)。

暗黙のうちにまたは明示的に ALL を指定すると、重複する値は削除されません。

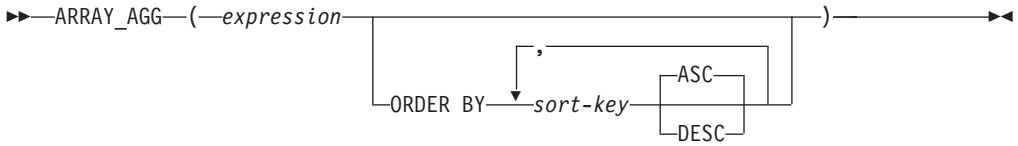
他の SQL インプリメンテーションとの互換性のため、集約関数で DISTINCT の同義語として UNIQUE を指定できます。

集約関数で、式を使用することができます。以下に例を示します。

```
SELECT MAX(BONUS + 1000)
  INTO :TOP_SALESREP_BONUS
  FROM EMPLOYEE
  WHERE COMM > 5000
```

集約関数は、スキーマ名 (例えば SYSIBM.COUNT(*)) で修飾することができます。

ARRAY_AGG



スキーマは SYSIBM です。

ARRAY_AGG 関数は、エレメントのセットを配列に集約します。式のデータ・タイプは、CREATE TYPE (配列) ステートメントで指定できるデータ・タイプでなければなりません (SQLSTATE 42884)。

sort-key を指定すると、配列に集約されるエレメントの順序が決まります。 *sort-key* が指定されていない場合、配列内でのエレメントの順序付けは非決定論的です。 *sort-key* を指定せず、同じ SELECT 節の中で ARRAY_AGG を複数回指定した場合、配列内のエレメントは、ARRAY_AGG の結果ごとに同じ順序付けが使用されます。

1 つの SELECT 節に、*sort-key* を指定する XMLAGG または ARRAY_AGG が複数回出現する場合、すべてのソート・キーは同一でなければなりません (SQLSTATE 428GZ)。

ARRAY_AGG 関数は、以下の特定のコンテキストの SQL プロシージャー内でのみ指定できます (SQLSTATE 42887)。

- SELECT INTO ステートメントの選択リスト
- スクロール可能でないカーソルの定義の全選択の選択リスト
- SET ステートメントの右側にあるスカラー副照会の選択リスト

ARRAY_AGG は OLAP 関数の一部としては使用できず、 ARRAY_AGG を使用する SELECT ステートメントには ORDER BY 節または DISTINCT 節を含めることができず、SELECT 節または HAVING 節は、副照会を含めること、または SQL 関数を呼び出すことはできません。

例:

- 次の DDL が与えられているものとします。

```
CREATE TYPE PHONELIST AS DECIMAL(10, 0)ARRAY[10]

CREATE TABLE EMPLOYEE (
  ID          INTEGER NOT NULL,
  PRIORITY    INTEGER NOT NULL,
  PHONENUMBER DECIMAL(10, 0),
  PRIMARY KEY(ID, PRIORITY))
```

  従業員に連絡を取れる連絡先番号の優先順位付けされたリストを戻す SELECT INTO ステートメントを使用するプロシージャーを作成します。

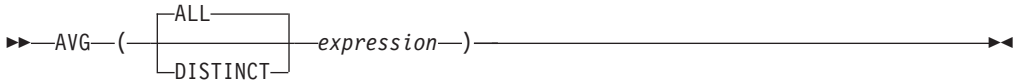
```
CREATE PROCEDURE GETPHONENUMBERS
  (IN  EMPID   INTEGER,
   OUT NUMBERS PHONELIST)
BEGIN
  SELECT ARRAY_AGG(PHONENUMBER ORDER BY PRIORITY)
```

```
    INTO NUMBERS
  FROM EMPLOYEE
  WHERE ID = EMPID;
END
```

任意の順序で従業員の連絡先番号のリストを戻す SET ステートメントを使用するプロシージャーを作成します。

```
CREATE PROCEDURE GETPHONENUMBERS
   (IN  EMPID   INTEGER,
    OUT NUMBERS PHONELIST)
BEGIN
  SET NUMBERS =
    (SELECT ARRAY_AGG(PHONENUMBER)
    FROM EMPLOYEE
    WHERE ID = EMPID);
END
```

AVG



スキーマは SYSIBM です。

AVG 関数は、一連の数値の平均値を戻します。

引数の値は数値 (組み込みタイプのみ) でなければならず、その合計は、結果のデータ・タイプの範囲内になければなりません。ただし、10 進数の結果データ・タイプは例外です。 10 進数の結果の場合、合計は、精度 31 および引数値と同一の位取りの 10 進数データ・タイプでサポートされている範囲内になければなりません。結果は NULL 値の場合もあります。

結果のデータ・タイプは、引数値のデータ・タイプと同じです。ただし、以下の場合を除きます。

- 引数値が短精度整数 (small integer) の場合、結果は長精度整数 (large integer) になります。
- 引数値が単精度浮動小数点の場合、結果は倍精度浮動小数点になります。
- 引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(34) になります。

引数値のデータ・タイプが精度 *p* で位取りが *s* の 10 進数の場合、結果の精度は 31、位取りは 31-*p* +*s* となります。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値は除去されます。数値的に等しい 10 進浮動小数点値に関して DISTINCT 節を解釈するとき、値の中の有効数字桁数は考慮されません。例えば、10 進浮動小数点数 123.00 が、10 進浮動小数点数 123 と区別されることはありません。照会から戻される数の表記は、検出された表記のいずれかになります (例えば 123.00 か 123 のいずれか)。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの平均値になります。

値が加算される順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

結果のタイプが整数であれば、平均値の小数部分は失われます。

例:

- PROJECT 表を使用して、部門 (DEPTNO) D11 におけるプロジェクトの平均スタッフ数 (PRSTAFF) を、ホスト変数 AVERAGE(decimal(5,2)) に設定します。

```
SELECT AVG(PRSTAFF)
  INTO :AVERAGE
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

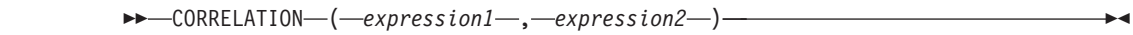
  サンプル表を使用してこの例を実行すると、結果として AVERAGE には、 4.25 (つまり、17/4) が設定されます。

- PROJECT 表を使用して、ホスト変数 ANY_CALC (decimal(5,2)) を、部門 (DEPTNO) 'D11' の中でのプロジェクトのスタッフ・レベル (PRSTAFF) の固有値についての平均値に設定します。

```
SELECT AVG(DISTINCT PRSTAFF)
  INTO :ANY_CALC
  FROM PROJECT
  WHERE DEPTNO = 'D11'
```

  サンプル表を使用してこの例を実行すると、結果として ANY_CALC は 4.66 (つまり 14/3) に設定されます。

CORRELATION



スキーマは SYSIBM です。

CORRELATION 関数は、数値の組の集合に関する相関係数を戻します。

引数には、数値を指定しなければなりません。

引数のいずれかが 10 進浮動小数点数である場合、結果は DECFLOAT(34) になります。そうでない場合、結果は倍精度浮動小数点数になります。 結果は NULL 値の場合もあります。値が NULL 値でない場合、結果は -1 から 1 になります。

この関数は、引数の値から導出されたペアの集合 (*expression1*, *expression2*) から、*expression1* または *expression2* のどちらかが NULL であるすべてのペアを除外したものに対して適用されます。

この関数が空のセットに適用された場合や、 STDDEV(*expression1*) または STDDEV(*expression2*) のどちらかがゼロに等しい場合、結果は NULL 値になります。それ以外の場合、結果はそのセット内にある値ペアの相関係数になります。結果は、以下の式と等しくなります。

```
COVARIANCE(expression1,expression2)/
(STDDEV(expression1)*
STDDEV(expression2))
```

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

CORRELATION の代わりに CORR を指定できます。

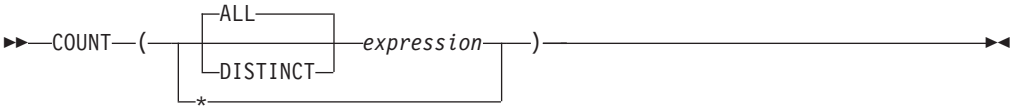
例:

- EMPLOYEE 表を使用して、ホスト変数 CORRLN (倍精度の浮動小数点数) を、部署 (WORKDEPT) 'A00' の従業員の給与と賞与の間に見られる相関に設定します。

  ```
  SELECT CORRELATION(SALARY, BONUS)
    INTO :CORRLN
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  サンプル表を使用した場合、CORRLN はおよそ 9.99853953399538E-001 に設定されます。

COUNT



スキーマは SYSIBM です。

COUNT 関数は、行の集合または値の集合内にある行または値の数を戻します。

DISTINCT が指定された場合、*expression* の結果データ・タイプは、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB、XML、これらの型の特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。そうでない場合、*expression* の結果データ・タイプはどのようなデータ・タイプにでもできます。

この関数の結果は長精度整数 (large integer) です。結果が NULL 値になることはありません。

COUNT(*) の引数は行の集合になります。結果は、集合の行の数です。 NULL 値 (NULL 値) のみから成る行もカウントに組み入れられます。

COUNT(DISTINCT *expression*) の引数は、値の集合です。この関数は、引数の値から NULL 値と重複値を除いた値の集合に対して適用されます。結果は、その集合の中の異なる非 NULL 値の数です。

COUNT(*expression*) または COUNT(ALL *expression*) の引数は、値の集合です。この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。結果は、その集合の中の NULL でない値の数です (重複値も含む)。

例:

- EMPLOYEE 表を使用して、 SEX 列の値が F である行数をホスト変数 FEMALE(int) に設定します。

  ```
  SELECT COUNT(*)
    INTO :FEMALE
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

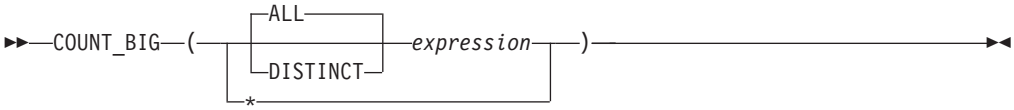
  サンプル表を使用してこの例を実行すると、結果として FEMALE に 13 が設定されます。

- EMPLOYEE 表を使用して、女性社員が少なくとも 1 人所属している部門 (WORKDEPT) の数を、ホスト変数 FEMALE_IN_DEPT (int) に設定します。

  ```
  SELECT COUNT(DISTINCT WORKDEPT)
    INTO :FEMALE_IN_DEPT
    FROM EMPLOYEE
    WHERE SEX = 'F'
  ```

  サンプル表を使用した場合、結果として FEMALE は 5 に設定されます。 (少なくとも 1 人の女性がいる部門は、A00、C01、D11、D21、および E11 です。)

COUNT_BIG



スキーマは SYSIBM です。

COUNT_BIG 関数は、一連の行または値の行の数または値の数を戻します。これは COUNT とほぼ同じですが、結果が整数の最大値より大きくなる場合がある点が異なります。

DISTINCT が指定された場合、*expression* の結果データ・タイプは、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB、XML、これらの型の特殊タイプ、または構造化タイプとすることはできません (SQLSTATE 42907)。そうでない場合、*expression* の結果データ・タイプはどのようなデータ・タイプにでもできます。

関数の結果は、精度 31 および位取り 0 の 10 進数です。結果を NULL にすることはできません。

COUNT_BIG(*) の引数は、行の集合です。結果は、集合の行の数です。 NULL 値 (NULL 値) のみから成る行もカウントに組み入れられます。

COUNT_BIG(DISTINCT *expression*) の引数は、値の集合です。この関数は、引数の値から NULL 値と重複値を除いた値の集合に対して適用されます。結果は、その集合の中の異なる非 NULL 値の数です。

COUNT_BIG(*expression*) または COUNT_BIG(ALL *expression*) の引数は、値の集合です。この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。結果は、その集合の中の NULL でない値の数です (重複値も含む)。

例:

- COUNT の例を参照して、COUNT を COUNT_BIG に置き換えてください。結果のデータ・タイプ以外は、同じ結果になります。
- アプリケーションによっては、COUNT を使用する必要があるにもかかわらず、最大整数よりも大きい値をサポートする必要がある場合があります。これは、ソースから派生されたユーザー定義関数を使用し、 SQL パスを設定して行うことができます。以下の一連のステートメントは、 COUNT_BIG に基づいて COUNT(*) をサポートするソースからの派生関数を作成して、精度 15 の 10 進数を戻す方法を示しています。 COUNT_BIG に基づくソースからの派生関数が、ここに示されている照会などの後続のステートメントで使用されるように、 SQL パスが設定されます。

  ```
  CREATE FUNCTION RICK.COUNT() RETURNS DECIMAL(15,0)
    SOURCE SYSIBM.COUNT_BIG();
  SET CURRENT PATH RICK, SYSTEM PATH;
  SELECT COUNT(*) FROM EMPLOYEE;
  ```

  ソースからの派生関数は、 COUNT(*) をサポートするパラメーターを指定せずに定義されていることに注意してください。これが有効なのは、関数 COUNT を指

定し、関数の使用時に関数をスキーマ名で修飾しない場合だけです。 COUNT 以外の名前を使用して COUNT(*) と同じ結果を得るためには、パラメーターを指定せずに関数を呼び出します。たがって、RICK.COUNT が RICK.MYCOUNT として定義されている場合、照会は以下のように書く必要があります。

```
SELECT MYCOUNT() FROM EMPLOYEE;
```

特定の列についてカウントを取る場合、ソースからの派生関数はその列のタイプを指定する必要があります。以下のステートメントによって作成されたソースからの派生関数は、 CHAR 列を引数として取り、COUNT_BIG を使用してカウントを実行します。

```
CREATE FUNCTION RICK.COUNT(CHAR()) RETURNS DOUBLE
  SOURCE SYSIBM.COUNT_BIG(CHAR());
SELECT COUNT(DISTINCT WORKDEPT) FROM EMPLOYEE;
```

COVARIANCE

スキーマは SYSIBM です。

COVARIANCE 関数は、数値の組の集合に関する (集団) 共分散を戻します。

引数には、数値を指定しなければなりません。

引数のいずれかが 10 進浮動小数点数である場合、結果は DECFLOAT(34) になります。そうでない場合、結果は倍精度浮動小数点数になります。 結果は NULL 値の場合もあります。

この関数は、引数の値から導出されたペアの集合 (*expression1*, *expression2*) から、*expression1* または *expression2* のどちらかが NULL であるすべてのペアを除外したものに対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセット内の値ペアの共分散になります。結果は、次のようにして割り出されます。

1. avgexp1 を AVG(*expression1*) の結果に、 avgexp2 を AVG(*expression2*) の結果にします。
2. COVARIANCE(*expression1*, *expression2*) の結果は、 AVG(*expression1* - avgexp1) * (*expression2* - avgexp2) になります。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

COVARIANCE の代わりに COVAR を指定できます。

例:

- EMPLOYEE 表を使用して、ホスト変数 COVARNCE (倍精度の浮動小数点) を、部署 (WORKDEPT) 'A00' の従業員の給与と賞与の間に見られる共分散に設定します。

  ```
  SELECT COVARIANCE(SALARY, BONUS)
    INTO :COVARNCE
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  サンプル表を使用した場合、COVARNCE はおよそ 1.68888888888889E+006 に設定されます。

GROUPING

スキーマは SYSIBM です。

GROUPING 関数は、グループ化集合およびスーパー・グループに関連して使用され、 GROUP BY 応答セットで戻された行が、 *expression* によって表される列を除外し、グループ化集合によって生成された行であるか否かを示す値を戻します。

引数はどのようなタイプでも構いませんが、 GROUP BY 節の項目でなければなりません。

この関数の結果は短精度整数 (small integer) です。結果は以下のいずれかの値に設定されます。

1 戻された行の *expression* の値は NULL 値であり、しかもその行はスーパー・グループによって生成されました。生成されたその行は、GROUP BY 式の小計の値を求めるのに使用することができます。

0 値は上記以外です。

例:

以下の照会を行うと、

```
SELECT SALES_DATE, SALES_PERSON,
    SUM(SALES) AS UNITS_SOLD,
    GROUPING(SALES_DATE) AS DATE_GROUP,
    GROUPING(SALES_PERSON) AS SALES_GROUP
  FROM SALES
  GROUP BY CUBE (SALES_DATE, SALES_PERSON)
  ORDER BY SALES_DATE, SALES_PERSON
```

結果は次のようになります。

```
SALES_DATE SALES_PERSON    UNITS_SOLD  DATE_GROUP  SALES_GROUP
---------- --------------- ----------- ----------- -----------
12/31/1995 GOUNOT                    1           0           0
12/31/1995 LEE                       6           0           0
12/31/1995 LUCCHESSI                 1           0           0
12/31/1995 -                         8           0           1
03/29/1996 GOUNOT                   11           0           0
03/29/1996 LEE                      12           0           0
03/29/1996 LUCCHESSI                 4           0           0
03/29/1996 -                        27           0           1
03/30/1996 GOUNOT                   21           0           0
03/30/1996 LEE                      21           0           0
03/30/1996 LUCCHESSI                 4           0           0
03/30/1996 -                        46           0           1
03/31/1996 GOUNOT                    3           0           0
03/31/1996 LEE                      27           0           0
03/31/1996 LUCCHESSI                 1           0           0
03/31/1996 -                        31           0           1
04/01/1996 GOUNOT                   14           0           0
04/01/1996 LEE                      25           0           0
04/01/1996 LUCCHESSI                 4           0           0
04/01/1996 -                        43           0           1
-          GOUNOT                   50           1           0
```

```
-          LEE                    91          1          0
-          LUCCHESSI              14          1          0
-          -                     155          1          1
```

アプリケーションは、DATE_GROUP の値が 0 であり、 SALES_GROUP の値が 1 であることによって、 SALES_DATE の小計行を認識することができます。SALES_PERSON の小計行は、DATE_GROUP の値が 1 であり、 SALES_GROUP の値が 0 であることによって認識することができます。合計行は、DATE_GROUP と SALES_GROUP の両方の値が 1 であることによって認識することができます。

MAX

スキーマは SYSIBM です。

MAX 関数は、値の集合の最大値を戻します。

引数値は、 LONG ストリング以外の任意の組み込みタイプにすることができます。

expression の結果データ・タイプは、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB にすることはできません。また、これらのタイプの特殊タイプや構造化タイプにすることもできません (SQLSTATE 42907)。

結果のデータ・タイプ、長さ、およびコード・ページは、引数値のデータ・タイプ、長さ、およびコード・ページと同じです。結果は、派生値と見なされ、NULL 値の場合もあります。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの中の最大値になります。

DISTINCT を指定しても結果に影響しないので、指定しないようにしてください。これは、他の関係システムとの互換性の目的で組み込まれています。

例:

- EMPLOYEE 表を使用して、月額の給与の最高額 (SALARY/12) の値をホスト変数 MAX_SALARY (decimal(7,2)) に設定します。

  ```
  SELECT MAX(SALARY) / 12
    INTO :MAX_SALARY
    FROM EMPLOYEE
  ```

  サンプル表を使用すると、結果として MAX_SALARY は 4395.83 に設定されます。

- PROJECT 表を使用して、照合シーケンスの最後にくるプロジェクト名 (PROJNAME) をホスト変数 LAST_PROJ (char(24)) に設定します。

  ```
  SELECT MAX(PROJNAME)
    INTO :LAST_PROJ
    FROM PROJECT
  ```

  サンプル表を使用すると、結果として LAST_PROJ は 'WELD LINE PLANNING' に設定されます。

- 前の例と同様に、プロジェクト名にホスト変数 PROJSUPP を連結した場合に照合シーケンスで最後になるプロジェクト名を、ホスト変数 LAST_PROJ (char(40)) に設定します。 PROJSUPP は '_Support' であり、データ・タイプは char(8) です。

```
SELECT MAX(PROJNAME CONCAT PROJSUPP)
  INTO :LAST_PROJ
  FROM PROJECT
```

サンプル表を使用した場合、結果として LAST_PROJ は 'WELD LINE PLANNING_SUPPORT' に設定されます。

MIN

MIN 関数は、値の集合の最小値を戻します。

引数値は、 LONG ストリング以外の任意の組み込みタイプにすることができます。

expression の結果データ・タイプは、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB にすることはできません。また、これらのタイプの特殊タイプや構造化タイプにすることもできません (SQLSTATE 42907)。

結果のデータ・タイプ、長さ、およびコード・ページは、引数値のデータ・タイプ、長さ、およびコード・ページと同じです。結果は、派生値と見なされ、NULL 値の場合もあります。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセット中の最小値になります。

DISTINCT を指定しても結果に影響しないので、指定しないようにしてください。これは、他の関係システムとの互換性の目的で組み込まれています。

例:

- EMPLOYEE 表を使用して、部門 (WORKDEPT) 'D11' の社員に対する最高と最低歩合 (COMM) の差をホスト変数 COMM_SPREAD (decimal(7,2)) に設定します。

  ```
  SELECT MAX(COMM) - MIN(COMM)
    INTO :COMM_SPREAD
    FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
  ```

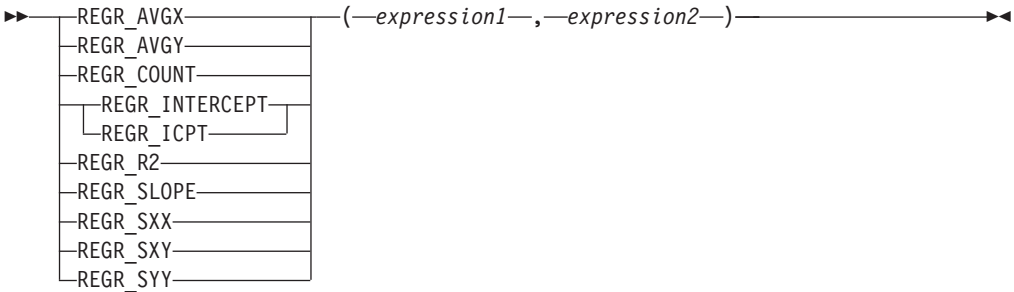
  サンプル表を使用すると、結果として COMM_SPREAD は 1118 (つまり 2580 - 1462) に設定されます。

- PROJECT 表を使用して、最初に完了するように予定されたプロジェクトの予定終了日 (PRENDATE) をホスト変数 (FIRST_FINISHED (char(10))) に設定します。

  ```
  SELECT MIN(PRENDATE)
    INTO :FIRST_FINISHED
    FROM PROJECT
  ```

  サンプル表を使用すると、結果として FIRST_FINISHED は '1982-09-15' に設定されます。

回帰関数



スキーマは SYSIBM です。

回帰関数は、通常の最小二乗法による回帰直線 (形式 y = a * x + b) を数値ペアの集合に当てはめることをサポートします。各ペアの最初のエレメント (*expression1*) は、従属変数の値 (つまり、"y 値") と解釈されます。各ペアの 2 番目のエレメント (*expression2*) は、独立変数の値 (つまり、"x 値") と解釈されます。

関数 REGR_COUNT は、回帰直線をフィッティングするために使用された NULL ではない数字のペアの数を戻します (下記参照)。

REGR_INTERCEPT (または REGR_ICPT) 関数は、回帰直線の y 切片 (前述の式では "b") を戻します。

REGR_R2 関数は、回帰に関する判別の係数 ("R 二乗" または "適合度" ともいう) を戻します。

REGR_SLOPE 関数は、直線の傾き (前述の式ではパラメーター "a") を戻します。

REGR_AVGX、REGR_AVGY、REGR_SXX、REGR_SXY、および REGR_SYY 関数は数量を戻します。そのデータを使用すれば、回帰モデルの質と統計としての有効性を評価するために必要な各種の診断統計を計算できます (下記参照)。

引数には、数値を指定しなければなりません。

REGR_COUNT の結果のデータ・タイプは整数です。残りの関数の場合、引数のいずれかが DECFLOAT(*n*) であれば、結果のデータ・タイプは DECFLOAT(34) になります。そうでない場合、結果のデータ・タイプは倍精度浮動小数点になります。いずれかの引数が特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。

結果は NULL 値の場合もあります。値が NULL 値でない場合、REGR_R2 の結果は 0 から 1 になります。 REGR_SXX と REGR_SYY の結果はどちらも正になります。

各関数は、 *expression1* または *expression2* のどちらかが NULL であるペアの除外によって、引数の値から導出されたペアの集合 (*expression1*, *expression2*) へ適用されます。

集合が NULL ではなく、かつ VARIANCE(*expression2*) が正の場合、REGR_COUNT は NULL ではない数字のペアの数を集合に戻し、その他の関数は次のように定義された結果を戻します。

```
REGR_SLOPE(expression1,expression2) =
COVARIANCE(expression1,expression2)/VARIANCE(expression2)

REGR_INTERCEPT(expression1, expression2) =
AVG(expression1) - REGR_SLOPE(expression1, expression2) * AVG(expression2)

REGR_R2(expression1, expression2) =
POWER(CORRELATION(expression1, expression2), 2) if VARIANCE(expression1)>0
REGR_R2(expression1, expression2) = 1 if VARIANCE(expression1)=0

REGR_AVGX(expression1, expression2) = AVG(expression2)

REGR_AVGY(expression1, expression2) = AVG(expression1)

REGR_SXX(expression1, expression2) =
REGR_COUNT(expression1, expression2) * VARIANCE(expression2)

REGR_SYY(expression1, expression2) =
REGR_COUNT(expression1, expression2) * VARIANCE(expression1)

REGR_SXY(expression1, expression2) =
REGR_COUNT(expression1, expression2) * COVARIANCE(expression1, expression2)
```

集合が空ではなく、かつ VARIANCE(*expression2*) がゼロに等しい場合、回帰直線は無限の傾きになるか、未定義の状態になります。その場合、関数 REGR_SLOPE、REGR_INTERCEPT、および REGR_R2 はそれぞれ NULL 値を戻し、その他の関数は上記のように定義された戻り値を戻します。集合が空の場合は、REGR_COUNT はゼロを戻し、その他の関数は NULL 値を戻します。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

回帰関数はすべて、1 回のデータ・パススルーでまとめて計算されます。一般に、回帰関数を使用して回帰分析に必要な統計を計算した方が、 AVERAGE、VARIANCE、 COVARIANCE などの通常の列関数を使用して同じ計算を実行するよりも効率的です。

線形回帰分析に関係する一般的な診断統計についても、上記の関数を使用して計算できます。以下に例を示します。

調整を加えた R2
: 1 - ((1 - REGR_R2) * ((REGR_COUNT - 1) / (REGR_COUNT - 2)))

標準誤差
: SQRT((REGR_SYY-(POWER(REGR_SXY,2)/REGR_SXX))/(REGR_COUNT-2))

二乗和の合計
: REGR_SYY

二乗和の回帰
: POWER(REGR_SXY,2) / REGR_SXX

二乗和の残差
: (二乗和の合計) - (回帰二乗和)

傾きについての t 統計
: REGR_SLOPE * SQRT(REGR_SXX) / (標準誤差)

y 切片についての t 統計

REGR_INTERCEPT/(標準誤差)*SQRT((1/REGR_COUNT)+(POWER (REGR_AVGX,2)/REGR_SXX))

例:

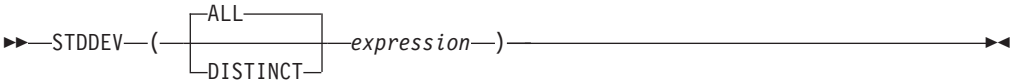
- EMPLOYEE 表を使用して、部門 (WORKDEPT) 'A00' の従業員の賞与を表す回帰直線 (通常の最小二乗法による) を、従業員の給与の線形関数として計算します。ホスト変数 SLOPE、ICPT、RSQR (いずれも倍精度浮動小数点数) をそれぞれ、回帰直線の判別の傾き、切片、係数に設定します。また、ホスト変数 AVGSAL を部門 'A00' の従業員の平均給与、ホスト変数 AVGBONUS を部門 'A00' の従業員の平均賞与に設定します。さらに、ホスト変数 CNT (整数) を、部門 'A00' の従業員のうち、給与データと賞与データが両方とも存在している従業員の数に設定します。その他の回帰統計はホスト変数 SXX、SYY、および SXY に格納します。

```
SELECT REGR_SLOPE(BONUS,SALARY), REGR_INTERCEPT(BONUS,SALARY),
REGR_R2(BONUS,SALARY), REGR_COUNT(BONUS,SALARY),
REGR_AVGX(BONUS,SALARY), REGR_AVGY(BONUS,SALARY),
REGR_SXX(BONUS,SALARY), REGR_SYY(BONUS,SALARY),
REGR_SXY(BONUS,SALARY)
INTO :SLOPE, :ICPT,
:RSQR, :CNT,
:AVGSAL, :AVGBONUS,
:SXX, :SYY,
:SXY
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'
```

サンプル表を使用する場合、ホスト変数は以下の概数値に設定されます。

```
SLOPE:  +1.71002671916749E-002
ICPT:  +1.00871888623260E+002
RSQR:  +9.99707928128685E-001
CNT:  3
AVGSAL:  +4.28333333333333E+004
AVGBONUS:  +8.33333333333333E+002
SXX:  +2.96291666666667E+008
SYY:  +8.66666666666667E+004
SXY:  +5.06666666666667E+006
```

STDDEV



スキーマは SYSIBM です。

STDDEV 関数は、一連の数値の標準偏差 (*n*) を戻します。 STDDEV の計算に使用される公式は以下のとおりです。

```
STDDEV = SQRT(VARIANCE)
```

上記で、SQRT(VARIANCE) は分散の平方根です。

引数には、数値を指定しなければなりません。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点値になります。結果は NULL 値の場合もあります。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値は除去されます。数値的に等しい 10 進浮動小数点値に関して DISTINCT 節を解釈するとき、値の中の有効数字桁数は考慮されません。例えば、10 進浮動小数点数 123.00 が、10 進浮動小数点数 123 と区別されることはありません。照会から戻される数の表記は、検出された表記のいずれかになります (例えば 123.00 か 123 のいずれか)。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの値の標準偏差になります。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

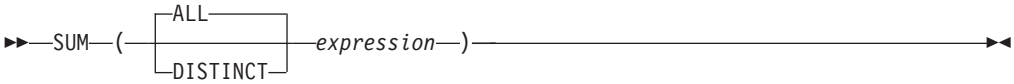
例:

- EMPLOYEE 表を使用して、ホスト変数 DEV (倍精度の浮動小数点) を部署 (WORKDEPT) 'A00' の従業員の給与の標準偏差に設定します。

  ```
  SELECT STDDEV(SALARY)
    INTO :DEV
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  DEV は、概算値 9938.00 の数値に設定されます。

SUM



スキーマは SYSIBM です。

SUM 関数は、一連の数値の合計値を戻します。

引数の値は数値 (組み込みタイプのみ) でなければならず、その合計は、結果のデータ・タイプの範囲内になければなりません。

結果のデータ・タイプは、引数値のデータ・タイプと同じです。ただし、以下の場合を除きます。

- 引数値が短精度整数 (small integer) の場合、結果は長精度整数 (large integer) になります。
- 引数値が単精度浮動小数点の場合、結果は倍精度浮動小数点になります。
- 引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(34) になります。

引数値のデータ・タイプが 10 進数の場合、結果の精度は 31、位取りは引数値の位取りと同じになります。結果は NULL 値の場合もあります。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値も除去されます。数値的に等しい 10 進浮動小数点値に関して DISTINCT 節を解釈するとき、値の中の有効数字桁数は考慮されません。例えば、10 進浮動小数点数 123.00 が、10 進浮動小数点数 123 と区別されることはありません。照会から戻される数の表記は、検出された表記のいずれかになります (例えば 123.00 か 123 のいずれか)。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの値の合計になります。

値を集計する順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

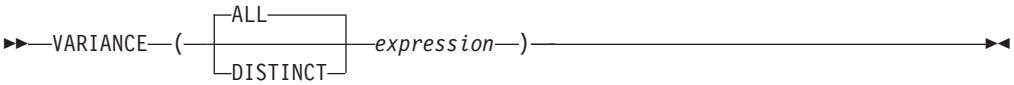
例:

- EMPLOYEE 表を使用して、事務職員 (JOB='CLERK') に支払われるボーナス (BONUS) の総額をホスト変数 JOB_BONUS (decimal(9,2)) に設定します。

```
SELECT SUM(BONUS)
  INTO :JOB_BONUS
  FROM EMPLOYEE
  WHERE JOB = 'CLERK'
```

  サンプル表を使用すると、結果として JOB_BONUS は 2800 に設定されます。

VARIANCE



スキーマは SYSIBM です。

VARIANCE 関数は、一連の数値の差異を戻します。

引数には、数値を指定しなければなりません。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点値になります。結果は NULL 値の場合もあります。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。 DISTINCT を指定すると、重複する値は除去されます。数値的に等しい 10 進浮動小数点値に関して DISTINCT 節を解釈するとき、値の中の有効数字桁数は考慮されません。例えば、10 進浮動小数点数 123.00 が、10 進浮動小数点数 123 と区別されることはありません。照会から戻される数の表記は、検出された表記のいずれかになります (例えば 123.00 か 123 のいずれか)。

この関数が空のセットに適用されると、結果は NULL 値になります。それ以外の場合、結果はそのセットの値の差異になります。

値が加算される順序は定義されていませんが、すべての中間結果は結果のデータ・タイプの範囲内になければなりません。

VARIANCE の代わりに VAR を指定できます。

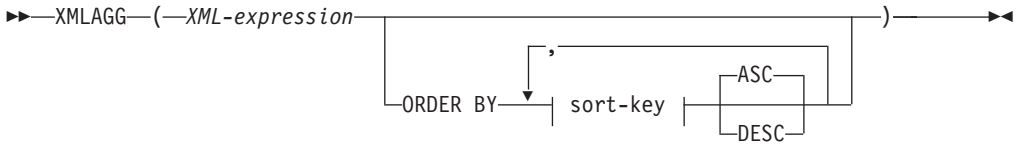
例:

- EMPLOYEE 表を使用して、ホスト変数 VARNCE (倍精度の浮動小数点) を部署 (WORKDEPT) 'A00' の従業員の給与の差異に設定します。

  ```
  SELECT VARIANCE(SALARY)
    INTO :VARNCE
    FROM EMPLOYEE
    WHERE WORKDEPT = 'A00'
  ```

  サンプル表を使用した場合、結果として VARNCE はおよそ 98763888.88 に設定されます。

XMLAGG



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLAGG 関数は、それぞれの非 NULL 値の項目を XML 値のセットに含む XML シーケンスを戻します。

XML-expression
データ・タイプ XML の式を指定します。

ORDER BY
集合内の処理対象の、同じグループ化集合に属する行の順序を指定します。ORDER BY 節を省略した場合や、ORDER BY が列データの ORDER BY を特定できない場合、同一のグループ化集合内の行は任意に順序付けられます。

sort-key
ソート・キーは、列名または *sort-key-expression* のどちらでも構いません。ソート・キーが定数の場合、ソート・キーは出力列の位置を (通常の ORDER BY 節におけるように) 参照しませんが、これは単なる定数でしかなく、ソート・キーではないことを意味することに注意してください。

結果のデータ・タイプは XML です。

この関数は、引数の値から NULL 値を除いて求めた値の集合に対して適用されます。

XML-expression 引数が NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。値のセットが空の場合、結果は NULL 値になります。それ以外の場合、結果は各値の項目をセットに含む XML シーケンスになります。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**: XML 値関数のネストの外部レベルの結果は、XMLSERIALIZE 関数の引数でなければなりません。
2. **OLAP 式でのサポート**: XMLAGG を、OLAP 集約関数の列関数として使用することはできません (SQLSTATE 42601)。

例:

注: XMLAGG は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- 姓別にソートされた従業員のリストを含む、各部門の部門エレメントを構成します。

```
SELECT XMLSERIALIZE(
  CONTENT XMLELEMENT(
    NAME "Department", XMLATTRIBUTES(
```

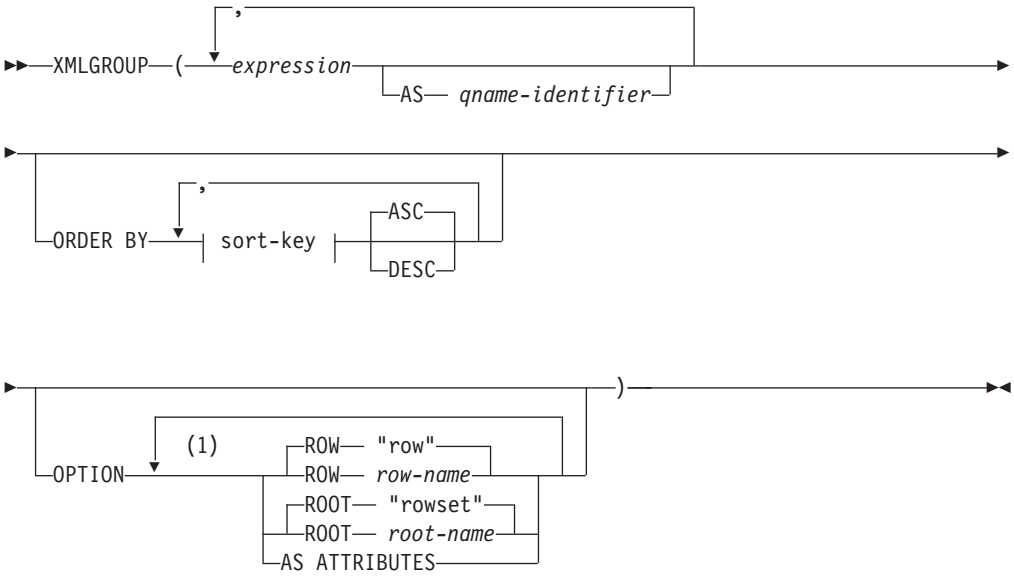
```
        E.WORKDEPT AS "name"
      ),
      XMLAGG(
        XMLELEMENT(
          NAME "emp", E.LASTNAME
        )
        ORDER BY E.LASTNAME
      )
    )
    AS CLOB(110)
  )
  AS "dept_list"
  FROM EMPLOYEE E
  WHERE E.WORKDEPT IN ('C01','E21')
  GROUP BY WORKDEPT
```

この照会は、次のような結果を生成します。

```
dept_list
----------------------...
<Department name="C01">
  <emp>KWAN</emp>
  <emp>NICHOLLS</emp>
  <emp>QUINTANA</emp>
</Department>
<Department name="E21">
  <emp>GOUNOT</emp>
  <emp>LEE</emp>
  <emp>MEHTA</emp>
  <emp>SPENSER</emp>
</Department>
```

XMLGROUP

XMLGROUP 関数は、XQuery 文書ノードを 1 つ持つ XML 値を戻します。これには最上位エレメント・ノードが 1 つ含まれています。これは、各行が行のサブエレメントにマップされる行のグループから単一ルートを持つ XML 文書を戻す集約式です。



注:

1 同じ節を複数回指定することはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

expression
生成される各 XML エレメント・ノード (または生成される各属性の値) の内容を式によって指定します。データ・タイプ *expression*を、構造化タイプとすることはできません (SQLSTATE 42884)。式には任意の SQL 式を指定できます。式が単純な列参照でない場合、*qname-identifier* を指定する必要があります。

AS *qname-identifier*
SQL ID として XML エレメント名または属性名を指定します。*qname-identifier* は、XML 修飾名の形式であるかまたは QName でなければなりません (SQLSTATE 42634)。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。名前が修飾される場合は、名前空間の接頭部をその有効範囲内で宣言する必要があります (SQLSTATE 42635)。 *qname-identifier* が指定されない場合、 *expression* は列名でなければなりません (SQLSTATE 42703)。エレメント名または属性名は、列名から QName への完全にエスケープしたマッピングを使用する列名から作成されます。

OPTION
XML 値を構成するための追加オプションを指定します。 OPTION 節を指定しない場合、デフォルトの動作が適用されます。

ROW *row-name*
各行のマップ先のエレメントの名前を指定します。オプションが指定されない場合のデフォルトのエレメント名は "row" です。

ROOT *root-name*
ルート・エレメント・ノードの名前を指定します。オプションが指定されない場合のデフォルトのルート・エレメント名は "rowset" です。

AS ATTRIBUTES
各式を列名または *qname-identifier* (属性名としての役割を果たす) を使用して属性値にマップすることを指定します。

ORDER BY
集合内の処理対象の、同じグループ化集合に属する行の順序を指定します。ORDER BY 節を省略した場合や、ORDER BY が列データの ORDER BY を特定できない場合、同一のグループ化集合内の行は任意に順序付けられます。

sort-key
ソート・キーは、列名または *sort-key-expression* のどちらでも構いません。ソート・キーが定数の場合、ソート・キーは出力列の位置を (通常の ORDER BY 節におけるように) 参照しませんが、これは単なる定数でしかなく、ソート・キーではないことを意味することに注意してください。

注

デフォルトの動作は、結果セットと XML 値の間の単純なマッピングを定義します。以下は、関数の動作に関して当てはまるいくつかの追加注意事項です。

- デフォルトで、各行は "row" という名前の XML エレメントに変換され、各列はネストされたエレメントに変換されます。その際、エレメント名として列名が使用されます。
- ヌル処理の動作は NULL ON NULL です。列の値が NULL の場合、そのマップ先のサブエレメントは空になります。すべての列の値が NULL の場合、行エレメントは生成されません。
- BLOB および FOR BIT DATA データ・タイプのバイナリー・コード化スキームは base64Binary エンコードです。
- デフォルトで、グループの行に対応するエレメントは、"rowset" という名前のルート・エレメントの子です。
- ルート・エレメントの行サブエレメントの順序は、照会結果セットに行が戻される順序と同じです。
- XML の結果を、単一ルートを持つ整形式 XML 文書とするために、文書ノードが暗黙的にルート・エレメントに追加されます。

例

整数列 C1 および C2 を持つ次のような表 T1 があるとします。列にはリレーショナル形式で格納された数値データが入っています。

```
C1          C2
----------- -----------
          1           2
          -           2
```

```
1           -
-           -

4 record(s) selected.
```

- 以下の例は、XMLGroup 照会とデフォルトの動作による出力断片が示されています。表を表すために 1 つの最上位エレメントがその中で使用されています。 :

```
SELECT XMLGROUP(C1, C2)FROM T1
```

```
<rowset>
   <row>
      <C1>1</C1>
      <C2>2</C2>
   </row>
   <row>
      <C2>2</C2>
   </row>
   <row>
      <C1>1</C1>
   </row>
</rowset>

  1 record(s) selected.
```

- 以下の例は、XMLGroup 照会と属性を中心としたマッピングによる出力断片を示しています。リレーショナル・データは前例のようにネストされたエレメントとして現れておらず、エレメント属性にマップされています。

```
SELECT XMLGROUP(C1, C2 OPTION AS ATTRIBUTES) FROM T1
```

```
<rowset>
   <row C1="1" C2="2"/>
   <row C2="2"/>
   <row C1="1"/>
</rowset>

  1 record(s) selected.
```

- 以下の例は、XMLGroup 照会とデフォルトの <rowset> ルート・エレメントが <document> によって置き換えられ、デフォルトの <row> エレメントが <entry> によって置き換えられた出力断片を示しています。列 C1 と C2 が <column1> と <column2> エレメントで返され、戻りセットは列 C1 で順序付けられます。

```
SELECT XMLGROUP(
   C1 AS "column1", C2 AS "column2"
   ORDER BY C1 OPTION ROW "entry" ROOT "document")
FROM T1
```

```
<document>
   <entry>
      <column1>1</column1>
      <column2>2</column2>
   </entry>
   <entry>
      <column1>1</column1>
   </entry>
   <entry>
      <column2>2</column2>
    </entry>
</document>
```

スカラー関数

スカラー関数は、式を使用できる個所であればどこにでも使用することができます。ただし、式と列関数の使用に適用される制約事項は、スカラー関数の中で式または列関数を使用する場合にも適用されます。例えば、スカラー関数の引数を列関数にすることができるのは、スカラー関数が使用されるコンテキストで列関数の使用が許されている場合だけです。

列関数の使用法に関する制約事項がスカラー関数に適用されないのは、スカラー関数が、値の集合ではなく、単一の値を対象にするからです。

次の SELECT ステートメントの結果には、部門 D01 の従業員の数と同じ数の行が入っています。

```
SELECT EMPNO, LASTNAME, YEAR(CURRENT DATE - BRTHDATE)
  FROM EMPLOYEE
  WHERE WORKDEPT = 'D01'
```

スカラー関数は、スキーマ名 (例えば SYSIBM.CHAR(123)) で修飾することができます。

Unicode データベースでは、文字ストリングまたは GRAPHIC ストリングを受け入れるスカラー関数はすべて、変換をサポートされている任意のストリング・タイプを受け入れます。

ABS または ABSVAL

スキーマは SYSIBM です。

この関数は、バージョン 7.1 のフィックスパック 2 から有効になりました。 ABS 関数の SYSFUN バージョン (または ABSVAL) 関数は引き続き使用可能です。

引数の絶対値を返します。引数は、任意の組み込み数値データ・タイプにすることができます。

結果のデータ・タイプと長さ属性は、引数と同じになります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。引数が SMALLINT、 INTEGER、または BIGINT の最大負数値であると、その結果はオーバーフロー・エラーになります。

例:

```
ABS(-51234)
```

は値 51234 の INTEGER を返します。

ACOS

スキーマは SYSIBM です。 (ACOS 関数の SYSFUN バージョンは引き続き使用可能です。)

引数のアークコサイン (逆余弦) の角度を戻します (ラジアン単位)。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

例:

ホスト変数 ACOSINE は、0.070737202 の値をもつ DECIMAL(10,9) ホスト変数であると仮定します。

```
SELECT ACOS(:ACOSINE)
  FROM SYSIBM.SYSDUMMY1
```

このステートメントは、近似値 1.49 を戻します。

ASCII

スキーマは SYSFUN です。

引数の左端の文字の ASCII コード値を整数として戻します。

引数は、任意の組み込み文字ストリング・タイプにすることができます。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。 VARCHAR の場合、最大長は 4000 バイトです。CLOB の場合、最大長は 1048576 バイトです。 LONG VARCHAR は、関数による処理に必要な CLOB に変換されます。

関数の結果は常に INTEGER になります。

結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

ASIN

スキーマは SYSIBM です。 (ASIN 関数の SYSFUN バージョンは引き続き使用可能です。)

引数のアークサイン (逆正弦) の角度を戻します (ラジアン単位)。

引数は、(DECFLOAT を除く) 任意の組み込み数値タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

ATAN

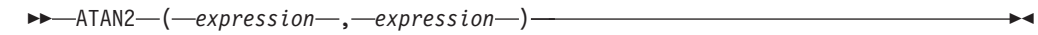
スキーマは SYSIBM です。 (ATAN 関数の SYSFUN バージョンは引き続き使用可能です。)

引数のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

ATAN2



スキーマは SYSIBM です。 (ATAN2 関数の SYSFUN バージョンは引き続き使用可能です。)

x 座標および y 座標のアークタンジェント (逆正接) の角度を戻します (ラジアン単位)。 x 座標および y 座標はそれぞれ、最初と 2 番目の引数によって指定されます。

最初と 2 番目の引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。いずれの引数も、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

ATANH

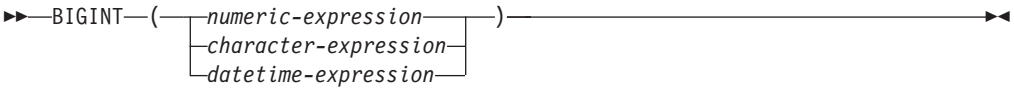
スキーマは SYSIBM です。

引数に対する双曲線アークタンジェント (逆正接) の値を戻します。引数はラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

BIGINT



スキーマは SYSIBM です。

BIGINT 関数は、数値、文字ストリング、日付、時刻、またはタイム・スタンプを 64 ビットで表した整数を、整数定数の形で戻します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

numeric-expression
組み込み数値データ・タイプの値を返す式。

引数が*数値式* の場合、結果は、引数を 64 ビット整数 (big integer) の列または変数に割り当てた場合と同じ数値になります。引数の整数部分が 64 ビット整数の範囲内にない場合は、エラーになります。引数に小数部分がある場合は、切り捨てられます。

character-expression
文字定数の最大長以下の長さの文字ストリング値を戻す式。先行ブランクと末尾ブランクは削除されます。その結果のストリングは、 SQL 整数定数を形成するための規則に従うものでなければなりません (SQLSTATE 22018)。文字ストリングとして、LONG ストリングを使うことはできません。

引数が文字式 の場合、結果は、対応する整数定数を 64 ビット整数 (big integer) の列または変数に割り当てた場合の数値と同じになります。

datetime-expression
次のデータ・タイプのいずれかの式。

- DATE。結果は、日付を *yyyymmdd* で表した BIGINT 値になります。
- TIME。結果は、時間を *hhmmss* で表した BIGINT 値になります。
- TIMESTAMP。結果は、タイム・スタンプを *yyyymmddhhmmss* で表した BIGINT 値になります。タイム・スタンプのマイクロ秒の部分は、結果には入っていません。

関数の結果は 64 ビット整数です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例:

- ORDERS_HISTORY 表から、注文の数を数えて結果を 64 ビット整数値として戻します。

```
SELECT BIGINT (COUNT_BIG(*))
  FROM ORDERS_HISTORY
```

- EMPLOYEE 表を使用して、アプリケーションでさらに処理を行うために、EMPNO 列を 64 ビット整数形式として選択します。

```
SELECT BIGINT (EMPNO) FROM EMPLOYEE
```

- RECEIVED (timestamp) 列に、 '1988-12-22-14.07.21.136421' に相当する内部値が入っていると想定します。

```
BIGINT(RECEIVED)
```

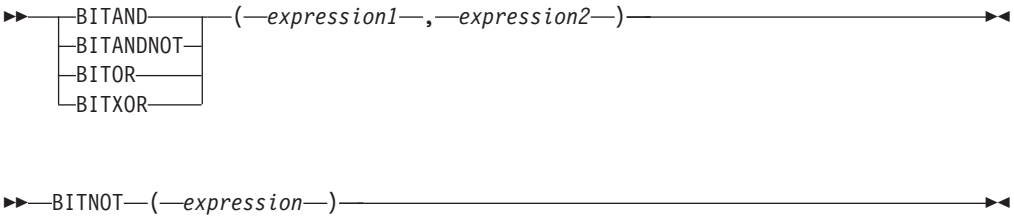
結果は、19 881 222 140 721 の値になります。

- STARTTIME (time) 列に、 '12:03:04' に相当する内部値が入っていると想定します。

```
BIGINT(STARTTIME)
```

結果は 120 304 の値になります。

BITAND、BITANDNOT、 BITOR、BITXOR、および BITNOT



スキーマは SYSIBM です。

これらのビット単位関数は、入力引数の整数値の「2 の補数」表記で処理し、結果を入力引数のデータ・タイプに基づくデータ・タイプで対応する基数 10 (10 進数) の整数値として返します。

表 31. ビット操作関数

関数	説明	結果の 2 の補数表現内のビットは、以下のとおりです。
BITAND	ビット単位 AND 演算を実行します。	両方の引数内の対応するビットが 1 である場合は、1 のみ。
BITANDNOT	2 番目の引数内にある、最初の引数のすべてのビットをクリアします。	2 番目の引数内の対応するビットが 1 の場合はゼロ。そうでない場合、結果は最初の引数内の対応するビットからコピーされます。
BITOR	ビット単位 OR 演算を実行します。	両方の引数内の対応するビットがゼロでない限り、1。
BITXOR	ビット単位排他 OR 演算を実行します。	両方の引数内の対応するビットが同じでない限り、1。
BITNOT	ビット単位 NOT 演算を実行します。	引数内の対応するビットと反対のもの。

引数は、データ・タイプ SMALLINT、INTEGER、BIGINT、または DECFLOAT により表される整数値でなければなりません。タイプ DECIMAL、REAL、または DOUBLE の引数は、DECFLOAT へのキャストです。値は切り捨てられて整数になります。

ビット操作関数は、SMALLINT の場合は最大で 16 ビット、INTEGER の場合は 32 ビット、BIGINT の場合は 64 ビット、および DECFLOAT の場合は 113 ビットまで操作できます。サポートされる DECFLOAT 値の範囲には -2^{112} から $2^{112}-1$ までの整数が含まれ、NaN や INFINITY などの特殊値はサポートされません (SQLSTATE 42815)。 2 つの引数が異なるデータ・タイプを持つ場合、より少ないビットをサポートする引数が、より多くのビットをサポートする引数のデータ・タイプを持つ値にキャストされます。このキャストは、負の値に設定されるビットに影響を与えます。例えば、SMALLINT 値としての -1 は、1 に設定された 16 ビットを持ち、これは INTEGER 値にキャストされると、1 に設定された 32 ビットを持ちます。

2 つの引数を持つ関数の結果は、プロモーション用のデータ・タイプ優先順位リスト内で最高位の引数のデータ・タイプを持ちます。いずれかの引数が DECFLOAT である場合、結果のデータ・タイプは DECFLOAT(34) です。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

BITNOT 関数の結果は、入力引数として同じデータ・タイプを持ちます。ただし DECIMAL、REAL、DOUBLE、または DECFLOAT(16) は DECFLOAT(34) を戻します。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

データ・タイプ別の、および異なるハードウェア・プラットフォーム上での内部表記の相違が原因で、BIT 関数の結果および引数の内部表記を表示または比較するための関数 (HEX など) またはホスト言語構造の使用は、データ・タイプ依存であり、移植不可です。データ・タイプおよびプラットフォームに依存せずに BIT 関数の結果および引数を表示または比較する方法は、実際の整数値を使用することです。

値のビットの切り替えには、BITXOR 関数の使用を推奨します。ビットをクリアするには、BITANDNOT 関数を使用します。 BITANDNOT(val, pattern) は、BITAND(val, BITNOT(pattern)) よりも効率的に操作を行います。

例:

以下の例は、INTEGER タイプの PROPERTIES 列を持つ ITEM 表に基づいています。

- 3 番目のプロパティー・ビットが設定されるすべての項目を戻します。

```
SELECT ITEMID FROM ITEM
  WHERE BITAND(PROPERTIES, 4) = 4
```

- 4 番目または 6 番目のプロパティー・ビットが設定されるすべての項目を戻します。

```
SELECT ITEMID FROM ITEM
  WHERE BITAND(PROPERTIES, 40) <> 0
```

- ID が 3412 の項目の 12 番目のプロパティーをクリアします。

```
UPDATE ITEM
    SET PROPERTIES = BITANDNOT(PROPERTIES, 2048)
  WHERE ITEMID = 3412
```

- ID が 3412 の項目の 5 番目のプロパティーを設定します。

```
UPDATE ITEM
    SET PROPERTIES = BITOR(PROPERTIES, 16)
  WHERE ITEMID = 3412
```

- ID が 3412 の項目の 11 番目のプロパティーを切り替えます。

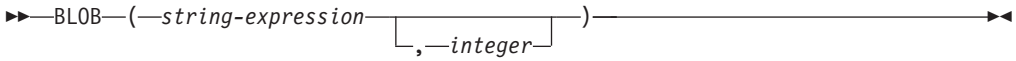
```
UPDATE ITEM
    SET PROPERTIES = BITXOR(PROPERTIES, 1024)
  WHERE ITEMID = 3412
```

- 2 番目のビットだけがオンになっている 16 ビット値のすべてのビットを切り替えます。

```
VALUES BITNOT(CAST(2 AS SMALLINT))
```

  -3 を (データ・タイプ SMALLINT で) 戻します。

BLOB



スキーマは SYSIBM です。

BLOB 関数は、任意のタイプのストリングの BLOB 表記を戻します。

string-expression
文字ストリング、GRAPHIC ストリング、またはバイナリー・ストリングの値をもつストリング式。

integer
結果の BLOB データ・タイプの長さ属性を指定する整数値。 *integer* を指定しない場合、結果の長さ属性は入力の長さと同じになります。ただし、入力が GRAPHIC ストリングの場合は除きます。その場合、結果の長さ属性は入力の長さの 2 倍になります。

関数の結果は BLOB です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例

- TOPOGRAPHIC_MAP という名前の BLOB 列と、 MAP_NAME という名前の VARCHAR 列をもつ表を使用して、 'Pellow Island' というストリングの入ったマップ (MAP) を探し出し、実際のマップの先頭にマップ名を連結した単一のバイナリー・ストリングを戻します。

```
SELECT BLOB(MAP_NAME  CONCAT ':  ') CONCAT TOPOGRAPHIC_MAP
  FROM ONTARIO_SERIES_4
  WHERE TOPOGRAPHIC_MAP LIKE BLOB('%Pellow Island%')
```

CARDINALITY

スキーマは SYSIBM です。

CARDINALITY 関数は、配列のエレメント数を示すタイプ BIGINT の値を戻します。

引数は、配列データ・タイプの SQL プロシージャーの変数またはパラメーターか、配列データ・タイプへのパラメーター・マーカーのキャスト仕様のいずれかとすることができます。 CARDINALITY 関数により戻される値は、配列に割り当てられたエレメントが含まれる副索引の中で最も高位のものです。これには、NULL 値が割り当てられたエレメントも含まれます。

この関数は、配列が空であった場合には 0 を戻します。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例:

タイプ INT_ARRAY が次のように定義されていると想定します。

```
CREATE TYPE INT_ARRAY
  AS INTEGER ARRAY[100]
```

以下の SQL PL コードのフラグメントの SET ステートメントは、変数 LEN に値 4 を割り当てます。

```
BEGIN
DECLARE LEN INTEGER;
DECLARE MYARRAY INT_ARRAY;

SET INT_ARRAY = ARRAY[0,0,1,1];
SET LEN = CARDINALITY(MYARRAY);
END
```

CEILING

スキーマは SYSIBM です。 (CEILING 関数の SYSFUN バージョンは引き続き使用可能です。)

引数よりも大きいか、または等しい整数で、最小の値を戻します。

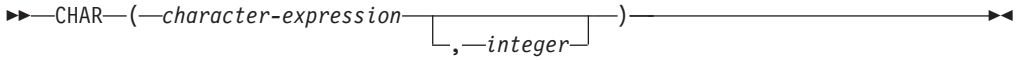
引数は、任意の組み込み数値タイプにすることができます。引数が DECIMAL の場合は位取りは 0 になる以外は、関数の結果のデータ・タイプと長さ属性は、引数と同じになります。たとえば、データ・タイプが DECIMAL(5,5) の引数は DECIMAL(5,0) を戻します。

引数が NULL になる可能性があるか、または引数が 10 進浮動小数点数ではなく、データベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

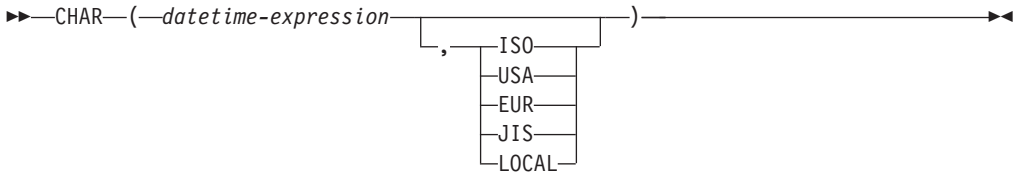
CEILING の代わりに CEIL を指定できます。

CHAR

文字→文字:



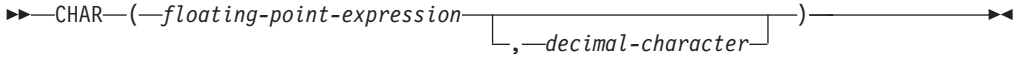
日付/時刻→文字:



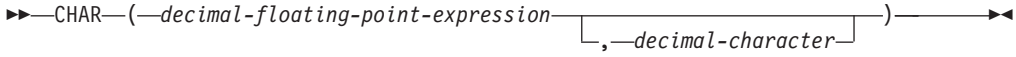
整数→文字:

10 進数→文字:

浮動小数点数→文字:



10 進浮動小数点数→文字:



スキーマは SYSIBM です。キーワードが関数シグニチャーで使用されている場合、関数名を修飾名で指定することはできません。 SYSFUN.CHAR(*floating-point-expression*) シグニチャーは、引き続き使用可能です。その場合、小数点文字はロケールに依存するため、データベース・サーバーのロケールに応じてピリオドまたはコンマが戻されます。

CHAR 関数は、以下の固定長文字ストリング表記を戻します。

- 文字ストリング (最初の引数がいずれかのタイプの文字ストリングの場合)
- 日付/時刻 (最初の引数が日付、時刻、またはタイム・スタンプの場合)
- 整数 (最初の引数が SMALLINT、INTEGER、または BIGINT の場合)
- 10 進数 (最初の引数が 10 進数の場合)
- 倍精度浮動小数点 (最初の引数が DOUBLE または REAL の場合)

- 10 進浮動小数点数 (最初の引数が DECFLOAT の場合)

最初の引数は組み込みデータ・タイプでなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

注: CAST 式を使用してストリング式を戻すこともできます。

関数の結果は、固定長文字ストリングです。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL の場合には、結果も NULL 値です。

文字→文字

character-expression
CHAR、VARCHAR、LONG VARCHAR、または CLOB のいずれかのデータ・タイプの値を戻す式。

integer
結果の固定長文字ストリングの長さ属性。値は 0 から 254 の範囲でなければなりません。

文字式の長さが結果の長さ属性より短い場合、結果の長さになるまでブランクが結果に埋められます。文字式の長さが結果の長さ属性より長い場合、切り捨てが行われます。その場合、切り捨てられた文字がすべてブランクで、文字式が LONG ストリング (LONG VARCHAR または CLOB) でない限り、警告 (SQLSTATE 01004) が戻されます。

日付/時刻→文字

datetime-expression
次の 3 つのデータ・タイプのいずれかの式。

日付 結果は、2 番目の引数によって指定された形式の日付の文字ストリング表記になります。結果の長さは 10 文字です。 2 番目の引数が指定され、その値が有効な値でない場合には、エラーが戻されます (SQLSTATE 42703)。

時刻 結果は、2 番目の引数によって指定された形式の時刻の文字ストリング表記になります。結果の長さは 8 文字です。2 番目の引数が指定され、その値が有効でない場合には、エラーが戻されます (SQLSTATE 42703)。

タイム・スタンプ
結果は、タイム・スタンプの文字ストリング表記になります。結果の長さは 26 文字です。2 番目の引数は適用されないので、指定してはなりません (SQLSTATE 42815)。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

整数→文字

integer-expression
整数データ・タイプの値 (SMALLINT、INTEGER または BIGINT のいずれか) を戻す式。

結果は、SQL 整数定数の形式による引数の文字ストリング表記になります。結果は、引数内の有効桁数を表す *n* 個の文字で構成されます。引数が負の場合は、負符号 (-) が前に付けられます。結果は、左揃えになります。

- 最初の引数が短精度整数 (small integer) の場合、その結果の長さは 6 になります。
- 最初の引数が長精度整数 (large integer) の場合、その結果の長さは 11 になります。
- 最初の引数が 64 ビット整数 (big integer) の場合、その結果の長さは 20 になります。

結果内のバイト数が、結果に定義されていた長さ未満の場合、結果の右側にブランクが埋められます。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

10 進数→文字

decimal-expression
10 進数データ・タイプの値を戻す式。別の精度と位取りが必要であれば、まず DECIMAL スカラー関数を使用して変更を行うことができます。

decimal-character
結果文字ストリングの中で 10 進数を区切るために使用する 1 バイト文字定数を指定します。文字定数を数字、正符号 (+)、負符号 (-)、またはブランク文字にすることはできません (SQLSTATE 42815)。デフォルトはピリオド (.) 文字です。

結果は、引数の固定長文字ストリング表記になります。結果は、小数点文字と *p* 桁の数字からなります (*p* は *decimal-expression* の精度)。引数が負の場合は、前に負符号が付けられます。結果の長さは 2+*p* です (*p* は *decimal-expression* の精度)。つまり、正の値には常に 1 個の末尾のブランクが付けられます。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

浮動小数点数→文字

floating-point-expression
浮動小数点データ・タイプ (DOUBLE または REAL) である値を戻す式。

decimal-character
結果文字ストリングの中で 10 進数を区切るために使用する 1 バイト文字定数を指定します。文字定数を数字、正符号 (+)、負符号 (-)、またはブランク文字にすることはできません (SQLSTATE 42815)。デフォルトはピリオド (.) 文字です。

結果は、浮動小数点定数形式の引数の固定長文字ストリング表記になります。結果の長さは 24 文字です。引数が負である場合、結果の先頭の文字は負符号 (-) になります。それ以外の場合、最初の文字は数字になります。引数値がゼロの場合、結果は 0E0 になります。それ以外の場合の結果は、

decimal-character と一連の数字が後に続くゼロ以外の 1 桁の数字で小数部が構成されることで引数の値を表すことのできる最小の文字数になります。結果のバイト数が 24 未満の場合、結果の右側にブランクが埋められます。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

10 進浮動小数点数→文字

decimal-floating-point-expression
10 進浮動小数点データ・タイプ (DECFLOAT) である値を戻す式。

decimal-character
結果文字ストリングの中で 10 進数を区切るために使用する 1 バイト文字定数を指定します。文字定数を数字、正符号 (+)、負符号 (-)、またはブランク文字にすることはできません (SQLSTATE 42815)。デフォルトはピリオド (.) 文字です。

結果は、引数の固定長文字ストリング表記になります。結果の長さは 42 文字です。結果の文字数が 42 未満の場合、長さが 42 文字になるように、結果の右側にブランク文字が埋められます。

例:

- PRSTDATE 列には、1988-12-25 に相当する内部値が入っているとします。以下の関数は、値 '12/25/1988' を戻します。

```
CHAR(PRSTDATE, USA)
```

- STARTING 列には 17:12:30 に相当する内部値が入っており、ホスト変数 HOUR_DUR (decimal(6,0)) は、050000 (すなわち 5 時間) の値をもった時刻期間であると仮定します。以下の関数は、値 '5:12 PM' を戻します。

```
CHAR(STARTING, USA)
```

以下の関数は、値 '10:12 PM' を戻します。

```
CHAR(STARTING + :HOUR_DUR, USA)
```

- RECEIVED 列 (TIMESTAMP) には、列 PRSTDATE と列 STARTING を組み合わせたものに相当する内部値が入っているとします。以下の関数は、値 '1988-12-25-17.12.30.000000' を戻します。

```
CHAR(RECEIVED)
```

- LASTNAME 列は VARCHAR(15) と定義されています。以下の関数は、10 バイトの長さの固定長文字ストリングでこの列内に値を戻します。 10 バイトを超える長さ (末尾ブランクを除く) の LASTNAME 値は切り捨てられて、警告が戻されます。

```
SELECT CHAR(LASTNAME,10) FROM EMPLOYEE
```

- EDLEVEL 列は SMALLINT と定義されています。以下の関数は、固定長文字ストリングでこの列内に値を戻します。 EDLEVEL 値が 18 であれば、CHAR(6) 値'18 ' ('18' の後に 4 つのブランクが続く) で戻されます。

```
SELECT CHAR(EDLEVEL) FROM EMPLOYEE
```

- SALARY 列は、9 の精度と 2 の位取りをもった DECIMAL と定義されています。現行値 (18357.50) は、小数点文字としてコンマを使って表示されることになります (18357,50)。以下の関数は、値 '00018357,50' を戻します。

```
CHAR(SALARY, ',')
```

- SALARY 列内の値が 20000.25 から減算されて、デフォルトの小数点文字付きで表示されることになります。以下の関数は、値 '-0001642.75' を戻します。

```
CHAR(20000.25 - SALARY)
```

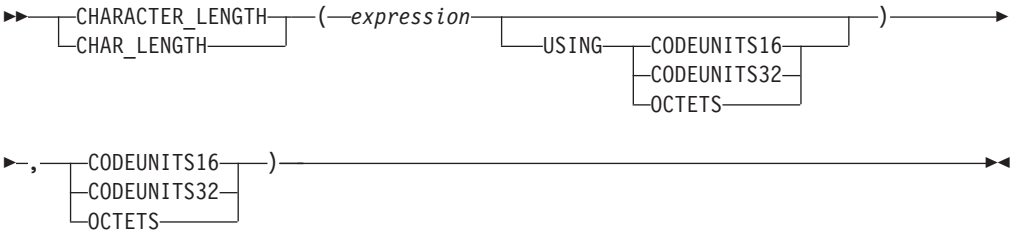
- ホスト変数 SEASONS_TICKETS は INTEGER と定義されていて、 10000 の値をもっていると想定します。以下の関数は、値 '10000.00 ' を戻します。

```
CHAR(DECIMAL(:SEASONS_TICKETS,7,2))
```

- ホスト変数 DOUBLE_NUM は DOUBLE と定義されていて、 -987.654321E-35 の値をもっていると想定します。以下の関数は、値 '-9.87654321E-33' を戻します。結果のデータ・タイプは CHAR(24) であるので、結果には 9 つの末尾ブランクがあります。

```
CHAR(:DOUBLE_NUM)
```

CHARACTER_LENGTH



スキーマは SYSIBM です。

CHARACTER_LENGTH 関数は、指定されたストリング単位で *expression* の長さを戻します。

expression
組み込み文字または GRAPHIC ストリングの値を戻す式。

CODEUNITS16、CODEUNITS32、または OCTETS
結果のストリング単位を指定します。 CODEUNITS16 は、結果が 16 ビット UTF-16 コード単位で表現されることを指定します。 CODEUNITS32 は、結果が 32 ビット UTF-32 コード単位で表現されることを指定します。 OCTETS は、結果がバイト単位で表現されることを指定します。

ストリング単位が CODEUNITS16 または CODEUNITS32 と指定され、*expression* がバイナリー・ストリングまたはビット・データである場合は、エラーが戻されます (SQLSTATE 428GC)。ストリング単位が OCTETS と指定され、*expression* がバイナリー・ストリングである場合は、エラーが戻されます (SQLSTATE 42815)。 CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

文字および GRAPHIC ストリングの長さには、末尾ブランクが含まれます。可変長ストリングの長さは、最大長ではなく実際の長さです。

例:

- NAME が VARCHAR(128) 列であり、Unicode UTF-8 でエンコードされ、値 'Jürgen' を含んでいるとします。以下の 2 つの照会は、値 6 を戻します。

```
SELECT CHARACTER_LENGTH(NAME, CODEUNITS32)
FROM T1 WHERE NAME = 'Jürgen'

SELECT CHARACTER_LENGTH(NAME, CODEUNITS16)
FROM T1 WHERE NAME = 'Jürgen'
```

  以下の 2 つの照会は、値 7 を戻します。

```
SELECT CHARACTER_LENGTH(NAME, OCTETS)
FROM T1 WHERE NAME = 'Jürgen'

SELECT LENGTH(NAME)
FROM T1 WHERE NAME = 'Jürgen'
```

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽のト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'
UTF-32BE	X'0001D11E'	X'0000004E'	X'00000303'	X'00000041'	X'00000042'

変数 UTF8_VAR に、ストリングの UTF-8 表現が格納されると想定します。

```
SELECT CHARACTER_LENGTH(UTF8_VAR, CODEUNITS16),
  CHARACTER_LENGTH(UTF8_VAR, CODEUNITS32),
  CHARACTER_LENGTH(UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

それぞれ、値 6、5、9 を戻します。

変数 UTF16_VAR に、ストリングの UTF-16BE 表現が格納されると想定します。

```
SELECT CHARACTER_LENGTH(UTF16_VAR, CODEUNITS16),
   CHARACTER_LENGTH(UTF16_VAR, CODEUNITS32),
   CHARACTER_LENGTH(UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

それぞれ、値 6、5、12 を戻します。

CHR

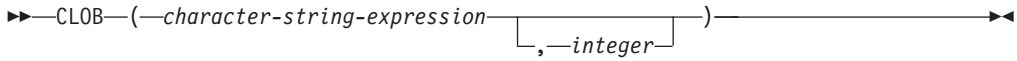
スキーマは SYSFUN です。

引数で指定される ASCII コード値の文字を戻します。 *expression* が 0 であれば、結果はブランク文字になります (X'20')。

引数は INTEGER または SMALLINT のいずれかです。引数の値は 0 から 255 の範囲でなければなりません。そうでない場合、戻り値は 255 に対応する ASCII コード値の文字になります。

関数の結果は CHAR(1) です。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

CLOB



スキーマは SYSIBM です。

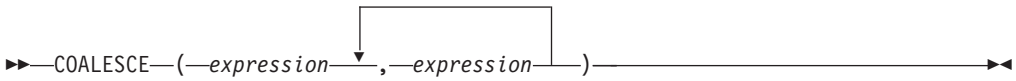
CLOB 関数は、文字ストリング・タイプの CLOB 表記を戻します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

character-string-expression
文字ストリングである値を戻す式。式を FOR BIT DATA として定義される文字ストリングとすることはできません (SQLSTATE 42846)。

integer
結果の CLOB データ・タイプの長さ属性を指定する整数値。値は 0 から 2,147,483,647 の範囲でなければなりません。 *integer* の値を指定しない場合、結果の長さは、最初の引数の長さと同じになります。

関数の結果は CLOB です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

COALESCE



スキーマは SYSIBM です。

COALESCE は、その値が NULL 値以外の最初の引数を戻します。

引数は指定された順序で評価され、関数の結果は NULL 値以外の最初の引数になります。結果は、すべての引数が NULL 値の場合にのみ NULL 値になります。選択された引数は、必要に応じて結果の属性に変換されます。

引数は互いに互換性がなければなりません。また、組み込みまたはユーザー定義のどちらのデータ・タイプでもかまいません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。)

例:

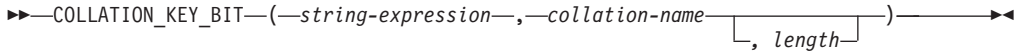
- DEPARTMENT 表のすべての行のすべての値を選択する場合に、部門の管理者 (MGRNO) が欠落しているときには (つまり NULL 値なら)、 'ABSENT' という値を戻すようにします。

```
SELECT DEPTNO, DEPTNAME, COALESCE(MGRNO, 'ABSENT'), ADMRDEPT
  FROM DEPARTMENT
```

- EMPLOYEE 表のすべての行から従業員番号 (EMPNO) と給与 (SALARY) を選択する場合に、給与が欠落していれば (つまり NULL 値なら)、値としてゼロを戻すようにします。

```
SELECT EMPNO, COALESCE(SALARY, 0)
  FROM EMPLOYEE
```

COLLATION_KEY_BIT



スキーマは SYSIBM です。

COLLATION_KEY_BIT 関数によって、指定した *collation-name* 内の *string-expression* の照合キーを表す VARCHAR FOR BIT DATA ストリングが返されます。

2 つのストリングの COLLATION_KEY_BIT の結果では、指定した *collation-name* 内のそれらの順序を判別するためのバイナリー比較を行うことができます。比較が意味を持つためには、使用される結果が同じ *collation-name* に基づいている必要があります。

string-expression
照合キーが判別される対象の CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC ストリングを返す式。 *string-expression* が CHAR または VARCHAR の場合、式を FOR BIT DATA にしないでください (SQLSTATE 429BM)。

string-expression が UTF-16 でない場合、この関数では *string-expression* の UTF-16 へのコード・ページ変換が実行されます。コード・ページ変換の結果に少なくとも 1 つの置換文字が含まれている場合は、この関数により 1 つ以上の置換文字を持つ UTF-16 ストリングの照合キーが返され、SQLCA 内の警告標識 SQLWARN8 が 'W' に設定されます。

collation-name
照合キーを判別する際に使用する照合を指定する文字定数。 *collation-name* の値には大文字と小文字の区別がなく、またこの値は 「*国際化対応ガイド*」の『Unicode 照合アルゴリズムに基づく照合』または「*国際化対応ガイド*」の『Unicode データの言語対応型の照合』に記載されているものの 1 つである (SQLSTATE 42616) 必要があります。

length
結果の長さ属性をバイト単位で指定する式。指定した場合は、*length* は 1 から 32 672 の範囲の整数である必要があります (SQLSTATE 42815)。

length の値を指定しない場合、結果の長さは以下のように決定されます。

表 32. 結果の長さの決定

ストリング引数のデータ・タイプ	結果のデータ・タイプの長さ
CHAR(*n*) または VARCHAR(*n*)	12*n* バイトと 32 672 バイトのうちの最小値
GRAPHIC(*n*) または VARGRAPHIC(*n*)	12*n* バイトと 32 672 バイトのうちの最小値

length を指定したかどうかに関係なく、照合キーの長さが結果のデータ・タイプの長さより長い場合は、エラー (SQLSTATE 42815) が戻ります。照合キーの実際の結果の長さは、UTF-16 への変換後は *string-expression* の長さの約 6 倍です。

string-expression が空ストリングの場合は、結果はゼロ以外の長さを持つ可能性がある有効な照合キーです。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

次の照会では、ドイツ語の言語認識照合をコード・ページ 923 で使用して、従業員が姓で順序付けられます。

```
SELECT FIRSTNME, LASTNAME
  FROM EMPLOYEE
  ORDER BY COLLATION_KEY_BIT (LASTNAME, 'SYSTEM_923_DE')
```

次の照会では、ケベック州にある従業員の部門を検索するために言語文化的に正しい比較が使用されます。

```
SELECT E.WORKDEPT
  FROM EMPLOYEE AS E INNER JOIN SALES AS S
  ON COLLATION_KEY_BIT(E.LASTNAME, 'UCA400R1_LFR') =
    COLLATION_KEY_BIT(S.SALES_PERSON, 'UCA400R1_LFR')
  WHERE S.REGION = 'Quebec'
```

COMPARE_DECFLOAT

スキーマは SYSIBM です。

COMPARE_DECFLOAT 関数は、2 つの引数が等しいか順不同であるか、あるいは一方の引数が他方より大きいかどうかを示す SMALLINT 値を戻します。

expression1
組み込み数値データ・タイプの値を返す式。引数が DECFLOAT(34) ではない場合、処理のために DECFLOAT(34) に論理的に変換されます。

expression2
組み込み数値データ・タイプの値を返す式。引数が DECFLOAT(34) ではない場合、処理のために DECFLOAT(34) に論理的に変換されます。

expression1 の値は *expression2* の値と比較され、以下の規則に従って結果が戻されます。

- 両方の引数が有限である場合、比較は代数的であり、10 進浮動小数点減算の手順に従います。差異がいずれかの符号のちょうどゼロである場合、2 つの引数は等価になります。 ゼロ以外の差異が正である場合、最初の引数は 2 番目の引数より大きくなります。ゼロ以外の差異が負である場合、最初の引数は 2 番目の引数より小さくなります。
- 正のゼロおよび負のゼロは等しいものとして比較されます。
- 正の無限大は、正の無限大と等しいものとして比較されます。
- 正の無限大は、すべての有限数値より大きいものとして比較されます。
- 負の無限大は、負の無限大と等しいものとして比較されます。
- 負の無限大は、すべての有限数値より小さいものとして比較されます。
- 数値比較は厳密に行われます。結果は範囲と精度が無制限であるかのように、有限のオペランドに対して判別されます。オーバーフロー状態またはアンダーフロー状態が発生することはありません。
- どちらかの引数が NaN または sNaN (正または負) である場合、結果は順不同です。

結果値は、次のようになります。

- 0: 引数が厳密に等しい場合
- 1: *expression1* が *expression2* より小さい場合
- 2: *expression1* が *expression2* より大きい場合
- 3: 引数が順不同である場合

この関数の結果は SMALLINT 値となります。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

例:

- 以下の例は、さまざまな 10 進浮動小数点値の入力を与えられた場合に COMPARE_DECFLOAT 関数によって戻される値を示しています。

```
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(2.17)) = 0
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(2.170)) = 2
COMPARE_DECFLOAT(DECFLOAT(2.170), DECFLOAT(2.17)) = 1
COMPARE_DECFLOAT(DECFLOAT(2.17), DECFLOAT(0.0)) = 2
COMPARE_DECFLOAT(INFINITY,INFINITY) = 0
COMPARE_DECFLOAT(INFINITY,-INFINITY) = 2
COMPARE_DECFLOAT(DECFLOAT(-2),INFINITY) = 1
COMPARE_DECFLOAT(NAN,NAN) = 3
COMPARE_DECFLOAT(DECFLOAT(-0.1),SNAN) = 3
```

CONCAT

注:

1 CONCAT の同義語として || を使用できます。

スキーマは SYSIBM です。

2 つのストリング引数を連結した値を戻します。この 2 つの引数のタイプには互換性がなければなりません。

関数の結果はストリングで、長さは、2 つの引数の長さの合計です。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

COS

スキーマは SYSIBM です。 (COS 関数の SYSFUN バージョンは引き続き使用可能です。)

引数に対するコサイン (余弦) の値を戻します。引数はラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

COSH

スキーマは SYSIBM です。

引数に対する双曲線コサイン (余弦) の値を戻します。引数はラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

COT

スキーマは SYSIBM です。 (COT 関数の SYSFUN バージョンは引き続き使用可能です。)

引数に対するコタンジェント (余接) の値を戻します。引数はラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

DATAPARTITIONNUM

スキーマは SYSIBM です。

DATAPARTITIONNUM 関数は、行が置かれているデータ・パーティションのシーケンス番号 (SYSDATAPARTITIONS.SEQNO) を戻します。データ・パーティションは範囲別にソートされ、シーケンス番号は 0 から始まります。たとえば、範囲が最低のデータ・パーティションに置かれている行の場合、DATAPARTITIONNUM 関数から 0 が戻されます。

引数は、表内の任意の列の修飾された名前または無修飾の名前でなければなりません。行レベルの情報が戻されるので、どの列が指定されるかに関係なく、結果は同じです。該当の列は、どのようなデータ・タイプであっても構いません。

column-name がビューの列を参照する場合、そのビューの列の式は、基礎となる基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

結果のデータ・タイプは INTEGER であり、NULL 値にはなりません。

この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

DATAPARTITIONNUM 関数は、チェック制約内、または生成列の定義で使用することはできません (SQLSTATE 42881)。 DATAPARTITIONNUM 関数は、マテリアライズ照会表 (MQT) 定義の中でも使用できません (SQLSTATE 428EC)。

例:

- ```
  SELECT DATAPARTITIONNUM (EMPNO)
  FROM EMPLOYEE
  ```

  DATAPARTITIONNUM によって戻されたシーケンス番号 (たとえば 0) を、他の SQL ステートメント (たとえば、ALTER TABLE...DETACH PARTITION) 内で使用できるデータ・パーティション名に変換するときは、SYSCAT.DATAPARTITIONS カタログ・ビューを照会することができます。以下の例で説明されているとおり、DATAPARTITIONNUM から取得された SEQNO を WHERE 節に組み込みます。

  ```
  SELECT DATAPARTITIONNAME
  FROM SYSCAT.DATAPARTITIONS
  WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
  ```

  上記の結果は、値 'PART0' になります。
  ```

DATE

スキーマは SYSIBM です。

DATE 関数は、値から日付を戻します。

引数は、日付、タイム・スタンプ、3652059 以下の正の整数、日付またはタイム・スタンプの有効なストリング表記、または CLOB、LONG VARCHAR、DBCLOB、または LONG VARGRAPHIC ではない長さ 7 文字のストリングのいずれかでなければなりません。

Unicode データベースだけが、日付またはタイム・スタンプの GRAPHIC ストリング表現である引数をサポートします。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

引数が長さ 7 のストリングの場合、 *yyyynnn* という形式の有効な日付を表していなければなりません。ここで、*yyyy* は年を示す数字、 *nnn* は年間通算日を示す 001 から 366 までの数字です。

関数の結果は日付です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果はその値の日付部分です。
- 引数が数値の場合
  - 結果は、0001 (1 月 1 日) から数えて *n* -1 日後の日付です (*n* は数字の整数部分)。
- 引数が長さ 7 のストリングの場合
  - 結果は、そのストリングで表された日付になります。

例:

列 RECEIVED (タイム・スタンプ) には、 '1988-12-25-17.12.30.000000' に相当する内部値が入っているものとします。

- 以下の例の結果は、'1988-12-25' の内部表記になります。

```
DATE(RECEIVED)
```

- 以下の例の結果は、'1988-12-25' の内部表記になります。

```
DATE('1988-12-25')
```

- 以下の例の結果は、'1988-12-25' の内部表記になります。

```
DATE('25.12.1988')
```

- 以下の例の結果は、'0001-02-04' の内部表記になります。

```
DATE(35)
```

DAY

スキーマは SYSIBM です。

DAY 関数は、値の日の部分を戻します。

引数は、日付、タイム・スタンプ、日付期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の日の部分 (1 から 31 の整数) になります。
- 引数が日付期間またはタイム・スタンプ期間の場合
  - 結果は、値の日の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例:

- PROJECT 表を使用して、 WELD LINE PLANNING プロジェクト (PROJNAME) の終了予定日 (PRENDATE) をホスト変数 END_DAY (短精度整数) に設定します。

  ```
  SELECT DAY(PRENDATE)
    INTO :END_DAY
    FROM PROJECT
    WHERE PROJNAME = 'WELD LINE PLANNING'
  ```

  サンプル表を使用した場合、結果として END_DAY は 15 に設定されます。
- 列 DATE1 (日付) には、2000-03-15 に相当する内部値が入っていて、列 DATE2 (日付) には、1999-12-31 に相当する内部値が入っているとします。

  ```
  DAY(DATE1 - DATE2)
  ```

  結果は、15 の値になります。

DAYNAME

スキーマは SYSFUN です。

データベースの開始時点のロケールに基づいて、引数の日の部分の曜日名から成る大文字小文字混合文字ストリング (たとえば、 Friday) を戻します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は VARCHAR(100) です。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

DAYOFWEEK

引数の曜日を 1 から 7 の範囲の整数値として戻します。1 は日曜日を表します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

DAYOFWEEK_ISO

スキーマは SYSFUN です。

引数の曜日を 1 から 7 の範囲の整数値として戻します。1 は月曜日を表します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

DAYOFYEAR

スキーマは SYSFUN です。

引数の年間通算日を、1 から 366 の範囲の整数値として戻します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

DAYS

スキーマは SYSIBM です。

DAYS 関数は、日付の整数表記を戻します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

結果は、1 月 1 日 (0001) から *D* までの日数に、 1 を加えた数になります (*D* は、引数に DATE 関数を適用した場合の結果となる日付)。

例:

- PROJECT 表を使用して、プロジェクト (PROJNO) 'IF2000' に要する見積日数 (PRENDATE - PRSTDATE) をホスト変数 EDUCATION_DAYS (整数) に設定します。

  ```
  SELECT DAYS(PRENDATE) - DAYS(PRSTDATE)
    INTO :EDUCATION_DAYS
    FROM PROJECT
    WHERE PROJNO = 'IF2000'
  ```

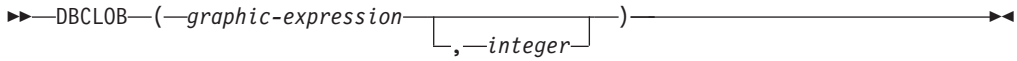
  結果として EDUCATION_DAYS は 396 に設定されます。

- PROJECT 表を使用して、ホスト変数 TOTAL_DAYS (int) に、部署 (DEPTNO) 'E21' のすべてのプロジェクトについての経過日数見積もり (PRENDATE - PRSTDATE) の合計を設定します。

  ```
  SELECT SUM(DAYS(PRENDATE) - DAYS(PRSTDATE))
    INTO :TOTAL_DAYS
    FROM PROJECT
    WHERE DEPTNO = 'E21'
  ```

  サンプル表を使用した場合、結果として TOTAL_DAYS は 1584 に設定されます。

DBCLOB



スキーマは SYSIBM です。

DBCLOB 関数は、GRAPHIC ストリング・タイプの DBCLOB 表記を戻します。

Unicode データベースでは、指定した引数が文字ストリングであると、まず GRAPHIC ストリングに変換されてから、関数が実行されます。最後の文字が高サロゲートになるように出力ストリングが切り捨てられた場合、そのサロゲートは次のいずれかになります。

- 指定した引数が文字ストリングの場合は現状のままになる。
- 指定した引数が GRAPHIC ストリングの場合はブランク文字 (X'0020') に変換される。

今後のリリースでこの動作は変更される可能性があります。

関数の結果は DBCLOB です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

graphic-expression
GRAPHIC ストリング値を戻す式。

integer
結果の DBCLOB データ・タイプの長さ属性を指定する整数値。値は 0 から 1 073 741 823 の範囲でなければなりません。 *integer* を指定しない場合、結果の長さは、最初の引数の長さと同じになります。

DBPARTITIONNUM

スキーマは SYSIBM です。

DBPARTITIONNUM 関数は、行のデータベース・パーティション番号を戻します。たとえば、SELECT 節で使用すると、結果セット内の各行のデータベース・パーティション番号を戻します。

引数は、表内の任意の列の修飾された名前または無修飾の名前でなければなりません。行レベルの情報が戻されるので、どの列が指定されるかに関係なく、結果は同じです。該当の列は、どのようなデータ・タイプであっても構いません。

column-name がビューの列を参照する場合、そのビューの列の式は、基礎となる基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

DBPARTITIONNUM 関数によってデータベース・パーティション番号が戻される特定の行 (および表) は、この関数を使用する SQL ステートメントのコンテキストから判別されます。

遷移変数および表に戻されるデータベース・パーティション番号は、分散キー列の現行遷移値から得られます。たとえば、挿入前トリガーにおいて、新しい遷移変数の現行値があれば、関数は予想データベース・パーティション番号を戻します。ただし、分散キー列の値はそれ以後の挿入前トリガーによって変更される場合があります。したがって、データベースに挿入される時点での行の最終データベース・パーティション番号は、予測値とは異なるかもしれません。

結果のデータ・タイプは INTEGER であり、NULL 値にはなりません。
`db2nodes.cfg` ファイルがない場合、結果は 0 になります。

この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

DBPARTITIONNUM 関数は、複製された表、チェック制約内、または生成された列の定義で使用することはできません (SQLSTATE 42881)。

以前のバージョンのDB2 との互換性: DBPARTITIONNUM の代わりに NODENUMBER を指定できます。

例:

- EMPLOYEE 表内の指定された従業員の行が、DEPARTMENT 表内の従業員の部門についての記述とは異なるデータベース・パーティションにあるインスタンス数をカウントします。

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- 2 つの表の行が同じデータベース・パーティションにあるようにするため、EMPLOYEE および DEPARTMENT の表を結合します。

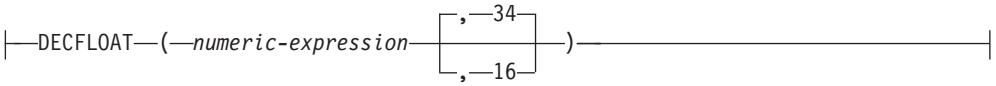
```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- EMPLOYEE 表で BEFORE トリガーを使用して、EMPINSERTLOG1 という表に、EMPLOYEE 表の従業員番号と新しい行の予想データベース・パーティション番号を記録します。

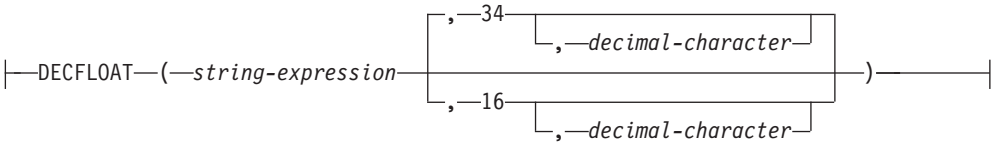
```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

DECFLOAT

数値から 10 進浮動小数点数へ:



文字から 10 進浮動小数点数へ:



スキーマは SYSIBM です。

DECFLOAT 関数は、数値の 10 進浮動小数点表記、または数値のストリング表記を戻します。

numeric-expression
組み込み数値データ・タイプの値を返す式。

string-expression
長さが文字定数の最大長 (4000 バイト) 以下の文字ストリング値を戻す式。CLOB または LONG VARCHAR データ・タイプにすることはできません。先行ブランクと末尾のブランクは、ストリングから除去されます。この結果のサブストリングは、SQL 整数、10 進数、浮動小数点数、または 10 進浮動小数点定数を形成するための規則に準拠していなければならず (SQLSTATE 22018)、さらに 42 バイト以下でなければなりません (SQLSTATE 42820)。

34 または 16
結果の精度の桁数を指定します。デフォルトは 34 です。

decimal-character
character-expression の小数部分と整数部分とを区切るために使用する 1 バイト文字定数を指定します。この文字には、数字、プラス (+)、マイナス (-)、またはブランクを使用できず、 *character-expression* の中に最高で 1 回しか使用することができません。

結果は、CAST(*string-expression* AS DECFLOAT(*n*)) または CAST(*numeric-expression* AS DECFLOAT(*n*)) の結果と同じ数値になります。先行ブランクと末尾のブランクは、ストリングから除去されます。

関数の結果は、精度の桁数を暗黙的または明示的に指定した 10 進浮動小数点数になります。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL であれば、結果は NULL 値になります。

必要な場合は、ソースはターゲットの精度に丸められます。 CURRENT DECFLOAT ROUNDING MODE 特殊レジスターは、丸めモードを決定します。

注: アプリケーションの移植性を高めるには、CAST の指定を使用します。

注: すべての数値は、整数、10 進数、または浮動小数点定数として解釈されてから、10 進浮動小数点数にキャストされます。浮動小数点定数を使用すると、丸めエラーの原因となるので、使用しないように強くお勧めします。代わりに、ストリングから、10 進浮動小数点数バージョンの DECFLOAT 関数を使用してください。

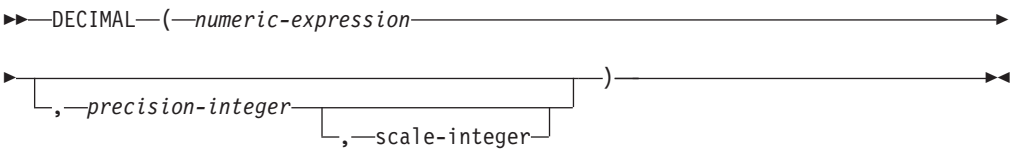
例:

- EMPLOYEE 表の EDLEVEL 列 (データ・タイプ = SMALLINT) の選択リストに DECFLOAT データ・タイプが必ず戻されるように EDLEVEL 関数を使用します。選択リストには、EMPNO 列も入れておいてください。

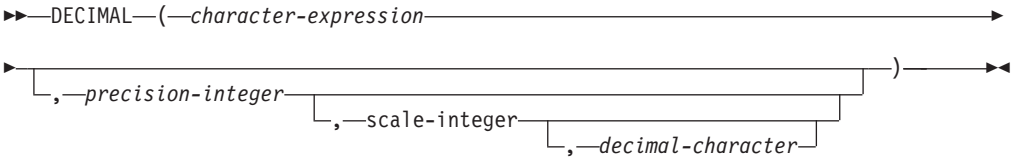
```
SELECT EMPNO, DECFLOAT(EDLEVEL,16)
FROM EMPLOYEE
```

DECIMAL

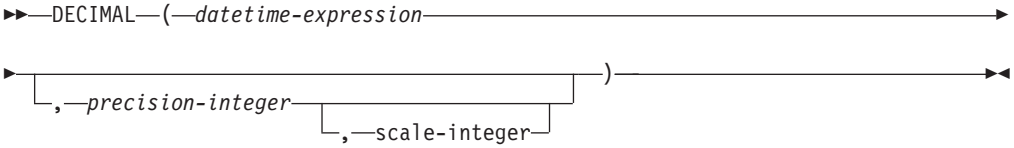
数値→ 10 進数:



文字→ 10 進数:



日付/時刻→ 10 進数:



スキーマは SYSIBM です。

DECIMAL 関数は、以下の 10 進表記を戻します。

- 数値
- 10 進数の文字ストリング表記
- 整数の文字ストリング表記
- 浮動小数点数の文字ストリング表記
- 10 進浮動小数点数の文字ストリング表記
- 日付/時刻 (引数が日付、時刻、またはタイム・スタンプの場合)

Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は、精度 *p*、位取り *s* の 10 進数になります (*p* と *s* はそれぞれ 2 番目と 3 番目の引数)。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL であれば、結果は NULL 値になります。

数値→ 10 進数

numeric-expression
数値データ・タイプの値を戻す式。

precision-integer
1 から 31 の範囲の値の整数定数。

precision-integer のデフォルト値は、 *numeric-expression* のデータ・タイプによって異なります。

- 10 進浮動小数点数の場合は 31
- 浮動小数点および 10 進数の場合は 15
- 64 ビット整数の場合は 19
- 長精度整数の場合は 11
- 短精度整数の場合は 5

scale-integer
0 から *precision-integer* の値までの範囲の整数定数。デフォルトはゼロです。

結果は、最初の引数が精度 *p*、位取り *s* の 10 進数の列または変数に割り当てられた場合と同じ数になります (*p* と *s* は、それぞれ 2 番目と 3 番目の引数)。数値の整数部分を表すために必要な有効 10 進桁数が、 *p* - *s* より大きい場合はエラーになります。

文字→ 10 進数

character-expression
長さが文字定数の最大長 (4000 バイト) 以下の文字ストリングである値を戻す式。 CLOB または LONG VARCHAR データ・タイプにすることはできません。先行ブランクと末尾のブランクは、ストリングから除去されます。この結果のサブストリングは、 SQL 整数または 10 進定数を形成するための規則に準拠していなければなりません (SQLSTATE 22018)。

定数 *decimal-character* のコード・ページに一致させるために必要であれば、 *character-expression* はデータベース・コード・ページに変換されます。

precision-integer
結果の精度を指定する整数定数 (値の範囲は 1 から 31)。この指定がない場合のデフォルト値は 15 です。

scale-integer
結果の位取りを指定する整数定数 (値の範囲は 0 から *precision-integer*)。 この指定がない場合のデフォルト値は 0 です。

decimal-character
character-expression の小数部分と整数部分とを区切るために使用する 1 バイト文字定数を指定します。この文字には、数字、プラス (+)、マイナス (-)、またはブランクを使用できず、 *character-expression* の中に最高で 1 回しか使用することができません (SQLSTATE 42815)。

結果は、精度 *p*、位取り *s* の 10 進数になります (*p* と *s* は、それぞれ 2 番目と 3 番目の引数)。小数点より右側の数字の桁数が、位取りより多い場合、 10 進数の終わりから数字が切り捨てられます。 *character-expression* の小数点文字の左側にある有効数字 (数値の整数部分) の桁数が *p* - *s* よりも多い場合は、エラーになります (SQLSTATE 22003)。*decimal-character* 引数に別の値が指定されている場合、サブストリングに使われているデフォルトの小数点文字は無効になります (SQLSTATE 22018)。

日付/時刻→ 10 進数

datetime-expression
次のデータ・タイプのいずれかの式。

- DATE。結果は、日付を *yyyymmdd* で表した DECIMAL(8,0) 値になります。
- TIME。結果は、時間を *hhmmss* で表した DECIMAL(6,0) 値になります。
- TIMESTAMP。結果は、タイム・スタンプを *yyyymmddhhmmss.nnnnnn* で表した DECIMAL(20,6) 値になります。

この関数を使ってユーザーは、精度を指定したり、精度と位取りを指定したりすることができます。ただし、精度を指定しないと、位取りを指定することはできません。 (precision,scale) のデフォルト値は、DATE の場合は (8,0)、 TIME の場合は (6,0)、TIMESTAMP の場合は (20,6) です。

結果は、精度 *p*、位取り *s* の 10 進数になります (*p* と *s* は、それぞれ 2 番目と 3 番目の引数)。小数点より右側の数字の桁数が、位取りより多い場合、終わりから数字が切り捨てられます。 *datetime-expression* の小数点文字の左側にある有効数字 (数値の整数部分) の桁数が *p* - *s* よりも多い場合は、エラーになります (SQLSTATE 22003)。

DECIMAL の代わりに DEC を指定できます。

例:

- EMPLOYEE 表の EDLEVEL 列 (データ・タイプ = SMALLINT) の選択リストに (精度が 5 で、位取りが 2 の) DECIMAL データ・タイプが必ず戻されるように DECIMAL 関数を使用します。選択リストには、EMPNO 列も入れておいてください。

```
SELECT EMPNO, DECIMAL(EDLEVEL,5,2)
  FROM EMPLOYEE
```

- ホスト変数 PERIOD のタイプが INTEGER であるとします。その値を日付期間として使用するためには、それを decimal(8,0) として「キャスト」する必要があります。

```
SELECT PRSTDATE + DECIMAL(:PERIOD,8)
  FROM PROJECT
```

- SALARY 列の更新内容が、小数点文字にコンマを使用した文字ストリングとして、ウィンドウから入力されるものとします (ユーザーは、たとえば 21400,50 と入力します)。アプリケーションが妥当性検査を行った後、これを、CHAR(10) として定義されたホスト変数 newsalary に設定します。

```
UPDATE STAFF
  SET SALARY = DECIMAL(:newsalary, 9, 2, ',')
  WHERE ID = :empid;
```

  newsalary の値は 21400.50 になります。

- 値にデフォルト値の小数点文字 (.) を追加します。

```
DECIMAL('21400,50', 9, 2, '.')
```

  この例では、小数点文字としてピリオド (.) を指定しているのに、第 1 引数の中で区切り文字としてコンマ (,) が使われているため、エラーになります。

- STARTING (time) 列に、 '12:10:00' に相当する内部値が入っていると想定します。

  ```
  DECIMAL(STARTING)
  ```

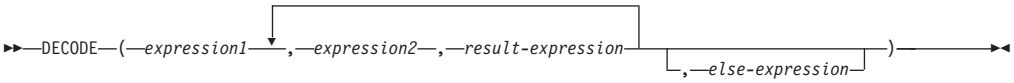
  結果は、121 000 の値になります。
- RECEIVED (timestamp) 列に、 '1988-12-22-14.07.21.136421' に相当する内部値が入っていると想定します。

  ```
  DECIMAL(RECEIVED)
  ```

  結果は、19 881 222 140 721.136421 の値になります。
- 以下の表は、各種の日時入力値の 10 進数結果とその結果の精度と位取りを示しています。

DECIMAL(引数)	精度と位取り	結果
DECIMAL(2000-03-21)	(8,0)	20000321
DECIMAL(2000-03-21, 10)	(10,0)	20000321
DECIMAL(2000-03-21, 12, 2)	(12,2)	20000321.00
DECIMAL(12:02:21)	(6,0)	120221
DECIMAL(12:02:21, 10)	(10,0)	120221
DECIMAL(12:02:21, 10, 2)	(10,2)	120221.00
DECIMAL(2000-03-21-12.02.21.123456)	(20, 6)	20000321120221.123456
DECIMAL(2000-03-21-12.02.21.123456, 23)	(23, 6)	20000321120221.123456
DECIMAL(2000-03-21-12.02.21.123456, 23, 4)	(23, 4)	20000321120221.1234

DECODE



スキーマは SYSIBM です。

DECODE 関数は、それぞれの *expression2* を *expression1* と比較します。 *expression1* が *expression2* と等しいか、または *expression1* と *expression2* の両方が NULL の場合は、その次の *result-expresssion* の値が返されます。 *expression2* が *expression1* に一致しない場合、*else-expression* の値が返されます。これら以外の場合は、NULL 値が返されます。

DECODE 関数は、NULL 値の処理を除いては、CASE 式と似ています。

- *expression1* の NULL 値は、対応する *expression2* の NULL 値と一致します。
- NULL キーワードが DECODE 関数内で引数として使用される場合、それは適切なデータ・タイプへのキャストでなければなりません。

DECODE 式の結果タイプを決定する規則は、対応する CASE 式に基づいています。

例:

次のような DECODE 式を考慮します。

```
DECODE (c1, 7, 'a', 6, 'b', 'c')
```

これは以下の CASE 式と同じ結果となります。

```
CASE c1
  WHEN 7 THEN 'a'
  WHEN 6 THEN 'b'
  ELSE 'c'
END
```

同様に、次のような DECODE 式を考慮します。

```
DECODE (c1, var1, 'a', var2, 'b')
```

c1、var1、および var2 を NULL 値にできる場合、以下の CASE 式と同じ結果になります。

```
CASE
  WHEN c1 = var1 OR (c1 IS NULL AND var1 IS NULL) THEN 'a'
  WHEN c1 = var2 OR (c1 IS NULL AND var2 IS NULL) THEN 'b'
  ELSE NULL
END
```

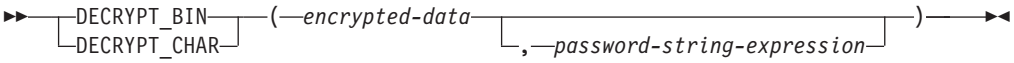
次のような照会について考慮します。

```
SELECT ID, DECODE(STATUS, 'A', 'Accepted',
                          'D', 'Denied',
                          CAST(NULL AS VARCHAR(1)), 'Unknown',
                          'Other')
FROM CONTRACTS
```

次に示すのは、CASE 式を使用した同じステートメントです。

```
SELECT ID,
  CASE
    WHEN STATUS = 'A' THEN 'Accepted'
    WHEN STATUS = 'D' THEN 'Denied'
    WHEN STATUS IS NULL THEN 'Unknown'
    ELSE 'Other'
  END
FROM CONTRACTS
```

DECRYPT_BIN および DECRYPT_CHAR



スキーマは SYSIBM です。

DECRYPT_BIN 関数と DECRYPT_CHAR 関数はどちらも、 *encrypted-data* 暗号化解除の結果である値を返します。暗号化解除に使用されるパスワードは、*password-string-expression*、または SET ENCRYPTION PASSWORD ステートメントで割り当てられた暗号化パスワード値のいずれかです。 DECRYPT_BIN および DECRYPT_CHAR 関数は、 ENCRYPT 関数を使って暗号化された値のみを暗号化解除できます (SQLSTATE 428FE)。

encrypted-data
CHAR FOR BIT DATA 値または VARCHAR FOR BIT DATA 値を、暗号化された完全なデータ・ストリングとして戻す式です。データ・ストリングは、ENCRYPT 関数を使って暗号化されたものでなければなりません。

password-string-expression
少なくとも 6 バイトで 127 バイトを超えない CHAR または VARCHAR 値を返す式です (SQLSTATE 428FC)。この式は、データの暗号化に使用されたパスワードと同じでなければなりません (SQLSTATE 428FD)。パスワード引数が NULL、または与えられていない場合、SET ENCRYPTION PASSWORD ステートメントによってセッションに割り当てられた暗号化パスワード値を使用してデータが暗号化解除されます (SQLSTATE 51039)。

DECRYPT_BIN 関数の結果は VARCHAR FOR BIT DATA です。DECRYPT_CHAR 関数の結果は VARCHAR です。 *encrypted-data* にヒントが組み込まれている場合、そのヒントは関数によって返されません。結果の長さ属性は、*encrypted-data* のデータ・タイプの長さマイナス 8 バイトになります。関数によって実際に返される長さの値は、暗号化されたオリジナル・ストリングの長さに一致します。暗号化ストリングを超えるバイトが *encrypted-data* に入っている場合、それらのバイトは関数によって返されません。

最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL の場合には、結果も NULL 値です。

データが暗号化されたときのコード・ページとは異なるコード・ページを使用する、別のシステムでデータが暗号化解除された場合、暗号化解除された値をデータベース・コード・ページに変換するとき、長さが超過してしまう可能性があります。この場合、*encrypted-data* 値をより大きなバイト数の VARCHAR ストリングに cast する必要があります。

例:

- SET ENCRYPTION PASSWORD ステートメントを使用して、セッションの暗号化パスワードを設定します。

```
CREATE TABLE EMP (SSN VARCHAR(24) FOR BIT DATA);
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
SELECT DECRYPT_CHAR(SSN)
  FROM EMP;
```

この照会は、値 '289-46-8832' を返します。

- 暗号化パスワードを明示的に渡します。

```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832','Ben123','');
SELECT DECRYPT_CHAR(SSN,'Ben123')
  FROM EMP;
```

この照会は、値 '289-46-8832' を返します。

DEGREES

スキーマは SYSIBM です。 (DEGREES 関数の SYSFUN バージョンは引き続き使用可能です。)

DEGREES 関数は、引数の度単位の角度を戻します。引数はラジアン単位の角度です。

引数は、任意の組み込み数値データ・タイプにすることができます。引数が 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で実行されます。それ以外の場合は、関数による処理のために引数が倍精度浮動小数点数に変換されます。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点数になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例:

- RAD は、3.142 の値を持つ DECIMAL(4,3) ホスト変数であると仮定します。

```
VALUES DEGREES(:RAD)
```

  これは概算値 180.0 を戻します。

DEREF

DEREF 関数は引数のターゲット・タイプのインスタンスを戻します。

引数は、有効範囲が定義された参照データ・タイプが指定されている任意の値にすることができます (SQLSTATE 428DT)。

結果の静的データ・タイプは、引数のターゲット・タイプです。結果の動的データ・タイプは、引数のターゲット・タイプのサブタイプです。結果は NULL 値の場合もあります。 *expression* が NULL 値か、または *expression* が突き合わせする OID がターゲット表にない参照の場合、結果は NULL 値になります。

結果は参照のターゲット・タイプのサブタイプのインスタンスです。結果は参照値と突き合わせするオブジェクト ID がある参照の、ターゲット表またはターゲット・ビューの行を検出することにより決定されます。この行のタイプによって結果の動的タイプが決定されます。結果のタイプがターゲット表の副表の行またはターゲット・ビューのサブビューの行に基づく場合があるため、ステートメントの許可 ID にはターゲット表とその副表のすべて、またはターゲット・ビューとそのサブビューのすべての SELECT 特権が必要です (SQLSTATE 42501)。

例:

EMPLOYEE はタイプ EMP の表であり、そのオブジェクト ID 列は EMPID であるとします。次の照会は、EMPLOYEE 表 (およびその副表) の行ごとに、タイプ EMP (またはそのサブタイプのいずれか) のオブジェクトを戻します。この照会では、EMPLOYEE およびその副表すべてに対する SELECT 権限が必要です。

```
SELECT DEREF(EMPID) FROM EMPLOYEE
```

DIFFERENCE

スキーマは SYSFUN です。

ストリングへの SOUNDEX 関数の適用に基づいて、 2 つのストリングの音の差を示す 0 から 4 の値を戻します。 4 の値は可能な限り最良の音の一致を示します。

引数は、CHAR または VARCHAR いずれかの文字ストリング (4000 バイトを超えない) にすることができます。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。関数は、渡されるデータが UTF-8 でエンコードされている場合でも、ASCII 文字であるものとして解釈します。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例:

```
VALUES (DIFFERENCE('CONSTRAINT','CONSTANT'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONSTANT')),
(DIFFERENCE('CONSTRAINT','CONTRITE'),SOUNDEX('CONSTRAINT'),
SOUNDEX('CONTRITE'))
```

この例では、以下を戻します。

```
1           2    3
----------- ---- ----
          4 C523 C523
          2 C523 C536
```

最初の行で、SOUNDEX の語は同じ結果になりますが、2 行目の語は類似性があるにすぎません。

DIGITS

スキーマは SYSIBM です。

DIGITS 関数は、数値の文字ストリング表記を戻します。

引数は、SMALLINT、 INTEGER、 BIGINT または DECIMAL の型の値を戻す式でなければなりません。

引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

この関数の結果は、引数の位取りに関係なく、引数の絶対値を表す固定長文字ストリングになります。結果には、符号も小数点文字も示されません。結果は、必要に応じてストリングを埋めるための先行ゼロの付いた数字だけで構成されます。ストリングの長さは次のとおりです。

- 引数が短精度整数 (small integer) の場合は 5
- 引数が長精度整数 (large integer) の場合は 10
- 引数が 64 ビット整数 (big integer) の場合は 19
- 引数が精度 *p* の 10 進数の場合は *p*

例:

- 表 TABLEX に、INTCOL という INTEGER 列があり、その値が 10 桁の数値であるとします。列 INTCOL に入っている最初の 4 桁の数字からなる、異なる 4 文字の組み合わせすべてのリストを作成します。

  ```
  SELECT DISTINCT SUBSTR(DIGITS(INTCOL),1,4)
    FROM TABLEX
  ```

- COLUMNX のデータ・タイプが DECIMAL(6,2) であり、その値の 1 つが -6.28 であると想定します。この値に対して次の関数を実行すると、

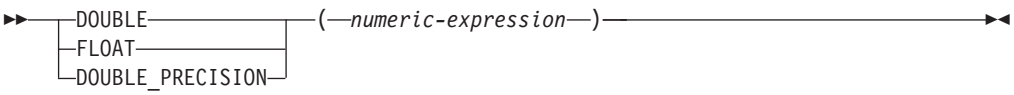
  ```
  DIGITS(COLUMNX)
  ```

  値 '000628' が戻されます。

  この結果は、ストリングをこの長さまで埋めるための先行ゼロの付いた、長さ 6 (列の精度) のストリングになります。符号も小数点文字も結果には示されません。

DOUBLE

数値 → 倍精度:



文字ストリング → 倍精度:

スキーマは SYSIBM です。ただし、DOUBLE(*string-expression*) のスキーマは SYSFUN です。

DOUBLE 関数は、以下に対応する浮動小数点数を戻します。

- 引数が数式の場合は、数値。
- 引数が文字ストリング式の場合は、数値の文字ストリング表記。

Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

数値 → 倍精度

numeric-expression
引数は、組み込み数値データ・タイプの値を返す式です。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

結果は、引数が倍精度浮動小数点の列、または変数に割り当てられた場合の結果と同じ数値になります。

文字ストリング → 倍精度

string-expression
引数のタイプは、数値定数形式の CHAR または VARCHAR にすることができます。引数に先行ブランクや後続ブランクがあっても、それは無視されます。

関数の結果は倍精度浮動小数点数になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

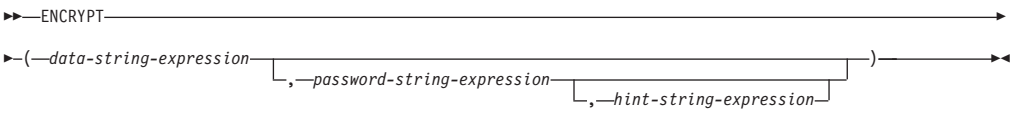
結果は、ストリングが定数と見なされ、倍精度浮動小数点の列または変数に割り当てられる場合の結果と同じ数値になります。

例:

EMPLOYEE 表を使用して、歩合がゼロではない従業員の給与と歩合の比率を計算します。関係する列 (SALARY と COMM) のデータ・タイプは DECIMAL です。結果が範囲外にならないようにするため、DOUBLE を SALARY に適用して、除算が浮動小数点数で実行されるようにします。

```
SELECT EMPNO, DOUBLE(SALARY)/COMM
  FROM EMPLOYEE
  WHERE COMM > 0
```

ENCRYPT



スキーマは SYSIBM です。

ENCRYPT 関数は、*data-string-expression* 暗号化の結果である値を返します。暗号化に使用されるパスワードは、*password-string-expression*、または SET ENCRYPTION PASSWORD ステートメントで割り当てられた暗号化パスワード値のいずれかです。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

data-string-expression
暗号化する CHAR または VARCHAR 値を返す式です。 *data-string-expression* のデータ・タイプの長さ属性は、 *hint-string-expression* 引数がなければ 32663 に、 *hint-string-expression* 引数が指定されていれば 32631 に制限されています (SQLSTATE 42815)。

password-string-expression
少なくとも 6 バイトで 127 バイトを超えない CHAR または VARCHAR 値を返す式です (SQLSTATE 428FC)。この値は、*data-string-expression* を暗号化するために使用されるパスワードを表します。パスワード引数が NULL、または与えられていない場合、SET ENCRYPTION PASSWORD ステートメントによってセッションに割り当てられた暗号化パスワード値を使用してデータが暗号化されます (SQLSTATE 51039)。

hint-string-expression
データ所有者がパスワードを思い出す (たとえば、'Ocean' が 'Pacific' を思い出すヒントになります) ために役立つ、32 バイトまでの CHAR または VARCHAR 値を返す式です。ヒントの値が与えられると、そのヒントは結果に組み込まれ、GETHINT 関数を使用して取り出すことができます。この引数が NULL、または与えられていない場合、ヒントは結果に組み込まれません。

関数の結果データ・タイプは VARCHAR FOR BIT DATA です。

- オプションのヒント・パラメーターが指定されている場合、結果の長さ属性は、暗号化されていないデータの長さ属性 + 8 バイト + 次の 8 バイト境界までのバイト数 + ヒントの長さの 32 バイトと等しくなります。
- オプションのヒント・パラメーターが指定されていない場合、結果の長さ属性は、暗号化されていないデータの長さ属性 + 8 バイト + 次の 8 バイト境界までのバイト数と等しくなります。

最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL の場合には、結果も NULL 値です。

暗号化された結果が *data-string-expression* 値よりも長くなることに注意してください。そのため、暗号化された値を割り当てる際、その暗号化された値全体が入るだけの十分なサイズでターゲットを宣言してください。

注:

- ***暗号化アルゴリズム:*** 内部暗号化アルゴリズムは埋め込み付きの RC2 ブロック暗号で、128 ビットの秘密鍵は MD5 メッセージ・ダイジェストによってパスワードから派生します。
- ***暗号化パスワードおよびデータ:*** パスワードの管理はユーザーの責任です。データが暗号化されると、そのデータを暗号化解除するために使用できるのは、暗号化するときに使用したパスワードだけです (SQLSTATE 428FD)。

  暗号化された結果には、NULL 終止符およびその他の印刷不能文字が組み込まれることがあります。推奨されているデータ長よりも短い長さへの割り当てまたは cast は、暗号化解除の失敗やデータ脱落の原因となります。ブランクは、短すぎる列に保管するときに切り捨てられる有効な暗号化データ値です。
- ***暗号化データの管理:*** 暗号化データは、 ENCRYPT 関数に対応する暗号化解除関数をサポートしているサーバーでのみ暗号化解除できます。そのため、暗号化データの入った列の複製は、 DECRYPT_BIN または DECRYPT_CHAR 関数をサポートしているサーバーで行わなければなりません。

例:

- SET ENCRYPTION PASSWORD ステートメントを使用して、セッションの暗号化パスワードを設定します。

```
CREATE TABLE EMP (SSN VARCHAR(24) FOR BIT DATA);
SET ENCRYPTION PASSWORD = 'Ben123';
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832');
```

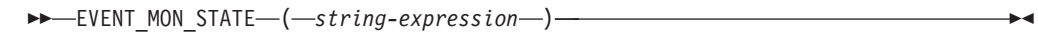
- 暗号化パスワードを明示的に渡します。

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Ben123');
```

- パスワードのヒントを定義します。

```
INSERT INTO EMP(SSN) VALUES ENCRYPT('289-46-8832','Pacific','Ocean');
```

EVENT_MON_STATE



スキーマは SYSIBM です。

EVENT_MON_STATE 関数は、イベント・モニターの現在の状態を戻します。

引数は、結果のタイプが CHAR または VARCHAR で、値がイベント・モニターの名前であるストリング式です。指定したイベント・モニターが SYSCAT.EVENTMONITORS カタログ表にない場合は、 SQLSTATE 42704 が戻されます。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

結果は、次のいずれかの値の整数になります。

0 イベント・モニターは非アクティブ状態です。

1 イベント・モニターはアクティブです。

引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例:

次の例は、定義済みのイベント・モニターすべてを選択して、各モニターがアクティブか非アクティブかを示すものです。

```
SELECT EVMONNAME,
  CASE
    WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive'
    WHEN EVENT_MON_STATE(EVMONNAME) = 1 THEN 'Active'
  END
  FROM SYSCAT.EVENTMONITORS
```

EXP

スキーマは SYSIBM です。 (EXP 関数の SYSFUN バージョンは引き続き使用可能です。)

EXP 関数は、引数で指定された累乗の自然対数 (e) の底である値を戻します。EXP 関数と LN 関数は互いに逆演算です。

引数は、組み込み数値データ・タイプの値を返す式でなければなりません。引数が 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で実行されます。それ以外の場合は、関数による処理のために引数が倍精度浮動小数点数に変換されます。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点数になります。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例:

- E は、3.453789832 の値をもつ DECIMAL(10,9) ホスト変数であると仮定します。

  ```
  VALUES EXP(:E)
  ```

  これは、DOUBLE 値 +3.16200000069145E+001 を戻します。

FLOAT

スキーマは SYSIBM です。

FLOAT 関数は、数値の浮動小数点数表記を戻します。 FLOAT は DOUBLE の同義語です。

FLOOR

スキーマは SYSIBM です。 (FLOOR 関数の SYSFUN バージョンは引き続き使用可能です。)

引数よりも小さいか等しい整数で、最大の整数値を戻します。

引数が DECIMAL の場合は位取りは 0 になる以外は、関数の結果のデータ・タイプと長さ属性は、引数と同じになります。たとえば、データ・タイプが DECIMAL(5,5) の引数は DECIMAL(5,0) を戻します。

引数が NULL になる可能性があるか、または引数が 10 進浮動小数点数ではなく、データベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

例:

- FLOOR 関数を使用して、小数点以下の任意の桁数までの切り捨てを行います。

```
SELECT FLOOR(SALARY)
  FROM EMPLOYEE
```

- 正の数値と負の数値の両方に FLOOR 関数を使用します。

```
VALUES FLOOR(3.5), FLOOR(3.1),
  FLOOR(-3.1), FLOOR(-3.5)
```

この例はそれぞれ、3.、3.、-4.、および -4. を戻します。

GENERATE_UNIQUE

スキーマは SYSIBM です。

GENERATE_UNIQUE 関数は、同一関数の他の実行と比較してユニークである、 13 バイト長のビット・データ文字ストリング (CHAR(13) FOR BIT DATA) を戻します。(システムの刻時機構は、関数が実行されるデータベース・パーティション番号とともに、内部の世界標準時 (UTC) タイム・スタンプを生成するのに使用されます。実際のシステム刻時機構をリバースへ動かす調整を行うと、値が重複する場合があります。) この関数は、deterministic 関数ではないものと定義されます。

この関数には引数がありません (空の括弧を指定する必要があります)。

関数の結果は、内部形式の世界標準時 (UTC)、および関数が処理されたデータベース・パーティション番号から成る固有値になります。結果が NULL 値になることはありません。

この関数の結果を使用して、表内の固有値を用意することができます。後続の各値は直前の値より大きくなり、表で使用できる順序列を提供します。値には、関数が実行されたデータベース・パーティションの番号が組み込まれ、それにより、複数のデータベース・パーティションにまたがってパーティション化された表も、ある順序列の固有値を持つことになります。この順序列は、関数が実行された時刻に基づいています。

この関数は、特殊レジスター CURRENT TIMESTAMP を使用する場合とは異なります。この特殊レジスターの場合、複数行の挿入ステートメントまたは全選択を伴う挿入ステートメントの各行について固有値が生成されます。

この関数の結果の一部であるタイム・スタンプ値は、 GENERATE_UNIQUE の結果を引数にする TIMESTAMP スカラー関数を使用して決定することができます。

例:

- 行ごとにユニークな列から成る表を作成します。 GENERATE_UNIQUE 関数を使用してこの列を移植します。 UNIQUE_ID 列には、列をビット・データ文字ストリングとして識別するために "FOR BIT DATA" が指定されていることに注意してください。

  ```
  CREATE TABLE EMP_UPDATE
    (UNIQUE_ID CHAR(13) FOR BIT DATA,
    EMPNO CHAR(6),
    TEXT VARCHAR(1000))
    INSERT INTO EMP_UPDATE
      VALUES (GENERATE_UNIQUE(), '000020', 'Update entry...'),
      (GENERATE_UNIQUE(), '000050', 'Update entry...')
  ```

  この表には、行ごとに固有な ID があります。ただし、UNIQUE_ID 列が、常に GENERATE_UNIQUE を使用して設定されている場合です。これは、表にトリガーを導入することによって行うことができます。

```
CREATE TRIGGER EMP_UPDATE_UNIQUE
  NO CASCADE BEFORE INSERT ON EMP_UPDATE
  REFERENCING NEW AS NEW_UPD
  FOR EACH ROW
  SNEW_UPD.UNIQUE_ID = GENERATE_UNIQUE()
```

このトリガーを定義すると、以下のように最初の列を指定せずに上記の INSERT ステートメントを出すことができます。

```
INSERT INTO EMP_UPDATE (EMPNO, TEXT)
  VALUES ('000020', 'Update entry 1...'),
  ('000050', 'Update entry 2...')
```

行が EMP_UPDATE に追加された時点のタイム・スタンプ (UTC における) は、以下を使用して戻すことができます。

```
SELECT TIMESTAMP (UNIQUE_ID), EMPNO, TEXT
  FROM EMP_UPDATE
```

したがって、表内のタイム・スタンプ列を行の挿入時に記録する必要はありません。

GETHINT

スキーマは SYSIBM です。

パスワード・ヒントが *encrypted-data* 内で見つかった場合、 GETHINT 関数をそのヒントを返します。パスワード・ヒントとは、データ所有者がパスワードを思い出すのに役立つ語句です。たとえば 'Ocean' は、'Pacific' を思い出すヒントになります。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

encrypted-data
完全な、暗号化データ・ストリングである CHAR FOR BIT DATA または VARCHAR FOR BIT DATA 値を返す式です。データ・ストリングは、ENCRYPT 関数を使って暗号化されたものでなければなりません (SQLSTATE 428FE)。

関数の結果は VARCHAR(32) です。結果は NULL になることがあります。ヒント・パラメーターが ENCRYPT 関数によって *encrypted-data* に追加されなかった場合、または最初の引数が NULL の場合には、結果が NULL 値になります。

例:

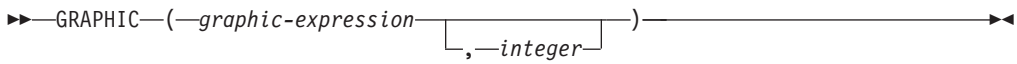
この例では、暗号化パスワード 'Pacific' を思い出すことができるよう、ヒント 'Ocean' が保管されます。

```
INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Pacific','Ocean');
SELECT GETHINT(SSN)
  FROM EMP;
```

返される値は 'Ocean' です。

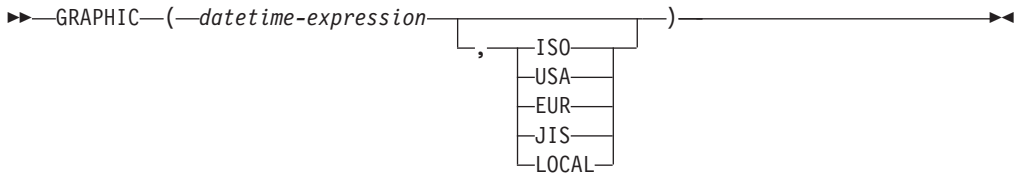
GRAPHIC

GRAPHIC → GRAPHIC:



文字 → GRAPHIC:

日付/時刻→GRAPHIC:



スキーマは SYSIBM です。キーワードが関数シグニチャーで使用されている場合、関数名を修飾名で指定することはできません。

GRAPHIC 関数は、以下を表す固定長の GRAPHIC ストリングを戻します。

- GRAPHIC ストリング (最初の引数がいずれかのタイプの GRAPHIC ストリングの場合)
- 文字ストリング。最初の引数がいずれかの型の文字ストリングの場合には、 1 バイト文字は 2 バイト文字に変換されます。
- 日付/時刻 (Unicode データベースのみ) (最初の引数が日付、時刻、またはタイム・スタンプの場合)

Unicode データベースでは、指定した引数が文字ストリングであると、まず GRAPHIC ストリングに変換されてから、関数が実行されます。最後の文字が高サロゲートになるように出力ストリングが切り捨てられた場合、そのサロゲートはブランク文字 (X'0020') に変換されます。この動作を過信しないでください。今後のリリースで変更される可能性があるからです。

この関数の結果は、固定長 GRAPHIC ストリング (GRAPHIC データ・タイプ) になります。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL であれば、結果は NULL 値になります。

GRAPHIC → GRAPHIC:

graphic-expression
GRAPHIC ストリング値を戻す式。

integer
結果の GRAPHIC データ・タイプの長さ属性を指定する整数値。値は 1 から 127 の範囲でなければなりません。値を指定しない場合の結果の長さ属性は、最初の引数の長さ属性と同じになります。

文字→ GRAPHIC:

character-expression
値が LONG VARCHAR または CLOB 以外の文字ストリングのデータ・タイプでなければならず、しかも 16336 バイトを最大長としてもつ式。

結果の長さ属性は、引数の長さ属性と同じになります。

日付/時刻 → GRAPHIC

datetime-expression
次の 3 つのデータ・タイプのいずれかの式。

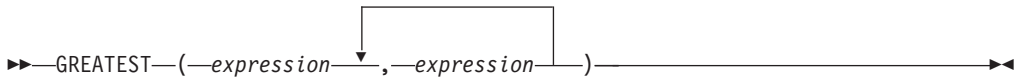
日付 結果は、2 番目の引数によって指定された形式の日付の GRAPHIC ストリング表記になります。結果の長さは 10 文字です。 2 番目の引数が指定され、その値が有効な値でない場合には、エラーが戻されます (SQLSTATE 42703)。

時刻 結果は、2 番目の引数によって指定された形式の時刻の GRAPHIC ストリング表記になります。結果の長さは 8 文字です。2 番目の引数が指定され、その値が有効でない場合には、エラーが戻されます (SQLSTATE 42703)。

タイム・スタンプ
結果は、タイム・スタンプの GRAPHIC ストリング表記になります。結果の長さは 26 文字です。2 番目の引数は適用されないので、指定してはなりません (SQLSTATE 42815)。

ストリングのコード・ページは、アプリケーション・サーバーのデータベースのコード・ページになります。

GREATEST



スキーマは SYSIBM です。

GREATEST 関数は、値の集合の最大値を戻します。

引数は互換性がなければならず、それぞれの引数は、次のもの以外のデータ・タイプの値を戻す式でなければなりません: ARRAY、LOB、LONG VARCHAR、LONG VARGRAPHIC、XML、これらのタイプのいずれかに基づく特殊タイプ、構造化タイプ (SQLSTATE 42815)。この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

選択された引数は、必要に応じて結果の属性に変換されます。結果の属性は、結果データ・タイプの規則に基づくすべてのオペランドにより決定されます。

この関数の結果は最大の引数値となります。少なくとも引数の 1 つが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

GREATEST スカラー関数は MAX スカラー関数の同義語です。

例:

表 T1 に 3 つの列 C1、C2、および C3 があり、それぞれ順に 1、7、および 4 の値があるとします。以下の照会を行うと、

```
SELECT GREATEST (C1, C2, C3) FROM T1
```

7 が戻されます。

列 C3 に 4 の代わりに値 NULL があると、この同じ照会で NULL が戻されます。

HASHEDVALUE

スキーマは SYSIBM です。

HASHEDVALUE 関数は、パーティション関数を行の分散キー値に適用することによって入手された行の分散マップ索引を戻します。例えば、SELECT 節で使用すると、その SELECT ステートメントの結果の生成に使用された表の各行の分散マップ索引を戻します。

遷移変数および表に戻される分散マップ索引は、分散キー列の現行遷移値から派生します。例えば、挿入前トリガーにおいて、新しい遷移変数の現行値があれば、関数は予測分散マップ索引を戻します。ただし、分散キー列の値はそれ以後の挿入前トリガーによって変更される場合があります。したがって、データベースに挿入される時点で、行の最終分散マップ索引は予測値と異なるかもしれません。

引数は、表内の列の修飾された名前または無修飾の名前でなければなりません。該当の列は、どのようなデータ・タイプであっても構いません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。)
column-name がビューの列を参照する場合、その列のビューの式は基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

HASHEDVALUE 関数によって分散マップ索引が戻される特定の行 (および表) は、この関数を使用する SQL ステートメントのコンテキストから判別されます。

結果のデータ・タイプは、0 から 4095 の範囲の INTEGER です。分散キーのない表の場合、結果は常に 0 になります。NULL 値が戻されることはありません。行レベルの情報が戻されるので、どの列が表に指定されるかに関係なく、結果は同じです。

HASHEDVALUE 関数は、複製された表、チェック制約内、または生成列の定義で使用することはできません (SQLSTATE 42881)。

バージョン 8 より前の旧バージョンとの互換性を保つために、 HASHEDVALUE を関数名 PARTITION に置き換えても構いません。

例:

- 分散マップ索引が 100 であるすべての行について、EMPLOYEE 表から従業員番号 (EMPNO) をリストします。

```
SELECT EMPNO FROM EMPLOYEE
  WHERE HASHEDVALUE(PHONENO) = 100
```

- 表 EMPLOYEE で BEFORE トリガーを作成して、従業員の挿入の際には必ず、EMPINSERTLOG2 という表に従業員番号と新しい行の予測分散マップ索引を記録します。

```
CREATE TRIGGER EMPINSLOGTRIG2
  BEFORE INSERT ON EMPLOYEE
  REFERENCING NEW AW NEWTABLE
  FOR EACH ROW
  INSERT INTO EMPINSERTLOG2
    VALUES(NEWTABLE.EMPNO, HASHEDVALUE(NEWTABLE.EMPNO))
```

HEX

スキーマは SYSIBM です。

HEX 関数は、値の 16 進表記を文字ストリングとして戻します。

引数には、最大長 16 336 バイトの任意の組み込みデータ・タイプの値である式を使うことができます。

この関数の結果は文字ストリングです。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

コード・ページはデータベース・コード・ページになります。

結果は、16 進数字のストリングです。最初の 2 つは引数の最初のバイト、次の 2 つは引数の 2 番目のバイトを表します。以下同様です。引数が日付/時刻値または数値である場合、結果は引数の内部形式の 16 進表記になります。戻される 16 進表記は、関数が実行されるアプリケーション・サーバーによって異なる場合があります。違いが生じる場合としては、次のような場合があります。

- EBCDIC サーバーに対する ASCII クライアントまたは ASCII サーバーに対する EBCDIC クライアントで、文字ストリング引数を指定して HEX 関数を実行したとき。
- クライアント・システムとサーバー・システムとで数値のバイト・オーダーが異なる場合に、 HEX 関数に数値引数を指定したとき (場合によります)。

結果のタイプと長さは、文字ストリング引数のタイプと長さによって異なります。

- 文字ストリング
  - 127 以下の固定長
    - 結果は、引数について定義されている長さの 2 倍の長さの固定長文字ストリングになります。
  - 127 を超える固定長
    - 結果は、引数について定義されている長さの 2 倍の長さの可変長文字ストリングになります。
  - 可変長
    - 結果は、引数について定義されている最大長さの 2 倍を最大長とする可変長文字ストリングになります。
- GRAPHIC ストリング
  - 63 以下の固定長
    - 結果は、引数について定義されている長さの 4 倍の長さの固定長文字ストリングになります。
- 63 を超える固定長
  - 結果は、引数について定義されている長さの 4 倍の長さの可変長文字ストリングになります。

- 可変長
  - 結果は、引数について定義されている最大の長さの 4 倍を最大長とする可変長文字ストリングになります。

例:

以下の例では、DB2 for AIX アプリケーション・サーバーを使用することを前提にしています。

- DEPARTMENT 表を使用して、ホスト変数 HEX_MGRNO (char(12)) に、'PLANNING' 部門 (DEPTNAME) の管理者番号 (MGRNO) の 16 進表記を設定します。

```
SELECT HEX(MGRNO)
  INTO :HEX_MGRNO
  FROM DEPARTMENT
  WHERE DEPTNAME = 'PLANNING'
```

  サンプル表を使用した場合、HEX_MGRNO は '303030303230' に設定されます (文字値は '000020')。
- COL_1 が、データ・タイプ char(1)、値 'B' の列であるとします。英字 'B' の 16 進数表記は X'42' です。 HEX(COL_1) は 2 バイトの長さのストリング '42' を戻します。
- COL_3 が、データ・タイプ decimal(6,2)、値 40.1 の列であるとします。 10 進数値 40.1 の内部表記に HEX 関数を適用した結果は、8 バイトの長さのストリング '0004010C' になります。

HOUR

スキーマは SYSIBM です。

HOUR 関数は、値の時の部分を戻します。

引数は、時刻、タイム・スタンプ、時刻期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が、時刻、タイム・スタンプ、または時刻やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の時間の部分 (0 から 24 の整数) になります。
- 引数が時刻期間またはタイム・スタンプ期間の場合
  - 結果は、値の時の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例:

CL_SCHED サンプル表を使用して、午後に始まるすべてのクラスを選択します。

```
SELECT * FROM CL_SCHED
  WHERE HOUR(STARTING) BETWEEN 12 AND 17
```

IDENTITY_VAL_LOCAL

スキーマは SYSIBM です。

IDENTITY_VAL_LOCAL 関数は、非 deterministic 関数であり、ID 列に割り当てられた最新の値を戻します。この割り当ては、VALUES 節を使用した単一の INSERT ステートメントの結果として発生したものです。関数には入力パラメーターはありません。

対応する ID 列の実際のデータ・タイプに関係なく、結果は DECIMAL(31,0) です。

関数によって返される値は、最新の単一行挿入操作で識別されている表の ID 列に割り当てられた値です。 INSERT ステートメントには、ID 列の入った表の VALUES 節が入っていなければなりません。また、INSERT ステートメントは、同じレベルで発行される必要もあります。つまり、値は、次に割り当てられる値に置き換えられるまでは、割り当てられたレベルでローカルに使用できなければなりません。 (レベル、トリガーやルーチンが呼び出されるたびに新しく開始されます。)

割り当てられる値は (ID 列が GENERATED BY DEFAULT で定義される場合は) ユーザーによって提供されるか、あるいはデータベース・マネージャーで生成された ID 値が提供されます。

VALUES 節をともなう単一行 INSERT ステートメントが、現行処理レベルで ID 列の入った表に対して発行されていない場合は、関数は NULL 値を戻します。

関数の結果が以下のステートメントによって影響を受けることはありません。

- ID 列のない表の、VALUES 節をともなう単一行 INSERT ステートメント
- VALUES 節をともなう複数行 INSERT ステートメント
- fullselect を持つ INSERT ステートメント
- ROLLBACK TO SAVEPOINT ステートメント

注:

- INSERT ステートメントの VALUES 節内の式は、挿入操作のターゲット列の割り当ての前に評価されます。そのため、INSERT ステートメントの VALUES 節にある IDENTITY_VAL_LOCAL 関数の呼び出しでは、前の挿入操作からの ID 列として、最新の割り当て値が使用されます。 ID 列の入った表の VALUES 節をともなう単一行 INSERT ステートメントが、 IDENTITY_VAL_LOCAL 関数と同じレベル内で実行されていない場合、関数は NULL 値を戻します。
- トリガーが定義されている表の ID 列の値は、ID 列のトリガー遷移変数を参照することにより、トリガー内で判別できます。
- 挿入トリガーのトリガー条件から IDENTITY_VAL_LOCAL 関数を呼び出した結果は、 NULL 値になります。
- 複数の BEFORE または AFTER 挿入トリガーが 1 つの表について存在することが可能です。この場合、各トリガーは別々に処理され、 IDENTITY_VAL_LOCAL 関数を使用して、あるトリガー処置によって割り当てられている値を別のトリガ

一処置に使用することはできません。概念上、複数のトリガー処置が同じレベルで定義されている場合でも、これは当てはまります。

- 一般的に、BEFORE 挿入トリガーの本体に IDENTITY_VAL_LOCAL 関数を使用することはお勧めしません。 BEFORE 挿入トリガーのトリガー処置から IDENTITY_VAL_LOCAL 関数を呼び出した結果は、NULL 値になります。トリガーが定義されている表の ID 列の値は、BEFORE 挿入トリガーのトリガー処置から IDENTITY_VAL_LOCAL 関数を呼び出すことによって得ることはできません。ただし、ID 列のトリガー遷移変数を参照することにより、その ID 列の値をトリガー処置で得ることができます。
- AFTER 挿入トリガーのトリガー処置から IDENTITY_VAL_LOCAL 関数を呼び出した結果は、ID 列の入った表の VALUES 節をともなう、同じトリガー処置で呼び出された最新の単一行挿入操作で識別されている表の ID 列に割り当てられた値になります。 (これは、トリガーの挿入後、FOR EACH ROW と FOR EACH STATEMENT の両方に適用されます。) ID 列の入った表の VALUES 節をともなう単一行 INSERT ステートメントが、 IDENTITY_VAL_LOCAL 関数呼び出しの前に、同じトリガー処置の中で実行されなかった場合、関数は NULL 値を返します。
- IDENTITY_VAL_LOCAL 関数は deterministic 関数ではないため、カーソルの SELECT ステートメント内でのこの関数の呼び出しの結果は、各 FETCH ステートメントによって異なります。
- 割り当て値は、ID 列に実際に割り当てられる値 (つまり、その後に続く SELECT ステートメントで戻される値です)。この値は必ずしも、INSERT ステートメントの VALUES 節で提供される値、あるいはデータベース・マネージャーによって生成される値とは限りません。割り当て値は、 ID 列に関連するトリガー遷移変数の、トリガー挿入前の本体内の SET 遷移変数ステートメントに指定されている値にすることができます。
- VALUES 節をともなう単一行 INSERT を ID 列と一緒に表に入れるのに失敗した後の関数によって戻される値は予測不能です。その値は、失敗した挿入操作の前に呼び出された関数から戻された値である場合もあれば、あるいは、続く挿入操作に割り当てられる値の場合もあります。戻される実際の値は、失敗のロケーションにより異なるので、予測不能です。

例:

例 1: T1 および T2 という 2 つの表を作成します。それぞれに C1 という名前の ID 列があります。表 T2 の ID シーケンスを 10 から開始します。 C2 のいくつかの値を T1 に挿入します。

```
CREATE TABLE T1
  (C1 INTEGER GENERATED ALWAYS AS IDENTITY,
   C2 INTEGER)

CREATE TABLE T2
  (C1 DECIMAL(15,0) GENERATED BY DEFAULT AS IDENTITY (START WITH 10),
   C2 INTEGER)

INSERT INTO T1 (C2) VALUES (5)

INSERT INTO T1 (C2) VALUES (6)

SELECT * FROM T1
```

この照会は次の値を返します。

```
C1          C2
----------- -----------
          1           5
          2           6
```

単一行を表 T2 に挿入します。ここで、列 C2 は IDENTITY_VAL_LOCAL 関数から値を得ます。

```
INSERT INTO T2 (C2) VALUES (IDENTITY_VAL_LOCAL())

SELECT * FROM T2
```

この照会は次の値を返します。

```
C1                C2
----------------- -----------
              10.           2
```

例 2: トリガーに関係するネストされた環境では、ID 列がより低いレベルで割り当てられていても、ある特定のレベルで割り当てられている ID 値を検索するには IDENTITY_VAL_LOCAL 関数を使用してください。 3 つの表 EMPLOYEE、EMP_ACT、および ACCT_LOG があると仮定します。 EMP_ACT および ACCT_LOG 表に挿入をさらに行う AFTER 挿入トリガーが EMPLOYEE に定義されています。

```
CREATE TABLE EMPLOYEE
  (EMPNO SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1000),
   NAME CHAR(30),
   SALARY DECIMAL(5,2),
   DEPTNO SMALLINT)

CREATE TABLE EMP_ACT
  (ACNT_NUM SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 1),
   EMPNO SMALLINT)

CREATE TABLE ACCT_LOG
  (ID SMALLINT GENERATED ALWAYS AS IDENTITY (START WITH 100),
   ACNT_NUM SMALLINT,
   EMPNO SMALLINT)

CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
  REFERENCING NEW AS NEW_EMP
  FOR EACH ROW
  BEGIN ATOMIC
    INSERT INTO EMP_ACT (EMPNO) VALUES (NEW_EMP.EMPNO);
    INSERT INTO ACCT_LOG (ACNT_NUM, EMPNO)
      VALUES (IDENTITY_VAL_LOCAL(), NEW_EMP.EMPNO);
  END
```

最初にトリガーされる挿入操作は、行を EMP_ACT 表に挿入します。このステートメントは、EMPLOYEE 表の EMPNO 列のトリガー遷移変数を使用して、EMPLOYEE 表の EMPNO 列の ID 値を EMP_ACT 表の EMPNO 列にコピーするよう指示します。 INSERT ステートメントはネストのこのレベルでは発行されていないので、EMPLOYEE 表の EMPNO 列に割り当てられている値を得るために、IDENTITY_VAL_LOCAL 関数を使用することはできません。
IDENTITY_VAL_LOCAL 関数が EMP_ACT 表の INSERT ステートメントの VALUES 節で呼び出された場合、NULL 値が返されます。 EMP_ACT 表に対する挿入操作の結果、ACNT_NUM 列の新しい ID 値も生成されます。

2 番目にトリガーされる挿入操作は、行を ACCT_LOG 表に挿入します。このステートメントは IDENTITY_VAL_LOCAL 関数を呼び出し、トリガー処置の前の挿入操作で EMP_ACT 表の ACNT_NUM 列に割り当てられた ID 値を、ACCT_LOG 表の ACNT_NUM 列にコピーするよう指示します。 EMPNO 列は、EMPLOYEE 表の EMPNO 列と同じ値が割り当てられています。

以下の INSERT ステートメントおよびすべてのトリガー・アクションが処理された後に:

```
INSERT INTO EMPLOYEE (NAME, SALARY, DEPTNO)
  VALUES ('Rupert', 989.99, 50)
```

3 つの表の内容は、以下のようになります。

```
SELECT EMPNO, SUBSTR(NAME,1,10) AS NAME, SALARY, DEPTNO
  FROM EMPLOYEE

EMPNO  NAME       SALARY  DEPTNO
------ ---------- ------- ------
  1000 Rupert      989.99     50

SELECT ACNT_NUM, EMPNO
  FROM EMP_ACT

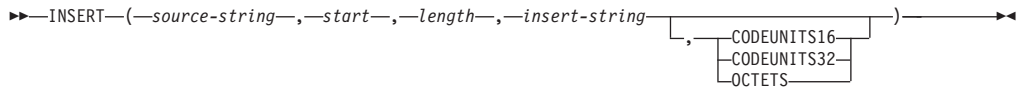
ACNT_NUM EMPNO
-------- ------
       1   1000

SELECT * FROM ACCT_LOG

ID     ACNT_NUM EMPNO
------ -------- ------
   100        1   1000
```

IDENTITY_VAL_LOCAL 関数の結果は、同じネスト・レベルで ID 列に割り当てられた最新の値になります。オリジナル INSERT ステートメント、およびトリガー処置すべてが処理された後、 EMPLOYEE 表の EMPNO 列に割り当てられている値が 1000 であるため、 IDENTITY_VAL_LOCAL 関数は値 1000 を返します。

INSERT



スキーマは SYSIBM です。 INSERT 関数の SYSFUN バージョンは引き続き使用可能です。

INSERT 関数は、*source-string* の *start* から *length* バイトを削除し、*insert-string* を挿入したストリングを戻します。

INSERT 関数は、length 引数が必須である点を除けば、OVERLAY 関数と同じです。

source-string
ソース・ストリングを指定する式。この式は、組み込みの文字ストリング・データ・タイプまたは GRAPHIC ストリング・データ・タイプの値を戻す必要があります (SQLSTATE 22011)。

start
整数値を戻す式。この整数値では、ソース・ストリングから指定のバイトを削除する開始点を指定します (この開始点は、別のストリングの挿入を開始する開始点でもあります)。この整数値は、1 から、*source-string* の長さに 1 を加算した値までの範囲でなければなりません (SQLSTATE 42815)。 OCTETS を指定した場合に、結果が GRAPHIC データであれば、値は、1 から、*source-string* の長さ属性の 2 倍に 1 を加算した値までの範囲にある奇数でなければなりません (SQLSTATE 428GC)。

length
ソース・ストリングから削除する (指定のストリング単位による) コード単位の数を指定する式。削除の開始点は、*start* で指定する位置になります。値は、正整数またはゼロでなければなりません (SQLSTATE 22011)。OCTETS が指定され、結果が GRAPHIC データである場合、値は偶数またはゼロでなければなりません (SQLSTATE 428GC)。

insert-string
source-string に挿入するストリングを指定する式。挿入の開始点は、*start* で指定する位置になります。この式は、*source-string* との互換性がある組み込みの文字ストリング・データ・タイプまたは GRAPHIC ストリング・データ・タイプの値を戻す必要があります (SQLSTATE 22011)。

CODEUNITS16、CODEUNITS32、または OCTETS
start および *length* のストリング単位を指定します。

CODEUNITS16 は、*start* および *length* を 16 ビットの UTF-16 コード単位で表すことを指定します。CODEUNITS32 は、*start* および *length* を 32 ビットの UTF-32 コード単位で表すことを指定します。OCTETS は、*start* および *length* をバイト単位で表すことを指定します。

ストリング単位として CODEUNITS16 または CODEUNITS32 を指定した場合に、結果がバイナリー・ストリングまたはビット・データであれば、エラーが戻されます (SQLSTATE 428GC)。ストリング単位として OCTETS を指定した場

合に、*insert-string* と *source-string* がバイナリー・ストリングであれば、エラーが戻されます (SQLSTATE 42815)。ストリング単位として OCTETS を指定すると、操作は、*source-string* のコード・ページで実行されます。ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果が GRAPHIC データであれば、*start* と *length* は 2 バイト単位の長さになり、それ以外の場合は、バイト単位になります。
CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

結果のデータ・タイプは、*source-string* と *insert-string* のデータ・タイプによって異なります。サポートされているタイプの組み合わせを以下の表にまとめます。

表 33. source-string と insert-string のデータ・タイプに対応した関数結果のデータ・タイプ

source-string	*insert-string*	**結果**
CHAR または VARCHAR	CHAR または VARCHAR	VARCHAR
GRAPHIC または VARGRAPHIC	GRAPHIC または VARGRAPHIC	VARGRAPHIC
CLOB	CHAR、VARCHAR、または CLOB	CLOB
DBCLOB	GRAPHIC、VARGRAPHIC、または DBCLOB	DBCLOB
CHAR または VARCHAR	CHAR FOR BIT DATA または VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHAR FOR BIT DATA または VARCHAR FOR BIT DATA	CHAR、VARCHAR、CHAR FOR BIT DATA、または VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
Unicode データベースの場合のみ:		
CHAR または VARCHAR	GRAPHIC または VARGRAPHIC	VARCHAR
GRAPHIC または VARGRAPHIC	CHAR または VARCHAR	VARGRAPHIC
CLOB	GRAPHIC、VARGRAPHIC、または DBCLOB	CLOB
DBCLOB	CHAR、VARCHAR、または CLOB	DBCLOB

source-string の長さを 0 にすることができます。この場合、(上記の *start* の境界が示すとおり) *start* が 1 でなければなりません。関数の結果は、*insert-string* のコピーになります。

insert-string の長さを 0 にすることも可能です。この結果として、*start* と *length* で指定したコード単位が *source-string* から削除されます。

結果の長さ属性は、*source-string* の長さ属性と *insert-string* を加算した値になります。結果の実際の長さは、A1 - MIN((A1 - V2 + 1), V3) + A4 で、各変数の意味は以下のとおりです。

- A1 は *source-string* の実際の長さ

- V2 は *start* の値
- V3 は *length* の値
- A4 は *insert-string* の実際の長さ

結果のストリングの実際の長さが戻りデータ・タイプの最大値を超える場合、エラーが戻されます (SQLSTATE 54006)。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- 'INSERTING' というストリングから、'INSISTING'、'INSISERTING'、'INSTING' というストリングを作成するために、既存のテキストの中にテキストを挿入します。

```
SELECT INSERT('INSERTING',4,2,'IS'),
  INSERT('INSERTING',4,0,'IS'),
  INSERT('INSERTING',4,2,'')
  FROM SYSIBM.SYSDUMMY1
```

- 'INSERTING' というストリングから、'XXINSERTING'、'XXNSERTING'、'XXSERTING'、'XXERTING' というストリングを作成するために、1 という開始点を使用して既存のテキストの前にテキストを挿入します。

```
SELECT INSERT('INSERTING',1,0,'XX'),
  INSERT('INSERTING',1,1,'XX'),
  INSERT('INSERTING',1,2,'XX'),
  INSERT('INSERTING',1,3,'XX')
  FROM SYSIBM.SYSDUMMY1
```

- 'ABCABC' というストリングから、'ABCABCXX' というストリングを作成するために、既存のテキストの後にテキストを挿入します。ソース・ストリングの長さは 6 文字なので、開始点を 7 (つまり、ソース・ストリングの長さに 1 を加算した値) に設定します。

```
SELECT INSERT('ABCABC',7,0,'XX')
  FROM SYSIBM.SYSDUMMY1
```

- ストリング 'Hegelstraße' を 'Hegelstrasse' に変更します。

```
SELECT INSERT('Hegelstraße',10,1,'ss',CODEUNITS16)
  FROM SYSIBM.SYSDUMMY1
```

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽のト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

変数 UTF8_VAR および UTF16_VAR に、ストリングの UTF-8 表記および UTF-16BE 表記がそれぞれ含まれているとします。 INSERT 関数を使用して、Unicode ストリング '&N~AB' に 'C' を挿入します。

```
SELECT INSERT(UTF8_VAR, 1, 4, 'C', CODEUNITS16),
  INSERT(UTF8_VAR, 1, 4, 'C', CODEUNITS32),
  INSERT(UTF8_VAR, 1, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'CAB'、'CB'、'CN~AB' という値がそれぞれ戻されます。

```
SELECT INSERT(UTF8_VAR, 5, 1, 'C', CODEUNITS16),
  INSERT(UTF8_VAR, 5, 1, 'C', CODEUNITS32),
  INSERT(UTF8_VAR, 5, 1, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~CB'、'&N~AC'、'&C~AB' という値がそれぞれ戻されます。

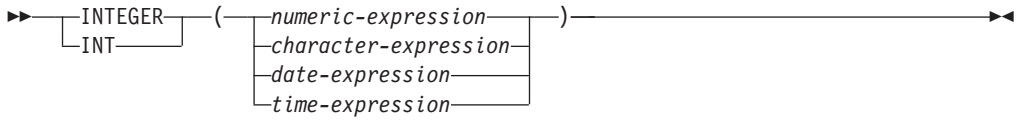
```
SELECT INSERT(UTF16_VAR, 1, 4, 'C', CODEUNITS16),
  INSERT(UTF16_VAR, 1, 4, 'C', CODEUNITS32),
  INSERT(UTF16_VAR, 1, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'CAB'、'CB'、'CN~AB' という値がそれぞれ戻されます。

```
SELECT INSERT(UTF16_VAR, 5, 2, 'C', CODEUNITS16),
  INSERT(UTF16_VAR, 5, 1, 'C', CODEUNITS32),
  INSERT(UTF16_VAR, 5, 4, 'C', OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~C'、'&N~AC'、'&CAB' という値がそれぞれ戻されます。

INTEGER



スキーマは SYSIBM です。

INTEGER 関数は、数値、文字ストリング、日付、または時刻を表す整数を、整数定数の形で戻します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

numeric-expression
組み込み数値データ・タイプの値を返す式。

引数が*数値式* の場合、結果は、引数を長精度整数 (large integer) の列または変数に割り当てた場合と同じ数値になります。引数の整数部分が整数の範囲内にない場合、エラーになります。引数に小数部分がある場合は、切り捨てられます。

character-expression
文字定数の最大長以下の長さの文字ストリング値を戻す式。先行ブランクと末尾ブランクは削除されます。その結果のストリングは、 SQL 整数定数を形成するための規則に従うものでなければなりません (SQLSTATE 22018)。文字ストリングとして、LONG ストリングを使うことはできません。

引数が文字式 の場合、結果は、対応する整数定数を長精度整数の列または変数に割り当てた場合の数値と同じになります。

date-expression
DATE データ・タイプの値を戻す式。結果は、日付を *yyyymmdd* で表した INTEGER 値になります。

time-expression
TIME データ・タイプの値を戻す式。結果は、時間を *hhmmss* で表した INTEGER 値になります。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例:

- EMPLOYEE 表を使用して、給与 (SALARY) を学歴 (EDLEVEL) 値で除算した値を示したリストを選択します。計算では小数部をすべて切り捨てます。リストには、計算に使用される値と従業員番号 (EMPNO) も入っていなければなりません。リストは、計算値の降順に配列する必要があります。

```
SELECT INTEGER (SALARY / EDLEVEL), SALARY, EDLEVEL, EMPNO
  FROM EMPLOYEE
  ORDER BY 1 DESC
```

- EMPLOYEE 表を使用して、アプリケーションでさらに処理を行うために、EMPNO 列を整数形式として選択します。

```
SELECT INTEGER(EMPNO) FROM EMPLOYEE
```

- BIRTHDATE (date) 列に、 '1964-07-20' に相当する内部値が入っていると想定します。

  ```
  INTEGER(BIRTHDATE)
  ```

  結果は 19 640 720 の値になります。
- STARTTIME (time) 列に、 '12:03:04' に相当する内部値が入っていると想定します。

  ```
  INTEGER(STARTTIME)
  ```

  結果は 120 304 の値になります。

JULIAN_DAY

スキーマは SYSFUN です。

紀元前 4713 年 1 月 1 日 (ユリウス暦の起点) からの経過日数を表す整数値を引数で指定された日付値に戻します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

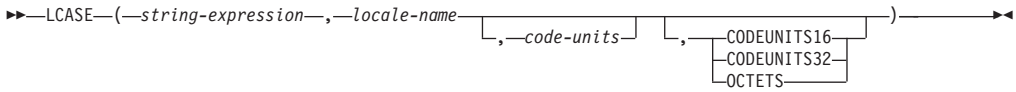
LCASE

スキーマは SYSIBM です。

LCASE 関数は、すべての SBCS 文字が小文字に変換されたストリングを戻します。

LCASE は LOWER の同義語です。

LCASE (ロケール依存)

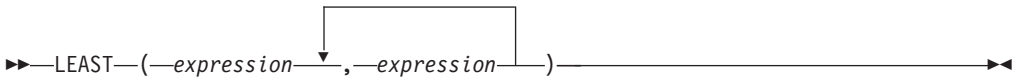


スキーマは SYSIBM です。

LCASE 関数は、指定したロケールに関連付けられた規則を使用して、すべての文字が小文字に変換されたストリングを戻します。

LCASE は LOWER の同義語です。

LEAST



スキーマは SYSIBM です。

LEAST 関数は、値の集合の最小値を戻します。

引数は互換性がなければならず、それぞれの引数は、次のもの以外のデータ・タイプの値を戻す式でなければなりません: ARRAY、LOB、LONG VARCHAR、LONG VARGRAPHIC、XML、これらのタイプのいずれかに基づく特殊タイプ、構造化タイプ (SQLSTATE 42815)。この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

選択された引数は、必要に応じて結果の属性に変換されます。結果の属性は、結果データ・タイプの規則に基づくすべてのオペランドにより決定されます。

この関数の結果は最小の引数値となります。少なくとも引数の 1 つが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

LEAST スカラー関数は MIN スカラー関数の同義語です。

例:

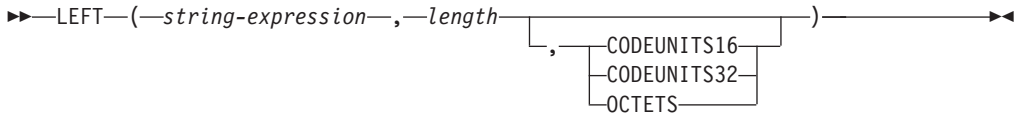
表 T1 に 3 つの列 C1、C2、および C3 があり、それぞれ順に 1、7、および 4 の値があるとします。以下の照会を行うと、

```
SELECT LEAST (C1, C2, C3) FROM T1
```

1 が戻されます。

列 C3 に 4 の代わりに値 NULL があると、この同じ照会で NULL が戻されます。

LEFT



スキーマは SYSIBM です。 LEFT 関数の SYSFUN バージョンは引き続き使用可能です。

LEFT 関数は、*string-expression* の左端にある長さ *length* のストリングを戻します (長さは、指定のストリング単位での長さになります)。*string-expression* が文字ストリングである場合、結果は文字ストリングです。 *string-expression* が GRAPHIC ストリングである場合、結果は GRAPHIC ストリングです。

string-expression
結果を取り出すストリングを指定する式。 この式は、組み込みの文字ストリング・データ・タイプまたは GRAPHIC ストリング・データ・タイプの値を戻す必要があります (SQLSTATE 42815)。*string-expression* のサブストリングは、*string-expression* のゼロ個以上の連続したコード・ポイントです。

length
結果の長さを指定する式。この結果は、組み込みの整数データ・タイプでなければなりません (SQLSTATE 42815)。値は、暗黙的または明示的に指定された単位で表される 0 から *string-expression* の長さまでの範囲になければなりません (SQLSTATE 22011)。ただし、例外が 1 つあります。ストリング単位を明示的に指定せずに値を定数として指定する場合、値は暗黙的ストリング単位での *string-expression* の長さ属性を超えることができます。 OCTETS を指定した場合に、結果が GRAPHIC データであれば、値は、0 から *string-expression* の長さ属性の 2 倍までの範囲にある偶数でなければなりません (SQLSTATE 428GC)。

CODEUNITS16、CODEUNITS32、または OCTETS
length のストリング単位を指定します。

CODEUNITS16 を指定すると、*length* は、16 ビットの UTF-16 コード単位の長さになります。CODEUNITS32 を指定すると、*length* は、32 ビットの UTF-32 コード単位の長さになります。OCTETS を指定すると、*length* は、バイト単位の長さになります。

ストリング単位として CODEUNITS16 または CODEUNITS32 を指定した場合に、*string-expression* がバイナリー・ストリングまたはビット・データであれば、エラーが戻されます (SQLSTATE 428GC)。ストリング単位として OCTETS を指定した場合に、*string-expression* が GRAPHIC ストリングであれば、*length* は偶数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 428GC)。ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果が GRAPHIC データであれば、*length* は 2 バイト単位の長さになり、それ以外の場合は、バイト単位になります。CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

string-expression の右側には必要な数の埋め込み文字が埋め込まれ、*string-expression* の指定のサブストリングが常に存在するようになります。埋め込み用の文字は、埋め込みが行われるコンテキストでストリングに埋め込みを適用するための文字と同じです。埋め込みの詳細については、『ストリング割り当て』と『割り当てと比較』を参照してください。

この関数の結果は可変長ストリングであり、その長さ属性は *length* とストリング単位を指定する方法によって異なります。定数を使用して *length* が指定されていない場合、またはストリング単位が明示的に指定された場合、長さ属性は *string-expression* の長さ属性と同じになります。定数を使用して *length* が指定され、ストリング単位が指定されていない場合、長さ属性は *length* と *string-expression* の長さ属性のうち長い方になります。結果のデータ・タイプは、次のように *string-expression* のデータ・タイプによって決まります。

- *string-expression* が CHAR または VARCHAR の場合は VARCHAR
- *string-expression* が CLOB の場合は CLOB
- *string-expression* が GRAPHIC または VARGRAPHIC の場合は VARGRAPHIC
- *string-expression* が DBCLOB の場合は DBCLOB

結果の実際の長さ (ストリング単位) は、*length* です。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- 変数 ALPHA の値が 'ABCDEF' であるとします。以下のステートメント:

  ```
  SELECT LEFT(ALPHA,3)
    FROM SYSIBM.SYSDUMMY1
  ```

  ALPHA の左端の 3 文字である 'ABC' が戻されます。

- VARCHAR(50) で定義されている変数 NAME の値が 'KATIE AUSTIN'、整変数 FIRSTNAME_LEN の値が 5 であると想定して、以下のステートメントを実行します。

  ```
  SELECT LEFT(NAME,FIRSTNAME_LEN)
    FROM SYSIBM.SYSDUMMY1
  ```

  値 'KATIE' が戻されます。

- 以下のステートメントは、長さゼロのストリングを戻します。

  ```
  SELECT LEFT('ABCABC',0)
    FROM SYSIBM.SYSDUMMY1
  ```

- EMPLOYEE 表の FIRSTNME 列は、VARCHAR(12) として定義されています。'BROWN' というラストネームの従業員のファーストネームを検出し、そのファーストネームを 10 バイト・ストリングで戻すには、以下のようにします。

  ```
  SELECT LEFT(FIRSTNME, 10)
    FROM EMPLOYEE
    WHERE LASTNAME = 'BROWN'
  ```

  'DAVID' という値の後に 5 つのブランク文字が埋め込まれた VARCHAR(12) ストリングが戻されます。

- Unicode データベースでは、FIRSTNAME が VARCHAR(12) の列になっています。その値の 1 つは、6 文字のストリング 'Jürgen' です。 FIRSTNAME が以下の値を持つ場合:

```
Function...                      Returns...

LEFT(FIRSTNAME,2,CODEUNITS32)   'Jü' -- x'4AC3BC'
LEFT(FIRSTNAME,2,CODEUNITS16)   'Jü' -- x'4AC3BC'
LEFT(FIRSTNAME,2,OCTETS)        'J'  -- x'4A20', a truncated string
```

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽のト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

20 バイトの長さ属性が指定されている変数 UTF8_VAR に、ストリングの UTF-8 表現が格納されると想定します。

```
SELECT LEFT(UTF8_VAR, 2, CODEUNITS16),
  LEFT(UTF8_VAR, 2, CODEUNITS32),
  LEFT(UTF8_VAR, 2, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&'、'&N'、'bb' という値がそれぞれ戻されます ('b' はブランク文字です)。

```
SELECT LEFT(UTF8_VAR, 5, CODEUNITS16),
  LEFT(UTF8_VAR, 5, CODEUNITS32),
  LEFT(UTF8_VAR, 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~A'、'&N~AB'、'&N' という値がそれぞれ戻されます。

```
SELECT LEFT(UTF8_VAR, 10, CODEUNITS16),
  LEFT(UTF8_VAR, 10, CODEUNITS32),
  LEFT(UTF8_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~ABbbbb'、'&N~ABbbbbb'、'&N~ABb' という値がそれぞれ戻されます ('b' はブランク文字です)。

20 コード単位の長さ属性が指定されている変数 UTF16_VAR に、ストリングの UTF-16BE 表現が格納されると想定します。

```
SELECT LEFT(UTF16_VAR, 2, CODEUNITS16),
  LEFT(UTF16_VAR, 2, CODEUNITS32),
  HEX (LEFT(UTF16_VAR, 2, OCTETS))
FROM SYSIBM.SYSDUMMY1
```

'&'、'&N'、X'D834' という値がそれぞれ戻されます (X'D834' は一致しない上位サロゲートです)。

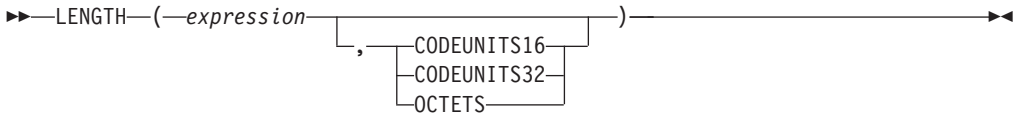
```
SELECT LEFT(UTF16_VAR, 5, CODEUNITS16),
  LEFT(UTF16_VAR, 5, CODEUNITS32),
  LEFT(UTF16_VAR, 6, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~A'、'&N~AB'、'&N' という値がそれぞれ戻されます。

```
SELECT LEFT(UTF16_VAR, 10, CODEUNITS16),
  LEFT(UTF16_VAR, 10, CODEUNITS32),
  LEFT(UTF16_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~ABbbbb'、'&N~ABbbbbb'、'&N~A' という値がそれぞれ戻されます ('b' はブランク文字です)。

LENGTH



スキーマは SYSIBM です。

LENGTH 関数は、暗黙的あるいは明示的なストリング単位で *expression* の長さを戻します。

expression
組み込みデータ・タイプの値を戻す式。 *expression* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。

CODEUNITS16、CODEUNITS32、または OCTETS
結果のストリング単位を指定します。 CODEUNITS16 は、結果が 16 ビット UTF-16 コード単位で表現されることを指定します。 CODEUNITS32 は、結果が 32 ビット UTF-32 コード単位で表現されることを指定します。 OCTETS は、結果がバイト単位で表現されることを指定します。

ストリング単位が明示的に指定され、*expression* がストリング・データでない場合、エラーが戻されます (SQLSTATE 428GC)。ストリング単位が CODEUNITS16 または CODEUNITS32 と指定され、*expression* がバイナリー・ストリングまたはビット・データである場合は、エラーが戻されます (SQLSTATE 428GC)。ストリング単位が OCTETS と指定され、*expression* がバイナリー・ストリングである場合は、エラーが戻されます (SQLSTATE 42815)。 CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果がグラフィック・データであれば、戻される値では、2 バイト単位の長さが指定されます。そうでない場合、戻される値では、バイト単位の長さが指定されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

文字および GRAPHIC ストリングの長さには、末尾ブランクが含まれます。バイナリー・ストリングの長さには、2 進ゼロが含まれます。可変長ストリングの長さは、最大長ではなく実際の長さです。それ以外のすべての値の場合の長さは、その値を表すために使用されるバイトの数になります。

- 短精度整数の場合は 2
- 長精度整数の場合は 4
- 精度 *p* の 10 進数の場合は (*p*/2)+1
- DECFLOAT(16) の場合は 8

- DECFLOAT(34) の場合は 16
- バイナリー・ストリングの場合は、そのストリングの長さ
- 文字ストリングの場合は、そのストリングの長さ
- 単精度浮動小数点の場合は 4
- 倍精度浮動小数点の場合は 8
- 日付の場合は 4
- 時刻の場合は 3
- タイム・スタンプの場合は 10

例:

- ホスト変数 ADDRESS が、'895 Don Mills Road' という値を持つ可変長文字ストリングであると想定します。

```
LENGTH(:ADDRESS)
```

  戻り値は 18 です。

- START_DATE が DATE タイプの列であると想定します。

```
LENGTH(START_DATE)
```

  戻り値は 4 です。

-

```
LENGTH(CHAR(START_DATE, EUR))
```

  戻り値は 10 です。

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽の ト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'
UTF-32BE	X'0001D11E'	X'0000004E'	X'00000303'	X'00000041'	X'00000042'

  変数 UTF8_VAR に、ストリングの UTF-8 表現が格納されると想定します。

```
SELECT LENGTH(UTF8_VAR, CODEUNITS16),
  LENGTH(UTF8_VAR, CODEUNITS32),
  LENGTH(UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  それぞれ、値 6、5、9 を戻します。

  変数 UTF16_VAR に、ストリングの UTF-16BE 表現が格納されると想定します。

```
SELECT LENGTH(UTF16_VAR, CODEUNITS16),
   LENGTH(UTF16_VAR, CODEUNITS32),
   LENGTH(UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

  それぞれ、値 6、5、12 を戻します。

LN

スキーマは SYSIBM です。 (LN 関数の SYSFUN バージョンは引き続き使用可能です。)

LN 関数は、数値の自然対数を戻します。 LN 関数と EXP 関数は互いに逆演算です。

引数は、組み込み数値データ・タイプの値を返す式でなければなりません。引数が 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で実行されます。それ以外の場合は、関数による処理のために引数が倍精度浮動小数点数に変換されます。引数の値は、ゼロより大きくなければなりません (SQLSTATE 22003)。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点数になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

代替構文: LN の代わりに LOG を指定できます。これは、以前のバージョンの DB2 との互換性のためにのみサポートされています。 LOG の代わりに LN を使用してください。一部のデータベース・マネージャーおよびアプリケーションは、数値の自然対数ではなく常用対数として LOG をインプリメントしているからです。

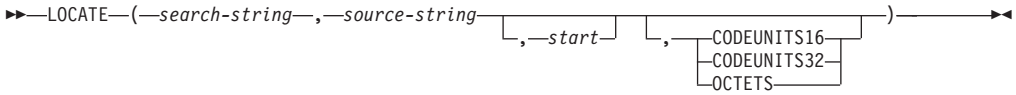
例:

- NATLOG は、31.62 の値を持つ DECIMAL(4,2) ホスト変数であると仮定します。

  ```
  VALUES LN(:NATLOG)
  ```

  これは概算値 3.45 を戻します。

LOCATE スカラー関数



スキーマは SYSIBM です。 LOCATE 関数の SYSFUN バージョンは引き続き使用可能です。ただし、データベースの照合に依存しているわけではありません。

LOCATE 関数は、あるストリング (*search-string* と呼ばれる) が別のストリング (*source-string* と呼ばれる) 内で、最初のオカレンスの開始位置を戻します。 *search-string* が見つからず、どちらの引数も NULL でない場合、結果はゼロになります。 *search-string* が見つかった場合、結果は 1 から *source-string* の実際の長さまでの数値になります。検索には、データベースの照合が使用されます。ただし、*search-string* または *source-string* が FOR BIT DATA で定義されている場合は除きます。その場合は、バイナリー比較によって検索が行われます。

オプションの *start* が指定されている場合、それは、*source-string* 中での検索が開始される文字位置を示します。オプションのストリング単位は、*start* と関数の結果がどの単位で表現されるかを表すために指定できます。

search-string の長さがゼロの場合、関数によって戻される結果は 1 です。それ以外の場合で、*source-string* の長さがゼロの場合、関数によって戻される結果は 0 です。それ以外の場合、

- *search-string* の値が *source-string* の値内の、連続する複数の位置にある同じ長さのサブストリングと等しい場合、関数によって戻される結果は、*source-string* 値内のその最初のサブストリングの開始位置になります。
- それ以外の場合、この関数によって戻される結果は 0 です。

search-string

検索の対象となるストリングを指定する式。この式は、実際の長さが 4000 バイトを超えない、組み込み文字ストリング・データ・タイプ、GRAPHIC ストリング・データ・タイプ、またはバイナリー・ストリング・データ・タイプである値を戻さなければなりません。式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを (CONCAT または || を使用して) 連結する式
- SQL プロシージャー・パラメーター

これらの規則は、LIKE 述部の *pattern-expression* に関して記述されるものと同様になります。

source-string
その中で検索が行われるストリングを指定する式。この式は、組み込み文字ストリング・データ・タイプ、GRAPHIC ストリング・データ・タイプ、バイナリー・ストリング・データ・タイプの値を戻す必要があります。この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数 (ロケーター変数またはファイル参照変数を含む)
- スカラー関数
- ラージ・オブジェクトのロケーター
- 列名
- 上記のいずれかを (CONCAT または || を使用して) 連結する式

start
検索が開始される *source-string* 内の位置を指定する式。この式は、ゼロ以上の整数である必要があります。 *start* が指定された場合、LOCATE 関数は以下のようになります。

```
POSITION(search-string,
  SUBSTRING(source-string, start, string-unit),
  string-unit) + start - 1
```

上記で、*string-unit* は、CODEUNITS16、CODEUNITS32、OCTETS のいずれかです。

start が指定されない場合、検索はソース・ストリングの先頭の位置から開始され、LOCATE 関数は以下のようになります。

```
POSITION(search-string, source-string, string-unit)
```

CODEUNITS16、CODEUNITS32、または OCTETS
start および結果のストリング単位を指定します。 CODEUNITS16 は、*start* および結果が 16 ビット UTF-16 コード単位で表現されることを指定します。 CODEUNITS32 は、*start* および結果が 32 ビット UTF-32 コード単位で表現されることを指定します。 OCTETS は、*start* および結果がバイト単位で表現されることを指定します。

ストリング単位が CODEUNITS16 または CODEUNITS32 と指定され、*search-string* または *source-string* がバイナリー・ストリングまたはビット・データである場合は、エラーが戻されます (SQLSTATE 428GC)。ストリング単位が OCTETS と指定され、*search-string* および *source-string* がバイナリー・ストリングである場合は、エラーが戻されます (SQLSTATE 42815)。

ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果がグラフィック・データであれば、*start* および戻り位置は 2 バイト単位で表現されます。それ以外の場合は、バイト単位で表現されます。

ロケールに依存する UCA ベースの照合がこの関数に使用される場合は、CODEUNITS16 オプションから最も適したパフォーマンスの特性を得られます。

CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

1 番目と 2 番目の引数は、互換性のあるストリング・タイプを持たなければなりません。互換性の詳細については、『ストリング変換についての規則』を参照してください。 Unicode データベースでは、一方のストリング引数が文字で (FOR BIT DATA ではない)、他方のストリング引数が GRAPHIC である場合、*search-string* は、処理のために *source-string* のデータ・タイプに変換されます。一方の引数が文字 FOR BIT DATA である場合、他方の引数は GRAPHIC であってはなりません (SQLSTATE 42846)。

この関数の結果は長精度整数 (large integer) です。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- ストリング 'DINING' 中に最初に現れる文字 'N' の位置を探す。

```
SELECT LOCATE('N', 'DINING')
FROM SYSIBM.SYSDUMMY1
```

  結果は値 3 になります。

- IN_TRAY という名前の表の中のすべての行について、RECEIVED 列と SUBJECT 列を選択し、NOTE_TEXT 列内のストリング 'GOOD' の開始位置を選択する。

```
SELECT RECEIVED, SUBJECT, LOCATE('GOOD', NOTE_TEXT)
FROM IN_TRAY
WHERE LOCATE('GOOD', NOTE_TEXT) <> 0
```

- ストリング 'Jürgen lives on Hegelstraße' 内の、文字 'ß' の位置を特定し、そのストリング内での位置を、CODEUNITS32 単位でホスト変数 LOCATION に設定する。

```
SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, CODEUNITS32)
```

  ホスト変数 LOCATION の値は 26 に設定されます。

- ストリング 'Jürgen lives on Hegelstraße' 内の、文字 'ß' の位置を特定し、そのストリング内での位置を、CODEUNITS16 単位でホスト変数 LOCATION に設定する。

```
SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, CODEUNITS16)
```

  ホスト変数 LOCATION の値は 26 に設定されます。

- ストリング 'Jürgen lives on Hegelstraße' 内の、文字 'ß' の位置を特定し、そのストリング内での位置を、OCTETS でホスト変数 LOCATION に設定する。

```
SET :LOCATION = LOCATE('ß', 'Jürgen lives on Hegelstraße', 1, OCTETS)
```

  ホスト変数 LOCATION の値は 27 に設定されます。

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽の ト音記号、'~' はスペースなしで続く表記のチルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

変数 UTF8_VAR に、ストリングの UTF-8 表現が格納されると想定します。

```
SELECT LOCATE('~', UTF8_VAR, CODEUNITS16),
  LOCATE('~', UTF8_VAR, CODEUNITS32),
  LOCATE('~', UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

それぞれ、値 4、3、6 を戻します。

変数 UTF16_VAR に、ストリングの UTF-16BE 表現が格納されると想定します。

```
SELECT LOCATE('~', UTF16_VAR, CODEUNITS16),
   LOCATE('~', UTF16_VAR, CODEUNITS32),
   LOCATE('~', UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

それぞれ、値 4、3、7 を戻します。

- 大/小文字を区別しない照合 UCA500R1_LEN_S1 で作成された Unicode データベース内で、'The quick brown fox' という句の中の 'Brown' という語の位置を検索します。

```
SET :LOCATION = LOCATE('Brown', 'The quick brown fox', CODEUNITS16)
```

ホスト変数 LOCATION の値は 11 に設定されます。

LOG10

スキーマは SYSIBM です。 (LOG10 関数の SYSFUN バージョンは引き続き使用可能です。)

LOG10 関数は、数値の常用対数 (底 10) を戻します。

引数は、組み込み数値データ・タイプの値を返す式でなければなりません。引数が 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で実行されます。それ以外の場合は、関数による処理のために引数が倍精度浮動小数点数に変換されます。引数の値は、ゼロより大きくなければなりません (SQLSTATE 22003)。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点数になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

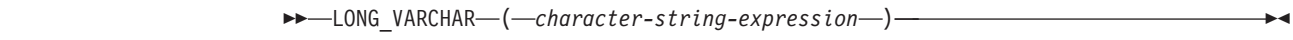
例:

- L は、31.62 の値を持つ DECIMAL(4,2) ホスト変数であると仮定します。

```
VALUES LOG10(:L)
```

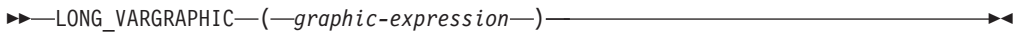
これは DOUBLE 値 +1.49996186559619E+000 を戻します。

LONG_VARCHAR



LONG_VARCHAR 関数は非推奨になっており、将来のリリースで除去される可能性があります。この関数は以前の DB2 バージョンと互換性があります。

LONG_VARGRAPHIC



LONG_VARGRAPHIC 関数は非推奨になっており、将来のリリースで除去される可能性があります。この関数は以前の DB2 バージョンと互換性があります。

LOWER

スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、 LONG VARCHAR 引数と CLOB 引数のサポートが引き続き有効です。)

LOWER 関数は、すべての SBCS 文字が小文字に変換されたストリングを戻します。 つまり、文字 A から Z は文字 a から z に変換され、その他の文字は小文字に相当するものがあればその文字に変換されます。例えば、コード・ページ 850 では、É は é にマップされます。結果の文字のコード・ポイント長が、元の文字のコード・ポイント長と同じでない場合、元の文字は変換されません。必ずすべての文字が変換されるわけではないので、LOWER(UPPER(*string-expression*)) の結果が LOWER(*string-expression*) と同じになるとは限りません。

引数は、値が CHAR または VARCHAR データ・タイプの式でなければなりません。

関数の結果のデータ・タイプと長さ属性は、引数と同じになります。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

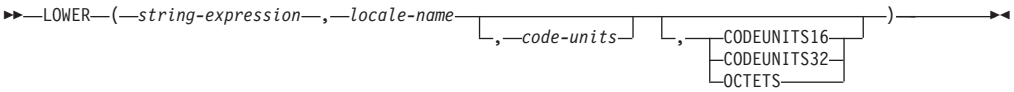
例:

EMPLOYEE 表の列 JOB の値で使用されている文字を小文字に変換します。

```
SELECT LOWER(JOB)
  FROM EMPLOYEE
  WHERE EMPNO = '000020';
```

結果は、値 'manager' になります。

LOWER (ロケール依存)



スキーマは SYSIBM です。

LOWER 関数は、指定したロケールに関連付けられた規則を使用して、すべての文字が小文字に変換されたストリングを戻します。

string-expression
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC ストリングを戻す式。*string-expression* が CHAR または VARCHAR の場合、この式は FOR BIT DATA であってはなりません (SQLSTATE 42815)。

locale-name
小文字への変換の規則を定義する、ロケールを指定する文字定数。*locale-name* の値は大/小文字の区別がなく、有効なロケールでなければなりません (SQLSTATE 42815)。有効なロケールとその命名については、『SQL および XQuery のロケール名』を参照してください。

code-units
結果内のコード単位の数を指定する整数定数。指定する場合、*code-units* は、結果が文字データの場合は 1 から 32672 までの間の整数、結果がグラフィック・データの場合は 1 から 16336 までの間の整数でなければなりません (SQLSTATE 42815)。 *code-units* が明示的に指定されないと、暗黙のうちに *string-expression* の長さ属性になります。 OCTETS が指定され、結果が GRAPHIC データである場合、*code-units* の値は偶数でなければなりません (SQLSTATE 428GC)。

CODEUNITS16、CODEUNITS32、または OCTETS
code-units のストリング単位を指定します。

CODEUNITS16 は、*code-units* が 16 ビット UTF-16 コード単位で表現されることを指定します。CODEUNITS32 は、*code-units* が 32 ビット UTF-32 コード単位で表現されることを指定します。OCTETS は、*code-units* がバイト単位で表現されることを指定します。

ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果が GRAPHIC データであれば、*code-units* は 2 バイト単位で表現されます。それ以外の場合は、バイト単位で表現されます。CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

関数の結果は、*string-expression* が CHAR または VARCHAR の場合は VARCHAR になり、*string-expression* が GRAPHIC または VARGRAPHIC の場合は VARGRAPHIC になります。

結果の長さ属性は、以下の表に示すように、*code-units* の暗黙的または明示的な値、暗黙的または明示的なストリング単位、および結果のデータ・タイプによって決まります。

表 34. ストリング単位および結果タイプの関数としての、LOWER の結果の長さ属性

ストリング単位	文字結果タイプ	GRAPHIC 結果タイプ
CODEUNITS16	MIN(*code-units* * 3, 32672)	*code-units*
CODEUNITS32	MIN(*code-units* * 4, 32672)	MIN(*code-units* * 2, 16336)
OCTETS	*code-units*	MIN(*code-units* / 2, 16336)

結果の実際の長さは、*string-expression* の長さより大きくなる場合があります。結果の実際の長さが結果の長さ属性より大きい場合は、エラーが返されます (SQLSTATE 42815)。結果内のコード単位の数が *code-units* の値を超えた場合は、エラーが返されます (SQLSTATE 42815)。

string-expression が UTF-16 ではない場合、この関数は *string-expression* の UTF-16 へのコード・ページ変換を実行し、次いで結果を UTF-16 から *string-expression* のコード・ページに変換します。どちらかのコード・ページ変換の結果として最低 1 つの置換文字が生じた場合、結果には置換文字が含まれることになり、警告が返されて (SQLSTATE 01517)、SQLCA の警告フラグ SQLWARN8 が「W」に設定されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- EMPLOYEE 表の列 JOB の値で使用されている文字を小文字に変換します。

```
SELECT LOWER(JOB, 'en_US')
  FROM EMPLOYEE
  WHERE EMPNO = '000020'
```

  結果は、値 'manager' になります。

- トルコ語のストリング内のすべての「I」文字について小文字を検索します。

```
VALUES LOWER('I■ıi', 'tr_TR', CODEUNITS16)
```

  結果は、'ıiii' というストリングになります。

LTRIM

スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、 LONG VARCHAR 引数と CLOB 引数のサポートが引き続き有効です。)

LTRIM 関数は、*string-expression* の先頭からブランクを除去します。

引数には CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC データ・タイプを使用できます。

- 引数が DBCS または EUC データベースの GRAPHIC ストリングである場合は、先行 2 バイト・ブランク文字が除去されます。
- 引数が Unicode データベースの GRAPHIC ストリングである場合は、先行 UCS-2 ブランクが除去されます。
- それ以外の場合は、先行 1 バイト・ブランクが除去されます。

この関数の結果のデータ・タイプは次のとおりです。

- *string-expression* のデータ・タイプが VARCHAR または CHAR の場合は VARCHAR になります。
- *string-expression* のデータ・タイプが VARGRAPHIC または GRAPHIC の場合は VARGRAPHIC になります。

戻される型の長さパラメーターは、引数のデータ・タイプの長さパラメーターと同じになります。

結果が文字ストリングである場合の実際の長さは、除去されるブランク文字のバイト数を *string-expression* から引いた値になります。結果が GRAPHIC ストリングである場合の実際の長さは、除去される 2 バイト・ブランク文字の数を *string-expression* から引いた値 (2 バイト文字単位) になります。すべての文字が除去された場合、結果は空になり、可変長ストリング (長さゼロ) が戻されます。

引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

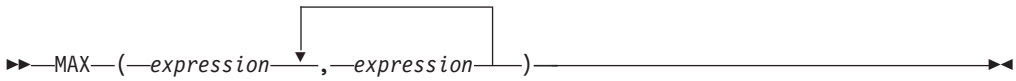
例:

ホスト変数 HELLO が CHAR(9) と定義されており、値は 'Hello' であるものとします。

```
VALUES LTRIM(:HELLO)
```

結果は 'Hello' になります。

MAX



スキーマは SYSIBM です。

MAX 関数は、値の集合の最大値を戻します。

引数は互換性がなければならず、それぞれの引数は、次のもの以外のデータ・タイプの値を戻す式でなければなりません: ARRAY、LOB、LONG VARCHAR、LONG VARGRAPHIC、XML、これらのタイプのいずれかに基づく特殊タイプ、構造化タイプ (SQLSTATE 42815)。この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

選択された引数は、必要に応じて結果の属性に変換されます。結果の属性は、結果データ・タイプの規則に基づくすべてのオペランドにより決定されます。

この関数の結果は最大の引数値となります。少なくとも引数の 1 つが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

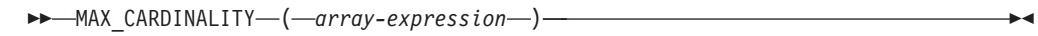
MAX スカラー関数は GREATEST スカラー関数の同義語です。

例:

従業員のボーナスを戻します。これは 500 または従業員の給与の 5% の、大きいほうの値です。

```
SELECT EMPNO, MAX(SALARY * 0.05, 500)
  FROM EMPLOYEE
```

MAX_CARDINALITY



スキーマは SYSIBM です。

MAX_CARDINALITY 関数は、配列に入れることができるエレメントの最大数を示すタイプ BIGINT の値を戻します。これは、ユーザー定義の配列タイプに対して CREATE TYPE ステートメントで指定したカーディナリティーです。

引数は、配列データ・タイプの SQL プロシージャーの変数またはパラメーターか、配列データ・タイプへのパラメーター・マーカーのキャスト仕様のいずれかとすることができます。

例:

タイプ PHONE_LIST が次のように定義されていると想定します。

```
CREATE TYPE PHONE_LIST
  AS INTEGER ARRAY[100]
```

配列 NUMBERS はタイプ PHONE_LIST です。次の SET ステートメントは、PHONE_LIST の定義に従って、変数 CARD に値 100 を割り当てます。

```
SET CARD = MAX_CARDINALITY(NUMBERS)
```

MICROSECOND

スキーマは SYSIBM です。

MICROSECOND 関数は、値のマイクロ秒の部分を戻します。

引数は、タイム・スタンプまたはタイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもないタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

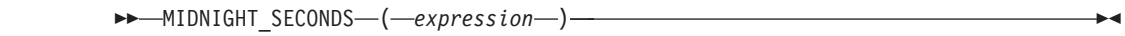
- 引数がタイム・スタンプまたはタイム・スタンプの有効なストリング表記の場合
  - 整数の範囲は 0 から 999999 となります。
- 引数が期間の場合
  - 結果には、-999999 から 999999 の間の整数値としてのマイクロ秒部分が反映されます。ゼロ以外の結果の符号は、引数と同じになります。

例:

- 表 TABLEA に、タイプが TIMESTAMP の TS1 および TS2 という 2 つの列が入っているものとします。 TS1 のマイクロ秒部分がゼロではなく、 TS1 と TS2 の秒部分が同じである行すべてを選択します。

```
SELECT * FROM TABLEA
  WHERE MICROSECOND(TS1) <> 0
    AND
  SECOND(TS1) = SECOND(TS2)
```

MIDNIGHT_SECONDS



スキーマは SYSFUN です。

午前 0 時から引数で指定した時刻値までの秒数を表す 0 から 86400 の範囲の整数値を戻します。

引数は、時刻またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例:

- 午前 0 時から 00:10:10 までの秒数、および午前 0 時から 13:10:10 までの秒数を求めます。

  ```
  VALUES (MIDNIGHT_SECONDS('00:10:10'), MIDNIGHT_SECONDS('13:10:10'))
  ```

  この例では、以下を戻します。

  ```
  1           2
  ----------- -----------
          610       47410
  ```

  1 分は 60 秒なので、午前 0 時から指定された時刻までは 610 秒です。 2 番目の例でも同じです。 1 時間は 3600 秒であり、1 分は 60 秒なので、指定された時刻から午前 0 時までは 47 410 秒です。

- 午前 0 時から 24:00:00 までの秒数、および午前 0 時から 00:00:00 までの秒数を求めます。

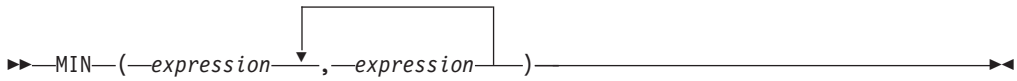
  ```
  VALUES (MIDNIGHT_SECONDS('24:00:00'), MIDNIGHT_SECONDS('00:00:00'))
  ```

  この例では、以下を戻します。

  ```
  1           2
  ----------- -----------
        86400           0
  ```

  これらの 2 つの値は、同じポイント・イン・タイムを表しているにもかかわらず、 MIDNIGHT_SECONDS 値が異なっていることに注意してください。

MIN



スキーマは SYSIBM です。

MIN 関数は、値の集合の最小値を戻します。

引数は互換性がなければならず、それぞれの引数は、次のもの以外のデータ・タイプの値を戻す式でなければなりません: ARRAY、LOB、LONG VARCHAR、LONG VARGRAPHIC、XML、これらのタイプのいずれかに基づく特殊タイプ、構造化タイプ (SQLSTATE 42815)。この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

選択された引数は、必要に応じて結果の属性に変換されます。結果の属性は、結果データ・タイプの規則に基づくすべてのオペランドにより決定されます。

この関数の結果は最小の引数値となります。少なくとも引数の 1 つが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

MIN スカラー関数は LEAST スカラー関数の同義語です。

例:

従業員のボーナスを戻します。これは 5000 または従業員の給与の 5% の、小さいほうの値です。

```
SELECT EMPNO, MIN(SALARY * 0.05, 5000)
  FROM EMPLOYEE
```

MINUTE

スキーマは SYSIBM です。

MINUTE 関数は、値の分の部分を戻します。

引数は、時刻、タイム・スタンプ、時刻期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が、時刻、タイム・スタンプ、または時刻やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の分の部分 (0 から 59 の整数) になります。
- 引数が時刻期間またはタイム・スタンプ期間の場合
  - 結果は、値の分の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例:

- CL_SCHED サンプル表を使用して、授業時間が 50 分未満のクラスを全選択します。

```
SELECT * FROM CL_SCHED
  WHERE HOUR(ENDING - STARTING) = 0
    AND
  MINUTE(ENDING - STARTING) < 50
```

MOD

スキーマは SYSFUN です。

最初の引数を 2 番目の引数で割った剰余を戻します。結果は、最初の引数が負である場合にのみ負になります。

関数の結果は次のとおりです。

- 両方の引数が SMALLINT の場合は SMALLINT になります。
- 一方の引数が INTEGER で、他方が INTEGER または SMALLINT の場合は INTEGER になります。
- 一方の引数が BIGINT で、他方が BIGINT、INTEGER または SMALLINT の場合は BIGINT になります。

結果は NULL 値になることがあります。いずれかの引数が NULL 値である場合、結果は NULL 値になります。

MONTH

スキーマは SYSIBM です。

MONTH 関数は、値の月の部分を戻します。

引数は、日付、タイム・スタンプ、日付期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の月の部分 (1 から 12 の整数) になります。
- 引数が日付期間またはタイム・スタンプ期間の場合
  - 結果は、値の月の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例:

- EMPLOYEE 表から、12 月に生まれた (BIRTHDATE) 社員の行を全選択します。

```
SELECT * FROM EMPLOYEE
  WHERE MONTH(BIRTHDATE) = 12
```

MONTHNAME

スキーマは SYSFUN です。

データベースの開始時点のロケールに基づいて、引数の月の部分の月名から成る大文字小文字混合文字ストリング (たとえば、 January) を戻します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は VARCHAR(100) です。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

MULTIPLY_ALT

スキーマは SYSIBM です。

MULTIPLY_ALT スカラー関数は、2 つの引数の積を 10 進数として返します。これは、引数の精度の合計が 31 を超えるときに特に合わせて、乗算の演算子の代わりとして用意されています。

引数には組み込み数値データ・タイプ (DECIMAL、BIGINT、INTEGER、または SMALLINT) を指定できます。

関数の結果は DECIMAL です。結果の精度と位取りは、以下のように決定されます。記号 *p* および *s* を使用して最初の引数の精度と位取りを、記号 *p'* および *s'* を使用して 2 番目の引数の精度と位取りを指定します。

- 精度は MIN(31, *p* + *p'*)
- 位取りは:
  - 両方の引数が 0 の場合は 0
  - *p* + *p'* が 31 以下であれば MIN(31, *s* + *s'*)
  - *p* + *p'* が 31 より大きい場合は MAX(MIN(3, *s* + *s'*), 31 - (*p* - *s* + *p'* - *s'*))

少なくとも 1 つの引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数の 1 つが NULL の場合、結果は NULL 値になります。

少なくとも 3 の位取りが必要で、精度の合計が 31 を超えるような 10 進数の計算を実行するときには、乗算演算子ではなく MULTIPLY_ALT 関数の使用が推奨されます。このような場合、内部計算が実行されるため、オーバーフローが回避されます。最終結果は、位取りを合わせるために必要に応じて切り捨てを使用して、結果データ・タイプに割り当てられます。最終結果のオーバーフローは、位取りが 3 のときはまだ起こり得ることに注意してください。

以下は、MULTIPLY_ALT と乗算演算子を使用した結果タイプの比較の例です。

引数タイプ 1	引数タイプ 2	MULTIPLY_ALT を使用した結果	乗算演算子を使用した結果
DECIMAL(31,3)	DECIMAL(15,8)	DECIMAL(31,3)	DECIMAL(31,11)
DECIMAL(26,23)	DECIMAL(10,1)	DECIMAL(31,19)	DECIMAL(31,24)
DECIMAL(18,17)	DECIMAL(20,19)	DECIMAL(31,29)	DECIMAL(31,31)
DECIMAL(16,3)	DECIMAL(17,8)	DECIMAL(31,9)	DECIMAL(31,11)
DECIMAL(26,5)	DECIMAL(11,0)	DECIMAL(31,3)	DECIMAL(31,5)
DECIMAL(21,1)	DECIMAL(15,1)	DECIMAL(31,2)	DECIMAL(31,2)

例:

最初の引数のデータ・タイプが DECIMAL(26,3)、 2 番目の引数のデータ・タイプが DECIMAL(9,8) の 2 つの値を乗算します。結果のデータ・タイプは DECIMAL(31,7) です。

```
values multiply_alt(98765432109876543210987.654,5.43210987)
1
--------------------------------
536504678578875294857887.5277415
```

これら 2 つの数値の積の全体は 536504678578875294857887.52774154498 ですが、最後の 4 桁は、結果のデータ・タイプの位取りに一致させるために切り捨てられます。同じ値を使って乗算演算子を使用すると、算術オーバーフローが発生します。これは、結果のデータ・タイプが DECIMAL(31,11) で、結果の値の小数点の左側が 24 桁になるものの、結果のデータ・タイプが 20 桁しかサポートしないためです。

NORMALIZE_DECFLOAT

スキーマは SYSIBM です。

NORMALIZE_DECFLOAT 関数は、最も単純な形式に設定された入力引数に等しい 10 進浮動小数点値を戻します。つまり、係数に後続ゼロがあるゼロ以外の数値であり、それらのゼロは除去します。これには、適切な 10 の累乗で係数を除算し、それに応じて指数を調整することによって数値を正規形で表記することが必要になる場合があります。ゼロ値は、指数が 0 に設定されています。

expression
組み込み数値データ・タイプの値を返す式。タイプ SMALLINT、INTEGER、REAL、DOUBLE、または DECIMAL(*p*, *s*) の引数は、処理のために DECFLOAT(16) に変換されますが、ここでは *p* <= 16 です。タイプ BIGINT または DECIMAL(*p*, *s*) の引数は、処理のために DECFLOAT(34) に変換されますが、ここでは *p* > 16 です。

10 進浮動小数点数への変換後の式のデータ・タイプが DECFLOAT(16) である場合、関数の結果は DECFLOAT(16) 値になります。そうでない場合には、関数の結果は DECFLOAT(34) 値になります。引数が特殊 10 進浮動小数点値である場合、結果は同じ特殊 10 進浮動小数点値です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例:

- 以下の例は、さまざまな 10 進浮動小数点値の入力を与えられた場合に NORMALIZE_DECFLOAT 関数によって戻される値を示しています。

```
NORMALIZE_DECFLOAT(DECFLOAT(2.1)) = 2.1
NORMALIZE_DECFLOAT(DECFLOAT(-2.0)) = -2
NORMALIZE_DECFLOAT(DECFLOAT(1.200)) = 1.2
NORMALIZE_DECFLOAT(DECFLOAT(-120)) = -1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(120.00)) = 1.2E+2
NORMALIZE_DECFLOAT(DECFLOAT(0.00)) = 0
NORMALIZE_DECFLOAT(-NAN) = -NaN
NORMALIZE_DECFLOAT(-INFINITY) = -Infinity
```

NULLIF

スキーマは SYSIBM です。

NULLIF 関数は、引数が等しい場合は NULL 値を戻し、それ以外の場合には最初の引数の値を戻します。

2 つの引数は比較可能でなければなりません。それらは、組み込みデータ・タイプ (LONG ストリング以外) か、まったく異なるデータ・タイプ (LONG ストリングに基づくもの以外) のいずれかにすることができます。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての互換データ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。) 結果の属性は、最初の引数の属性になります。

NULLIF(e1,e2) を使用した結果は、次の式を使用した結果と同じになります。

```
CASE WHEN e1=e2 THEN NULL ELSE e1 END
```

一方または両方の引数が NULL で、e1=e2 が不明と評価されると、 CASE 式はこれを真ではないと見なします。したがって、この場合、NULLIF は最初の引数の値を戻します。

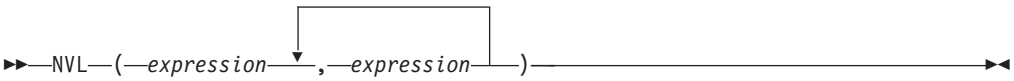
例:

- ホスト変数 PROFIT、CASH、および LOSSES のデータ・タイプが DECIMAL で、値がそれぞれ 4500.00、500.00、および 5000.00 であるとします。

  ```
  NULLIF (:PROFIT + :CASH , :LOSSES )
  ```

  NULL 値が戻されます。

NVL

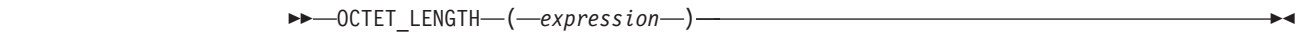


スキーマは SYSIBM です。

NVL 関数は、NULL 値以外の最初の引数を戻します。

NVL は COALESCE の同義語です。

OCTET_LENGTH



スキーマは SYSIBM です。

OCTET_LENGTH 関数は、オクテット (バイト) で *expression* の長さを戻します。

expression
組み込みストリング・データ・タイプの値を戻す式。

関数の結果は INTEGER になります。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

文字または GRAPHIC ストリングの長さには、末尾ブランクが含まれます。バイナリー・ストリングの長さには、2 進ゼロが含まれます。可変長ストリングの長さは、最大長ではなく実際の長さです。

移植性を高めるために、データ・タイプ DECIMAL(31) の結果を受け入れられるようにアプリケーションをコーディングしてください。

例:

- 表 T1 に、C1 という名前の GRAPHIC(10) 列があるとします。

  ```
  SELECT OCTET_LENGTH(C1) FROM T1
  ```

  戻り値は 20 です。
- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽の ト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

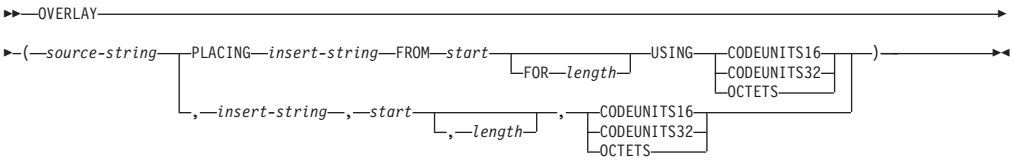
	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

  変数 UTF8_VAR および UTF16_VAR に、ストリングの UTF-8 表記および UTF-16BE 表記がそれぞれ含まれているとします。

  ```
  SELECT OCTET_LENGTH(UTF8_VAR),
    OCTET_LENGTH(UTF16_VAR)
  FROM SYSIBM.SYSDUMMY1
  ```

  これは値 9 および 12 をそれぞれ戻します。

OVERLAY



スキーマは SYSIBM です。

OVERLAY 関数は、*source-string* 内の *start* から始めて、指定されたコード単位の *length* が削除され、*insert-string* が挿入されたストリングを返します。

source-string
ソース・ストリングを指定する式。この式は、組み込みの文字ストリング・データ・タイプまたは GRAPHIC ストリング・データ・タイプの値を戻す必要があります (SQLSTATE 42815)。

insert-string
source-string に挿入するストリングを指定する式。挿入の開始点は、*start* で指定する位置になります。この式は、*source-string* との互換性がある組み込みの文字ストリング・データ・タイプまたは GRAPHIC ストリング・データ・タイプの値を戻す必要があります (SQLSTATE 22011)。 *insert-string* のコード・ページが *source-string* のコード・ページと異なる場合は、*insert-string* は *source-string* のコード・ページに変換されます。

start
整数値を戻す式。この整数値では、ソース・ストリングから指定のバイトを削除する開始点を指定します (この開始点は、別のストリングの挿入を開始する開始点でもあります)。この整数値は、1 から、*source-string* の長さに 1 を加算した値までの範囲でなければなりません (SQLSTATE 42815)。OCTETS を指定した場合に、結果が GRAPHIC データであれば、値は、1 から、*source-string* の長さ属性の 2 倍に 1 を加算した値までの範囲にある奇数でなければなりません (SQLSTATE 428GC)。

length
ソース・ストリングから削除する (指定のストリング単位による) コード単位の数を指定する式。削除の開始点は、*start* で指定する位置になります。値は、正整数またはゼロでなければなりません (SQLSTATE 22011)。OCTETS が指定され、結果が GRAPHIC データである場合、値は偶数またはゼロでなければなりません (SQLSTATE 428GC)。

length を指定しないことは、1 の値を指定することと同じです。ただし、OCTETS を指定して結果が GRAPHIC データである場合を除きます (この場合は、*length* を指定しないことは 2 の値を指定することと同じです。)。

CODEUNITS16、CODEUNITS32、または OCTETS
start および *length* のストリング単位を指定します。

CODEUNITS16 は、*start* および *length* を 16 ビットの UTF-16 コード単位で表すことを指定します。 CODEUNITS32 は、*start* および *length* を 32 ビットの UTF-32 コード単位で表すことを指定します。 OCTETS は、*start* および *length* をバイト単位で表すことを指定します。

ストリング単位として CODEUNITS16 または CODEUNITS32 を指定した場合に、結果がバイナリー・ストリングまたはビット・データであれば、エラーが戻されます (SQLSTATE 428GC)。ストリング単位として OCTETS を指定した場合に、*insert-string* と *source-string* がバイナリー・ストリングであれば、エラーが戻されます (SQLSTATE 42815)。ストリング単位として OCTETS を指定すると、操作は、*source-string* のコード・ページで実行されます。
CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

結果のデータ・タイプは、*source-string* と *insert-string* のデータ・タイプによって異なります。サポートされているタイプの組み合わせを以下の表にまとめます。

表 35. source-string と insert-string のデータ・タイプに対応した関数結果のデータ・タイプ

source-string	*insert-string*	**結果**
CHAR または VARCHAR	CHAR または VARCHAR	VARCHAR
GRAPHIC または VARGRAPHIC	GRAPHIC または VARGRAPHIC	VARGRAPHIC
CLOB	CHAR、VARCHAR、または CLOB	CLOB
DBCLOB	GRAPHIC、VARGRAPHIC、または DBCLOB	DBCLOB
CHAR または VARCHAR	CHAR FOR BIT DATA または VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHAR FOR BIT DATA または VARCHAR FOR BIT DATA	CHAR、VARCHAR、CHAR FOR BIT DATA、または VARCHAR FOR BIT DATA	VARCHAR FOR BIT DATA
Unicode データベースの場合のみ:		
CHAR または VARCHAR	GRAPHIC または VARGRAPHIC	VARCHAR
GRAPHIC または VARGRAPHIC	CHAR または VARCHAR	VARGRAPHIC
CLOB	GRAPHIC、VARGRAPHIC、または DBCLOB	CLOB
DBCLOB	CHAR、VARCHAR、または CLOB	DBCLOB

source-string の長さを 0 にすることができます。この場合、(上記の *start* の境界が示すとおり) *start* が 1 でなければなりません。関数の結果は、*insert-string* のコピーになります。

insert-string の長さを 0 にすることも可能です。この結果として、*start* と *length* で指定したコード単位が *source-string* から削除されます。

結果の長さ属性は、*source-string* の長さ属性と *insert-string* を加算した値になります。結果の実際の長さは、A1 - MIN((A1 - V2 + 1), V3) + A4 で、各変数の意味は以下のとおりです。

- A1 は *source-string* の実際の長さ

- V2 は *start* の値
- V3 は *length* の値
- A4 は *insert-string* の実際の長さ

結果のストリングの実際の長さが戻りデータ・タイプの最大値を超える場合、エラーが戻されます (SQLSTATE 54006)。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- 'INSERTING' というストリングから、'INSISTING'、'INSISERTING'、'INSTING' というストリングを作成するために、既存のテキストの中にテキストを挿入します。

```
SELECT OVERLAY('INSERTING','IS',4,2,OCTETS),
  OVERLAY('INSERTING','IS',4,0,OCTETS),
  OVERLAY('INSERTING','',4,2,OCTETS)
  FROM SYSIBM.SYSDUMMY1
```

- 'INSERTING' というストリングから、'XXINSERTING'、'XXNSERTING'、'XXSERTING'、'XXERTING' というストリングを作成するために、1 という開始点を使用して既存のテキストの前にテキストを挿入します。

```
SELECT OVERLAY('INSERTING','XX',1,0,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,1,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,2,CODEUNITS16)),
  OVERLAY('INSERTING','XX',1,3,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- 'ABCABC' というストリングから、'ABCABCXX' というストリングを作成するために、既存のテキストの後にテキストを挿入します。ソース・ストリングの長さは 6 文字なので、開始点を 7 (つまり、ソース・ストリングの長さに 1 を加算した値) に設定します。

```
SELECT OVERLAY('ABCABC','XX',7,0,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- ストリング 'Hegelstraße' を 'Hegelstrasse' に変更します。

```
SELECT OVERLAY('Hegelstraße','ss',10,1,CODEUNITS16))
  FROM SYSIBM.SYSDUMMY1
```

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽の ト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

変数 UTF8_VAR および UTF16_VAR に、ストリングの UTF-8 表記および UTF-16BE 表記がそれぞれ含まれているとします。 OVERLAY 関数を使用して、'C' を Unicode ストリング '&N~AB' に挿入します。

```
SELECT OVERLAY(UTF8_VAR, 'C', 1, CODEUNITS16),
  OVERLAY(UTF8_VAR, 'C', 1, CODEUNITS32),
  OVERLAY(UTF8_VAR, 'C', 1, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

それぞれ値 'C?N~AB'、'CN~AB'、および 'CbbbN~AB' が返されます。ここで '?' は X'EDB49E' を表し、中間の UTF-16 形式の X'DD1E' に対応します。また 'bbb' により UTF-8 の不完全な文字 X'9D849E' が置き換えられます。

```
SELECT OVERLAY(UTF8_VAR, 'C', 5, CODEUNITS16),
  OVERLAY(UTF8_VAR, 'C', 5, CODEUNITS32),
  OVERLAY(UTF8_VAR, 'C', 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

それぞれ値 '&N~CB'、'&N~AC'、および '&N~AB' が返されます。

```
SELECT OVERLAY(UTF16_VAR, 'C', 1, CODEUNITS16),
  OVERLAY(UTF16_VAR, 'C', 1, CODEUNITS32)
FROM SYSIBM.SYSDUMMY1
```

それぞれ値 'C?N~AB' および 'CN~AB' が返されます。ここで '?' により、不一致の低位サロゲート U+DD1E が表されます。

```
SELECT OVERLAY(UTF16_VAR, 'C', 5, CODEUNITS16),
  OVERLAY(UTF16_VAR, 'C', 5, CODEUNITS32)
FROM SYSIBM.SYSDUMMY1
```

それぞれ値 '&N~CB' および '&N~AC' が返されます。

PARAMETER

PARAMETER 関数は、db2-fn:sqlquery 関数の呼び出しの一部として XQuery によって値が動的に提供される SQL ステートメント内の位置を表します。

スキーマは SYSIBM です。

integer-constant は、db2-fn:sqlquery の引数の値の位置のインデックスです。値の範囲は、1 から db2-fn:sqlquery SQL ステートメントで指定された引数の総数までででなければなりません (SQLSTATE 42815)。

PARAMETER 関数は、db2-fn:sqlquery 関数の呼び出しの一部として XQuery によって値が動的に提供される SQL ステートメント内の位置を表します。 PARAMETER 関数の引数は、db2-fn:sqlquery 関数の実行時に PARAMETER 関数に代入される値を決定します。 PARAMETER 関数によって提供される値は、同じ SQL ステートメント内で複数回参照できます。

この関数は、XQuery 式の db2-fn:sqlquery 関数のストリング・リテラル引数に含まれる全選択でのみ使用可能です (SQLSTATE 42887)。

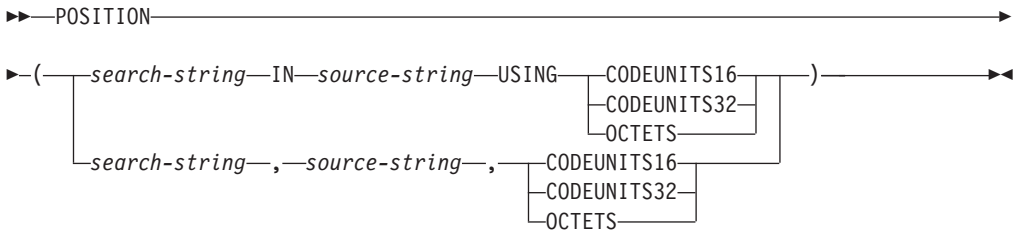
例:

以下の例の db2-fn:sqlquery 関数呼び出しでは、1 つの PARAMETER 関数呼び出しと XQuery 式 `$po/@OrderDate` (注文日属性) が使用されます。 PARAMETER 関数呼び出しは、注文日属性の値を戻します。

```
xquery
declare default element namespace "http://posample.org";
for $po in db2-fn:xmlcolumn('PURCHASEORDER.PORDER')/PurchaseOrder,
  $item in $po/item/partid
for $p in db2-fn:sqlquery(
  "select description from product where promostart < PARAMETER(1)",
    $po/@OrderDate )
where $p//@pid = $item
return
<RESULT>
  <PoNum>{data($po/@PoNum)}</PoNum>
  <PartID>{data($item)} </PartID>
  <PoDate>{data($po/@OrderDate)}</PoDate>
</RESULT>
```

この例は、プロモーション開始日より後に販売されたすべての部品に関する購入 ID、部品 ID、および購入日を戻します。

POSITION スカラー関数



スキーマは SYSIBM です。

POSITION 関数は、あるストリング (*search-string* と呼ばれる) の、別のストリング (*source-string* と呼ばれる) の中での、最初の出現箇所の開始位置を戻します。*search-string* が見つからず、いずれの引数も NULL でない場合、結果は 0 です。*search-string* が見つかった場合、結果として、1 から *source-string* の実際の長さまでの数が、明示的に指定されたストリング単位で表記されます。検索には、データベースの照合が使用されます。ただし、*search-string* または *source-string* が FOR BIT DATA で定義されている場合は除きます。その場合は、バイナリー比較によって検索が行われます。

source-string の実際の長さが 0 の場合、関数の結果は 0 です。*search-string* の実際の長さが 0 で、*source-string* が NULL 以外の場合、関数の結果は 1 です。

search-string
検索の対象となるストリングを指定する式。この式は、実際の長さが 4000 バイトを超えない、組み込み文字ストリング・データ・タイプ、GRAPHIC ストリング・データ・タイプ、またはバイナリー・ストリング・データ・タイプである値を戻さなければなりません。式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを (CONCAT または || を使用して) 連結する式
- SQL プロシージャー・パラメーター

これらの規則は、LIKE 述部の *pattern-expression* に関して記述されるものと同様になります。

source-string
その中で検索が行われるストリングを指定する式。この式は、組み込み文字ストリング・データ・タイプ、GRAPHIC ストリング・データ・タイプ、バイナリー・ストリング・データ・タイプの値を戻す必要があります。この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター

- ホスト変数 (ロケーター変数またはファイル参照変数を含む)
- スカラー関数
- ラージ・オブジェクトのロケーター
- 列名
- 上記のいずれかを (CONCAT または || を使用して) 連結する式

CODEUNITS16、CODEUNITS32、または OCTETS
結果のストリング単位を指定します。 CODEUNITS16 は、結果が 16 ビット UTF-16 コード単位で表現されることを指定します。 CODEUNITS32 は、結果が 32 ビット UTF-32 コード単位で表現されることを指定します。 OCTETS は、結果がバイト単位で表現されることを指定します。

ストリング単位が CODEUNITS16 または CODEUNITS32 と指定され、*search-string* または *source-string* がバイナリー・ストリングまたはビット・データである場合は、エラーが戻されます (SQLSTATE 428GC)。ストリング単位が OCTETS と指定され、*search-string* および *source-string* がバイナリー・ストリングである場合は、エラーが戻されます (SQLSTATE 42815)。

ロケールに依存する UCA ベースの照合がこの関数に使用される場合は、CODEUNITS16 オプションから最も適したパフォーマンスの特性を得られます。

CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

1 番目と 2 番目の引数は、互換性のあるストリング・タイプを持たなければなりません。互換性の詳細については、『ストリング変換についての規則』を参照してください。 Unicode データベースでは、一方のストリング引数が文字で (FOR BIT DATA ではない)、他方のストリング引数が GRAPHIC である場合、*search-string* は、処理のために *source-string* のデータ・タイプに変換されます。一方の引数が文字 FOR BIT DATA である場合、他方の引数は GRAPHIC であってはなりません (SQLSTATE 42846)。

この関数の結果は長精度整数 (large integer) です。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- ストリングを含む IN_TRAY 表内のすべての行の NOTE_TEXT 列内で、RECEIVED 列、SUBJECT 列、およびストリング 'GOOD BEER' の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, POSITION('GOOD BEER', NOTE_TEXT, OCTETS)
FROM IN_TRAY
WHERE POSITION('GOOD BEER', NOTE_TEXT, OCTETS) <> 0
```

- 文字 'ß' の位置をストリング 'Jürgen lives on Hegelstraße' から見つけ、ストリング内で CODEUNITS32 単位を尺度として、ホスト変数 LOCATION をその位置で設定します。

```
SET :LOCATION = POSITION(
  'ß', 'Jürgen lives on Hegelstraße', CODEUNITS32
)
```

ホスト変数 LOCATION の値は 26 に設定されます。

- 文字 'ß' の位置をストリング 'Jürgen lives on Hegelstraß' から見つけ、ストリング内で OCTETS を尺度として、ホスト変数 LOCATION をその位置で設定します。

```
SET :LOCATION = POSITION(
  'ß', 'Jürgen lives on Hegelstraße', OCTETS
)
```

ホスト変数 LOCATION の値は 27 に設定されます。

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽のト音記号、'~' はスペースなしで続く表記のチルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

変数 UTF8_VAR に、ストリングの UTF-8 表現が格納されると想定します。

```
SELECT POSITION('N', UTF8_VAR, CODEUNITS16),
  POSITION('N', UTF8_VAR, CODEUNITS32),
  POSITION('N', UTF8_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

これは値 3、2、および 5 をそれぞれ戻します。

変数 UTF16_VAR に、ストリングの UTF-16BE 表現が格納されると想定します。

```
SELECT POSITION('B', UTF16_VAR, CODEUNITS16),
   POSITION('B', UTF16_VAR, CODEUNITS32),
   POSITION('B', UTF16_VAR, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

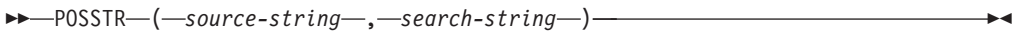
これは値 6、5、および 11 をそれぞれ戻します。

- 大/小文字を区別しない照合 UCA500R1_LEN_S1 で作成された Unicode データベース内で、'The quick brown fox' という句の中の 'Brown' という語の位置を検索します。

```
SET :LOCATION = POSITION('Brown', 'The quick brown fox', CODEUNITS16)
```

ホスト変数 LOCATION の値は 11 に設定されます。

POSSTR スカラー関数



スキーマは SYSIBM です。

POSSTR 関数は、あるストリング (*source-string*、ソース・ストリングと呼ばれる) の中で、別のストリング (*search-string*、検索ストリングと呼ばれる) の最初の出現箇所の開始位置を戻します。 *search-string* の位置を示す数値は、1 から始まります (0 ではない)。

この関数の結果は長精度整数 (large integer) です。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

source-string
探索が行われるロケーションとしてのソース・ストリングを指定します。

この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数 (ロケーター変数またはファイル参照変数を含む)
- スカラー関数
- ラージ・オブジェクトのロケーター
- 列名
- 上記のいずれかを連結する式

search-string
検索対象のストリングを指定する式。

この式は、以下のいずれかによって指定できます。

- 定数
- 特殊レジスター
- グローバル変数
- ホスト変数
- 上記のいずれかをオペランドとするスカラー関数
- 上記のいずれかを連結する式
- SQL プロシージャー・パラメーター

以下の制約があります。

- 式のエレメントに、LONG VARCHAR、CLOB、LONG VARGRAPHIC、または DBCLOB のタイプを使うことはできません。また、BLOB ファイル参照変数は使用できません。
- *search-string* の実際の長さは 4000 バイトを超えてはなりません。

以下に、無効なストリング式またはストリングの例を示します。

- SQL ユーザー定義関数パラメーター

- トリガー遷移変数
- 動的コンパウンド・ステートメント内のローカル変数

Unicode データベースでは、一方の引数が文字で (FOR BIT DATA ではない)、他方の引数が GRAPHIC である場合、*search-string* は、処理のために *source-string* のデータ・タイプに変換されます。一方の引数が文字 FOR BIT DATA である場合、他方の引数は GRAPHIC であってはなりません (SQLSTATE 42846)。

search-string と *source-string* には、いずれも、ゼロ個以上の連続した位置があります。ストリングが文字ストリングまたはバイナリー・ストリングの場合、 1 つの位置は 1 バイトを表します。ストリングが GRAPHIC ストリングの場合、位置は GRAPHIC (DBCS) 文字になります。

POSSTR 関数は混合データ・ストリングを受け入れます。ただし、 POSSTR は、厳密にバイト・カウント単位で計算し、データベース照合と、1 バイト文字とマルチバイト文字の間の変更は感知しません。

以下の規則が適用されます。

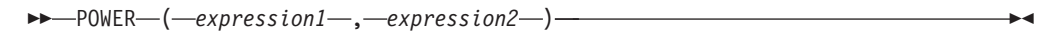
- *source-string* と *search-string* のデータ・タイプには、互換性がある必要があります。そうでない場合、エラーになります (SQLSTATE 42884)。
  - *source-string* が文字ストリングの場合、 *search-string* は CLOB または LONG VARCHAR 以外の文字ストリングでなければならず、実際の長さが 32672 バイト以下でなければなりません。
  - *source-string* が GRAPHIC ストリングの場合、 *search-string* は DBCLOB または LONG VARGRAPHIC 以外の GRAPHIC ストリングでなければならず、実際の長さが 2 バイト文字で 16336 個以下でなければなりません。
  - *source-string* がバイナリー・ストリングである場合、*search-string* は、実際の長さが 32672 バイト以下のバイナリー・ストリングでなければなりません。
- *search-string* の長さがゼロの場合、この関数によって戻される結果は 1 です。
- それ以外の場合は、次のとおりです。
  - *source-string* の長さがゼロの場合、関数によって戻される結果はゼロです。
  - それ以外の場合は、次のとおりです。
    - *search-string* が *source-string* の値のうち、連続する複数の位置の同じ長さのサブストリングに等しい場合、この関数によって戻される結果は、*source-string* 値の中でそのようなサブストリングのうち最初の開始位置になります。
    - それ以外の場合、この関数によって戻される結果は 0 です。

例

- IN_TRAY 表の項目のうち、 NOTE_TEXT 列に 'GOOD BEER' という語句が入っている項目について、 RECEIVED 列と SUBJECT 列、およびその語句の開始位置を選択します。

```
SELECT RECEIVED, SUBJECT, POSSTR(NOTE_TEXT, 'GOOD BEER')
  FROM IN_TRAY
  WHERE POSSTR(NOTE_TEXT, 'GOOD BEER') <> 0
```

POWER



スキーマは SYSIBM です。 (POWER 関数の SYSFUN バージョンは引き続き使用可能です。)

POWER 関数は、最初の引数を 2 番目の引数に累乗した結果を戻します。

引数は、任意の組み込み数値データ・タイプにすることができます。 DECIMAL および REAL の引数は倍精度浮動小数点数に変換されます。いずれかの引数が 10 進浮動小数点数の場合は、関数による処理のために引数が DECFLOAT(34) に変換されます。

関数の結果は次のとおりです。

- 両方の引数が INTEGER または SMALLINT の場合は INTEGER になります。
- 一方の引数が BIGINT で、他方が BIGINT、INTEGER または SMALLINT の場合は BIGINT になります。
- 一方の引数が 10 進浮動小数点数の場合は DECFLOAT(34) になります。いずれかの引数が DECFLOAT で、以下の陳述の 1 つが真であれば、結果は NAN および無効演算条件になります。
  - 両方の引数がゼロである
  - 2 番目の引数に、ゼロ以外の小数部分がある
  - 2 番目の引数が 9 桁を超えている
  - 2 番目の引数が INFINITY である
- それ以外の場合は DOUBLE になります。

引数が特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。

結果は NULL 値になることがあります。いずれかの引数が NULL 値である場合、結果は NULL 値になります。

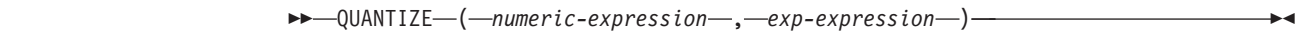
例:

- ホスト変数 HPOWER は、値が 3 の整数であると仮定します。

```
VALUES POWER(2,:HPOWER)
```

  これは値 8 を戻します。

QUANTIZE



スキーマは SYSIBM です。

QUANTIZE 関数は、値 (丸め以外) において等しく、*numeric-expression* と符号が等しく、*exp-expression* の指数と等しい指数を持つ、10 進浮動小数点値を戻します。桁数 (16 または 34) は、*numeric-expression* の桁数と同じです。

numeric-expression
組み込み数値データ・タイプの値を返す式。引数が 10 進浮動小数点値ではない場合、処理のために DECFLOAT(34) に変換されます。

exp-expression
組み込み数値データ・タイプの値を返す式。引数が 10 進浮動小数点値ではない場合、処理のために DECFLOAT(34) に変換されます。*exp-expression* が *numeric-expression* のスケール変更のパターン例として使用されます。*exp-expression* の符号および係数は、無視されます。

結果の係数は、*numeric-expression* の係数から派生します。必要な場合 (指数が増加している場合) には、丸められるか、10 の累乗で乗算されるか (指数が減少している場合)、または変更されないままです (指数が既に *exp-expression* の指数と等しい場合)。

CURRENT DECFLOAT ROUNDING MODE 特殊レジスターは、丸めモードを決定します。

10 進浮動小数点データ・タイプの他の算術演算とは異なり、量子化演算の後に係数の長さが *exp-expression* によって指定された精度より大きくなる場合、結果は NaN になり、警告が戻されます (SQLSTATE 0168D)。 これにより、警告条件がない場合には、QUANTIZE の結果の指数は常に *exp-expression* の指数と等しくなります。

- どちらかの引数が NaN の場合は、NaN が戻されます。
- どちらかの引数が sNaN の場合は、NaN が戻され、警告が戻されます (SQLSTATE 01565)。
- 両方の引数が無限大 (正または負) である場合は、最初の引数と同じ符号の無限大が戻されます。
- 一方の引数が無限大 (正または負) であり、他方の引数が無限大ではない場合、NaN が戻され、警告が戻されます (SQLSTATE 0168D)。

両方の引数が DECFLOAT(16) である場合、関数の結果は DECFLOAT(16) 値になります。そうでない場合には、関数の結果は DECFLOAT(34) 値になります。結果は NULL 値になることがあります。いずれかの引数が NULL 値である場合、結果は NULL 値になります。

例:

- 以下の例は、さまざまな 10 進浮動小数点値の入力が与えられ、ROUND_HALF_UP の丸めモードを取った場合に QUANTIZE 関数によって戻される値を示しています。

```
QUANTIZE(2.17, DECFLOAT(0.001)) = 2.170
QUANTIZE(2.17, DECFLOAT(0.01)) = 2.17
QUANTIZE(2.17, DECFLOAT(0.1)) = 2.2
QUANTIZE(2.17, DECFLOAT('1E+0')) = 2
QUANTIZE(2.17, DECFLOAT('1E+1')) = 0E+1
QUANTIZE(2, DECFLOAT(INFINITY)) = NaN    -- warning
QUANTIZE(0, DECFLOAT('1E+5')) = 0E+5
QUANTIZE(217, DECFLOAT('1E-1')) = 217.0
QUANTIZE(217, DECFLOAT('1E+0')) = 217
QUANTIZE(217, DECFLOAT('1E+1')) = 2.2E+2
QUANTIZE(217, DECFLOAT('1E+2')) = 2E+2
```

- 以下の例では、QUANTIZE 関数に対して値 -0 が返されます。 CHAR 関数が使用されているのは、クライアント・プログラムにより負符号 (-) が除去される可能性をなくすためです。

```
CHAR(QUANTIZE(-0.1, DECFLOAT(1))) = -0
```

QUARTER

スキーマは SYSFUN です。

引数に指定した日付が属する四半期を示す 1 から 4 の範囲の整数値を戻します。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

RADIANS

スキーマは SYSIBM です。 (RADIANS 関数の SYSFUN バージョンは引き続き使用可能です。)

RADIANS 関数は、度単位で表された引数のラジアン数を戻します。

引数は、任意の組み込み数値データ・タイプにすることができます。引数が 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で実行されます。それ以外の場合は、関数による処理のために引数が倍精度浮動小数点数に変換されます。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点数になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

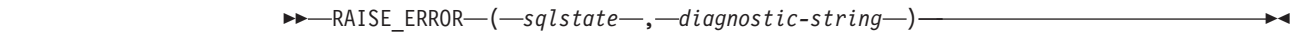
例:

- ホスト変数 HDEG は、値が 180 の INTEGER であると仮定します。以下のステートメント:

  ```
  VALUES RADIANS(:HDEG)
  ```

  は、値 +3.14159265358979E+000 を戻します。

RAISE_ERROR



スキーマは SYSIBM です。

RAISE_ERROR 関数は、指定された SQLSTATE、SQLCODE -438、および *diagnostic-string* のエラーが、この関数を備えたステートメントから戻されるようにします。 RAISE_ERROR 関数は、未定義データ・タイプでは常に NULL を戻します。Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

sqlstate
厳密に 5 バイトの文字ストリング。これは、長さ 5 と定義された CHAR 型、または長さ 5 以上と定義された VARCHAR 型でなければなりません。 *sqlstate* の値は、次のように、アプリケーション定義の SQLSTATE の規則に従っていなければなりません。

- 各文字は、数字 ('0' から '9')、またはアクセントのない大文字の英字 ('A' から 'Z') でなければなりません。
- SQLSTATE クラス (最初の 2 文字) は、 '00'、'01'、または '02' であってはなりません (これらの値はエラー・クラスではないので)。
- SQLSTATE クラス (最初の 2 文字) が文字 '0' から '6' または 'A' から 'H' で始まっている場合、サブクラス (最後の 3 文字) は 'I' から 'Z' の範囲の文字で始まっていなければなりません。
- SQLSTATE クラス (最初の 2 文字) が文字 '7'、'8'、'9'、または 'I' から 'Z' で始まっている場合、サブクラス (最後の 3 文字) として '0' から '9' または 'A' から 'Z' のいずれでも使用することができます。

SQLSTATE がこれらの規則に従っていない場合は、エラーになります (SQLSTATE 428B3)。

diagnostic-string
エラー条件を記述する最高 70 バイトの文字ストリングを戻すタイプ CHAR または VARCHAR の式。ストリングが 70 バイトを超える場合には切り捨てられます。

この関数を、結果データ・タイプの規則が適用されないコンテキスト (選択リストで単独の場合など) で使用するには、 Cast 指定を使用して、NULL 値の戻される値にデータ・タイプに割り当てる必要があります。 CASE 式は、RAISE_ERROR 関数を使う最適のロケーションといえます。

例:

従業員番号と学歴のリストを、学歴を Post Graduate、Graduate、および Diploma として示します。学歴が 20 を超える場合は、エラーになります。

```
SELECT EMPNO,
  CASE WHEN EDUCLVL < 16 THEN 'Diploma'
       WHEN EDUCLVL < 18 THEN 'Graduate'
       WHEN EDUCLVL < 21 THEN 'Post Graduate'
```

```
      ELSE RAISE_ERROR('70001',
        'EDUCLVL has a value greater than 20')
  END
FROM EMPLOYEE
```

RAND

スキーマは SYSFUN です。

RAND 関数は、0 から 1 の浮動小数点値を戻します。

式を指定すると、シード値として使用されます。式は、0 から 2147483647 までの値をもった組み込み SMALLINT または INTEGER データ・タイプでなければなりません。

結果のデータ・タイプは、倍精度の浮動小数点です。引数が NULL の場合、結果は NULL 値になります。

特定のシード値の場合、照会の実行のたびにその照会中の RAND 関数の特定のインスタンスに対して同じ一連の乱数が生成されます。シード値を指定しない場合は、同一セッション内での照会の実行のたびに別の一連の乱数が生成されます。セッションごとに異なる一連の乱数を生成するには、たとえば現在時刻に基づいてランダム・シードを指定します。

RAND は一律の結果を生じない関数です。

REAL

スキーマは SYSIBM です。

REAL 関数は、数値の単精度浮動小数点表記を戻します。

引数は、組み込み数値データ・タイプの値を返す式です。

関数の結果は単精度浮動小数点数になります。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

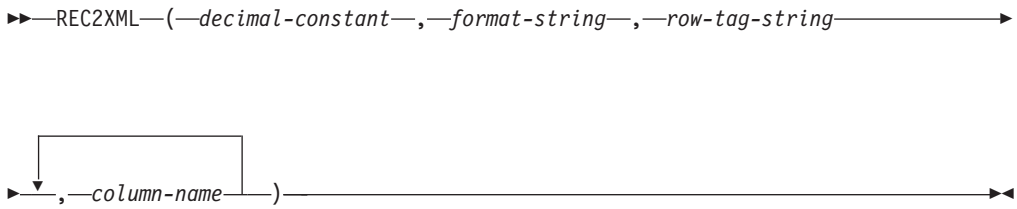
結果は、引数が単精度浮動小数点の列または変数に割り当てられた場合の結果と同じ数値になります。

例:

EMPLOYEE 表を使用して、歩合がゼロではない従業員の給与と歩合の比率を計算します。関係する列 (SALARY と COMM) のデータ・タイプは DECIMAL です。単精度浮動小数点の結果が必要です。したがって、除算が浮動小数点 (実際には倍精度) で実行されるように REAL が SALARY に適用され、次に単精度の浮動小数点で結果を戻すために REAL が式全体に適用されます。

```
SELECT EMPNO, REAL(REAL(SALARY)/COMM)
  FROM EMPLOYEE
  WHERE COMM > 0
```

REC2XML



スキーマは SYSIBM です。

REC2XML 関数は、XML タグで形式設定されて列名と列データを収めたストリングを戻します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

decimal-constant
列データ文字の置換用の拡張係数。この 10 進値は 0.0 より大きく、6.0 以下でなければなりません (SQLSTATE 42820)。

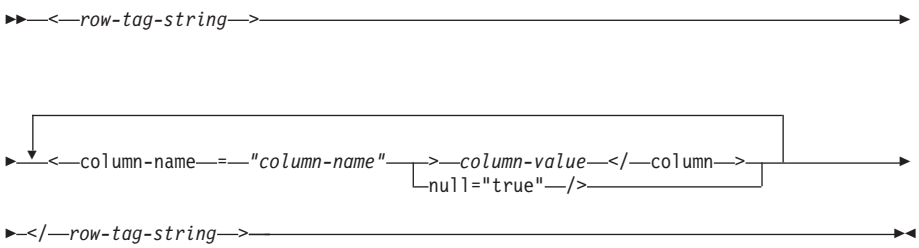
decimal-constant は、関数の結果の長さを計算するために使われます。文字データ・タイプのそれぞれの列ごとに、列の長さ属性が結果の長さに挿入される前に、長さ属性にこの拡張係数を掛けます。

拡張しないことを指定するには、値 1.0 を使用します。 1.0 より小さい値を指定すると、結果の長さが短く計算されます。結果ストリングの実際の長さが、関数の計算された結果の長さよりも長い場合には、エラーが発生します (SQLSTATE 22001)。

format-string
関数を実行する際、どのフォーマットを使用するかを指定するストリング定数。

format-string は大文字小文字を区別するため、以下に示す値が認識されるようにするには、大文字で指定する必要があります。

COLATTVAL または COLATTVAL_XML
これらのフォーマットは、列を属性値とするストリングを戻します。



列名は、有効な XML 属性値である場合と、そうでない場合があります。列名が有効な XML 属性値ではない場合、列名が結果ストリングに挿入される前に、列名の文字置換が行われます。

列の値は、有効な XML エレメント名である場合と、そうでない場合があります。 *format-string* に COLATTVAL を指定すると、列名が有効な XML エレメ

ント値ではない場合、列値が結果ストリングに挿入される前に、列値の文字置換が行われます。 *format-string* に COLATTVAL_XML を指定すると、列値の文字置換は行われません (ただし、列名の文字置換は行われます)。

row-tag-string
各行に使用するタグを指定するストリング定数。空ストリングを指定すると、値 'row' と想定されます。

1 つまたは複数のブランク文字の入ったストリングを指定すると、結果ストリングには、最初の *row-tag-string* も最後の *row-tag-string* も表示されません (不等号括弧の区切り文字を含む)。

column-name
表列の名前 (修飾子付きまたは修飾子なし)。列のデータ・タイプは、以下のいずれかでなければなりません (SQLSTATE 42815)。

- 数値 (SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE)
- 文字ストリング (CHAR、VARCHAR サブタイプ BIT DATA の文字ストリングは使用できません。)
- 日時 (DATE、TIME、TIMESTAMP)
- 上記のいずれかの型に基づくユーザー定義タイプ

同じ列名を 2 度以上指定することはできません (SQLSTATE 42734)。

関数の結果は VARCHAR です。最大長は 32672 バイトです (SQLSTATE 54006)。

以下のような呼び出しの場合、

REC2XML (dc, fs, rt, c_1, c_2, ..., c_n)

"fs" の値を "COLATTVAL" または "COLATTVAL_XML" のいずれかにすると、結果は次の式と同じになります。

```
'<' CONCAT rt CONCAT '>' CONCAT y1 CONCAT y2
CONCAT ... CONCAT yn CONCAT '</' CONCAT rt CONCAT '>'
```

ここで y_n は以下と同等です。

```
'<column name="' CONCAT xvcn CONCAT vn
```

さらに vn は以下と同等です。

```
'">' CONCAT rn CONCAT '</column>'
```

(列が非 NULL の場合)

```
'" null="true"/>'
```

(列値が NULL の場合)

xvc_n は、 c_n の列名のストリング表記と同等です。 479 ページの表 37 に示されたすべての文字は、対応する表記に置換されます。これによって、結果ストリングは必ず有効な XML 属性またはエレメント値のトークンになります。

r_n は、 479 ページの表 36 に示されているストリング表記と同等です。

表 36. 列値のストリング結果

c_n のデータ・タイプ	r_n
CHAR、VARCHAR	値はストリングです。 *format-string* が文字 "_XML" で終わらない場合、 c_n 内の各文字は、 表 37 に示される置換表記によって置換されます。長さ属性は、[dc] に [c_n の長さ属性] を乗算したものになります。
SMALLINT、 INTEGER、 BIGINT、 DECIMAL、 NUMERIC、 REAL、 DOUBLE	値は LTRIM(RTRIM(CHAR(c_n))) です。長さ属性は、CHAR(c_n) の結果の長さです。小数点文字は必ずピリオド ('.') 文字です。
DATE	値は CHAR(c_n,ISO) です。長さ属性は、CHAR(c_n,ISO) の結果の長さです。
TIME	値は CHAR(c_n,JIS) です。長さ属性は、CHAR(c_n,JIS) の結果の長さです。
TIMESTAMP	値は CHAR(c_n) です。長さ属性は、CHAR(c_n) の結果の長さです。

文字の置換:

format-string に指定される値によっては、列名を有効な XML 属性値にして、列値を有効な XML エレメント値にするために、列名と列値の一部の文字が置換されます。

表 37. XML 属性値およびエレメント値の文字置換

文字	置換
<	<
>	>
"	"
&	&
'	'

例:

注: REC2XML は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- サンプル・データベースの DEPARTMENT 表を使用して、部門 'D01' の部門表の行 (DEPTNAME 列と LOCATION 列を除く) を、 XML ストリングにフォーマット設定します。データの中には置換の必要な文字が入っていないため、拡張係数は 1.0 (拡張なし) です。さらに、この行の MGRNO 値が NULL であることに注意してください。

```
SELECT REC2XML (1.0, 'COLATTVAL', '', DEPTNO, MGRNO, ADMRDEPT)
  FROM DEPARTMENT
  WHERE DEPTNO = 'D01'
```

  この例は、以下の VARCHAR(117) ストリングを戻します。

```
<row>
<column name="DEPTNO">D01</column>
<column name="MGRNO" null="true"/>
<column name="ADMRDEPT">A00</column>
</row>
```

- 5 日制の大学のスケジュールで、'&43<FIE' という名前の授業を、 CLASS_CODE 列用の新しいフォーマットを使用して表 CL_SCHED に追加します。この例では REC2XML 関数を使用して、この新しい授業のデータの入った XML ストリングを形式設定します (授業の終了時刻を除く)。

  REC2XML 呼び出しの長さ属性 (下記を参照) に拡張係数 1.0 を掛けて、 128 とします ('<row>' と '</row>' のオーバーヘッドに 11、列名に 21、 '<column name='、'>'、'</column>'、二重引用符に合わせて 75、 CLASS_CODE データに 7、DAY データに 6、STARTING データに 8)。文字 '&' および '<' は置換されるので、拡張係数 1.0 では不十分でしょう。関数の長さ属性は、新しいフォーマットの CLASS_CODE データ用に 7 バイトから 14 バイトへの増加をサポートする必要があるでしょう。

  しかし、DAY 値が決して 1 桁より多くならないことがわかっているので、使用されない余分な 5 単位の長さが合計に加えられます。したがって、2 の増加のみを拡張で扱えばいいことになります。引数リストの中では CLASS_CODE が唯一の文字ストリング列ですから、拡張係数が適用される列データはこれだけです。長さを 2 だけ増加させるには、拡張係数 9/7 (約 1.2857) が必要でしょう。そこで、拡張係数 1.3 を使用します。

  ```
  SELECT REC2XML (1.3, 'COLATTVAL', 'record', CLASS_CODE, DAY, STARTING)
    FROM CL_SCHED
    WHERE CLASS_CODE = '&43<FIE'
  ```

  この例は、以下の VARCHAR(167) ストリングを戻します。

  ```
  <record>
  <column name="CLASS_CODE">&amp;43&lt;FIE</column>
  <column name="DAY">5</column>
  <column name="STARTING">06:45:00</column>
  </record>
  ```

- サンプル・データベースの EMP_RESUME 表に、新しい行がいくつか追加されたとします。新しい行は、履歴書を有効な (妥当な) XML ストリングとして保管します。文字置換が実行されないように、*format-string* には COLATTVAL_XML を使用します。履歴書の長さは、3500 バイトを超えることはありません。以下の照会を使用して、EMP_RESUME 表から履歴書の XML バージョンを選択し、それを XML 文書の一部としてフォーマット設定します。

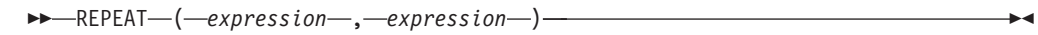
  ```
  SELECT REC2XML (1.0, 'COLATTVAL_XML', 'row', EMPNO, RESUME_XML)
    FROM (SELECT EMPNO, CAST(RESUME AS VARCHAR(3500)) AS RESUME_XML
      FROM EMP_RESUME
      WHERE RESUME_FORMAT = 'XML')
    AS EMP_RESUME_XML
  ```

  この例は、XML フォーマットの履歴書がある各従業員ごとに、行を戻します。戻される各行は、次のフォーマットのストリングになります。

  ```
  <row>
  <column name="EMPNO">{employee number}</column>
  <column name="RESUME_XML">{resume in XML}</column>
  </row>
  ```

ここで "{employee number}" は列の実際の EMPNO 値、 "{resume in XML}" は履歴書である実際の XML フラグメントのストリング値です。

REPEAT



スキーマは SYSFUN です。

2 番目の引数によって指定された回数だけ繰り返した最初の引数で構成される文字ストリングを戻します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

最初の引数は、文字ストリングまたはバイナリー・ストリング型です。 VARCHAR の場合、最大長は 4000 バイトです。 CLOB またはバイナリー・ストリングの場合、最大長は 1048576 バイトです。 2 番目の引数は SMALLINT または INTEGER にすることができます。

関数の結果は次のとおりです。

- 最初の引数が VARCHAR (4000 バイトを超えない) または CHAR である場合、VARCHAR(4000) になります。
- 最初の引数が CLOB または LONG VARCHAR の場合は CLOB(1M) になります。
- 最初の引数が BLOB の場合は BLOB(1M) になります。

結果は NULL 値になることがあります。いずれかの引数が NULL 値である場合、結果は NULL 値になります。

例:

- 句 'REPEAT THIS' を 5 回リストします。

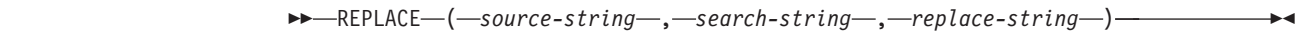
```
VALUES CHAR(REPEAT('REPEAT THIS', 5), 60)
```

この例では以下が戻されます。

```
1
------------------------------------------------------------
REPEAT THISREPEAT THISREPEAT THISREPEAT THISREPEAT THIS
```

前述のように、REPEAT 関数の出力は VARCHAR(4000) になります。上記の例の場合、 REPEAT の出力を 60 バイトまでに制限するために CHAR 関数が使用されています。

REPLACE



スキーマは SYSIBM です。 REPLACE 関数の SYSFUN バージョンは引き続き使用可能です。ただし、データベースの照合に依存しているわけではありません。

source-string 内に存在する *search-string* のすべての出現箇所を *replace-string* に置き換えます。*search-string* が *source-string* 内で検出されなければ、*search-string* が変更なしで戻されます。検索には、データベースの照合が使用されます。ただし、*source-string*、*search-string*、*replace-string* のいずれかが FOR BIT DATA で定義されている場合は除きます。その場合は、バイナリー比較によって検索が行われます。

source-string
ソース・ストリングを指定する式。この式は、CHAR、VARCHAR、GRAPHIC、VARGRAPHIC のいずれかのデータ・タイプの値を戻す必要があります。

search-string
ソース・ストリングから除去するストリングを指定する式。この式は、CHAR、VARCHAR、GRAPHIC、VARGRAPHIC のいずれかのデータ・タイプの値を戻す必要があります。

replace-string
置き換えストリングを指定する式。この式は、CHAR、VARCHAR、GRAPHIC、VARGRAPHIC のいずれかのデータ・タイプの値を戻す必要があります。この式が空ストリングであれば、ソース・ストリングから除去するストリングの代わりに置き換えるストリングが存在しないことになります。

各ストリングの実際の長さは、文字ストリングの場合は 32672 バイト以下、GRAPHIC ストリングの場合は 16336 バイト以下でなければなりません。この 3 つの引数のデータ・タイプには互換性がなければなりません。

source-string、*search-string*、*replace-string* のいずれかが FOR BIT DATA として定義されていれば、結果は VARCHAR FOR BIT DATA になります。*source-string* が文字ストリングであれば、結果は VARCHAR になります。*source-string* が GRAPHIC ストリングであれば、結果は VARGRAPHIC になります。1 つの引数が文字 FOR BIT DATA である場合、その他の引数は GRAPHIC であってはなりません (SQLSTATE 42846)。

結果の長さ属性は、引数によって異なります。

- *replace-string* の長さ属性が *search-string* の長さ属性以下であれば、結果の長さ属性は、*source-string* の長さ属性になります。
- *replace-string* の長さ属性が *search-string* の長さ属性より大きければ、結果の長さ属性は、結果のデータ・タイプによって以下のようになります。
  - VARCHAR の場合:
    - L1 < = 4000 であれば、結果の長さ属性は、`MIN(4000, (L3*(L1/L2)) + MOD(L1,L2))` になります。

- そうでなければ、結果の長さ属性は、MIN(32672, (L3*(L1/L2)) + MOD(L1,L2)) になります。

– VARGRAPHIC の場合:

- L1 < = 2000 であれば、結果の長さ属性は、MIN(2000, (L3*(L1/L2)) + MOD(L1,L2)) になります。
- そうでなければ、結果の長さ属性は、MIN(16336, (L3*(L1/L2)) + MOD(L1,L2)) になります。

ここで、

– L1 は *source-string* の長さ属性です。

– 検索ストリングがストリング定数である場合、L2 は *search-string* の長さ属性です。それ以外の場合、L2 は 1 です。

– L3 は *replace-string* の長さ属性です。

結果が文字ストリングであれば、結果の長さ属性は、32672 を超えてはなりません。結果が GRAPHIC ストリングであれば、結果の長さ属性は、16336 を超えてはなりません。

結果の実際の長さは、*source-string* の実際の長さに、 *replace-string* の実際の長さを *source-string* に存在する *search-string* のオカレンス数で乗算した値を加算し、*search-string* の長さを減算した値になります。

replace-string の実際の長さが戻りデータ・タイプの最大値を超えている場合は、エラーが戻されます。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- 'DINING' という語の文字 'N' のすべてのオカレンスを 'VID' で置き換えます。

```
VALUES CHAR (REPLACE ('DINING', 'N', 'VID'), 10)
```

結果は 'DIVIDIVIDG' というストリングになります。

- 大/小文字を区別しない照合 UCA500R1_LEN_S1 を使用する Unicode データベースで、'QUICK' という語を 'LARGE' という語で置き換えます。

```
VALUES REPLACE ('The quick brown fox', 'QUICK', 'LARGE')
```

結果はストリング 'The LARGE brown fox' になります。

RID_BIT および RID

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

RID_BIT および RID 関数は、行の行 ID (RID) をさまざまな形式で戻します。RID は、行を一意的に識別するために使用されます。それぞれの関数は、行に対して複数回呼び出されると異なる値を戻すことがあります。例えば、表に対して REORG ユーティリティーが実行された後に、 RID_BIT および RID 関数はそのユーティリティーの実行前に戻すはずであった値とは異なる値を、行に対して戻すことがあります。 RID_BIT および RID 関数は、決定論的なものではありません。RID_BIT 関数の結果には、RID 関数の場合とは異なり、それを不注意で異なる表に対して使用することを防止する表情報が含まれています。 RID 関数は、パーティション・データベース環境ではサポートされません。

table-designator
基本表、ビュー、またはネストされた表の式を一意的に識別します (SQLSTATE 42867)。 *table-designator* がビューまたはネストされた表の式を指定する場合、RID_BIT および RID 関数はビューまたはネストされた表の式の基本表の RID を戻します。指定されたビューまたはネストされた表の式は、外部副選択に 1 つだけの基本表を含んでいる必要があります (SQLSTATE 42867)。
table-designator は削除可能でなければなりません (SQLSTATE 42703)。削除可能なビューについて詳しくは、『CREATE VIEW』の『注』セクションを参照してください。

RID_BIT 関数の結果は VARCHAR (16) FOR BIT DATA です。結果は NULL 値の場合もあります。 RID 関数の結果は BIGINT です。結果は NULL 値の場合もあります。

注:

- アプリケーション内にオプティミスティック・ロックをインプリメントするには、 ROW CHANGE TOKEN 式によって戻される値を RID_BIT スカラー関数の引数として使用します。

例:

- EMPLOYEE 表から部門 20 の RID および従業員のラストネームを戻します。

```
SELECT RID_BIT (EMPLOYEE), ROW CHANGE TOKEN FOR EMPLOYEE, LASTNAME
  FROM EMPLOYEE
  WHERE DEPTNO = '20'
```

- 以下のように定義された表 EMP1 を想定します。

```
CREATE TABLE EMP1 (
  EMPNO   CHAR(6),
  NAME    CHAR(30),
  SALARY  DECIMAL(9,2),
  PICTURE BLOB(250K),
  RESUME  CLOB(32K)
)
```

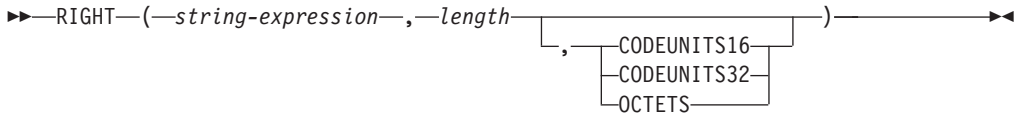
従業員番号 3500 に対応する行について、ホスト変数 HV_EMP_RID を RID_BIT 組み込みスカラー関数の値に設定し、 HV_EMP_RCT を ROW CHANGE TOKEN 式の値に設定します。

```
SELECT RID_BIT(EMP1), ROW CHANGE TOKEN FOR EMP1
  INTO :HV_EMP_RID, :HV_EMP_RCT FROM EMP1
  WHERE EMPNO = '3500'
```

その RID 値を使用して従業員およびユーザー定義関数 UPDATE_RESUME を識別して、従業員の給与を $1000 ずつ増加させてから従業員の履歴書を更新します。

```
UPDATE EMP1 SET
  SALARY = SALARY + 1000,
  RESUME = UPDATE_RESUME(:HV_RESUME)
WHERE RID_BIT(EMP1) = :HV_EMP_RID
AND ROW CHANGE TOKEN FOR EMP1 = :HV_EMP_RCT
```

RIGHT



スキーマは SYSIBM です。 RIGHT 関数の SYSFUN バージョンは引き続き使用可能です。

RIGHT 関数は、*string-expression* の右端にある長さ *length* のストリングを戻します (長さは、指定のストリング単位での長さになります)。*string-expression* が文字ストリングである場合、結果は文字ストリングです。 *string-expression* が GRAPHIC ストリングである場合、結果は GRAPHIC ストリングです。

string-expression
結果を取り出すストリングを指定する式。 この式は、組み込みの文字ストリング・データ・タイプまたは GRAPHIC ストリング・データ・タイプの値を戻す必要があります (SQLSTATE 42815)。*string-expression* のサブストリングは、*string-expression* のゼロ個以上の連続したコード・ポイントです。

length
結果の長さを指定する式。この結果は、組み込みの整数データ・タイプでなければなりません (SQLSTATE 42815)。値は、暗黙的または明示的に指定された単位で表される 0 から *string-expression* の長さまでの範囲になければなりません (SQLSTATE 22011)。ただし、例外が 1 つあります。ストリング単位を明示的に指定せずに値を定数として指定する場合、値は暗黙的ストリング単位での *string-expression* の長さ属性を超えることができます。 OCTETS を指定した場合に、結果が GRAPHIC データであれば、値は、0 から *string-expression* の長さ属性の 2 倍までの範囲にある偶数でなければなりません (SQLSTATE 428GC)。

CODEUNITS16、CODEUNITS32、または OCTETS
length のストリング単位を指定します。

CODEUNITS16 を指定すると、*length* は、16 ビットの UTF-16 コード単位の長さになります。CODEUNITS32 を指定すると、*length* は、32 ビットの UTF-32 コード単位の長さになります。OCTETS を指定すると、*length* は、バイト単位の長さになります。

ストリング単位として CODEUNITS16 または CODEUNITS32 を指定した場合に、*string-expression* がバイナリー・ストリングまたはビット・データであれば、エラーが戻されます (SQLSTATE 428GC)。ストリング単位として OCTETS を指定した場合に、*string-expression* が GRAPHIC ストリングであれば、*length* は偶数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 428GC)。ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果が GRAPHIC データであれば、*length* は 2 バイト単位の長さになり、それ以外の場合は、バイト単位になります。CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

string-expression の右側には必要な数の埋め込み文字が埋め込まれ、*string-expression* の指定のサブストリングが常に存在するようになります。埋め込み用の文字は、埋め込みが行われるコンテキストでストリングに埋め込みを適用するための文字と同じです。埋め込みの詳細については、『ストリング割り当て』と『割り当てと比較』を参照してください。

この関数の結果は可変長ストリングであり、その長さ属性は *length* とストリング単位を指定する方法によって異なります。定数を使用して *length* が指定されていない場合、またはストリング単位が明示的に指定された場合、長さ属性は *string-expression* の長さ属性と同じになります。定数を使用して *length* が指定され、ストリング単位が指定されていない場合、長さ属性は *length* と *string-expression* の長さ属性のうち長い方になります。結果のデータ・タイプは、次のように *string-expression* のデータ・タイプによって決まります。

- *string-expression* が CHAR または VARCHAR の場合は VARCHAR
- *string-expression* が CLOB の場合は CLOB
- *string-expression* が GRAPHIC または VARGRAPHIC の場合は VARGRAPHIC
- *string-expression* が DBCLOB の場合は DBCLOB

結果の実際の長さ (ストリング単位) は、*length* です。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- 変数 ALPHA の値が 'ABCDEF' であるとします。以下のステートメント:

  ```
  SELECT RIGHT(ALPHA,3)
    FROM SYSIBM.SYSDUMMY1
  ```

  ALPHA の右端の 3 文字である 'DEF' が戻されます。

- VARCHAR(50) で定義されている変数 NAME の値が 'KATIE AUSTIN'、整変数 LASTNAME_LEN の値が 6 であると想定して、以下のステートメントを実行します。

  ```
  SELECT RIGHT(NAME,LASTNAME_LEN)
    FROM SYSIBM.SYSDUMMY1
  ```

  値 'AUSTIN' が戻されます。

- 以下のステートメントは、長さゼロのストリングを戻します。

  ```
  SELECT RIGHT('ABCABC',0)
    FROM SYSIBM.SYSDUMMY1
  ```

- EMPLOYEE 表の FIRSTNME 列は、VARCHAR(12) として定義されています。'BROWN' というラストネームの従業員のファーストネームを検出し、そのファーストネームを 10 バイト・ストリングで戻すには、以下のようにします。

  ```
  SELECT RIGHT(FIRSTNME, 10)
    FROM EMPLOYEE
    WHERE LASTNAME = 'BROWN'
  ```

  'DAVID' という値の後に 5 つのブランク文字が埋め込まれた VARCHAR(12) ストリングが戻されます。

- Unicode データベースでは、FIRSTNAME が VARCHAR(12) の列になっています。その値の 1 つは、6 文字のストリング 'Jürgen' です。 FIRSTNAME が以下の値を持つ場合:

```
Function...                      Returns...

RIGHT(FIRSTNAME,5,CODEUNITS32)  'ürgen' -- x'C3BC7267656E'
RIGHT(FIRSTNAME,5,CODEUNITS16)  'ürgen' -- x'C3BC7267656E'
RIGHT(FIRSTNAME,5,OCTETS)       'rgen'  -- x'207267656E', a truncated string
```

- 以下の例は、Unicode ストリング '&N~AB' に対応します。 '&' は音楽のト音記号、'~' は結合チルド文字です。このストリングは、以下に、何種類かの Unicode エンコード方式で示してあります。

	'&'	'N'	'~'	'A'	'B'
UTF-8	X'F09D849E'	X'4E'	X'CC83'	X'41'	X'42'
UTF-16BE	X'D834DD1E'	X'004E'	X'0303'	X'0041'	X'0042'

20 バイトの長さ属性が指定されている変数 UTF8_VAR に、ストリングの UTF-8 表現が格納されると想定します。

```
SELECT RIGHT(UTF8_VAR, 2, CODEUNITS16),
  RIGHT(UTF8_VAR, 2, CODEUNITS32),
  RIGHT(UTF8_VAR, 2, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'AB'、'AB'、'AB' という値がそれぞれ戻されます。

```
SELECT RIGHT(UTF8_VAR, 5, CODEUNITS16),
  RIGHT(UTF8_VAR, 5, CODEUNITS32),
  RIGHT(UTF8_VAR, 5, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'?N~AB'、'&N~AB'、'N~AB' という値がそれぞれ戻されます ('?' は X'EDB49E' です)。

```
SELECT RIGHT(UTF8_VAR, 10, CODEUNITS16),
  RIGHT(UTF8_VAR, 10, CODEUNITS32),
  RIGHT(UTF8_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~ABbbbb'、'&N~ABbbbbb'、'&N~ABb' という値がそれぞれ戻されます ('b' はブランク文字です)。

20 コード単位の長さ属性が指定されている変数 UTF16_VAR に、ストリングの UTF-16BE 表現が格納されると想定します。

```
SELECT RIGHT(UTF16_VAR, 2, CODEUNITS16),
  RIGHT(UTF16_VAR, 2, CODEUNITS32),
  RIGHT(UTF16_VAR, 2, OCTETS))
FROM SYSIBM.SYSDUMMY1
```

'AB'、'AB'、'B' という値がそれぞれ戻されます。

```
SELECT RIGHT(UTF16_VAR, 5, CODEUNITS16),
  RIGHT(UTF16_VAR, 5, CODEUNITS32),
  RIGHT(UTF16_VAR, 6, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'?N~AB'、'&N~AB'、'~AB' という値がそれぞれ戻されます ('?' はスタンドアロン下位サロゲート X'DD1E' です)。

```
SELECT RIGHT(UTF16_VAR, 10, CODEUNITS16),
  RIGHT(UTF16_VAR, 10, CODEUNITS32),
  RIGHT(UTF16_VAR, 10, OCTETS)
FROM SYSIBM.SYSDUMMY1
```

'&N~ABbbbb'、'&N~ABbbbbb'、'?N~AB' という値がそれぞれ戻されます ('b' はブランク文字、'?' は X'DD1E' です)。

ROUND

スキーマは SYSIBM です。 (ROUND 関数の SYSFUN バージョンは引き続き使用可能です。)

ROUND 関数は、*expression2* が正の場合は小数点の右側、 *expression2* がゼロまたは負の場合は小数点の左側の、 *expression2* の桁数に丸められた *expression1* を返します。

expression1 が正の場合、5 以上の端数は、次に大きい正の数に丸められます。たとえば ROUND(3.5,0) = 4 です。 *expression1* が負の場合、5 以上の端数は、次に小さい負の数に丸められます。たとえば、ROUND(-3.5,0) = -4 のようになります。

expression1
組み込み数値データ・タイプの値を返す式。

expression2
短精度整数または長精度整数を返す式。*expression2* の値が負ではないときは、小数点の右側のその桁数に丸めることを指定します。 *expression2* の値が負のときは、小数点の左側の、 *expression2* の位置の絶対値に丸めることを指定します。

expression2 が負ではない場合、 expression1 は小数点の右側の、*expression2* の桁数の絶対値に丸められます。 *expression2* の値が *expression1* の位取りより大きい場合、 1 大きい精度を持つ結果値を除いて、値は変更されません。たとえば、 ROUND(748.58,5) = 748.58 のようになります。ここでの精度は 6 で、位取りは 2 のままです。

expression2 が負の場合、*expression1* は小数点の左側の、 *expression2* +1 の桁数の絶対値に丸められます。

負の *expression2* の絶対値が小数点の左側の桁数より大きい場合、結果は 0 になります。たとえば、ROUND(748.58,-4) = 0 のようになります。

結果のデータ・タイプおよび長さ属性は、最初の引数のデータ・タイプおよび長さ属性と同じになります。ただし、*expression1* が DECIMAL であり、精度が 31 より小さいときに精度が 1 増加する場合を除きます。

たとえば、データ・タイプが DECIMAL(5,2) の引数の結果は DECIMAL(6,2) になります。データ・タイプ DECIMAL(31,2) の引数は DECIMAL(31,2) になります。位取りは最初の引数の位取りと同じです。

引数が NULL か、または引数が 10 進浮動小数点数ではなく、データベースの構成で DFT_SQLMATHWARN が YES に設定されている場合、結果は NULL になります。引数が NULL の場合、結果は NULL 値になります。

この関数は、10 進浮動小数点引数についても、CURRENT DECFLOAT ROUNDING MODE 特殊レジスターの設定から影響を受けません。 ROUND の丸めの動作は、ROUND_HALF_UP の値に対応します。 10 進浮動小数点値の動作が CURRENT

DECFLOAT ROUNDING MODE 特殊レジスターにより指定された丸めモードに従うようにする場合には、代わりに QUANTIZE 関数を使用してください。

例:

値 873.726 を小数点以下 2, 1, 0, -1, -2, -3, および -4 桁に、それぞれ丸めます。

```
VALUES (
  ROUND(873.726, 2),
  ROUND(873.726, 1),
  ROUND(873.726, 0),
  ROUND(873.726,-1),
  ROUND(873.726,-2),
  ROUND(873.726,-3),
  ROUND(873.726,-4) )
```

この例は次の値を返します。

```
1         2         3         4         5         6         7
--------- --------- --------- --------- --------- --------- ---------
  873.730   873.700   874.000   870.000   900.000  1000.000     0.000
```

正と負の両方の数を使って計算します。

```
VALUES (
  ROUND(3.5, 0),
  ROUND(3.1, 0),
  ROUND(-3.1, 0),
  ROUND(-3.5,0) )
```

この例は次の値を返します。

```
1    2    3    4
---- ---- ---- ----
 4.0  3.0 -3.0 -4.0
```

10 進浮動小数点数 3.12350 の小数第 3 位までの丸めを計算します。

```
VALUES (
  ROUND(DECFLOAT('3.12350'), 3))
```

この例は次の値を返します。

```
1
-----
3.12400
```

RTRIM

スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、 LONG VARCHAR 引数と CLOB 引数のサポートが引き続き有効です。)

RTRIM 関数は、*string-expression* の末尾からブランクを除去します。

引数には CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC データ・タイプを使用できます。

- 引数が DBCS または EUC データベースの GRAPHIC ストリングである場合は、後続 2 バイト・ブランク文字が除去されます。
- 引数が Unicode データベースの GRAPHIC ストリングである場合は、後続 UCS-2 ブランクが除去されます。
- それ以外の場合は、後続 1 バイト・ブランクが除去されます。

この関数の結果のデータ・タイプは次のとおりです。

- *string-expression* のデータ・タイプが VARCHAR または CHAR の場合は VARCHAR になります。
- *string-expression* のデータ・タイプが VARGRAPHIC または GRAPHIC の場合は VARGRAPHIC になります。

戻される型の長さパラメーターは、引数のデータ・タイプの長さパラメーターと同じになります。

結果が文字ストリングである場合の実際の長さは、除去されるブランク文字のバイト数を *string-expression* から引いた値になります。結果が GRAPHIC ストリングである場合の実際の長さは、除去される 2 バイト・ブランク文字の数を *string-expression* から引いた値 (2 バイト文字単位) になります。すべての文字が除去された場合、結果は空になり、可変長ストリング (長さゼロ) が戻されます。

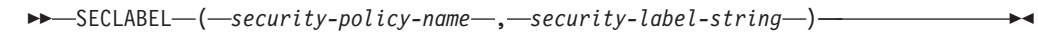
引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

例: ホスト変数 HELLO が CHAR(9) と定義されており、値は 'Hello ' であるものとします。

```
VALUES RTRIM(:HELLO)
```

結果は 'Hello' になります。

SECLABEL



スキーマは SYSIBM です。

SECLABEL 関数は、データ・タイプが DB2SECURITYLABEL の、名前の付いていないセキュリティー・ラベルを戻します。名前付きのセキュリティー・ラベルを作成する手間を省いて、特定のコンポーネント値を使ってセキュリティー・ラベルを挿入するには、SECLABEL 関数を使用します。

security-policy-name
現行のサーバーに存在するセキュリティー・ポリシーを指定するストリング (SQLSTATE 42704)。 ストリングは、文字または GRAPHIC ストリングの定数かホスト変数でなければなりません。

security-label-string
security-policy-name で指名されたセキュリティー・ポリシーに対してセキュリティー・ラベルの有効な表記を戻す式 (SQLSTATE 4274I)。式は組み込み CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC のいずれかのデータ・タイプの値を戻す必要があります。

例:

- 以下のステートメントは、CONTRIBUTIONS という名前のセキュリティー・ポリシーで保護されている表 REGIONS 内に行を挿入します。挿入する行のセキュリティー・ラベルは、SECLABEL 関数によって与えられます。セキュリティー・ポリシー CONTRIBUTIONS は、2 つのコンポーネントをもっています。与えられるセキュリティー・ラベルは、最初のコンポーネントに対してエレメント LIFE MEMBER をもち、2 番目のコンポーネントに対してエレメント BLUE および YELLOW を持ちます。

```
INSERT INTO REGIONS
  VALUES (SECLABEL('CONTRIBUTIONS', 'LIFE MEMBER:(BLUE,YELLOW)'),
    1, 'Northeast')
```

- 以下のステートメントは、3 つのコンポーネントを持つ TS_SECPOLICY という名前のセキュリティー・ポリシーで保護されている表 CASE_IDS 内に行を挿入します。セキュリティー・ラベルは、SECLABEL 関数によって提供されます。挿入されたセキュリティー・ラベルは、最初のコンポーネントに対してエレメント HIGH PROFILE、2 番目のコンポーネントに対して空値、3 番目のコンポーネントに対してエレメント G19 をもちます。

```
INSERT INTO CASE_IDS
  VALUES (SECLABEL('TS_SECPOLICY', 'HIGH PROFILE:():G19') , 3, 'KLB')
```

SECLABEL_BY_NAME

スキーマは SYSIBM です。

SECLABEL_BY_NAME 関数は、指定されたセキュリティー・ラベルを戻します。戻されたセキュリティー・ラベルは、DB2SECURITYLABEL のデータ・タイプをもっています。名前付きのセキュリティー・ラベルを挿入するには、この関数を使用します。

security-policy-name
現行のサーバーに存在するセキュリティー・ポリシーを指定するストリング (SQLSTATE 42704)。 ストリングは、文字または GRAPHIC ストリングの定数かホスト変数でなければなりません。

security-label-name
security-policy-name で指定されたセキュリティー・ポリシー用の、現行のサーバーに存在するセキュリティー・ラベルの名前を戻す式 (SQLSTATE 4274I)。 式は組み込み CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC のいずれかのデータ・タイプの値を戻す必要があります。

例:

- Tina というユーザーが、CONTRIBUTIONS という名前のセキュリティー・ポリシーで保護されている表 REGIONS 内に行を挿入しようとしています。 Tina は、EMPLOYEESECLABEL という名前のセキュリティー・ラベルで行を保護するつもりです。 CONTRIBUTIONS.EMPLOYEESECLABEL が不明の ID であるため、以下のステートメントは失敗します。

  ```
  INSERT INTO REGIONS
  VALUES (CONTRIBUTIONS.EMPLOYEESECLABEL, 1, 'Southwest')   -- incorrect
  ```

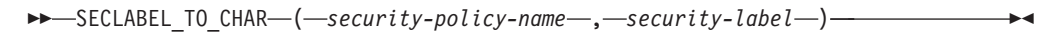
  最初の値がストリングであり、DB2SECURITYLABEL のデータ・タイプをもっていないので、以下のステートメントは失敗します。

  ```
  INSERT INTO REGIONS
  VALUES ('CONTRIBUTIONS.EMPLOYEESECLABEL', 1, 'Southwest') -- incorrect
  ```

  DB2SECURITYLABEL のデータ・タイプを持つセキュリティー・ラベルが SECLABEL_BY_NAME 関数から戻されるので、以下のステートメントは正常に完了します。

  ```
  INSERT INTO REGIONS
  VALUES (SECLABEL_BY_NAME('CONTRIBUTIONS', 'EMPLOYEESECLABEL'),
    1, 'Southwest')                                          -- correct
  ```

SECLABEL_TO_CHAR



スキーマは SYSIBM です。

SECLABEL_TO_CHAR 関数は、セキュリティー・ラベルを受け入れ、セキュリティー・ラベル内のすべてのエレメントを入れたストリングを戻します。このストリングは、セキュリティー・ラベル・ストリング・フォーマットになっています。

security-policy-name
現行のサーバーに存在するセキュリティー・ポリシーを指定するストリング (SQLSTATE 42704)。 ストリングは、文字または GRAPHIC ストリングの定数かホスト変数でなければなりません。

security-label
security-policy-name で指名されたセキュリティー・ポリシーに対して有効なセキュリティー・ラベル値を戻す式 (SQLSTATE 4274I)。式は組み込み SYSPROC.DB2SECURITYLABEL 特殊タイプの値を戻す必要があります。

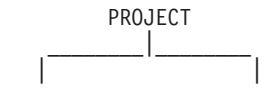
注

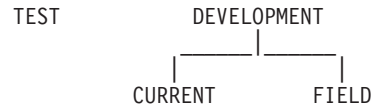
- ステートメントの許可 ID が、DB2SECURITYLABEL のデータ・タイプを持つ列から読み取られるセキュリティー・ラベルに対してこの関数を実行した場合、その許可 ID の LBAC 信用証明情報によって、関数の出力が影響を受けることがあります。そのような場合に、あるエレメントに対する読み取りアクセス権をその許可 ID がもっていないと、出力内にそのエレメントは組み入れられません。あるエレメントだけが入った (他のエレメントは入っていない) セキュリティー・ラベルによって保護されていたデータの読み取りが LBAC 信用証明情報によって許可されている場合、許可 ID はそのエレメントに読み取りアクセスすることができます。

  規則セット DB2LBACRULES の場合、読み取りアクセスできないエレメントを入れることができるのは、タイプ TREE のコンポーネントだけです。他のタイプのコンポーネントの場合、エレメントのいずれか 1 つが読み取りアクセスをブロックすると、行の読み取りがまったくできなくなります。したがって、この方法で除外されるエレメントを持つのは、タイプ TREE のコンポーネントだけです。

例:

- EMP 表には、RECORDNUM および LABEL の 2 つの列があります。RECORDNUM はデータ・タイプ INTEGER をもち、LABEL はタイプ DB2SECURITYLABEL をもっています。表 EMP は、セキュリティー・ポリシー DATA_ACCESSPOLICY によって保護されています。このポリシーは、DB2LBACRULES 規則セットを使用し、コンポーネントを 1 つだけ (TREE タイプの GROUPS) もっています。 GROUPS には、PROJECT、TEST、DEVELOPMENT、CURRENT、および FIELD という 5 つのエレメントがあります。以下のダイアグラムは、これらのエレメントの相互関係を示しています。





EMP 表には、以下のデータが収められています。

```
RECORDNUM  LABEL
---------  ----------------
        1  PROJECT
        2  (TEST, FIELD)
        3  (CURRENT, FIELD)
```

Djavan は、DEVELOPMENT エレメントだけが入った、読み取り用のセキュリティー・ラベルを保有しています。その意味するところは、Djavan は DEVELOPMENT、CURRENT、および FIELD エレメントに読み取りアクセスできるということです。

```
SELECT RECORDNUM, SECLABEL_TO_CHAR('DATA_ACCESSPOLICY', LABEL) FROM EMP
```

これは、以下のものを戻します。

```
RECORDNUM  LABEL
---------  ----------------
        2  FIELD
        3  (CURRENT, FIELD)
```

RECORDNUM 値が 1 の行は、出力には組み入れられません。なぜなら Djavan の LBAC 信用証明情報では、当人はこの行を読み取ることができないからです。RECORDNUM 値が 2 の行では、エレメント TEST は出力に組み入れられません。なぜなら、Djavan はこのエレメントに読み取りアクセスできないからです。セキュリティー・ラベル内の唯一のエレメントが TEST であったとしたら、Djavan はこの行にまったくアクセスできなかったはずです。 Djavan はエレメント CURRENT および FIELD には読み取りアクセスできるので、両方のエレメントが出力内に現れます。

次に、Djavan は、DB2LBACREADTREE 規則に対する免除を付与されます。それは、TREE タイプのコンポーネントのどのエレメントも、読み取りアクセスをブロックしないことを意味します。同じ照会が、以下を戻します。

```
RECORDNUM  LABEL
---------  ----------------
        1  PROJECT
        2  (TEST, FIELD)
        3  (CURRENT, FIELD)
```

今回は、すべての行とすべてのエレメントが出力に組み入れられています。なぜなら、免除によって Djavan はすべてのエレメントに読み取りアクセスできるようになったからです。

SECOND

スキーマは SYSIBM です。

SECOND 関数は、値の秒の部分を戻します。

引数は、時刻、タイム・スタンプ、時刻期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない時刻またはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

- 引数が、時刻、タイム・スタンプ、または時刻やタイム・スタンプの有効なストリング表記の場合
  - 結果は、値の秒の部分 (0 から 59 の整数) になります。
- 引数が時刻期間またはタイム・スタンプ期間の場合
  - 結果は、値の秒の部分 (-99 から 99 の整数) になります。ゼロ以外の結果の符号は、引数と同じになります。

例:

- ホスト変数 TIME_DUR (decimal(6,0)) の値が 153045 と想定します。

  ```
  SECOND(:TIME_DUR)
  ```

  戻り値は 45 です。
- RECEIVED (timestamp) 列に、 1988-12-25-17.12.30.000000 に相当する内部値が入っていると想定します。

  ```
  SECOND(RECEIVED)
  ```

  この例では 30 の値を戻します。

SIGN

スキーマは SYSIBM です。 (SIGN 関数の SYSFUN バージョンは引き続き使用可能です。)

引数の符号の標識を戻します。引数が負の場合は、-1 が戻されます。引数が -0 の 10 進浮動小数点値である場合、-0 の 10 進浮動小数点値が戻されます。引数がゼロの場合は、0 が戻されます。引数が正の場合には、1 が戻されます。

引数は、任意の組み込み数値データ・タイプにすることができます。 DECIMAL および REAL 値は、関数での処理用に倍精度浮動小数点数に変換されます。

関数の結果は次のとおりです。

- 引数が SMALLINT の場合は SMALLINT になります。
- 引数が INTEGER の場合は INTEGER になります。
- 引数が BIGINT の場合は BIGINT になります。
- 引数が DECFLOAT(*n*) の場合は DECFLOAT(*n*) になります。
- それ以外の場合は DOUBLE になります。

結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例:

- ホスト変数 PROFIT は、値が 50000 の長精度整数であると仮定します。

```
VALUES SIGN(:PROFIT)
```

これは値 1 を戻します。

SIN

スキーマは SYSIBM です。 (SIN 関数の SYSFUN バージョンは引き続き使用可能です。)

引数のサイン (正弦) の値を戻します。引数は、ラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

SINH

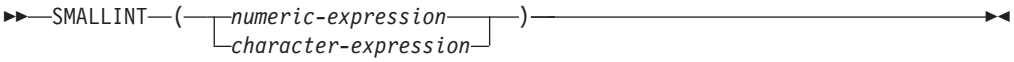
スキーマは SYSIBM です。

引数に対する双曲線サイン (正弦) の値を戻します。引数はラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

SMALLINT



スキーマは SYSIBM です。

SMALLINT 関数は、短精度整数 (small integer) 定数の形式の数値または文字ストリングの短精度整数 (small integer) 表記を戻します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

numeric-expression
組み込み数値データ・タイプの値を返す式。

引数が*数値式* の場合、結果は、引数を短精度整数 (small integer) の列または変数に割り当てた場合と同じ数値になります。引数の整数部分が短精度整数 (small integer) の範囲内にない場合、エラーになります。引数に小数部分がある場合は、切り捨てられます。

character-expression
文字定数の最大長以下の長さの文字ストリング値を戻す式。先行ブランクと末尾ブランクは削除されます。その結果のストリングは、 SQL 整数定数を形成するための規則に従うものでなければなりません (SQLSTATE 22018)。ただし、定数の値は短精度整数 (small integer) の範囲内になければなりません (SQLSTATE 22003)。文字ストリングとして、LONG ストリングを使うことはできません。

引数が*文字式* の場合、結果は、対応する整数定数を短精度整数 (small integer) の列または変数に割り当てた場合の数値と同じになります。

この関数の結果は短精度整数 (small integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

SOUNDEX

スキーマは SYSFUN です。

引数内の語の音を示す 4 文字コードを戻します。この結果は、他のストリングの音との比較に使用することができます。

引数は、 CHAR または VARCHAR (4000 バイトを超えない) のいずれかの文字ストリングです。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。関数は、渡されるデータが UTF-8 でエンコードされている場合でも、ASCII 文字であるものとして解釈します。

関数の結果は CHAR(4) です。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

SOUNDEX 関数は、音が分かっているが、正確なつづりが不明なストリングを検出する場合に有用です。文字および文字の組み合わせがどのように聞こえるかを前提とするものであり、類似する音の語の探索に役立ちます。比較は、直接行うことができますが、ストリングを引数として DIFFERENCE 関数へ渡すことによって行うこともできます。

例:

EMPLOYEE 表を使って、 'Loucesy' に似通った音の姓をもつ従業員の EMPNO および LASTNAME を見つけます。

```
SELECT EMPNO, LASTNAME FROM EMPLOYEE
  WHERE SOUNDEX(LASTNAME) = SOUNDEX('Loucesy')
```

この例では、以下を戻します。

```
EMPNO  LASTNAME
------ ---------------
000110 LUCCHESSI
```

SPACE

スキーマは SYSFUN です。

引数によって指定された長さのブランクで構成される文字ストリングを戻します。

引数は SMALLINT または INTEGER にすることができます。

関数の結果は VARCHAR(4000) になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

SQRT

スキーマは SYSIBM です。 (SQRT 関数の SYSFUN バージョンは引き続き使用可能です。)

SQRT 関数は、数値の平方根を戻します。

引数は、組み込み数値データ・タイプの値を返す式でなければなりません。引数が 10 進浮動小数点数の場合、演算は 10 進浮動小数点数で実行されます。それ以外の場合は、関数による処理のために引数が倍精度浮動小数点数に変換されます。

引数が DECFLOAT(*n*) の場合、結果は DECFLOAT(*n*) になります。それ以外の場合、結果は倍精度浮動小数点数になります。

引数が特殊 10 進浮動小数点値である場合、10 進浮動小数点数の一般算術演算の規則が適用されます。 206 ページの『10 進浮動小数点数のための一般算術演算規則』の『10 進浮動小数点数のための一般算術演算規則』を参照してください。

結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

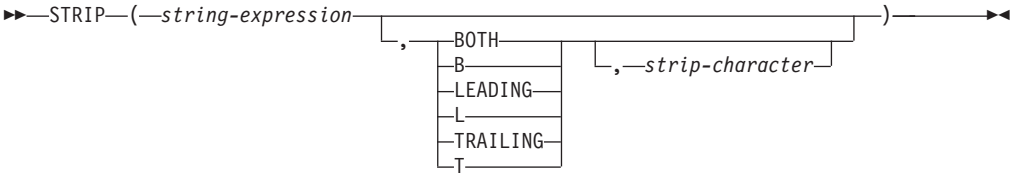
例:

- SQUARE は、9.0 の値をもつ DECIMAL(2,1) ホスト変数であると仮定します。

```
VALUES SQRT(:SQUARE)
```

これは概算値 3.00 を戻します。

STRIP



スキーマは SYSIBM です。キーワードが関数シグニチャーで使用されている場合、関数名を修飾名で指定することはできません。

STRIP 関数は、ブランク、または指定した別の文字のオカレンスを、ストリング式の末尾または先頭から除去します。

STRIP 関数は、TRIM スカラー関数と同じです。

string-expression
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC のいずれかのデータ・タイプの値を戻す式。

BOTH、LEADING、または TRAILING
文字をストリング式の先頭、末尾、または両端から除去するかどうかを指定します。この引数を指定しない場合、文字はストリングの末尾と先頭の両方から除去されます。

strip-character
除去する文字を指定する単一文字定数。 *strip-character* は、UTF-32 エンコードが単一文字であるすべての文字にすることができます。この文字のバイナリー表記が突き合わされます。

strip-character が指定されず、以下の場合:

- *string-expression* が DBCS GRAPHIC ストリングである場合、デフォルトの *strip-character* は DBCS のブランクです。そのコード・ポイントは、データベースのコード・ページに従属します。
- *string-expression* が UCS-2 GRAPHIC ストリングの場合、デフォルトの *strip-character* は UCS-2 のブランク (X'0020') です。
- その他の場合、デフォルトの *strip-character* は SBCS のブランク (X'20') です。

結果は、*string-expression* の長さ属性と同じ最大長を持つ、可変長ストリングです。結果の実際の長さは、*string-expression* の長さから、除去するバイト数を引いたものです。すべての文字が除去された場合、結果は空の可変長ストリングです。結果のコード・ページは、*string-expression* のコード・ページと同じです。

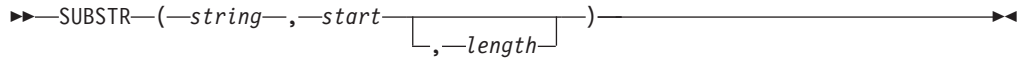
例:

- タイプ CHAR(9) のホスト変数 BALANCE が、値 '000345.50' を持つと想定します。

```
SELECT STRIP(:BALANCE, LEADING, '0'),
FROM SYSIBM.SYSDUMMY1
```

この例では、値 '345.50' を戻します。

SUBSTR



SUBSTR 関数は、ストリングのサブストリングを戻します。

string が文字ストリングの場合、関数の結果は、その最初の引数のコード・ページで示された文字ストリングになります。バイナリー・ストリングの場合には、関数の結果もバイナリー・ストリングになります。 GRAPHIC ストリングの場合は、関数の結果も最初の引数のコード・ページで示された GRAPHIC ストリングになります。最初の引数がホスト変数であると、結果のコード・ページは、データベースのコード・ページになります。 SUBSTR 関数のいずれかの引数が NULL 値の可能性がある場合、結果も NULL 値になる可能性があります。いずれかの引数が NULL 値の場合、結果は NULL 値になります。

string
結果を取り出すストリングを指定する式。

string が文字ストリングまたはバイナリー・ストリングの場合、 *string* のサブストリングは、ゼロ個以上の連続したバイトからなるストリング になります。 *string* が GRAPHIC ストリングの場合、*string* のサブストリングは、ゼロ個以上の連続する 2 バイト文字からなるストリング になります。

start
結果の最初のバイト位置 (文字ストリングまたはバイナリー・ストリングの場合)、あるいは結果の最初の文字の位置 (GRAPHIC ストリングの場合) を指定する式。 *start* は、*string* が固定長か可変長かに応じて、 1 から *string* の最大長までの整数でなければなりません (この範囲外の場合は SQLSTATE 22011)。データベース・コード・ページのコンテキスト内にあるバイト数として指定しなければなりません。アプリケーション・コード・ページのコンテキストで指定してはなりません。

length
結果の長さを指定する式。この式を指定する場合、 *length* は、0 から *n* の範囲のバイナリー整数でなければなりません。ただし、*n* は、 (*string* の長さ属性) - *start* + 1 です (その範囲外の場合は SQLSTATE 22011)。

length を明示的に指定した場合、 *string* の右側に必要な数のブランク文字 (文字ストリングの場合は 1 バイト、 GRAPHIC ストリングの場合は 2 バイト) が、また BLOB ストリングの場合には必要な数の 16 進ゼロ文字がそれぞれ効率的に付加されて、 *string* のうちの指定したサブストリングが常に存在するようにされます。 *length* のデフォルト値は、文字ストリングまたはバイナリー・ストリングの場合は、 *start* で指定されたバイト位置から *string* の最後のバイト位置までのバイト数、 GRAPHIC ストリングの場合には、 *start* で指定された文字位置から *string* の最後の文字位置までの 2 バイト文字の数です。ただし、*string* が可変長ストリングで、その長さが *start* 未満の場合、デフォルト値はゼロになり、結果は空ストリングになります。データベース・コード・ページのコンテキスト内にあるバイト数として指定しなければなりません。アプリケーション・コード・ページのコンテキストで指定してはなりません。 (例えば、デ

ータ・タイプ VARCHAR(18)、値 'MCKNIGHT' の列 NAME の場合、SUBSTR(NAME,10) では空ストリングが戻されます。)

表 38 に、入力のタイプと属性ごとに、 SUBSTR 関数の結果タイプと長さがどうなるかを示しています。

表 38. SUBSTR の結果のデータ・タイプと長さ

ストリング引数のデータ・タイプ	長さ引数	結果データ・タイプ
CHAR(A)	定数 (*l*<255)	CHAR(*l*)
CHAR(A)	指定しない。*start* 引数は定数	CHAR(A-*start*+1)
CHAR(A)	定数以外	VARCHAR(A)
VARCHAR(A)	定数 (*l*<255)	CHAR(*l*)
VARCHAR(A)	定数 (254<*l*<32673)	VARCHAR(*l*)
VARCHAR(A)	定数以外、または指定しない。	VARCHAR(A)
LONG VARCHAR	定数 (*l*<255)	CHAR(*l*)
LONG VARCHAR	定数 (254<*l*<4001)	VARCHAR(*l*)
LONG VARCHAR	定数 (*l*>4000)	LONG VARCHAR
LONG VARCHAR	定数以外、または指定しない。	LONG VARCHAR
CLOB(A)	定数 (*l*)	CLOB(*l*)
CLOB(A)	定数以外、または指定しない。	CLOB(A)
GRAPHIC(A)	定数 (*l*<128)	GRAPHIC(*l*)
GRAPHIC(A)	指定しない。*start* 引数は定数	GRAPHIC(A-*start*+1)
GRAPHIC(A)	定数以外	VARGRAPHIC(A)
VARGRAPHIC(A)	定数 (*l*<128)	GRAPHIC(*l*)
VARGRAPHIC(A)	定数 (127<*l*<16337)	VARGRAPHIC(*l*)
VARGRAPHIC(A)	定数以外	VARGRAPHIC(A)
LONG VARGRAPHIC	定数 (*l*<128)	GRAPHIC(*l*)
LONG VARGRAPHIC	定数 (127<*l*<2001)	VARGRAPHIC(*l*)
LONG VARGRAPHIC	定数 (*l*>2000)	LONG VARGRAPHIC
LONG VARGRAPHIC	定数以外、または指定しない。	LONG VARGRAPHIC
DBCLOB(A)	定数 (*l*)	DBCLOB(*l*)
DBCLOB(A)	定数以外、または指定しない。	DBCLOB(A)
BLOB(A)	定数 (*l*)	BLOB(*l*)
BLOB(A)	定数以外、または指定しない。	BLOB(A)

注: データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は、引き続きサポートされていますが、非推奨になっており、将来のリリースで除去される可能性があります。

string が固定長ストリングの場合に *length* を省略すると、暗黙に LENGTH(*string*) - *start* + 1 が指定されます。 *string* が可変長ストリングの場合に *length* を省略すると、暗黙にゼロまたは LENGTH(*string*) - *start* + 1 のいずれか大きい方が指定されます。

例:

- ホスト変数 NAME (VARCHAR(50)) の値が 'BLUE JAY' で、ホスト変数 SURNAME_POS (int) の値が 6 と想定します。

```
SUBSTR(:NAME, :SURNAME_POS)
```

値 'JAY' が戻されます。

```
SUBSTR(:NAME, :SURNAME_POS,1)
```

この例では 'J' の値を戻します。

- PROJECT 表から、語 'OPERATION' で始まるプロジェクト名 (PROJNAME) の行を全選択します。

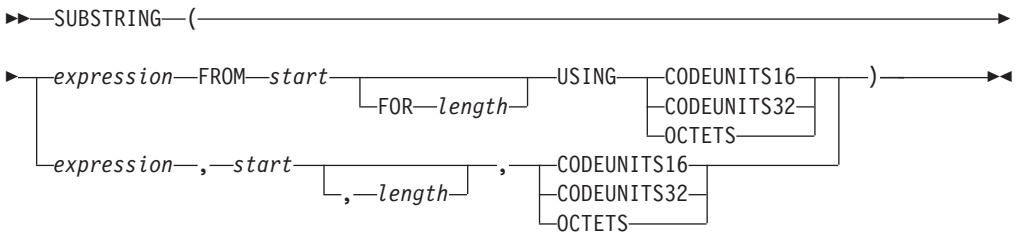
```
SELECT * FROM PROJECT
  WHERE SUBSTR(PROJNAME,1,10) = 'OPERATION '
```

定数の最後にあるスペースは、 OPERATIONS などの語で始まるものを除外するために必要です。

注:

1. 動的 SQL では、*string*、*start*、および *length* が、パラメーター・マーカー (疑問符 (?)) によって表される場合があります。 *string* にパラメーター・マーカーが使用されると、オペランドのデータ・タイプは VARCHAR になり、オペランドは NULL 可能になります。
2. 上記の結果定義には明確には述べられていませんが、 *string* が 1 バイト文字/マルチバイト文字混合ストリングの場合、 *start* と *length* の値によっては、結果にマルチバイト文字のフラグメントが入ることになる場合があります。 つまり、結果が 2 バイト文字の 2 番目のバイトから始まったり、 2 バイト文字の最初のバイトで終わったりする可能性があるということです。 SUBSTR 関数は、このようなフラグメント化の検出を行わず、またこのようなフラグメント化があっても特別な処理は何も行われません。

SUBSTRING



スキーマは SYSIBM です。

SUBSTRING 関数は、ストリングのサブストリングを戻します。

expression
組み込みストリング・データ・タイプの値を返す式。 *expression* が文字ストリングである場合、結果は文字ストリングです。 *expression* が GRAPHIC ストリングである場合、結果は GRAPHIC ストリングです。 *expression* がバイナリー・ストリングである場合、結果はバイナリー・ストリングです。

expression のサブストリングは、*expression* のゼロ以上の連続ストリング単位です。

start
結果の最初のストリング単位になる、*expression* 内の位置を指定する式。指定されたストリング単位で表記される *start* は、整数を戻さなければなりません。 *start* の値は、正、負、またはゼロにすることができます。値 1 は、結果の最初のストリング単位が、*expression* の最初のストリング単位であることを示します。 OCTETS が指定され、*expression* が GRAPHIC データである場合、*start* は奇数でなければなりません。そうでない場合、エラーが戻されます (SQLSTATE 428GC)。

length
結果のサブストリングの最大の実際の長さを指定する式。

expression が固定長ストリングの場合に *length* を省略すると、暗黙的に CHARACTER_LENGTH(*expression* USING *string-unit*) - *start* + 1 が指定されます。これは、*start* から *expression* の最後の位置までの、*string units* の数 (CODEUNITS16、CODEUNITS32、または OCTETS) です。*expression* が可変長ストリングの場合に *length* を省略すると、暗黙的に ゼロまたは CHARACTER_LENGTH(*expression* USING *string-unit*) - *start* + 1 の大きい方が指定されます。希望する長さがゼロの場合、結果は空ストリングです。

指定した場合、*length* は組み込み整数データ・タイプの値を戻す式でなければなりません。値はゼロ以上でなければなりません。*n* より大きい値が指定される場合、*n* が結果のサブストリングの長さとして使用されます。ここで、*n* は (*expression* の長さ属性) - *start* + 1 です。値は、明示的に指定された単位で表現されます。 OCTETS が指定され、*expression* が GRAPHIC データである場合、*length* は偶数でなければなりません (SQLSTATE 428GC)。

CODEUNITS16、CODEUNITS32、または OCTETS
start および *length* のストリング単位を指定します。 CODEUNITS16 は、*start* および *length* を 16 ビットの UTF-16 コード単位で表すことを指定します。

CODEUNITS32 は、*start* および *length* を 32 ビットの UTF-32 コード単位で表すことを指定します。 OCTETS は、*start* および *length* をバイト単位で表すことを指定します。

ストリング単位が CODEUNITS16 または CODEUNITS32 と指定され、*expression* がバイナリー・ストリングまたはビット・データである場合は、エラーが戻されます (SQLSTATE 428GC)。ストリング単位が OCTETS と指定され、*expression* がバイナリー・ストリングである場合は、エラーが戻されます (SQLSTATE 42815)。

CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

SUBSTRING 関数が OCTETS を使用して呼び出され、*source-string* がコード・ポイントごとに複数のバイトを必要とするコード・ページでエンコードされる場合 (混合または MBCS)、SUBSTRING 操作は、マルチバイト・コード・ポイントを分割し、結果のサブストリングは部分的なコード・ポイントで開始または終了する場合があります。これが発生する場合、関数は結果のバイト長を変更しない仕方で、先頭または末尾の部分的コード・ポイントのバイトをブランクで置き換えます。 (下記の関連する例を参照。)

結果の長さ属性は、*expression* の長さ属性と同じになります。関数のいずれかの引数が NULL 値の可能性がある場合、結果も NULL 値になる可能性があります。いずれかの引数が NULL 値の場合、結果は NULL 値になります。結果にいずれかの文字が埋め込まれることはありません。 *expression* の実際の長さが 0 である場合、結果の実際の長さも 0 になります。

注:

- 結果の長さ属性は、入力ストリング式の長さ属性と同じになります。この動作は SUBSTR 関数の動作とは異なります。その場合の長さ属性は、関数の *start* および *length* 引数から導出されます。

例:

- FIRSTNAME は、表 T1 の VARCHAR(12) 列です。その値の 1 つは、6 文字のストリング 'Jürgen' です。 FIRSTNAME が以下の値を持つ場合:

```
関数 ...                              戻り ...
------------------------------------  ------------------------------------
SUBSTRING(FIRSTNAME,1,2,CODEUNITS32)  'Jü' -- x'4AC3BC'
SUBSTRING(FIRSTNAME,1,2,CODEUNITS16)  'Jü' -- x'4AC3BC'
SUBSTRING(FIRSTNAME,1,2,OCTETS)       'J ' -- x'4A20' (a truncated string)
SUBSTRING(FIRSTNAME,8,CODEUNITS16)    ゼロの長さのストリング
SUBSTRING(FIRSTNAME,8,4,OCTETS)       ゼロの長さのストリング
```

- C1 は、表 T1 の VARCHAR(12) 列です。その値の 1 つはストリング 'ABCDEFG' です。 C1 が以下の値を持つ場合:

```
関数 ...                  戻り ...
-------------------------  --------------------
SUBSTRING(C1,-2,2,OCTETS)  ゼロの長さのストリング
SUBSTRING(C1,-2,4,OCTETS)  'A'
SUBSTRING(C1,-2,OCTETS)    'ABCDEFG'
SUBSTRING(C1,0,1,OCTETS)   ゼロの長さのストリング
```

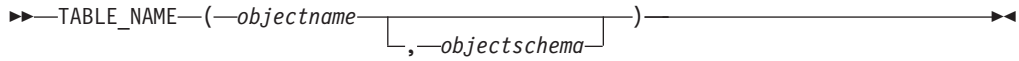
- 以下では、ストリングの長さの単位が OCTETS の場合、SUBSTRING が先頭または末尾の部分的マルチバイト・コード・ポイントのバイトをブランクで置き換

える例を示しています。 UTF8_VAR に Unicode ストリング '&N~AB' の UTF-8 表記が含まれていると想定します。ここで '&' は音楽の ト音記号、'~' は結合チルド記号文字です。

```
SUBSTRING(UTF8_VAR, 2, 5, OCTETS)
```

3 つのブランク・バイトが 'N' に先行し、1 つのブランク・バイトが 'N' に続きます。

TABLE_NAME



スキーマは SYSIBM です。

TABLE_NAME 関数は、別名チェーンが解決された後で検出されたオブジェクトの非修飾名を戻します。指定された *objectname* (および *objectschema*) が、解決の開始点として使用されます。開始点が別名を参照していない場合は、開始点の非修飾名が戻されます。結果の名前は、表、ビュー、または未定義オブジェクトのいずれかになります。Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

objectname
解決しようとする非修飾名 (通常は既存の別名) を表す文字式。 *objectname* は、CHAR または VARCHAR のデータ・タイプ、 1 バイト以上 129 バイト未満の長さでなければなりません。

objectschema
指定された *objectname* の解決前の値を修飾するのに使うスキーマを表す文字式。 *objectschema* は、CHAR または VARCHAR のデータ・タイプ、 1 バイト以上 129 バイト未満の長さでなければなりません。

objectschema を指定しない場合は、修飾子にデフォルトのスキーマが使用されます。

この関数の結果のデータ・タイプは VARCHAR(128) です。 *objectname* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *objectname* が NULL であれば、結果も NULL 値になります。 *objectschema* が NULL 値の場合は、デフォルトのスキーマ名が使用されます。結果は、非修飾名を表す文字ストリングになります。結果の名前は、次のいずれかを表す可能性があります。

表 *objectname* の値が、表名 (入力値が戻される) であったか、あるいは解決結果が表となり、その表名が戻されることになる別名であった。

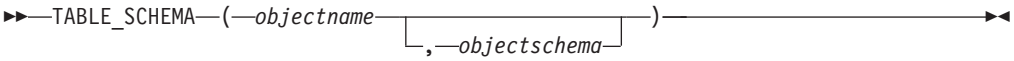
ビュー *objectname* の値が、ビュー名 (入力値が戻される) であったか、あるいは解決結果がビューとなり、そのビュー名が戻されることになる別名であった。

未定義オブジェクト
objectname の値が、未定義オブジェクト (入力値が戻される) であったか、あるいは解決結果が未定義オブジェクトとなり、その名前が戻されることになる別名であった。

したがって、NULL 以外の値がこの関数に指定された場合、結果名のオブジェクトが存在していなくても、常にその値が戻されます。

注: パーティション・データベース構成において、TABLE_SCHEMA および TABLE_NAME スカラー関数を使用するときにコーディネーター・パーティションとカタログ・パーティションの間で生じる不必要な通信を避けることによって、パフォーマンスを改善するには、代わりに BASE_TABLE 表関数を使用することができます。

TABLE_SCHEMA



スキーマは SYSIBM です。

TABLE_SCHEMA 関数は、別名チェーンが解決された後で検出されるオブジェクトのスキーマ名を戻します。指定された *objectname* (および *objectschema*) が、解決の開始点として使用されます。開始点が別名を参照していない場合は、開始点のスキーマ名が戻されます。結果のスキーマ名は、表、ビュー、または未定義オブジェクトのいずれかになります。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

objectname
解決しようとする非修飾名 (通常は既存の別名) を表す文字式。 *objectname* は、CHAR または VARCHAR のデータ・タイプ、 1 バイト以上 129 バイト未満の長さでなければなりません。

objectschema
指定された *objectname* の解決前の値を修飾するのに使うスキーマを表す文字式。 *objectschema* は、CHAR または VARCHAR のデータ・タイプ、 1 バイト以上 129 バイト未満の長さでなければなりません。

objectschema を指定しない場合は、修飾子にデフォルトのスキーマが使用されます。

この関数の結果のデータ・タイプは VARCHAR(128) です。 *objectname* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *objectname* が NULL であれば、結果も NULL 値になります。 *objectschema* が NULL 値の場合は、デフォルトのスキーマ名が使用されます。結果は、スキーマ名を表す文字ストリングになります。結果のスキーマは、次のいずれかのスキーマ名を表します。

表 *objectname* の値が、表名 (*objectschema* の入力値またはデフォルト値が戻される) であったか、あるいは解決結果が表となり、そのスキーマ名が戻されることになる別名であった。

ビュー *objectname* の値が、ビュー名 (*objectschema* の入力値またはデフォルト値が戻される) であったか、あるいは解決結果がビューとなり、そのスキーマ名が戻されることになる別名であった。

未定義オブジェクト
objectname の値が、未定義オブジェクト名 (*objectschema* の入力値またはデフォルト値が戻される) であったか、あるいは解決結果が未定義オブジェクトとなり、そのスキーマ名が戻されることになる別名であった。

したがって、NULL 以外の *objectname* 値がこの関数に指定された場合、結果名のスキーマ名でのオブジェクトが存在していなくても、常に値が戻されます。例えば、`TABLE_SCHEMA('DEPT', 'PEOPLE')` は、カタログ項目が見つからない場合には、'PEOPLE ' を戻します。

注: パーティション・データベース構成において、TABLE_SCHEMA および TABLE_NAME スカラー関数を使用するときにコーディネーター・パーティションとカタログ・パーティションの間で生じる不必要な通信を避けることによって、パフォーマンスを改善するには、代わりに BASE_TABLE 表関数を使用することができます。

例:

- PBIRD は、表 HEDGES.T1 に定義されている別名 PBIRD.A1 を使用して、SYSCAT.TABLES から指定した表の統計値を選択しようとしています。

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('A1')
  AND TABSCHEMA = TABLE_SCHEMA ('A1')
```

  HEDGES.T1 について要求された統計値が、カタログから取り出されます。

- HEDGES.X1 というオブジェクトの統計値を、 HEDGES.X1 を使用して SYSCAT.TABLES から選択します。 HEDGES.X1 が別名か表かが分からないため、 TABLE_NAME と TABLE_SCHEMA を使用します。

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('X1','HEDGES')
  AND TABSCHEMA = TABLE_SCHEMA ('X1','HEDGES')
```

  HEDGES.X1 が表であるとすると、 HEDGES.X1 について要求された統計がカタログから取り出されます。

- HEDGES.T2 に対して定義された別名 PBIRD.A2 を使用して、 SYSCAT.TABLES から指定した表の統計を選択しますが、HEDGES.T2 は存在していません。

```
SELECT NPAGES, CARD FROM SYSCAT.TABLES
  WHERE TABNAME = TABLE_NAME ('A2','PBIRD')
  AND TABSCHEMA = TABLE_SCHEMA ('A2',PBIRD')
```

  TABNAME = 'T2' および TABSCHEMA = 'HEDGES' である項目が SYSCAT.TABLES の中に見つからないため、このステートメントからは 0 個のレコードが戻されます。

- SYSCAT.TABLES 内の各項目の修飾名、および別名項目については最終参照名を選択します。

```
SELECT TABSCHEMA AS SCHEMA,  TABNAME AS NAME,
  TABLE_SCHEMA (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_SCHEMA,
  TABLE_NAME (BASE_TABNAME, BASE_TABSCHEMA) AS REAL_NAME
  FROM SYSCAT.TABLES
```

  このステートメントは、カタログ内の各オブジェクトの修飾名と、別名項目については最終参照名 (別名が解決された後の名前) を戻します。別名でないすべての項目については、 BASE_TABNAME および BASE_TABSCHEMA が NULL 値であるため、 REAL_SCHEMA 列と REAL_NAME 列は NULL 値になります。

TAN

スキーマは SYSIBM です。 (TAN 関数の SYSFUN バージョンは引き続き使用可能です。)

引数のタンジェント (正接) の値を戻します。引数は、ラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

TANH

スキーマは SYSIBM です。

引数に対する双曲線タンジェント (正接) の値を戻します。引数はラジアン単位の角度です。

引数は、(DECFLOAT を除く) 任意の組み込み数値データ・タイプにすることができます。引数は、関数での処理に必要な倍精度浮動小数点数に変換されます。

関数の結果は倍精度浮動小数点数になります。引数が NULL になる可能性があるか、またはデータベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

TIME

スキーマは SYSIBM です。

TIME 関数は、値から時刻を戻します。

引数は、時刻、タイム・スタンプであるか、または CLOB、LONG VARCHAR、DBCLOB、または LONG VARGRAPHIC ではない時刻またはタイム・スタンプの有効なストリング表記でなければなりません。

Unicode データベースだけが、時刻またはタイム・スタンプの GRAPHIC ストリング表現である引数をサポートします。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は時刻です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、引数のデータ・タイプに応じて以下のように異なります。

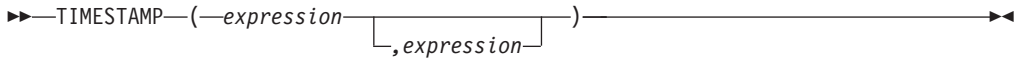
- 引数が時刻の場合
  - 結果は、指定した時刻になります。
- 引数がタイム・スタンプの場合
  - 結果はタイム・スタンプの時刻の部分になります。
- 引数がストリングの場合
  - 結果は、そのストリングによって表される時刻になります。

例:

- IN_TRAY サンプル表から、 (任意の日の) 現在の時刻よりも 1 時間後以降に受け取ったすべての注を選択します。

```
SELECT * FROM IN_TRAY
  WHERE TIME(RECEIVED) >= CURRENT TIME + 1 HOUR
```

TIMESTAMP



スキーマは SYSIBM です。

TIMESTAMP 関数は、1 つの値または 2 つの値からタイム・スタンプを戻します。

Unicode データベースだけが、日付、時刻、またはタイム・スタンプの GRAPHIC ストリング表現である引数をサポートします。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

引数に関する規則は、2 番目の引数を指定するか否かによって異なります。

- 引数を 1 つだけ指定した場合
  - タイム・スタンプ、タイム・スタンプの有効なストリング表記、あるいは CLOB、LONG VARCHAR、DBCLOB、または LONG VARGRAPHIC ではない長さ 14 のストリングのいずれかでなければなりません。

    長さ 14 のストリングは、有効な日付と時刻を *yyyyxxddhhmmss* という形式で表した数字のストリングであることが必要です (ここで、 *yyyy* は年、 *xx* は月、 *dd* は日、 *hh* は時、 *mm* は分、そして *ss* は秒を表します)
- 引数を 2 つとも指定する場合
  - 最初の引数は日付または日付の有効なストリング表記でなければならず、 2 番目の引数は時刻または時刻の有効なストリング表記でなければなりません。

関数の結果はタイム・スタンプです。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

その他の規則は、2 番目の引数を指定するか否かによって以下のように異なります。

- 引数を 2 つとも指定する場合
  - 結果は、最初の引数によって日付が指定され、 2 番目の引数によって時刻が指定されたタイム・スタンプです。タイム・スタンプのマイクロ秒部分はゼロです。
- 引数が 1 つだけ指定され、それがタイム・スタンプの場合
  - 結果は、指定したタイム・スタンプになります。
- 引数が 1 つだけ指定され、それがストリングの場合
  - 結果は、ストリングによって表されるタイム・スタンプになります。引数が長さ 14 のストリングの場合、タイム・スタンプのマイクロ秒部分はゼロになります。

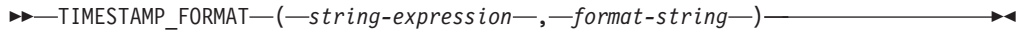
例:

- START_DATE (日付) 列が 1988-12-25 に等しい値、 START_TIME (時刻) 列が 17.12.30 に等しい値であると想定します。

```
TIMESTAMP(START_DATE, START_TIME)
```

この例は、値 '1988-12-25-17.12.30.000000' を戻します。

TIMESTAMP_FORMAT



スキーマは SYSIBM です。

TIMESTAMP_FORMAT 関数により、指定したフォーマットが使用されて入力ストリングの解釈に基づくタイム・スタンプが返されます。

string-expression

254 未満の (SQLSTATE 42815) 長さ属性を持つ CHAR または VARCHAR データ・タイプの値が返される式。 *string-expression* には、*format-string* で指定したフォーマットに対応するタイム・スタンプの各構成要素が含まれている必要があります。

format-string

254 未満の (SQLSTATE 22007) 長さ属性を持つ組み込み文字ストリングのデータ・タイプの値が返される式。 *format-string* には *string-expression* の解釈方法に関するテンプレートが含まれ、これを基にタイム・スタンプ値に変換されます。 *format-string* の内容は英大文字小文字混合で指定できます。

有効な *format-string* には少なくとも 1 つのフォーマット・エレメントを含める必要があり、タイム・スタンプのすべての構成要素に複数の指定を含めてはならず、また 表 39 (SQLSTATE 22007) に特に注記のない限り、フォーマット・エレメントの任意の組み合わせを含めることができます。例えば、*format-string* には YY と YYYY の両方を含めることはできません。これは、それらが *string-expression* の年の構成要素を解釈するためにともに使用されてしまうためです。下の表を参照して、どのフォーマット・エレメントを同時に指定できないかを確認してください。 2 つのフォーマット・エレメントは、オプションで以下の 1 つ以上の区切り文字で分離することができます。

- ダッシュ (-)
- ピリオド (.)
- スラッシュ (/)
- コンマ (,)
- アポストロフィ (')
- セミコロン (;)
- コロン (:)
- ブランク ( )

区切り文字は *format-string* の始めまたは終わりにも指定できます。 *string-expression* で指定する区切り文字は構成要素を区切るために使用するものであり、*format-string* で指定する区切り文字と一致する必要はありません。

表 39. TIMESTAMP_FORMAT 関数のフォーマット・エレメント

フォーマット・エレメント	タイム・スタンプの関連する構成要素	説明
DD	日	日 (01-31)。
DDD	月、日	年間通算日 (001-366)

表 39. TIMESTAMP_FORMAT 関数のフォーマット・エレメント (続き)

フォーマット・エレメント	タイム・スタンプの関連する構成要素	説明
FF[*n*]	マイクロ秒	小数秒 (0-999999)。数字 *n* は、*string-expression* 内に予想される桁数の指定に使用します。 *n* の有効値は 1 から 6 です。FF を指定することは、FF6 を指定することと同等です。FF フォーマット・エレメントに対応する *string-expression* 内の構成要素の後に区切り文字が続いている場合、またはそれが最後の構成要素である場合、小数秒の桁数はフォーマット・エレメントで指定した桁数よりも少ないことがあります。この場合は、指定した桁数の右側に数字のゼロが埋め込まれます。
HH	時	HH の動作は HH12 と同様です。
HH12	時	12 時間形式の時 (01-12)。AM がデフォルトの午前/午後の指定子です。
HH24	時	24 時間形式の時 (00-24)。
J	年、月、および日	ユリウス日 (紀元前 4713 年 1 月 1 日からの日数)。
MI	分	分 (00-59)。
MM	月	月 (01-12)。
NNNNNN	マイクロ秒	マイクロ秒 (000000-999999)。FF6 と同様。
RR	年	調整済み年の最後の 2 桁 (00-99)。
RRRR	年	4 桁の調整済み年 (0000-9999)。
SS	秒	秒 (00-59)。
SSSSS	時、分、および秒	直近の午前 0 時からの秒数 (00000-86400)。
Y	年	年の最後の 1 桁 (0-9)。現在の年の最初の 3 桁が、完全な 4 桁の年の判別に使用されます。

表 39. TIMESTAMP_FORMAT 関数のフォーマット・エレメント (続き)

フォーマット・エレメント	タイム・スタンプの関連する構成要素	説明
YY	年	年の最後の 2 桁 (00-99)。現在の年の最初の 2 桁が、完全な 4 桁の年の判別に使用されます。
YYY	年	年の最後の 3 桁 (000-999)。現在の年の最初の桁が、完全な 4 桁の年の判別に使用されます。
YYYY	年	4 桁の年 (0000-9999)。

RR と RRRR の各フォーマット・エレメントは、次の表に従って現在の年の左端の 2 桁に基づく 2 桁の値または 4 桁の値を生成するために、入力された値を調整することによって年の指定の解釈方法を変更するために使用することができます。

現在の年の最後の 2 桁	*string-expression* 内の 2 桁の年	タイム・スタンプの年の構成要素の最初の 2 桁
0-50	0-49	現在の年の最初の 2 桁
51-99	0-49	現在の年 + 1 の最初の 2 桁
0-50	50-99	現在の年 - 1 の最初の 2 桁
51-99	50-99	現在の年の最初の 2 桁

例えば、現在の年が 2007 の場合、フォーマット 'RR' の '86' は 1986 を意味しますが、現在の年が 2052 の場合は 2086 を意味します。

format-string にタイム・スタンプの以下の構成要素の 1 つに対してフォーマット・エレメントが含まれていない場合は、以下のデフォルトが使用されます。

タイム・スタンプの構成要素	デフォルト
年	現在の年
月	現在の月
日	01 (現在の月の最初の日)
時	00
分	00
秒	00
マイクロ秒	000000

format-string 内の対応するフォーマット・エレメントに有効数字桁数の最大数がないタイム・スタンプ値 (つまり、月、日、時、分、秒) のすべての構成要素に先行ゼロを指定できます。

タイム・スタンプの構成要素 (年、月、日、時、分、秒など) を表す *string-expression* のサブストリングには、対応するフォーマット・エレメントで指示されたタイム・スタンプのその構成要素の最大桁数より少ない桁数を含めることができます。入力のない桁はデフォルトのゼロになります。例えば、'YYYY-MM-DD HH24:MI:SS' の *format-string* を持つ場合、'999-3-9 5:7:2' の入力値では '0999-03-09 05:07:02' と同じ結果が生成されます。

関数の結果はタイム・スタンプです。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

注:

- ***決定論:*** TIMESTAMP_FORMAT 関数は、通常決定論的には定義されません。この関数が決定論的になるのは、以下の条件がすべて真の場合だけです。
  - *format-string* が定数である
  - *format-string* に現在の年を使用する必要のない年を完全に定義するフォーマット・エレメントが含まれている (フォーマット・エレメントの YYYY または J が含まれている)
  - *format-string* に現在の月を使用する必要のない月が定義されている (フォーマット・エレメントの MM または J が含まれている)
- ***代替構文:*** TO_DATE と TO_TIMESTAMP は、TIMESTAMP_FORMAT の同義語です。

例:

- 2000 年が始まる 1 秒前 (1999 年 12 月 31 日 23 時 59 分 59 秒) にあたる受信タイム・スタンプで、IN_TRAY 表に行を挿入します。

```
INSERT INTO IN_TRAY (RECEIVED)
  VALUES (TIMESTAMP_FORMAT('1999-12-31 23:59:59',
    'YYYY-MM-DD HH24:MI:SS'))
```

- アプリケーションで、日付情報のストリングが INDATEVAR という変数に受け取られます。この値は厳密にはフォーマット設定されていなく、また年に 2 桁または 4 桁の数字、月と日に 1 桁または 2 桁の数字が含まれています。日付の各構成要素はダッシュ (-) またはスラッシュ (/) 文字で分離され、日、月、および年の順序であることが予期されています。時間情報は、時 (24 時間形式) および分で構成され、通常コロンで分離されています。サンプル値としては、'15/12/98 13:48' や '9-3-2004 8:02' などがあります。このような値を IN_TRAY 表に挿入します。

```
INSERT INTO IN_TRAY (RECEIVED)
  VALUES (TIMESTAMP_FORMAT(:INDATEVAR,
    'DD/MM/RRRR HH24:MI'))
```

  フォーマットに RRRR を使用すると、2 桁と 4 桁の年の値が考慮され、現在の年に基づいて入力されていない最初の 2 桁が割り当てられます。 YYYY を使用すると、2 桁の年の入力値には先行ゼロが設定されます。また、スラッシュ区切り文字ではダッシュ文字も許可されます。現在の年が 2007 年とすると、サンプル値の結果として生成されるタイム・スタンプは以下のとおりです。

```
'15/12/98 13:48' --> 1998-12-15-13.48.00.000000
'9-3-2004 8:02'  --> 2004-03-09-08.02.00.000000
```

TIMESTAMP_ISO

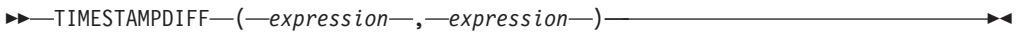
スキーマは SYSFUN です。

日付、時刻、またはタイム・スタンプの引数に基づいてタイム・スタンプ値を戻します。引数が日付の場合は、時間エレメントのすべてにゼロが入れられます。引数が時刻の場合、日付エレメントには CURRENT DATE 特殊レジスターの値が入れられ、時刻の小数エレメントにはゼロが入れられます。

引数は、日付、時刻、またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付、時刻、またはタイム・スタンプの有効な文字ストリング表現でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は TIMESTAMP になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

TIMESTAMPDIFF



スキーマは SYSFUN です。

2 つのタイム・スタンプ間の差に基づいて、最初の引数によって定義されたタイプのインターバル数の見積もりが戻されます。

最初の引数は INTEGER または SMALLINT のいずれかです。インターバル (最初の引数) の有効な値は次のとおりです。

1 秒の小数部

2 秒

4 分

8 時間

16 日

32 週

64 月

128 四半期

256 年

2 番目の引数は、2 つのタイム・スタンプの減算を行い、その結果を CHAR(22) に変換した結果です。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

以下の前提事項が、差の見積もりに使用されます。

- 1 年は 365 日である。
- 1 カ月は 30 日である。
- 1 日は 24 時間である。
- 1 時間は 60 分である。
- 1 分は 60 秒である。

上記の前提は、タイム・スタンプ期間である 2 番目の引数の情報を最初の引数で指定されたインターバル・タイプに変換する際に使用されます。戻される見積もりが、日数によって異なる場合があります。たとえば、 '1997-03-01-00.00.00' と '1997-02-01-00.00.00' の差の日数 (インターバル 16) が要求された場合、結果は 30 になります。これは、タイム・スタンプ相互間の差は 1 カ月であり、 1 カ月は 30 日であるという前提が適用されるからです。

例:

以下の例は、2 つのタイム・スタンプにはさまれた分数である 4277 を戻します。

```
TIMESTAMPDIFF(4,CHAR(TIMESTAMP('2001-09-29-11.25.42.483219') -
  TIMESTAMP('2001-09-26-12.07.58.065497')))
```

TO_CHAR

スキーマは SYSIBM です。

TO_CHAR 関数は、文字テンプレートを使ってフォーマットされたタイム・スタンプの文字表現を戻します。

TO_CHAR は VARCHAR_FORMAT の同義語です。

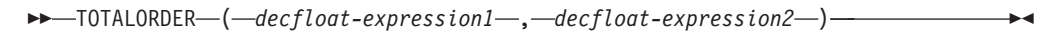
TO_DATE

スキーマは SYSIBM です。

TO_DATE 関数は、文字テンプレートを使って解釈された文字ストリングからタイム・スタンプを戻します。

TO_DATE は TIMESTAMP_FORMAT の同義語です。

TOTALORDER



スキーマは SYSIBM です。

TOTALORDER 関数は、2 つの引数の比較の順序を示す -1、0、または 1 の SMALLINT 値を戻します。

decfloat-expression1
組み込み数値データ・タイプの値を返す式。引数が DECFLOAT(34) ではない場合、処理のために DECFLOAT(34) に論理的に変換されます。

decfloat-expression2
組み込み数値データ・タイプの値を返す式。引数が 10 進浮動小数点値ではない場合、処理のために DECFLOAT(34) に変換されます。

数値比較は正確であり、結果は範囲と精度が無制限であるかのように、有限のオペランドに対して判別されます。 オーバーフロー状態またはアンダーフロー状態が発生することはありません。

一方の値が DECFLOAT(16) で、他方が DECFLOAT(34) の場合、比較される前に DECFLOAT(16) 値は DECFLOAT(34) に変換されます。

TOTALORDER 関数のセマンティクスは、IEEE 754R の全体的順序述部の規則に基づいています。 TOTALORDER は以下の値を戻します。

- *decfloat-expression1* が *decfloat-expression2* と比較して順序が低い場合、-1
- *decfloat-expression1* と *decfloat-expression2* の両方が同じ順序の場合、0
- *decfloat-expression1* が *decfloat-expression2* と比較して順序が高い場合、1

特殊値および有限数値の順序は、次のとおりです。

```
-NAN<-SNAN<-INFINITY<-0.10<-0.100<-0<0<0.100<0.10<INFINITY<SNAN<NAN
```

この関数の結果は SMALLINT 値となります。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

例:

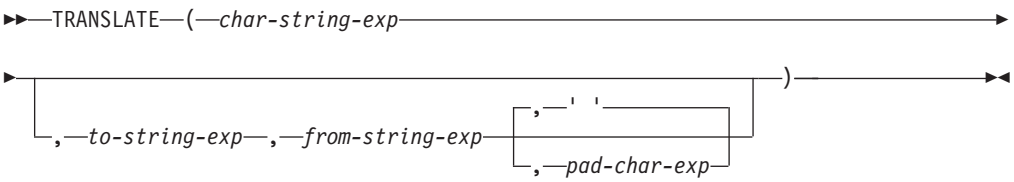
- 以下の例は、TOTALORDER 関数を使った 10 進浮動小数点値の比較を示しています。

```
TOTALORDER(-INFINITY, -INFINITY) = 0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.0)) = 0
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-1.00)) = -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(-0.5)) = -1
TOTALORDER(DECFLOAT(-1.0), DECFLOAT(0.5)) = -1
TOTALORDER(DECFLOAT(-1.0), INFINITY) = -1
TOTALORDER(DECFLOAT(-1.0), SNAN) = -1
TOTALORDER(DECFLOAT(-1.0), NAN) = -1
TOTALORDER(NAN, DECFLOAT(-1.0)) = 1
TOTALORDER(-NAN, -NAN) = 0
TOTALORDER(-SNAN, -SNAN) = 0
TOTALORDER(NAN, NAN) = 0
TOTALORDER(SNAN, SNAN) = 0
```

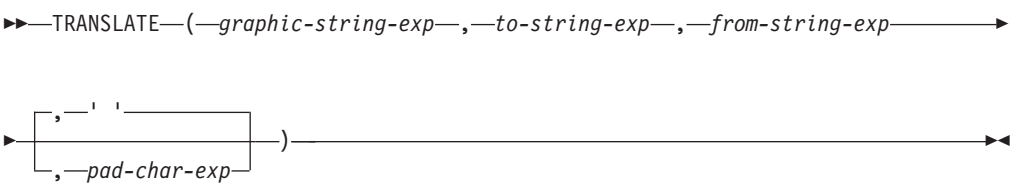
```
TOTALORDER(-1.0, -1.0) = 0
TOTALORDER(-1.0, -1.00) = -1
TOTALORDER(-1.0, -0.5) = -1
TOTALORDER(-1.0, 0.5) = -1
TOTALORDER(-1.0, INFINITY) = -1
TOTALORDER(-1.0, SNAN) = -1
TOTALORDER(-1.0, NAN) = -1
```

TRANSLATE スカラー関数

文字ストリング式:



GRAPHIC ストリング式:



スキーマは SYSIBM です。

TRANSLATE 関数は、ストリング式内の 1 つ以上の文字が他の文字に変換された可能性のある値を戻します。

この関数は、*from-string-exp* の中でも出現する *char-string-exp* または *graphic-string-exp* 内の文字すべてを、 *to-string-exp* 内の対応する文字に変換します。あるいは対応する文字がない場合は *pad-char-exp* で指定される埋め込み文字に変換します。

char-string-exp **or** *graphic-string-exp*
変換されるストリングを指定します。

to-string-exp
char-string-exp 内の特定の文字を、どのような文字のストリングに変換するかを指定します。

to-string-exp の値を指定せず、データ・タイプが GRAPHIC でない場合、*char-string-exp* 内の文字すべてが大文字変換されます。つまり、文字 a から z は文字 A から Z に変換され、他の文字は大文字に相当するものがあればその文字に変換されます。例えば、コード・ページ 850 では、é は É に変換されますが、ÿ は変換されません。これは、コード・ページ 850 に Ÿ が組み込まれていないためです。結果の文字のコード・ポイント長が、元の文字のコード・ポイント長と同じでない場合、元の文字は変換されません。

from-string-exp
char-string-exp の中に見つかった場合に *to-string-exp* の中の対応する文字に変換される文字のストリングを指定します。 *from-string-exp* に重複する文字が入っている場合は、最初に見つかった文字が使用され、重複は無視されます。
to-string-exp が *from-string-exp* より長い場合、余分な文字は無視されます。
to-string-exp を指定する場合は、*from-string-exp* も指定する必要があります。

pad-char-exp

to-string-exp が *from-string-exp* より短い場合に、 *to-string-exp* への埋め込みに使用する単一の文字を指定します。 *pad-char-exp* 引数は、長さ属性が 0 または 1 でなければなりません。長さゼロのストリングを指定すると、*from-string-exp* 内の文字は、*to-string-exp* 内に対応する文字がなければ、*char-string-exp* または *graphic-string-exp* から除去されます。値が指定されない場合には、1 バイトのブランク文字が想定されます。

引数には、データ・タイプ CHAR または VARCHAR のいずれかの文字ストリングか、あるいはデータ・タイプ GRAPHIC または VARGRAPHIC の GRAPHIC ストリングを指定できます。データ・タイプ LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、または DBCLOB は使用できません。

graphic-string-exp を指定する場合、 *pad-char-exp* だけがオプションです (値を指定しない場合、 2 バイトのブランク文字が想定されます)。埋め込み文字を含め、各引数は GRAPHIC データ・タイプでなければなりません。

結果のデータ・タイプおよびコード・ページは、最初の引数のデータ・タイプおよびコード・ページと同じになります。最初の引数がホスト変数であると、結果のコード・ページは、データベースのコード・ページになります。最初の引数以外のどの引数の場合も、それ自身または最初の引数が FOR BIT DATA と定義されて (この場合は変換は行われません) いない限り、結果のコード・ページに変換されます。

文字と GRAPHIC が同等のデータ・タイプであると見なされる Unicode データベースでは、以下の例外があります。

- 最初の引数以外のいずれかの引数が FOR BIT DATA である場合、結果のコード・ページは 1208 になります。
- どの引数も FOR BIT DATA ではない場合、結果のコード・ページは、一連の引数の中で最も頻繁に出現するコード・ページになります。
- どの引数も FOR BIT DATA ではなく、一連の引数の中で 2 つの異なるコード・ページが同等の頻度で出現する場合、結果のコード・ページは 1200 になります。

結果の長さ属性は、最初の引数と同じになります。引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

引数が CHAR または VARCHAR のデータ・タイプの場合、*to-string-exp* と *from-string-exp* の対応する文字は、同じバイト数でなければなりません (ストリングの長さがゼロの場合を除く)。例えば、1 バイト文字をマルチバイト文字に変換することや、マルチバイト文字を 1 バイト文字に変換することは無効です。*pad-char-exp* 引数が、有効なマルチバイト文字の第 1 バイトになることはありません (SQLSTATE 42815)。

文字のマッチングには、バイナリー比較が使用されます。データベースの照合は使用されません。

char-string-exp のみを指定した場合、 1 バイト文字は大文字変換され、マルチバイト文字はそのままになります。

例:

- ホスト変数 SITE (VARCHAR(30)) の値が 'Hanauma Bay' であると想定します。

```
TRANSLATE(:SITE)
```

この例では、値 'HANAUMA BAY' を戻します。

```
TRANSLATE(:SITE, 'j', 'B')
```

この例では、'Hanauma jay' が戻されます。

```
TRANSLATE(:SITE, 'ei', 'aa')
```

この例では、値 'Heneume Bey' が戻されます。

```
TRANSLATE(:SITE, 'bA', 'Bay', '%')
```

この例では、値 'HAnAumA bA%' が戻されます。

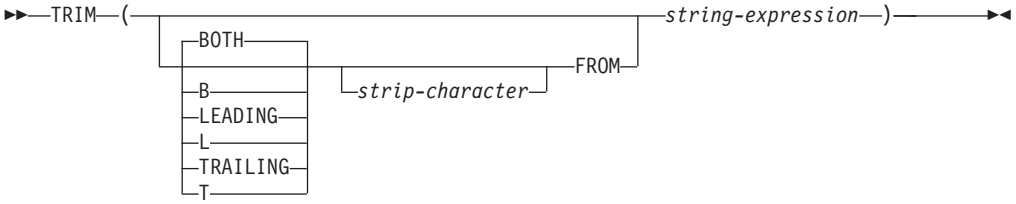
```
TRANSLATE(:SITE, 'r', 'Bu')
```

この例では、値 'Hana ma ray' が戻されます。

```
TRANSLATE(:SITE, 'r', 'Bu', '')
```

この例では、値 'Hanama ray' が戻されます。

TRIM



スキーマは SYSIBM です。キーワードが関数シグニチャーで使用されている場合、関数名を修飾名で指定することはできません。

TRIM 関数は、ブランク、または指定した別の文字のオカレンスを、ストリング式の末尾または先頭から除去します。

BOTH、LEADING、または TRAILING
文字をストリング式の先頭、末尾、または両端から除去するかどうかを指定します。この引数を指定しない場合、文字はストリングの末尾と先頭の両方から除去されます。

strip-character
除去する文字を指定する単一文字定数。 *strip-character* は、UTF-32 エンコードが単一文字であるすべての文字にすることができます。この文字のバイナリー表記が突き合わされます。

strip-character が指定されず、以下の場合:

- *string-expression* が DBCS GRAPHIC ストリングである場合、デフォルトの *strip-character* は DBCS のブランクです。そのコード・ポイントは、データベースのコード・ページに従属します。
- *string-expression* が UCS-2 GRAPHIC ストリングの場合、デフォルトの *strip-character* は UCS-2 のブランク (X'0020') です。
- その他の場合、デフォルトの *strip-character* は SBCS のブランク (X'20') です。

FROM *string-expression*
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC のいずれかのデータ・タイプの値を戻す式。

結果は、*string-expression* の長さ属性と同じ最大長を持つ、可変長ストリングです。結果の実際の長さは、*string-expression* の長さから、除去するバイト数を引いたものです。すべての文字が除去された場合、結果は空の可変長ストリングです。結果のコード・ページは、*string-expression* のコード・ページと同じです。

例:

- タイプ CHAR(9) のホスト変数 HELLO が、値 ' Hello' を持つと想定します。

```
SELECT TRIM(:HELLO),
  TRIM(TRAILING FROM :HELLO)
FROM SYSIBM.SYSDUMMY1
```

これは値 'Hello' および ' Hello' をそれぞれ戻します。

- タイプ CHAR(9) のホスト変数 BALANCE が、値 '000345.50' を持つと想定します。

```
SELECT TRIM(L '0' FROM :BALANCE),
FROM SYSIBM.SYSDUMMY1
```

この例では、値 '345.50' を戻します。

TRUNCATE

スキーマは SYSIBM です。 (TRUNCATE 関数の SYSFUN バージョンは引き続き使用可能です。)

expression2 が正の場合は小数点の右側、 *expression2* がゼロまたは負の場合は小数点の左側の、 *expression2* の桁数に丸められた *expression1* を返します。

expression1
組み込み数値データ・タイプの値を返す式。

expression2
短精度整数または長精度整数を返す式。整数の絶対値は、*expression2* が正の場合は小数点の右側、 *expression2* が負の場合は小数点の左側の結果桁数を指定します。

expression2 の絶対値が小数点の左側の桁数より大きい場合、結果は 0 になります。例を以下に示します。

```
TRUNCATE(748.58,-4) = 0
```

結果のデータ・タイプおよび長さ属性は、最初の引数のデータ・タイプおよび長さ属性と同じになります。

引数が NULL になる可能性があるか、または引数が 10 進浮動小数点数ではなく、データベース構成パラメーターで DFT_SQLMATHWARN が YES に設定されている場合には、結果は NULL になる可能性があります。引数が NULL の場合、結果は NULL 値になります。

TRUNCATE の代わりに TRUNC を指定できます。

例:

- EMPLOYEE 表を使って、最高給与額の社員の平均月給を計算します。結果の小数点より右側の 2 桁を切り捨てます。

  ```
  SELECT TRUNCATE(MAX(SALARY)/12,2)
    FROM EMPLOYEE;
  ```

  最高給与額の社員は年収が $52750.00 であるので、この例では 4395.83 が戻されます。

- 873.726 をそれぞれ小数点以下 2、1、0、-1、および -2 桁に切り捨てた数字が表示されます。

  ```
  VALUES (
    TRUNCATE(873.726,2),
    TRUNCATE(873.726,1),
    TRUNCATE(873.726,0),
    TRUNCATE(873.726,-1),
    TRUNCATE(873.726,-2),
    TRUNCATE(873.726,-3) );
  ```

  この例では、873.720、873.700、873.000、870.000、800.000、および 0.000 が戻されます。

- 873.726 を小数点以下 0 桁に切り捨てた 10 進浮動小数点数が表示されます。

  ```
  VALUES(TRUNCATE(DECFLOAT(873.726),0))
  ```

  は、値 873 を戻します。

TYPE_ID

スキーマは SYSIBM です。

TYPE_ID 関数は、*expression* の動的データ・タイプの内部タイプ ID を戻します。

引数はユーザー定義の構造化タイプでなければなりません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての構造化データ・タイプを引数として受け入れるので、別のユーザー定義タイプをサポートするための追加のシグニチャーを作成する必要はありません。)

この関数の結果のデータ・タイプは INTEGER です。 *expression* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。

TYPE_ID 関数が戻した値は、データベース間で移動することはできません。動的データ・タイプのタイプ・スキーマおよびタイプ名が同じであっても、値が異なる可能性があります。移植性を考慮してコーディングを行う場合、タイプ・スキーマおよびタイプ名の判別には TYPE_SCHEMA および TYPE_NAME 関数を使用してください。

例:

- ある表階層には、タイプ EMP のルート表 EMPLOYEE と、タイプ MGR の副表 MANAGER があります。別の表 ACTIVITIES は、 REF(EMP) SCOPE EMPLOYEE と定義されている WHO_RESPONSIBLE という列を収めています。 ACTIVITIES を参照するたびに、参照に対応する行の内部タイプ ID を表示します。

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_ID(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  DEREF 関数は、行に対応するオブジェクトを戻すときに使用します。

TYPE_NAME

スキーマは SYSIBM です。

TYPE_ID 関数は、*expression* の動的データ・タイプの非修飾名を戻します。

引数はユーザー定義の構造化タイプでなければなりません。 (この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての構造化データ・タイプを引数として受け入れるので、別のユーザー定義タイプをサポートするための追加のシグニチャーを作成する必要はありません。)

この関数の結果のデータ・タイプは VARCHAR(18) です。 *expression* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。 TYPE_SCHEMA 関数を使用して、 TYPE_NAME が戻したタイプ名のスキーマ名を判別してください。

例:

- ある表階層には、タイプ EMP のルート表 EMPLOYEE と、タイプ MGR の副表 MANAGER があります。別の表 ACTIVITIES は、 REF(EMP) SCOPE EMPLOYEE と定義されている WHO_RESPONSIBLE という列を収めています。 ACTIVITIES を参照するたびに、参照に対応する行のタイプを表示します。

  ```
  SELECT TASK, WHO_RESPONSIBLE->NAME,
      TYPE_NAME(DEREF(WHO_RESPONSIBLE)),
      TYPE_SCHEMA(DEREF(WHO_RESPONSIBLE))
    FROM ACTIVITIES
  ```

  DEREF 関数は、行に対応するオブジェクトを戻すときに使用します。

TYPE_SCHEMA

スキーマは SYSIBM です。

TYPE_SCHEMA 関数は、*expression* の動的データ・タイプのスキーマ名を戻します。

引数はユーザー定義の構造化タイプでなければなりません。この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべての構造化データ・タイプを引数として受け入れるので、別のユーザー定義タイプをサポートするための追加のシグニチャーを作成する必要はありません。

この関数の結果のデータ・タイプは VARCHAR(128) です。 *expression* が NULL になる可能性がある場合は、結果も NULL になる可能性があります。 *expression* が NULL であれば、結果も NULL 値になります。 TYPE_NAME 関数を使用して、 TYPE_SCHEMA が戻したスキーマ名に関連するタイプ名を判別してください。

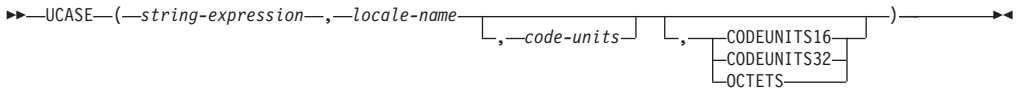
UCASE

スキーマは SYSIBM です。

UCASE 関数は、最初の引数 (*char-string-exp*) だけが指定されるという点を除けば、TRANSLATE 関数と同じです。

UCASE は UPPER の同義語です。

UCASE (ロケール依存)



スキーマは SYSIBM です。

UCASE 関数は、指定したロケールに関連付けられた規則を使用して、すべての文字が大文字に変換されたストリングを戻します。

UCASE は UPPER の同義語です。

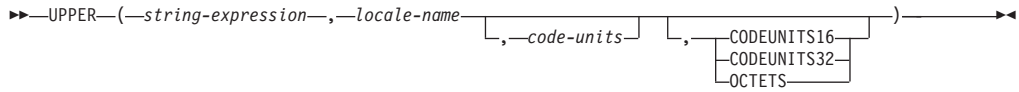
UPPER

スキーマは SYSIBM です。 (この関数の SYSFUN バージョンでは、上位互換性が引き続き有効です。)

UPPER 関数は、最初の引数 (*char-string-exp*) だけが指定されるという点を除けば、TRANSLATE 関数と同じです。

Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

UPPER (ロケール依存)



スキーマは SYSIBM です。

UPPER 関数は、指定したロケールに関連付けられた規則を使用して、すべての文字が大文字に変換されたストリングを戻します。

string-expression
CHAR、VARCHAR、GRAPHIC、または VARGRAPHIC ストリングを戻す式。*string-expression* が CHAR または VARCHAR の場合、この式は FOR BIT DATA であってはなりません (SQLSTATE 42815)。

locale-name
大文字への変換の規則を定義する、ロケールを指定する文字定数。*locale-name* の値は大/小文字の区別がなく、有効なロケールでなければなりません (SQLSTATE 42815)。有効なロケールとその命名については、『SQL および XQuery のロケール名』を参照してください。

code-units
結果内のコード単位の数を指定する整数定数。指定する場合、*code-units* は、結果が文字データの場合は 1 から 32672 までの間の整数、結果がグラフィック・データの場合は 1 から 16336 までの間の整数でなければなりません (SQLSTATE 42815)。 *code-units* が明示的に指定されないと、暗黙のうちに *string-expression* の長さ属性になります。 OCTETS が指定され、結果が GRAPHIC データである場合、*code-units* の値は偶数でなければなりません (SQLSTATE 428GC)。

CODEUNITS16、CODEUNITS32、または OCTETS
code-units のストリング単位を指定します。

CODEUNITS16 は、*code-units* が 16 ビット UTF-16 コード単位で表現されることを指定します。CODEUNITS32 は、*code-units* が 32 ビット UTF-32 コード単位で表現されることを指定します。OCTETS は、*code-units* がバイト単位で表現されることを指定します。

ストリング単位が明示的に指定されなければ、結果のデータ・タイプによって、使用される単位が決定されます。結果が GRAPHIC データであれば、*code-units* は 2 バイト単位で表現されます。それ以外の場合は、バイト単位で表現されます。CODEUNITS16、CODEUNITS32、および OCTETS の詳細については、『文字ストリング』の『組み込み関数のストリング単位』を参照してください。

関数の結果は、*string-expression* が CHAR または VARCHAR の場合は VARCHAR になり、*string-expression* が GRAPHIC または VARGRAPHIC の場合は VARGRAPHIC になります。

結果の長さ属性は、以下の表に示すように、*code-units* の暗黙的または明示的な値、暗黙的または明示的なストリング単位、および結果のデータ・タイプによって決まります。

表 40. ストリング単位および結果タイプの関数としての、UPPER の結果の長さ属性

ストリング単位	文字結果タイプ	GRAPHIC 結果タイプ
CODEUNITS16	MIN(*code-units* * 3, 32768)	*code-units*
CODEUNITS32	MIN(*code-units* * 4, 32768)	MIN(*code-units* * 2, 16336)
OCTETS	*code-units*	MIN(*code-units* / 2, 16336)

結果の実際の長さは、*string-expression* の長さより大きくなる場合があります。結果の実際の長さが結果の長さ属性より大きい場合は、エラーが返されます (SQLSTATE 42815)。結果内のコード単位の数が *code-units* の値を超えた場合は、エラーが返されます (SQLSTATE 42815)。

string-expression が UTF-16 ではない場合、この関数は *string-expression* の UTF-16 へのコード・ページ変換を実行し、次いで結果を UTF-16 から *string-expression* のコード・ページに変換します。どちらかのコード・ページ変換の結果として最低 1 つの置換文字が生じた場合、結果には置換文字が含まれることになり、警告が返されて (SQLSTATE 01517)、SQLCA の警告フラグ SQLWARN8 が「W」に設定されます。

引数のいずれかが NULL になる可能性がある場合、結果も NULL になる可能性があります。引数のいずれかが NULL の場合、その結果は NULL 値です。

例:

- EMPLOYEE 表の列 JOB の値で使用されている文字を大文字で返すようにします。

```
SELECT UPPER(JOB, 'en_US')
  FROM EMPLOYEE
  WHERE EMPNO = '000020'
```

  結果は、値 'MANAGER' になります。

- トルコ語のストリング内のすべての「I」文字について大文字を検索します。

```
VALUES UPPER('I■ıi', 'tr_TR', CODEUNITS16)
```

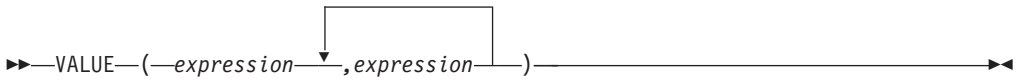
  結果は、ストリング 'I■I■' になります。

- ドイツ語の「ß」(鋭い S) の大文字形式を検出します。

```
VALUES UPPER('ß', 'de', 2, CODEUNITS16)
```

  結果は、ストリング 'SS' になります。この例では *code-units* を指定しなければならないことに注意してください。結果に含まれるコード単位のほうが、*string-expression* に含まれるものよりも多いためです。

VALUE



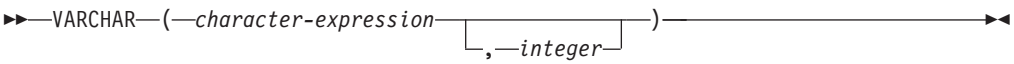
スキーマは SYSIBM です。

VALUE 関数は、NULL 値以外の最初の引数を戻します。

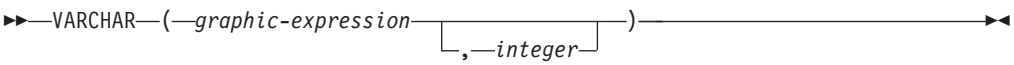
VALUE は COALESCE の同義語です。

VARCHAR

文字 → VARCHAR:



GRAPHIC → VARCHAR:



日付/時刻 → VARCHAR:

スキーマは SYSIBM です。

VARCHAR 関数は、以下の可変長文字ストリング表記を戻します。

- 文字ストリング (最初の引数がいずれかのタイプの文字ストリングの場合)
- GRAPHIC ストリング (Unicode データベースのみ)。これは、最初の引数がいずれかのタイプの GRAPHIC ストリングの場合です。
- 日付/時刻。これは、引数が日付、時刻、またはタイム・スタンプの場合です。

Unicode データベースでは、複数バイト文字を介して出力ストリングが途中で切り捨てられると、次のようになります。

- 入力が文字ストリングであった場合、部分的な文字は 1 つ以上のブランクに置き換えられます。
- 入力が GRAPHIC ストリングであった場合、部分的な文字は空ストリングに置き換えられます。

このどちらの動作も過信しないでください。今後のリリースで変更される可能性があるからです。

関数の結果は、可変長文字ストリングです。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL の場合には、結果も NULL 値です。

文字→ VARCHAR

character-expression
32672 バイトの最大長の文字ストリング・データ・タイプでなければならない値をもつ式。

integer
結果の可変長文字ストリングの長さ属性。値は 0 から 32672 の範囲でなければなりません。この引数を指定しない場合、結果の長さ属性は引数の長さ属性と同じになります。

GRAPHIC → VARCHAR

graphic-expression
値が LONG VARGRAPHIC または DBCLOB 以外の文字ストリングのデータ・タイプでなければならず、しかも 16336 個の 2 バイト文字を最大長としてもつ式。

integer
結果の可変長文字ストリングの長さ属性。値は 0 から 32672 の範囲でなければなりません。この引数を指定しない場合、結果の長さ属性は引数の長さ属性と同じになります。

日付/時刻 → VARCHAR

datetime-expression
日付、時刻、またはタイム・スタンプのいずれかのデータ・タイプでなければならない値をもつ式。

例:

- VARCHAR(8) と定義されたホスト変数 JOB_DESC (VARCHAR(8)) を、社員 Dolores Quintana に関するジョブ記述と同等の VARCHAR (JOB 列の値) に設定します。

```
SELECT VARCHAR(JOB)
  INTO :JOB_DESC
  FROM EMPLOYEE
  WHERE LASTNAME = 'QUINTANA'
```

VARCHAR_BIT_FORMAT

スキーマは SYSIBM です。

VARCHAR_BIT_FORMAT 関数は、文字テンプレートを使ってフォーマットされた文字ストリングのビット・ストリング表現を返します。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

character-expression
CLOB ではない組み込み文字ストリングである値を返す式 (SQLSTATE 42815)。必要な長さは、指定されたフォーマット・ストリングおよび値の解釈方法により決定されます。

format-string
character-expression の値の解釈方法に関するテンプレートの入った文字定数。

有効なフォーマット・ストリングには、`'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'` および `'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'` (SQLSTATE 42815) が含まれます。ここで 'x' または 'X' はそれぞれ、結果の 16 進数字 1 桁に対応します。

この関数の結果は、フォーマット・ストリングに基づいた長さ属性および実際の長さを持つ可変長文字ストリング FOR BIT DATA です。上にリストされている 2 つの有効なフォーマット・ストリングでは、長さ属性は 36 で、実際の長さは 16 バイトです。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL であれば、結果は NULL 値になります。

例

- UUID (Universal Unique Identifier) をそのバイナリー形式で表す

```
VARCHAR_BIT_FORMAT('d83d6360-1818-11db-9804-b622a1ef5492',
                   'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  戻される結果:

```
x'D83D6360181811DB9804B622A1EF5492'
```

- UUID (Universal Unique Identifier) をそのバイナリー形式で表す

```
VARCHAR_BIT_FORMAT('D83D6360-1818-11DB-9804-B622A1EF5492',
                   'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

  戻される結果:

```
x'D83D6360181811DB9804B622A1EF5492'
```

VARCHAR_FORMAT

スキーマは SYSIBM です。

VARCHAR_FORMAT 関数によって、指定した文字テンプレートに従ってフォーマットされたタイム・スタンプのストリング表記が返されます。

timestamp-expression

長さ属性が 254 未満の組み込みタイム・スタンプまたは組み込み文字ストリングのデータ・タイプの値が返される式。戻り値には、タイム・スタンプの有効なストリング表記 (SQLSTATE 42815 または SQLSTATE 22007) が含まれます。タイム・スタンプのストリング表記の有効なフォーマットについては、96 ページの『日付/時刻の値のストリング表記』を参照してください。

timestamp-expression が文字ストリングの場合は、値がタイム・スタンプに変換される前に先行ブランクおよび末尾ブランクが除去されます。

format-string

254 未満の (SQLSTATE 22007) 長さ属性を持つ組み込み文字ストリングのデータ・タイプの値が返される式。 *format-string* には、*timestamp-expression* のフォーマット方法に関するテンプレートが含まれます。

format-string を有効にするには、下にリストしたフォーマット・エレメント (SQLSTATE 22007) の組み合わせを含める必要があります。 2 つのフォーマット・エレメントは、オプションで以下の 1 つ以上の区切り文字で分離することができます。

- ダッシュ (-)
- ピリオド (.)
- スラッシュ (/)
- コンマ (,)
- アポストロフィ (')
- セミコロン (;)
- コロン (:)
- ブランク ( )

区切り文字は *format-string* の始めまたは終わりにも指定できます。

表 41. VARCHAR_FORMAT 関数のフォーマット・エレメント

フォーマット・エレメント	説明
CC	世紀 (01-99)。4 桁の年の最後の 2 桁がゼロの場合は、結果はこの年の最初の 2 桁です。 4 桁の年の最後の 2 桁がゼロでない場合は、結果はこの年の最初の 2 桁に 1 を足したものです。
DD	日 (01-31)。
DDD	年間通算日 (001-366)

表 41. VARCHAR_FORMAT 関数のフォーマット・エレメント (続き)

フォーマット・エレメント	説明
FF[*n*]	小数秒 (0-999999)。数字 *n* は、戻り値に含まれる桁数の指定に使用します。 *n* の有効値は 1 から 6 です。デフォルトは 6 です。
HH	HH の動作は HH12 と同様です。
HH12	12 時間形式の時 (01-12)。AM がデフォルトの午前/午後の指定子です。
HH24	24 時間形式の時 (00-24)。
IW	年の ISO 週番号 (01-53)。週は月曜日から始まり、7 日から成ります。第 1 週は、木曜日が含まれる年の第 1 週目であり、これは 1 月 4 日が含まれる第 1 週と同等です。
I	ISO 年 (0-9)。返される ISO 週番号に基づく年の最後の桁。
IY	ISO 年 (00-99)。返される ISO 週番号に基づく年の最後の 2 桁。
IYY	ISO 年 (000-999)。返される ISO 週番号に基づく年の最後の 3 桁。
IYYY	ISO 年 (0000-9999)。返される ISO 週番号に基づく 4 桁の年。
J	ユリウス日 (紀元前 4713 年 1 月 1 日からの日数)。
MI	分 (00-59)。
MM	月 (01-12)。
NNNNNN	マイクロ秒 (000000-999999)。FF6 と同様。
Q	四半期 (1-4)。例えば、1 月から 3 月までの月では 1 が返されます。
RR	RR の動作は YY と同様です。
RRRR	RRRR の動作は YYYY と同様です。
SS	秒 (00-59)。
SSSSS	直近の午前 0 時からの秒数 (00000-86400)。
W	月の週 (1-5)。第 1 週は、月の最初の日から始まり 7 日目で終わります。
WW	年の週 (01-53)。第 1 週は、1 月 1 日から始まり 1 月 7 日で終わります。
Y	年の最後の 1 桁 (0-9)。
YY	年の最後の 2 桁 (00-99)。
YYY	年の最後の 3 桁 (000-999)。
YYYY	4 桁の年 (0000-9999)。

結果は、*format-string* で指定したフォーマットの *timestamp-expression* の表記です。*format-string* は、オプションで 1 つ以上の区切り文字で分離できる一連のフォーマット・エレメントとして解釈されます。 *format-string* 内の文字のストリングは、

551 ページの表 41 にある一致する最長のフォーマット・エレメントとして解釈されます。同じ文字が含まれる 2 つのフォーマット・エレメントを区切り文字で区切らない場合は、この指定は、左から始めて上の表内の一致する最長のフォーマット・エレメントとして解釈され、このフォーマット・ストリングの残りに一致するものが見つかるまで続けられます。 例えば、'YYYYYYDD' のフォーマット・エレメントは 'YYYY'、'YY'、および 'DD' と解釈されます。

結果は、可変長文字ストリングです。長さ属性は、100 と *format-string* の長さ属性の大きい方です。また、*format-string* は、長さ属性や実際の結果の長さも決定します。結果のストリングは、結果の長さ属性を超えてはいけません (SQLSTATE 22007)。引数のいずれかが NULL 値になる可能性がある場合、結果も NULL 値になる可能性があります。引数のいずれかが NULL 値の場合、その結果は NULL 値です。

注:

- ***代替構文:*** TO_CHAR は VARCHAR_FORMAT の同義語です。

例:

- 名前が 'SYSU' で始まるすべてのシステム表の名前と作成タイム・スタンプを表示します。

```
SELECT VARCHAR(TABNAME, 20) AS TABLE_NAME,
  VARCHAR_FORMAT(CREATE_TIME, 'YYYY-MM-DD HH24:MI:SS')
    AS CREATION_TIME
  FROM SYSCAT.TABLES
  WHERE TABNAME LIKE 'SYSU%'
```

  この例では、以下を戻します。

```
TABLE_NAME          CREATION_TIME
------------------- -------------------
SYSUSERAUTH         2000-05-19 08:18:56
SYSUSEROPTIONS      2000-05-19 08:18:56
```

- 変数の TMSTAMP に 2007-03-09-14.07.38.123456 という値が設定されているとします。以下の各例では、いくつかの関数の呼び出しと結果のストリング値が示されています。各ケースの結果タイプは VARCHAR(100) です。

関数呼び出し	結果
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMISSFF3')	20070309020738123
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHH24MISS')	20070309140738
VARCHAR_FORMAT(TMSTAMP,'YYYYMMDDHHMI')	200703090207
VARCHAR_FORMAT(TMSTAMP,'DD/MM/YY')	09/03/07
VARCHAR_FORMAT(TMSTAMP,'MM-DD-YYYY')	03-09-2007
VARCHAR_FORMAT(TMSTAMP,'J')	2454169
VARCHAR_FORMAT(TMSTAMP,'Q')	1
VARCHAR_FORMAT(TMSTAMP,'W')	2
VARCHAR_FORMAT(TMSTAMP,'IW')	10
VARCHAR_FORMAT(TMSTAMP,'WW')	10

VARCHAR_FORMAT_BIT

スキーマは SYSIBM です。

VARCHAR_FORMAT_BIT 関数は、文字テンプレートを使ってフォーマットされたビット・ストリングの文字表現を戻します。

bit-data-expression
組み込み文字ストリング FOR BIT DATA データ・タイプである値を返す式 (SQLSTATE 42815)。必要な長さは、指定されたフォーマット・ストリングおよび値の解釈方法により決定されます。

format-string
結果のフォーマット方法に関するテンプレートの入った文字定数。

有効なフォーマット・ストリングには、'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx' および 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX' (SQLSTATE 42815) が含まれます。ここで 'x' または 'X' はそれぞれ、*bit-data-expression* に含まれる 16 進数字 1 桁に対応します。

この関数の結果は、フォーマット・ストリングに基づいた長さ属性および実際の長さを持つ可変長文字ストリングです。上にリストされている 2 つの有効なフォーマット・ストリングでは、長さ属性は 36 で、実際の長さは 36 バイトです。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL であれば、結果は NULL 値になります。

例

- UUID (Universal Unique Identifier) をその定様式で表す

```
VARCHAR_FORMAT_BIT(cast(x'd83d6360181811db9804b622a1ef5492'
                        as varchar(16) for bit data),
                   'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx')
```

  戻される結果:

```
'd83d6360-1818-11db-9804-b622a1ef5492'
```

- UUID (Universal Unique Identifier) をその定様式で表す

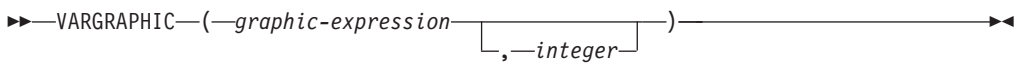
```
VARCHAR_FORMAT_BIT(cast(x'd83d6360181811db9804b622a1ef5492'
                        as char(16) for bit data),
                   'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')
```

  戻される結果:

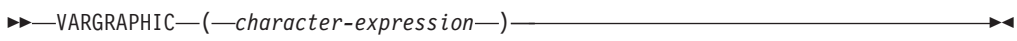
```
'D83D6360-1818-11DB-9804-B622A1EF5492'
```

VARGRAPHIC

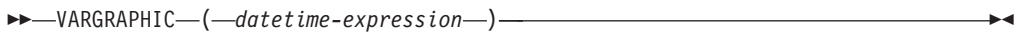
GRAPHIC → VARGRAPHIC:



文字→ VARGRAPHIC:



日付/時刻 → VARGRAPHIC:



スキーマは SYSIBM です。

VARGRAPHIC 関数は、以下の可変長 GRAPHIC ストリング表記を戻します。

- GRAPHIC ストリング (最初の引数がいずれかのタイプの GRAPHIC ストリングの場合)
- 文字ストリング。最初の引数がいずれかの型の文字ストリングの場合には、 1 バイト文字は 2 バイト文字に変換されます。
- 日付/時刻 (Unicode データベースのみ)。これは、引数が日付、時刻、またはタイム・スタンプの場合です。

Unicode データベースでは、指定した引数が文字ストリングであると、まず GRAPHIC ストリングに変換されてから、関数が実行されます。最後の文字が高サロゲートになるように出力ストリングが切り捨てられた場合、そのサロゲートはブランク文字 (X'0020') に変換されます。この動作を過信しないでください。今後のリリースで変更される可能性があるからです。

この関数の結果は、可変長 GRAPHIC ストリング (VARGRAPHIC データ・タイプ) です。最初の引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。最初の引数が NULL であれば、結果は NULL 値になります。

GRAPHIC → VARGRAPHIC

graphic-expression
GRAPHIC ストリング値を戻す式。

integer
結果の VARGRAPHIC データ・タイプの長さ属性を指定する整数値。値は 0 から 16336 の範囲でなければなりません。値を指定しない場合の結果の長さ属性は、最初の引数の長さ属性と同じになります。

GRAPHIC 式の長さが結果の長さ属性より長い場合、結果は切り捨てられます。その場合、切り捨てられた文字がすべてブランクで、 GRAPHIC 式が LONG ストリング (LONG VARGRAPHIC または DBCLOB) でない限り、警告 (SQLSTATE 01004) が戻されます。

文字→ VARGRAPHIC

character-expression
値が LONG VARCHAR または CLOB 以外の文字ストリングのデータ・タイプでなければならず、しかも 16336 バイトを最大長としてもつ式。

結果の長さ属性は、引数の長さ属性と同じになります。

character-expression 内の各 1 バイト文字は、結果においては、それに相当する 2 バイト表記または 2 バイト置換文字に変換されます。 *character-expression* 内の各 2 バイト文字は、それ以外の変換なしでマップされます。 2 バイト文字の最初の 1 バイトが *character-expression* の最後のバイトとして示される場合、それは 2 バイトの置換文字に変換されます。 *character-expression* 内の文字順序はそのまま維持されます。

Unicode データベースの場合、この関数は文字ストリングをオペランドのコード・ページから UCS-2 へと変換します。 2 バイト文字をはじめとして、オペランドのすべての文字が変換されます。 2 番目の引数の値を指定すると、結果のストリングに必要な長さが指定されます (UCS-2 文字)。

VARGRAPHIC 関数による 2 バイト・コード・ポイントへの変換は、オペランドのコード・ページに基づいています。

オペランドの 2 バイト文字は変換されません。その他の文字はすべて、それに対応する同等の 2 バイト文字に変換されます。対応する 2 バイトで相当するものが存在しない場合には、コード・ページ用の 2 バイト置換文字が使用されます。

1 つ以上の 2 バイト置換文字が結果内に戻されても警告やエラー・コードは生成されません。

日付/時刻 → VARGRAPHIC

datetime-expression
日付、時刻、またはタイム・スタンプのいずれかのデータ・タイプでなければならない値をもつ式。

WEEK

引数の年間通算週番号を 1 から 54 の範囲の整数値として戻します。週は日曜日から始まります。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

WEEK_ISO

スキーマは SYSFUN です。

引数の年間通算週番号を、1 から 53 の範囲の整数値で戻します。週は月曜日から始まり、常に 7 日から成ります。第 1 週は、木曜日の入った年の第 1 週目であり、それは 1 月 4 日の入った第 1 週と同等です。よって、年の初めの 3 日間までを、前の年の最終週と表す可能性があります。逆に、年の最後の 3 日間までを、翌年の最初の週と表す可能性もあります。

引数は、日付またはタイム・スタンプであるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

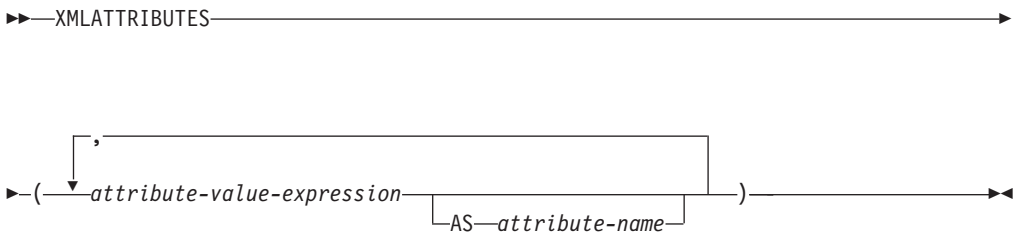
関数の結果は INTEGER になります。結果は NULL になる可能性があります。引数が NULL である場合、その結果は NULL 値になります。

例:

以下のリストは、WEEK_ISO および DAYOFWEEK_ISO の結果例です。

```
DATE       WEEK_ISO    DAYOFWEEK_ISO
---------- ----------- -------------
1997-12-28          52             7
1997-12-31           1             3
1998-01-01           1             4
1999-01-01          53             5
1999-01-04           1             1
1999-12-31          52             5
2000-01-01          52             6
2000-01-03           1             1
```

XMLATTRIBUTES



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLATTRIBUTES 関数は、引数から XML 属性を構成します。この関数は、XMLELEMENT 関数の引数としてのみ使用できます。結果は、それぞれの非 NULL 入力値の XQuery 属性ノードを含む XML シーケンスになります。

attribute-value-expression
結果が属性値になる式。データ・タイプ *attribute-value-expression*を、構造化タイプとすることはできません (SQLSTATE 42884)。式には任意の SQL 式を指定できます。式が単純な列参照でない場合、属性名を指定する必要があります。

attribute-name
属性名を指定します。この名前は SQL ID であり、XML 修飾名の形式かまたは QName でなければなりません (SQLSTATE 42634)。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。属性名を `xmlns` にしたり、その前に `xmlns:` を付けたりすることはできません。名前空間は、関数 XMLNAMESPACES を使って宣言します。重複した属性名を (暗黙的、明示的にかかわらず) 使用することはできません (SQLSTATE 42713)。

attribute-name が指定されない場合、*attribute-value-expression* は列名でなければなりません (SQLSTATE 42703)。属性名は、列名から XML 属性名への完全にエスケープしたマッピングを使用する列名から作成されます。

結果のデータ・タイプは XML です。 *attribute-value-expression* の結果が NULL になる可能性がある場合は、結果も NULL になる可能性があります。
attribute-value-expression の結果がすべて NULL であれば、結果も NULL 値になります。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**: BLOB データ・タイプ、および FOR BIT DATA と定義されている文字ストリング・データはサポートされません (SQLSTATE 42884)。

例:

注: XMLATTRIBUTES は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- エレメント、およびその属性を生成します。

```
SELECT E.EMPNO, XMLELEMENT(
  NAME "Emp",
  XMLATTRIBUTES(
    E.EMPNO, E.FIRSTNME ||' '|| E.LASTNAME AS "name"
```

```
  )
)
AS "Result"
FROM EMPLOYEE E WHERE E.EDLEVEL = 12
```

この照会は、次のような結果を生成します。

```
EMPNO  Result
000290 <Emp EMPNO="000290" name="JOHN PARKER"></Emp>
000310 <Emp EMPNO="000310" name="MAUDE SETRIGHT"></Emp>
200310 <Emp EMPNO="200310" name="MICHELLE SPRINGER"></Emp>
```

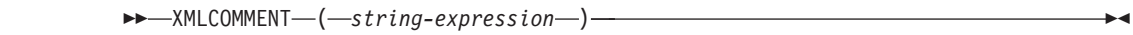
- エレメント、および QName で使用されないネーム・スペース宣言を生成します。属性値には接頭部が使用されます。

```
VALUES XMLELEMENT(
  NAME "size",
  XMLNAMESPACES(
    'http://www.w3.org/2001/XMLSchema-instance' AS "xsi",
    'http://www.w3.org/2001/XMLSchema' AS "xsd"
  ),
  XMLATTRIBUTES(
    'xsd:string' AS "xsi:type"
  ), '1'
)
```

この照会は、次のような結果を生成します。

```
<size xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:type="xsd:string">1</size>
```

XMLCOMMENT



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLCOMMENT 関数は、XQuery コメント・ノードを 1 つ持つ XML 値を戻します。その内容は入力引数です。

string-expression
値が文字ストリング・タイプ CHAR、VARCHAR、または CLOB を持つ式。*string-expression* の結果は構文解析され、XML 1.0 規則で指定されている XML コメントの要件との適合性が検査されます。 *string-expression* の結果は次の正規表現に従っていなければなりません。

```
((Char - '-') | ('-' (Char - '-')))*
```

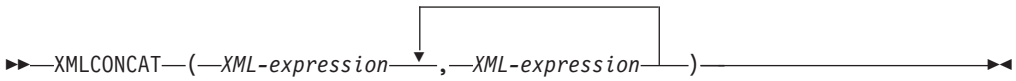
`Char` は、任意の Unicode 文字として定義されます。ただしサロゲート・ブロック X'FFFE' および X'FFFF' は除きます。基本的に、XML コメントに隣接する 2 つのハイフンを含めることはできません。また、XML コメントをハイフンで終了することもできません (SQLSTATE 2200S)。

結果のデータ・タイプは XML です。 *string-expression* の結果が NULL になる可能性がある場合、結果も NULL になる可能性があります。入力値が NULL であれば、結果も NULL 値になります。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**:
   XMLCOMMENT はサポートされません (SQLSTATE 42997)。

XMLCONCAT



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLCONCAT 関数は、可変数の XML 入力引数の連結を含むシーケンスを戻します。

XML-expression
データ・タイプ XML の式を指定します。

結果のデータ・タイプは XML です。結果は、非 NULL の入力 XML 値の連結を含む XML シーケンスになります。入力内の NULL 値は無視されます。いずれかの *XML-expression* の結果が NULL になる可能性がある場合は、結果も NULL になる可能性があります。各入力値の結果が NULL であれば、結果も NULL 値になります。

注:

1. **複数のデータベース・パーティションを持つデータベースでのサポート**: XML 関数のネストの外部レベルの結果は、XMLSERIALIZE 関数の引数でなければなりません (SQLSTATE 42997)。

例:

注: XMLCONCAT は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- ファーストネーム別にソートされた従業員のリストを含む、部門 A00 および B01 の部門エレメントを構成します。部門エレメントの直前に紹介コメントを含めます。

```
SELECT XMLCONCAT(
  XMLCOMMENT(
    'Confirm these employees are on track for their product schedule'
  ),
  XMLELEMENT(
    NAME "Department",
    XMLATTRIBUTES(
      E.WORKDEPT AS "name"
    ),
    XMLAGG(
      XMLELEMENT(
        NAME "emp", E.FIRSTNME
      )
    ORDER BY E.FIRSTNME
    )
  )
)
FROM EMPLOYEE E
WHERE E.WORKDEPT IN ('A00', 'B01')
GROUP BY E.WORKDEPT
```

この照会は、次のような結果を生成します。

```
<!--Confirm these employees are on track for their product schedule-->
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>DIAN</emp>
<emp>GREG</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
<!--Confirm these employees are on track for their product schedule-->
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

XMLDOCUMENT

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLDOCUMENT 関数は、XQuery 文書ノードを 1 つ持つ XML 値を戻します。これにはゼロ個以上の子ノードが含まれます。

XML-expression
XML 値を戻す式。 XML 値のシーケンス項目が属性ノードであってはなりません (SQLSTATE 10507)。

結果のデータ・タイプは XML です。 *XML-expression* の結果が NULL になる可能性がある場合、結果も NULL になる可能性があります。入力値が NULL であれば、結果も NULL 値になります。

その結果生成される文書ノードの子は、以下のステップの説明に従って構成されます。入力式はノードのシーケンスまたは原子値です。これはこのステップで内容シーケンスとして参照されます。

1. 内容シーケンスに文書ノードが含まれる場合、内容シーケンスの文書ノードはその文書ノードの子によって置き換えられます。
2. 内容シーケンス内の 1 つ以上の原子値の各隣接シーケンスは、各原子値をストリング (隣接する値の間にブランク文字を 1 つ挿入したもの) にキャストした結果を含むテキスト・ノードで置き換えられます。
3. 内容シーケンスの各ノードごとにノードの新規ディープ・コピーが構成されます。ノードのディープ・コピーとは、そのノードをルートとするサブツリー全体のコピーのことです (これにはノードそのものとその子孫が含まれます)。コピーされた各ノードは、新しいノード ID を持ちます。コピーされたエレメントおよび属性ノードは、それぞれのタイプのアノテーションを保持します。
4. 内容シーケンス内のノードは、新規文書ノードの子になります。

XMLDOCUMENT 関数は、XQuery が計算する文書コンストラクターを効果的に実行します。以下の関数があるとします。

```
XMLQUERY('document {$E}' PASSING BY REF XML-expression AS "E")
```

これは、以下と同じ意味になります。

```
XMLDOCUMENT( XML-expression )
```

ただし、*XML-expression* が NULL であり、XMLQUERY が空シーケンスを戻す場合 (XMLDOCUMENT の場合は NULL 値) は例外です。

注:

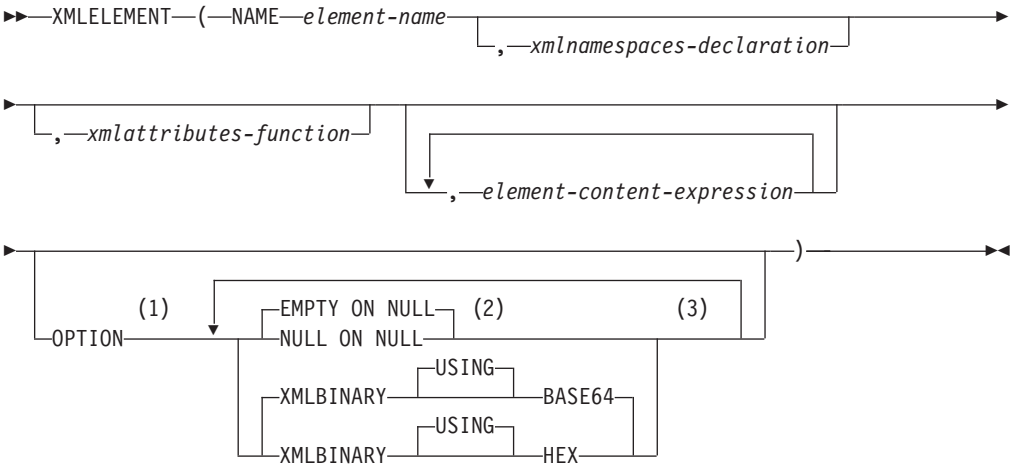
1. **複数のデータベース・パーティション・データベースでのサポート**:
   XMLDOCUMENT はサポートされません (SQLSTATE 42997)。

例:

- 作成された文書を XML 列に挿入します。

```
INSERT INTO T1 VALUES(
  123, (
    SELECT XMLDOCUMENT(
      XMLELEMENT(
        NAME "Emp", E.FIRSTNME || ' ' || E.LASTNAME, XMLCOMMENT(
          'This is just a simple example'
        )
      )
    )
    FROM EMPLOYEE E
    WHERE E.EMPNO = '000120'
  )
)
```

XMLELEMENT



注:

1 OPTION 節は、少なくとも 1 つの *xmlattributes-function* または *element-content-expression* が指定されている場合にのみ指定できます。

2 NULL ON NULL または EMPTY ON NULL は、少なくとも 1 つの *element-content-expression* が指定されている場合にのみ指定できます。

3 同じ節を複数回指定することはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLELEMENT 関数は、XQuery エレメント・ノードである XML 値を戻します。

NAME *element-name*
XML エレメントの名前を指定します。この名前は SQL ID であり、XML 修飾名の形式かまたは QName でなければなりません (SQLSTATE 42634)。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。名前が修飾される場合は、名前空間の接頭部をその有効範囲内で宣言する必要があります (SQLSTATE 42635)。

xmlnamespaces-declaration
XMLNAMESPACES 宣言の結果である XML 名前空間宣言を指定します。宣言される名前空間は、XMLELEMENT 関数の有効範囲内です。この名前空間は、それらが別の副選択内に現れるかどうかに関係なく、XMLELEMENT 関数内のネストされた XML 関数に適用されます。

xmlnamespaces-declaration が指定されない場合、名前空間宣言は構成されたエレメントとは関連付けられません。

xmlattributes-function
エレメントの XML 属性を指定します。この属性は、XMLATTRIBUTES 関数の結果です。

element-content-expression
生成される XML エレメント・ノードの内容を、式によって、または式リスト

によって指定します。データ・タイプ *element-content-expression*を、構造化タイプとすることはできません (SQLSTATE 42884)。式には任意の SQL 式を指定できます。

element-content-expression が指定されない場合、空ストリングがエレメントの内容として使用され、OPTION NULL ON NULL または EMPTY ON NULL を指定することはできません。

OPTION

XML エレメントを構成するための追加オプションを指定します。 OPTION 節を指定しない場合、デフォルトは EMPTY ON NULL XMLBINARY USING BASE64 です。この節は、*element-content-expression* で指定するネストされた XMLELEMENT の呼び出しに影響を与えません。

EMPTY ON NULL or NULL ON NULL

各 *element-content-expression* の値が NULL 値の場合に NULL 値を戻すか、あるいは空エレメントを戻すかを指定します。このオプションはエレメントの内容の NULL 処理にのみ影響し、属性値の NULL 処理には影響を及ぼしません。デフォルトは EMPTY ON NULL です。

EMPTY ON NULL

それぞれの *element-content-expression* の値が NULL であれば、空のエレメントが戻されます。

NULL ON NULL

それぞれの *element-content-expression* の値が NULL であれば、NULL 値が戻されます。

XMLBINARY USING BASE64 or XMLBINARY USING HEX

バイナリー入力データ、FOR BIT DATA 属性を持つ文字ストリング・データ、またはこれらのタイプのいずれかに基づく特殊タイプの想定エンコードを指定します。エンコードはエレメントの内容または属性値に適用されます。デフォルトは XMLBINARY USING BASE64 です。

XMLBINARY USING BASE64

想定エンコードが Base64 文字である (XML スキーマ・タイプ xs:base64Binary のエンコードに対して定義される) ことを指定します。Base64 エンコードは、US-ASCII の 65 文字のサブセット (10 個の数字、26 個の小文字、26 個の大文字、'+' および '/') を使用して、サブセット内の 1 つの印刷可能文字により、バイナリーまたはビット・データのすべての 6 ビットを表します。文字を普遍的に表せるようにこれらの文字が選ばれています。この方法を使うと、エンコード・データのサイズが元のバイナリーまたはビット・データより 33 % 大きくなります。

XMLBINARY USING HEX

想定エンコードが 16 進文字である (XML スキーマ・タイプ xs:hexBinary のエンコードに対して定義される) ことを指定します。 16 進数エンコードは、各バイト (8 ビット) を 2 つの 16 進文字で表します。この方法を使うと、エンコード・データが元のバイナリーまたはビット・データの 2 倍のサイズになります。

この関数は、エレメント名、オプションの名前空間宣言の集合、オプションの属性の集合、および XML エレメントの内容を構成するゼロ個以上の引数をとります。この結果は、XML エレメント・ノードを含む XML シーケンスまたは NULL 値です。

結果のデータ・タイプは XML です。いずれかの *element-content-expression* 引数が NULL の可能性がある場合、結果も NULL になる可能性があります。すべての *element-content-expression* 引数値が NULL で NULL ON NULL オプションが有効になっている場合、結果は NULL 値になります。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**: 関数は、バージョン 8 のものだけがサポートされます。XML 値関数のネストの外部レベルの結果は、XMLSERIALIZE 関数の引数でなければなりません。NULL 処理のオプションとバイナリー・エンコードのオプションは指定できません (SQLSTATE 42997)。 BLOB、および FOR BIT DATA として定義されている文字ストリング・データは指定できません (SQLSTATE 42884)。

   デフォルトのネーム・スペースを定義する別のエレメントの内容としてコピーされるエレメントを構成する場合、デフォルトのネーム・スペースはコピーされたエレメント内で明示的に宣言解除する必要があります。これは、新規の親エレメントからデフォルトのネーム・スペースを継承した結果として生じる可能性のあるエラーを避けるためです。事前定義ネーム・スペース接頭部 (「xs」、「xsi」、「xml」、および「sqlxml」) をその使用時に明示的に宣言する必要もあります。
2. **エレメント・ノードの構成**: 結果のエレメント・ノードは以下のように構成されます。
   a. *xmlnamespaces-declaration* は、構成されたエレメントの有効範囲内の名前空間のセットを追加します。それぞれの有効範囲内の名前空間は、名前空間接頭部 (またはデフォルトの名前空間) を、名前空間 URI と関連付けます。有効範囲内の名前空間は、エレメントの有効範囲内の QNames の解釈に使用できる名前空間接頭部のセットを定義します。
   b. xmlattributes-function を指定した場合、それが評価され、結果は属性ノードのシーケンスになります。
   c. それぞれの *element-content-expression* は評価され、結果は以下のようにノードのシーケンスに変換されます。
      - 結果のタイプが XML でない場合は、XML にマッピングされる *element-content-expression* の結果が内容である、XML テキスト・ノードに変換されます。このマッピングは、SQL データ値から XML データ値へのマッピングの規則に従います (『データ・タイプ間のキャスト』の非 XML 値から XML 値へのサポートされるキャストを説明する表を参照)。
      - 結果タイプが XML である場合、一般に結果は項目のシーケンスになります。そのシーケンス内のいくつかの項目は、文書ノードである場合があります。シーケンス内の各文書ノードは、その最上位の子のシーケンスによって置き換えられます。次いで結果のシーケンス内の各ノードに対して、その子と属性を組み込んだノードの新しいディープ・コピーが構成されます。コピーされた各ノードは、新しいノード ID を持ちます。コピーされ

たエレメントおよび属性ノードは、それぞれのタイプのアノテーションを保持します。シーケンス内で戻される 1 つ以上の原子値の隣接する各シーケンスごとに、新規テキスト・ノードが構成され、隣接値の間に単一ブランク文字が挿入された、ストリングに対する各原子値のキャストの結果が含まれます。内容のシーケンス内の隣接するテキスト・ノードは、ブランクを間に挟まずにその内容を連結して単一のテキスト・ノードにマージされます。連結後に、内容がゼロ長ストリングであるテキスト・ノードは、内容のシーケンスから削除されます。

d. XML 属性の結果のシーケンス、およびすべての *element-content-expression* 指定の結果のシーケンスは、内容シーケンスと呼ばれる 1 つのシーケンスに連結されます。内容シーケンス内の隣接するテキスト・ノードのすべてのシーケンスは、単一のテキスト・ノードにマージされます。すべての *element-content-expression* 引数が空ストリングである場合、または *element-content-expression* 引数が指定されていない場合、空のエレメントが戻されます。

e. 内容シーケンスには、属性ノードではないノードに続けて属性ノードを含めることはできません (SQLSTATE 10507)。内容シーケンス内の属性ノードは、新規エレメント・ノードの属性になります。これらの属性ノードの複数が同じ名前を持つことはできません (SQLSTATE 10503)。名前空間 URI が、構成されたエレメントの有効範囲内名前空間にない場合、名前空間宣言は、属性ノードの名前で使用されるすべての名前空間に対応して作成されます。

f. 内容シーケンス内のエレメント、テキスト、コメント、および処理命令ノードは、構成されたエレメント・ノードの子になります。

g. 構成されたエレメント・ノードには `xs:anyType` のタイプ・アノテーションが与えられ、その各属性には `xdt:untypedAtomic` のタイプ・アノテーションが与えられます。構成されたエレメント・ノードのノード名は、NAME キーワードの後に指定された element-name です。

3. **XMLELEMENT 内での名前空間の使用規則**: 名前空間の範囲に関する以下の規則を考慮してください。

   - XMLNAMESPACES declaration で宣言された名前空間は、XMLELEMENT 関数で構成されるエレメント・ノードの有効範囲内名前空間です。エレメント・ノードが直列化される場合、そのそれぞれの有効範囲内名前空間は名前空間属性として直列化されます。ただしこれは、エレメント・ノードの親の有効範囲内名前空間であったり、親エレメントも直列化されていたりするのではない場合に限ります。
   - XMLQUERY または XMLEXISTS が *element-content-expression* にある場合、名前空間は XMLQUERY または XMLEXISTS の XQuery 式の静的に既知の名前空間になります。静的に既知の名前空間は、XQuery 式内の QNames を解決するために使用されます。 XQuery プロローグが、XQuery 式の有効範囲内で、同じ接頭部を持つ名前空間を宣言する場合、プロローグ内で宣言される名前空間は、XMLNAMESPACES 宣言内で宣言された名前空間をオーバーライドします。
   - 構成されたエレメントの属性が *element-content-expression* に由来するものである場合、その名前空間は構成されたエレメントの有効範囲内名前空間としてまだ宣言されていない可能性があり、その場合には、それに対して新規名前空間

が作成されます。これが結果として競合になる場合、属性名の接頭部がすでに異なる URI に有効範囲内ネーム・スペースによりバインド済みであるということです。DB2 はそのような競合の原因にならない接頭部を生成し、属性名で使用されている接頭部は新規接頭部に変更され、ネーム・スペースがその新規接頭部に対して作成されます。生成される新規接頭部は、「db2ns-xx」というパターンに従います。ここで x は、A から Z、a から z、0 から 9 の範囲から選択された文字です。以下に例を示します。

```
VALUES XMLELEMENT(
  NAME "c", XMLQUERY(
    'declare namespace ipo="www.ipo.com"; $m/ipo:a/@ipo:b'
      PASSING XMLPARSE(
        DOCUMENT '<tst:a xmlns:tst="www.ipo.com" tst:b="2"/>'
      ) AS "m"
  )
)
```

これは、以下のものを戻します。

```
<c xmlns:tst="www.ipo.com" tst:b="2"/>
```

2 番目の例は、以下のようなものです。

```
VALUES XMLELEMENT(
  NAME "tst:c", XMLNAMESPACES(
    'www.tst.com' AS "tst"
  ),
  XMLQUERY(
    'declare namespace ipo="www.ipo.com"; $m/ipo:a/@ipo:b'
      PASSING XMLPARSE(
        DOCUMENT '<tst:a xmlns:tst="www.ipo.com" tst:b="2"/>'
      ) AS "m"
  )
)
```

これは、以下のものを戻します。

```
<tst:c xmlns:tst="www.tst.com" xmlns:db2ns-a1="www.ipo.com"
  db2ns-a1:b="2"/>
```

例:

注: XMLELEMENT は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- エレメントを NULL ON NULL オプションを使用して構成します。

```
SELECT E.FIRSTNME, E.LASTNAME, XMLELEMENT(
  NAME "Emp", XMLELEMENT(
    NAME "firstname", E.FIRSTNME
  ),
  XMLELEMENT(
    NAME "lastname", E.LASTNAME
  )
  OPTION NULL ON NULL
)
AS "Result"
FROM EMPLOYEE E
WHERE E.EDLEVEL = 12
```

この照会は、次のような結果を生成します。

```
FIRSTNME     LASTNAME        Emp
JOHN         PARKER          <Emp><firstname>JOHN</firstname>
                               <lastname>PARKER</lastname></Emp>
```

```
MAUDE        SETRIGHT        <Emp><firstname>MAUDE</firstname>
                               <lastname>SETRIGHT</lastname></Emp>
MICHELLE     SPRINGER        <Emp><firstname>MICHELLE</firstname>
                               <lastname>SPRINGER</lastname></Emp>
```

- 子エレメントとしてネストしたエレメントのリストを使用してエレメントを生成します。

```
SELECT XMLELEMENT(
  NAME "Department", XMLATTRIBUTES(
    E.WORKDEPT AS "name"
  ),
  XMLAGG(
    XMLELEMENT(
      NAME "emp", E.FIRSTNME
    )
    ORDER BY E.FIRSTNME
  )
)
AS "dept_list"
FROM EMPLOYEE E
WHERE E.WORKDEPT IN ('A00', 'B01')
GROUP BY WORKDEPT
```

この照会は、次のような結果を生成します。

```
dept_list
<Department name="A00">
<emp>CHRISTINE</emp>
<emp>SEAN</emp>
<emp>VINCENZO</emp>
</Department>
<Department name="B01">
<emp>MICHAEL</emp>
</Department>
```

- デフォルトの XML エレメント名前空間を指定し、副選択を使用して、ネストされた XML エレメントを作成します。

```
SELECT XMLELEMENT(
         NAME "root",
         XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
         XMLATTRIBUTES(cid),
            (SELECT
               XMLAGG(
                 XMLELEMENT(
                   NAME "poid", poid
                 )
               )
              FROM purchaseorder
              WHERE purchaseorder.custid = customer.cid
            )
       )
FROM customer
WHERE cid = '1002'
```

このステートメントは、以下のような、ルート・エレメント内でデフォルト・エレメント名前空間が宣言された XML 文書を戻します。

```
<root xmlns="http://mytest.uri" CID="1002">
  <poid>5000</poid>
  <poid>5003</poid>
  <poid>5006</poid>
</root>
```

- XML 名前空間を指定した共通表式を使用します。

共通表式を使用して XML エレメントを構成する際に、このエレメントを同じ SQL ステートメント内の他の場所で使用する場合は、エレメントの構成の一部として名前空間宣言を指定する必要があります。以下のステートメントは、PURCHASEORDER 表を使用して poid エレメントを作成する共通表式と、CUSTOMER 表を使用してルート・エレメントを作成する SELECT ステートメントの両方でデフォルトの XML 名前空間を指定します。

```
WITH tempid(id, elem) AS
  (SELECT custid, XMLELEMENT(NAME "poid",
     XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
       poid)
   FROM purchaseorder )
SELECT XMLELEMENT(NAME "root",
         XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
            XMLATTRIBUTES(cid),
            (SELECT XMLAGG(elem)
             FROM tempid
             WHERE tempid.id = customer.cid )
        )
FROM customer
WHERE cid = '1002'
```

このステートメントは、以下のような、ルート・エレメント内でデフォルト・エレメント名前空間が宣言された XML 文書を戻します。

```
<root xmlns="http://mytest.uri" CID="1002">
  <poid>5000</poid>
  <poid>5003</poid>
  <poid>5006</poid>
</root>
```

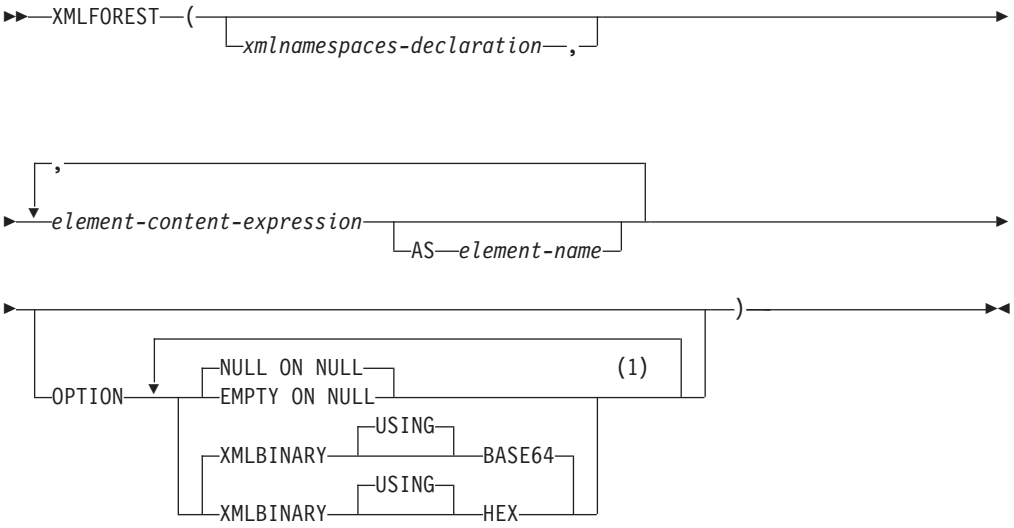
以下のステートメントでは、CUSTOMER 表を使用してルート・エレメントを作成する SELECT ステートメントのみでデフォルト・エレメント名前空間が宣言されています。

```
WITH tempid(id, elem) AS
    (SELECT custid, XMLELEMENT(NAME "poid", poid)
     FROM purchaseorder )
SELECT XMLELEMENT(NAME "root",
         XMLNAMESPACES(DEFAULT 'http://mytest.uri'),
            XMLATTRIBUTES(cid),
            (SELECT XMLAGG(elem)
             FROM tempid
             WHERE tempid.id = customer.cid )
       )
FROM customer
WHERE cid = '1002'
```

このステートメントは、以下のような、ルート・エレメント内でデフォルト・エレメント名前空間が宣言された XML 文書を戻します。poid エレメントはデフォルト・エレメント名前空間宣言のない共通表式内で作成されるので、poid エレメントのデフォルト・エレメント名前空間は定義されません。XML 文書内で、poid エレメントのデフォルト・エレメント名前空間は空ストリング "" に設定されます。その理由は、poid エレメントのデフォルト・エレメント名前空間は定義されず、poid エレメントはルート・エレメント xmlns="http://mytest.uri" のデフォルト・エレメント名前空間に所属しないからです。

```
<root xmlns="http://mytest.uri" CID="1002">
  <poid xmlns="">5000</poid>
  <poid xmlns="">5003</poid>
  <poid xmlns="">5006</poid>
</root>
```

XMLFOREST



注:

1 同じ節を複数回指定することはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLFOREST 関数は、一連の XQuery エレメント・ノードである XML 値を戻します。

xmlnamespaces-declaration
XMLNAMESPACES 宣言の結果である XML 名前空間宣言を指定します。宣言される名前空間は、XMLFOREST 関数の有効範囲内にあります。名前空間は、別の副選択の中で現れるかどうかにかかわらず、XMLFOREST 関数内でネストされているすべての XML 関数に対して適用されます。

xmlnamespaces-declaration を指定しないと、名前空間宣言と構成されたエレメントとの関連付けが行われません。

element-content-expression
生成される XML エレメント・ノードの内容を式によって指定します。データ・タイプ *element-content-expression*を、構造化タイプとすることはできません (SQLSTATE 42884)。式には任意の SQL 式を指定できます。式が単純な列参照でない場合、エレメント名を指定する必要があります。

AS *element-name*
SQL ID として XML エレメント名を指定します。エレメント名は、 XML 修飾名の形式であるかまたは QName (SQLSTATE 42634) でなければなりません。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。名前が修飾される場合は、名前空間の接頭部をその有効範囲内で宣言する必要があります (SQLSTATE 42635)。 *element-name* が指定されない場合、*element-content-expression* は列名でなければなりません (SQLSTATE 42703、SQLCODE -206)。エレメント名は、列名から QName への完全にエスケープしたマッピングを使用する列名から作成されます。

OPTION

XML エレメントを構成するための追加オプションを指定します。 OPTION 節を指定しない場合、デフォルトは NULL ON NULL XMLBINARY USING BASE64 です。この節は、*element-content-expression* で指定するネストされた XMLELEMENT の呼び出しに影響を与えません。

EMPTY ON NULL or NULL ON NULL

各 *element-content-expression* の値が NULL 値の場合に NULL 値を戻すか、あるいは空エレメントを戻すかを指定します。このオプションはエレメントの内容の NULL 処理にのみ影響し、属性値の NULL 処理には影響を及ぼしません。デフォルトは NULL ON NULL です。

EMPTY ON NULL

それぞれの *element-content-expression* の値が NULL であれば、空のエレメントが戻されます。

NULL ON NULL

それぞれの *element-content-expression* の値が NULL であれば、NULL 値が戻されます。

XMLBINARY USING BASE64 or XMLBINARY USING HEX

バイナリー入力データ、FOR BIT DATA 属性を持つ文字ストリング・データ、またはこれらのタイプのいずれかに基づく特殊タイプに対して想定されるエンコードを指定します。エンコードはエレメントの内容または属性値に適用されます。デフォルトは XMLBINARY USING BASE64 です。

XMLBINARY USING BASE64

想定エンコードが Base64 文字である (XML スキーマ・タイプ xs:base64Binary のエンコードに対して定義される) ことを指定します。Base64 エンコードは、US-ASCII の 65 文字のサブセット (10 個の数字、26 個の小文字、26 個の大文字、'+' および '/') のうち、1 つの印刷可能文字を示すことにより、バイナリーまたはビット・データの任意の 6 ビットを表します。文字を普遍的に表せるようにこれらの文字が選ばれています。この方法を使うと、エンコード・データのサイズが元のバイナリーまたはビット・データより 33 % 大きくなります。

XMLBINARY USING HEX

想定エンコードが 16 進文字である (XML スキーマ・タイプ xs:hexBinary のエンコードに対して定義される) ことを指定します。 16 進数エンコードは、各バイト (8 ビット) を 2 つの 16 進文字で表します。この方法を使うと、エンコード・データが元のバイナリーまたはビット・データの 2 倍のサイズになります。

この関数は、任意指定のネーム・スペース宣言のセット、および名前とエレメントの内容を構成する 1 つ以上の引数 (エレメント・ノードが 1 つ以上ある場合) を取ります。結果は、一連の XQuery エレメント・ノードまたは NULL 値を含む XML シーケンスとなります。

結果のデータ・タイプは XML です。 *element-content-expression* 引数のいずれかに NULL の可能性がある場合、結果も NULL になる可能性があります。すべての *element-content-expression* 引数値が NULL で NULL ON NULL オプションが有効になっている場合、結果は NULL 値になります。

XMLFOREST 関数は XMLCONCAT および XMLELEMENT を使って表現できます。例えば、以下の 2 つの式は意味的には同じです。

```
XMLFOREST(xmlnamespaces-declaration, arg1 AS name1, arg2 AS name2 ...)
```

```
XMLCONCAT(
  XMLELEMENT(
    NAME name1, xmlnamespaces-declaration, arg1
  ),
  XMLELEMENT(
    NAME name2, xmlnamespaces-declaration, arg2
  )
  ...
)
```

注:

1. **複数のデータベース・パーティション・データベースでのサポート**: 関数は、バージョン 8 のものだけがサポートされます。XML 値関数のネストの外部レベルの結果は、XMLSERIALIZE 関数の引数でなければなりません。NULL 処理のオプションとバイナリー・エンコードのオプションは指定できません (SQLSTATE 42997)。 BLOB、および FOR BIT DATA として定義されている文字ストリング・データは指定できません (SQLSTATE 42884)。

   デフォルトのネーム・スペースを定義する別のエレメントの内容としてコピーされるエレメントを構成する場合、デフォルトのネーム・スペースはコピーされたエレメント内で明示的に宣言解除する必要があります。これは、新規の親エレメントからデフォルトのネーム・スペースを継承した結果として生じる可能性のあるエラーを避けるためです。事前定義ネーム・スペース接頭部 (「xs」、「xsi」、「xml」、および「sqlxml」) をその使用時に明示的に宣言する必要もあります。

例:

注: XMLFOREST は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- デフォルトのネーム・スペースを持つエレメントのフォレストを構成します。

```
SELECT EMPNO,
  XMLFOREST(
    XMLNAMESPACES(
      DEFAULT 'http://hr.org', 'http://fed.gov' AS "d"
    ),
    LASTNAME, JOB AS "d:job"
  )
AS "Result"
FROM EMPLOYEE
WHERE EDLEVEL = 12
```

  この照会は、次のような結果を生成します。

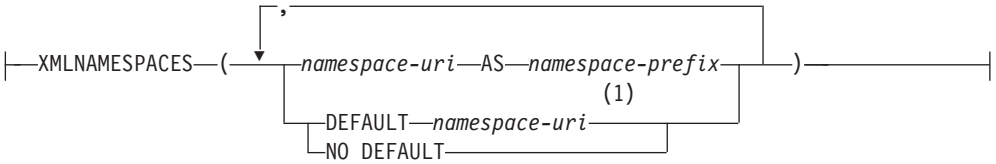
```
EMPNO  Result
000290 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">PARKER
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>

000310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SETRIGHT
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
```

```
200310 <LASTNAME xmlns="http://hr.org" xmlns:d="http://fed.gov">SPRINGER
  </LASTNAME>
<d:job xmlns="http://hr.org" xmlns:d="http://fed.gov">OPERATOR</d:job>
```

XMLNAMESPACES

xmlnamespaces-declaration:



注:

1 DEFAULT または NO DEFAULT は、XMLNAMESPACES の引数内で 一度しか指定できません。

スキーマは SYSIBM です。宣言名を修飾名で指定することはできません。

XMLNAMESPACES 宣言は、引数からネーム・スペース宣言を作成します。この宣言は、特定の関数、たとえば XMLELEMENT、XMLFOREST、および XMLTABLE の引数としてのみ使用できます。結果は、それぞれの非 NULL 入力値の有効範囲内のネーム・スペースを含む 1 つ以上の XML ネーム・スペース宣言となります。

namespace-uri
SQL 文字ストリング定数に、ネーム・スペースの URI を指定します。 *namespace-prefix* と併せて使用する場合には、この文字ストリング定数を空にしてはなりません (SQLSTATE 42815)。

namespace-prefix
ネーム・スペースの接頭部を指定します。この接頭部は SQL ID であり、XML NCName の形式でなければなりません (SQLSTATE 42634)。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。接頭部を `xml` または `xmlns` にすることはできず、ネーム・スペース宣言リストの中で固有でなければなりません (SQLSTATE 42635)。

DEFAULT *namespace-uri*
このネーム・スペース宣言の有効範囲内で使用するデフォルトのネーム・スペースを指定します。 *namespace-uri* はネストされる有効範囲内の別の DEFAULT 宣言または NO DEFAULT 宣言によってオーバーライドされなければ、有効範囲内の非修飾名に適用されます。

NO DEFAULT
このネーム・スペース宣言の有効範囲内ではデフォルトのネーム・スペースが使用されないことを指定します。ネストされる有効範囲内の DEFAULT 宣言によってオーバーライドされなければ、有効範囲内のデフォルトのネーム・スペースはありません。

結果のデータ・タイプは XML です。結果は、各指定ネーム・スペースの XML ネーム・スペース宣言となります。結果が NULL 値になることはありません。

例:

注: XMLNAMESPACES は、出力の中にブランク・スペースまたは改行文字を挿入しません。例の出力はすべて、読みやすくするために書式を整えています。

- adm:employee という名前の XML エレメントと、 XML 属性 adm:department を生成します。このどちらにも、 adm を接頭部としてもつネーム・スペースが関連付けられます。

```
SELECT EMPNO, XMLELEMENT(
  NAME "adm:employee", XMLNAMESPACES(
    'http://www.adm.com' AS "adm"
  ),
  XMLATTRIBUTES(
    WORKDEPT AS "adm:department"
  ),
LASTNAME
)
FROM EMPLOYEE
WHERE JOB = 'ANALYST'
```

この照会は、次のような結果を生成します。

```
000130 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  QUINTANA</adm:employee>
000140 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  NICHOLLS</adm:employee>
200140 <adm:employee xmlns:adm="http://www.adm.com" adm:department="C01">
  NATZ</adm:employee>
```

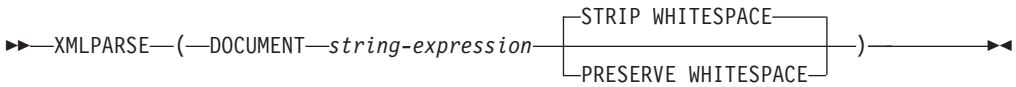
- デフォルトの名前空間に関連付けられた「employee」という名前の XML エレメントと、デフォルトの名前空間を使用するサブエレメント「department」を持ったデフォルトの名前空間を使用しない「job」という名前のサブエレメントを作成します。

```
SELECT EMP.EMPNO, XMLELEMENT(
  NAME "employee", XMLNAMESPACES(
    DEFAULT 'http://hr.org'
  ),
  EMP.LASTNAME, XMLELEMENT(
    NAME "job", XMLNAMESPACES(
      NO DEFAULT
    ),
    EMP.JOB, XMLELEMENT(
      NAME "department", XMLNAMESPACES(
        DEFAULT 'http://adm.org'
      ),
      EMP.WORKDEPT
    )
  )
)
FROM EMPLOYEE EMP
WHERE EMP.EDLEVEL = 12
```

この照会は、次のような結果を生成します。

```
000290 <employee xmlns="http://hr.org">PARKER<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
000310 <employee xmlns="http://hr.org">SETRIGHT<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
200310 <employee xmlns="http://hr.org">SPRINGER<job xmlns="">OPERATOR
  <department xmlns="http://adm.org">E11</department></job></employee>
```

XMLPARSE



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLPARSE 関数は、引数を XML 文書として構文解析し、XML 値を戻します。

DOCUMENT
構文解析される文字ストリング式が、XML 名前空間勧告による変更を受けた XML 1.0 に準拠する整形式 XML 文書に評価されることを指定します (SQLSTATE 2200M)。

string-expression
文字ストリングまたは BLOB 値を戻す式を指定します。パラメーター・マーカーを使用する場合は、サポートされるデータ・タイプの 1 つにこれを明示的にキャストする必要があります。

STRIP WHITESPACE or PRESERVE WHITESPACE
入力引数内の空白を保持するかどうかを指定します。どちらも指定されない場合、デフォルトは STRIP WHITESPACE です。

STRIP WHITESPACE
最長で 1000 バイトの空白文字だけを含むテキスト・ノードを除去することを指定します (最も近い収容エレメントが属性 xml:space='preserve' を持たない場合)。 1000 バイトを超える空白で始まるテキスト・ノードがある場合、エラーが戻されます (SQLSTATE 54059)。

CDATA セクション内の空白文字も、このオプションから影響を受けます。DTD のエレメントに DOCTYPE 宣言があっても、エレメントのコンテンツ・モデルは空白を除去するかどうかを決めるためには使用されません。。

PRESERVE WHITESPACE
すべての空白を保持することを指定します (最も近い収容エレメントが属性 xml:space='default' を持つ場合でも同様です)。

結果のデータ・タイプは XML です。 *string-expression* の結果が NULL になる可能性がある場合、結果も NULL になる可能性があります。 *string-expression* の結果が NULL であれば、結果も NULL 値になります。

注:

1. **複数のデータベース・パーティションを持つデータベースでのサポート**: XMLPARSE はサポートされません (SQLSTATE 42997)。
2. **入力ストリングのエンコード**: 入力ストリングには、XML 文書内の文字のエンコードを識別する XML 宣言を含めることができます。ストリングを文字ストリングとして XMLPARSE 関数に渡す場合、これはデータベース・サーバーでコード・ページに変換されます。このコード・ページは、発信元のコード・ページや XML 宣言で識別されるエンコードとは異なる場合があります。

そのため、アプリケーションは文字ストリングの入力を持つ XMLPARSE を直接使用することは避けるべきです。アプリケーションはホスト変数を直接使用して XML 文書を含むストリングを送信する必要があります。これは、外部コード・ページと XML 宣言内のエンコードの間の一致を保つためです。この状況で XMLPARSE を使用する必要がある場合には、引数として BLOB タイプを指定します。これにより、コード・ページの変換を避けられます。

3. **DTD の処理**: 外部文書タイプ定義 (DTD) およびエンティティーをデータベース内に登録する必要があります。内部および外部 DTD は両方とも、有効な構文であるか検査されます。構文解析処理中に、以下のアクションも実行されます。
   - 内部および外部 DTD により定義されるデフォルト値が適用される。
   - エンティティー参照およびパラメーター・エンティティーがそれぞれの拡張フォームによって置換される。
   - 内部 DTD と外部 DTD が同一のエレメントを定義する場合、エラーが戻される (SQLSTATE 2200M)。
   - 内部 DTD と外部 DTD が同一のエンティティーまたは属性を定義する場合、内部定義が選択される。

   構文解析後、内部 DTD およびエンティティーは、外部 DTD およびエンティティーの参照と同様に、保管された値の表記では保存されません。
4. **非 UTF-8 データベースでの文字変換**: XML 文書が非 Unicode データベース・サーバーに構文解析されるとき、文字データ・タイプのホスト変数かパラメーター・マーカーから文書が渡されるか、または文字ストリング・リテラルから文書が渡される場合に、コード・ページ変換が行われます。 タイプ XML、BLOB または FOR BIT DATA (CHAR FOR BIT DATA または VARCHAR FOR BIT DATA) のホスト変数またはパラメーター・マーカーを使用した XML 文書の構文解析は、コード・ページ変換を行いません。文字データ・タイプが使用されるときには、XML 文書内のすべての文字がターゲット・データベース・コード・ページで一致するコード・ポイントを持っていることを注意深く確認してください。そうでない場合、置換文字が導入される場合があります。構成パラメーター **enable_xmlchar** を使用すると、非 Unicode データベースに保管されている XML データの整合性を確保する助けが得られます。このパラメーターを「NO」に設定すると、文字データ・タイプからの XML 文書の挿入をブロックします。BLOB および FOR BIT DATA データ・タイプは、これらのデータ・タイプを使用してデータベースに受け渡される文書がコード・ページ変換を行わないため、引き続き許可されています。

例

PRESERVE WHITESPACE オプションを使用すると、表に挿入された XML 文書内の空白文字 (description エレメント内の空白文字を含む) を保持します。

```
INSERT INTO PRODUCT VALUES ('100-103-99','Tool bag',14.95,NULL,NULL,NULL,
XMLPARSE( DOCUMENT
  '<produce xmlns="http://posample.org" pid="100-103-99">
    <description>
     <name>Tool bag</name>
     <details>
      Super Deluxe tool bag:
      - 26 inches long, 12 inches wide
      - Curved padded handle
      - Locking latch
      - Reinforced exterior pockets
```

```
    </details>
    <price>14.95</price>
    <weight>3 kg</weight>
   </description>
  </product>' PRESERVE WHITESPACE ));
```

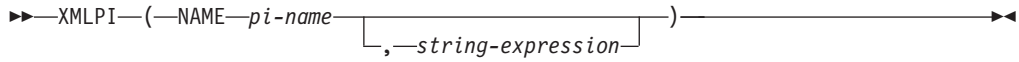
以下の SELECT ステートメントを実行します。

```
SELECT XMLQUERY ('$d/*:product/*:description/*:details' passing DESCRIPTION as "d" )
FROM PRODUCT WHERE PID = '100-103-99' ;
```

これにより、空白文字を使用して details エレメントが戻ります。

```
<details xmlns="http://posample.org">
  Super Deluxe tool bag:
  - 26 inches long, 12 inches wide
  - Curved padded handle
  - Locking latch
  - Reinforced exterior pockets
</details>
```

XMLPI



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLPI 関数は、XQuery 処理命令ノードを 1 つ持つ XML 値を戻します。

NAME *pi-name*
処理命令の名前を指定します。この名前は SQL ID であり、XML NCName の形式でなければなりません (SQLSTATE 42634)。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。どのような大/小文字の組み合わせにしても、この名前を「xml」にすることはできません (SQLSTATE 42634)。

string-expression
文字ストリングである値を戻す式。結果のストリングは UTF-8 に変換されます。これは、XML 1.0 規則で指定されている XML 処理命令の内容に従っていなければなりません (SQLSTATE 2200T)。

- ストリングにサブストリング「?>」を含めてはなりません。このサブストリングは処理命令を終了させるからです。
- ストリングの各文字には任意の Unicode 文字を使用できます。ただし、サロゲート・ブロック X'FFFE' および X'FFFF' は除きます。

結果として生成されるストリングは、構成される処理命令ノードの内容になります。

結果のデータ・タイプは XML です。 *string-expression* の結果が NULL になる可能性がある場合、結果も NULL になる可能性があります。 *string-expression* の結果が NULL であれば、結果も NULL 値になります。 *string-expression* が空ストリングであるかまたは指定されない場合、空の処理命令ノードが戻されます。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**: XMLPI はサポートされません (SQLSTATE 42997)。

例:

- XML 処理命令ノードを生成します。

```
SELECT XMLPI(
  NAME "Instruction", 'Push the red button'
)
FROM SYSIBM.SYSDUMMY1
```

この照会は、次のような結果を生成します。

```
<?Instruction Push the red button?>
```

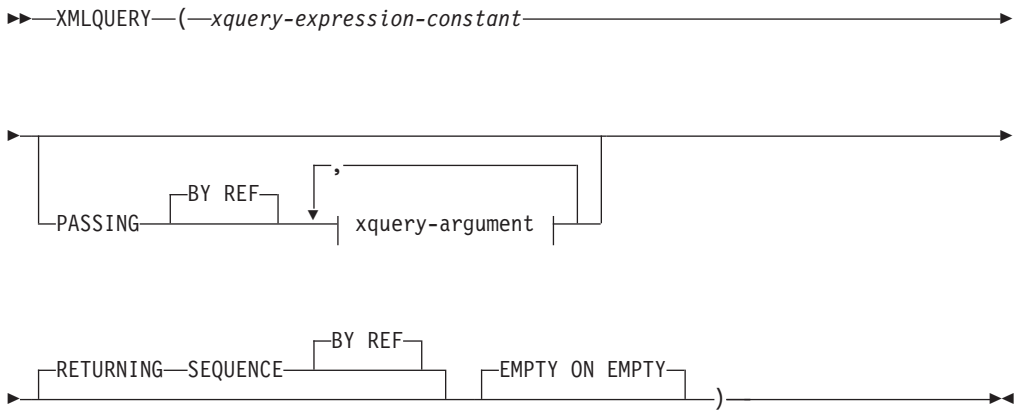
- 空の XML 処理命令ノードを生成します。

```
SELECT XMLPI(
  NAME "Warning"
)
FROM SYSIBM.SYSDUMMY1
```

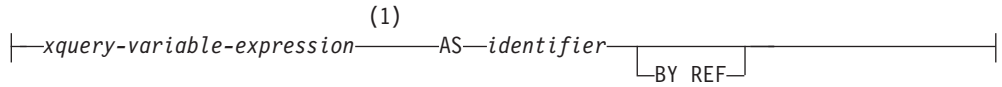
この照会は、次のような結果を生成します。

```
<?Warning ?>
```

XMLQUERY



xquery-argument:



注:

1 式のデータ・タイプを DECFLOAT にすることはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

場合によっては指定した入力引数を XQuery 変数として使用して、XMLQUERY 関数は XML 値を XQuery 式の評価から戻します。

xquery-expression-constant
サポートされる XQuery 言語構文を使用して、XQuery 式として解釈される SQL 文字ストリング定数を指定します。定数ストリングは、XQuery ステートメントとして構文解析される前に UTF-8 に変換されます。 XQuery 式は、オプション・セットの入力 XML 値を使用して実行し、XMLQUERY 式の値としても戻される出力シーケンスを戻します。 *xquery-expression-constant* の値は、空ストリングまたはブランク文字のストリングにすることはできません (SQLSTATE 10505)。

PASSING
入力値、およびそれらの値を *xquery-expression-constant* で指定された XQuery 式に渡す方法を指定します。デフォルトでは、関数が呼び出された有効範囲内にあるすべての固有の列名が、列の名前を変数名として使用して XQuery 式に暗黙的に渡されます。指定の xquery-argument 内の *identifier* が有効範囲内の列名と一致する場合、明示的な xquery-argument はその暗黙的な列をオーバーライドして XQuery 式に渡されます。

BY REF
デフォルトの受け渡しメカニズムを、データ・タイプ XML の任意の *xquery-variable-expression* または戻り値の参照によると指定します。 XML 値を参照で渡す場合、XQuery の評価は、入力ノード・ツリーがあればそれを使用します。その場合は指定された入力式から直接、元のノードの ID お

よび文書順序を含めすべてのプロパティーを保持したまま使用します。 2 つの引数が同じ XML 値を渡す場合、その 2 つの入力引数の間に含まれている何らかのノードに関係するノード ID 比較および文書順序比較は、同じ XML ノード・ツリー内のノードを参照する場合があります。

この節は、非 XML 値の受け渡しには影響を与えません。非 XML 値は、XML へのキャスト中に値の新規コピーを作成します。

xquery-argument

xquery-expression-constant により指定された XQuery 式に渡される引数を指定します。引数は、値およびその値が渡される方法を指定します。引数には、評価される SQL 式が組み込まれます。

- 結果の値は、XML 型である場合、*input-xml-value* になります。 NULL の XML 値は、XML の空シーケンスに変換されます。
- 結果の値は、XML 型でない場合、XML データ・タイプにキャスト可能でなければなりません。 NULL 値は、XML の空シーケンスに変換されます。変換される値は、*input-xml-value* になります。

xquery-expression-constant が評価されるとき、XQuery 変数は *input-xml-value* と等しい値、および AS 節により指定された名前で示されます。

xquery-variable-expression

実行中に *xquery-expression-constant* により指定された XQuery 式が使用できる値を持つ SQL 式を指定します。式には、シーケンス参照 (SQLSTATE 428F9) または OLAP 関数 (SQLSTATE 42903) を含めることはできません。式のデータ・タイプを DECFLOAT にすることはできません。

AS *identifier*

xquery-variable-expression により生成された値が、*xquery-expression-constant* に XQuery 変数として渡されることを指定します。変数名は *identifier* になります。 XQuery 言語の変数名に先行する先頭のドル記号 ($) は、*identifier* には含められません。 identifier は有効な XQuery 変数名でなければならず、XML NCName に制限されます (SQLSTATE 42634)。 identifier は、長さが 128 バイトを超えてはなりません。同じ PASSING 節内の 2 つの引数が同じ identifier を使用することはできません (SQLSTATE 42711)。

BY REF

XML 入力値が参照により渡されるように指示します。 XML 値を参照で渡す場合、XQuery の評価は、入力ノード・ツリーがあればそれを使用します。その場合は指定された入力式から直接、元のノードの ID および文書順序を含めすべてのプロパティーを保持したまま使用します。2 つの引数が同じ XML 値を渡す場合、その 2 つの入力引数の間に含まれている何らかのノードに関係するノード ID 比較および文書順序比較は、同じ XML ノード・ツリー内のノードを参照する場合があります。 BY REF が *xquery-variable-expression* に続いて指定されない場合、XML 引数は、PASSING キーワードに続く構文により提供されるデフォルトの受け渡しメカニズムによって渡されます。このオプション

は、非 XML 値に指定することはできません。非 XML 値が渡される場合、値は XML に変換されます。このプロセスによりコピーが作成されます。

RETURNING SEQUENCE
XMLQUERY 式がシーケンスを戻すことを指示します。

BY REF
XQuery 式の結果を参照により戻すことを指示します。この値にノードが含まれる場合、XQuery 式の戻り値を使用する式は、元のノードの ID および文書順序を含めすべてのノードのプロパティーを保持したまま、ノード参照を直接受け取ります。参照されるノードは、そのノード・ツリー内で接続されたままです。
BY REF 節が指定されず、PASSING が指定されている場合、デフォルトの受け渡しメカニズムが使用されます。 BY REF が指定されず、PASSING も指定されていない場合、デフォルトの受け渡しメカニズムは BY REF です。

EMPTY ON EMPTY
XQuery 式の処理による空のシーケンスの結果を、空のシーケンスとして戻すことを指定します。

結果のデータ・タイプは XML であり、NULL にはできません。

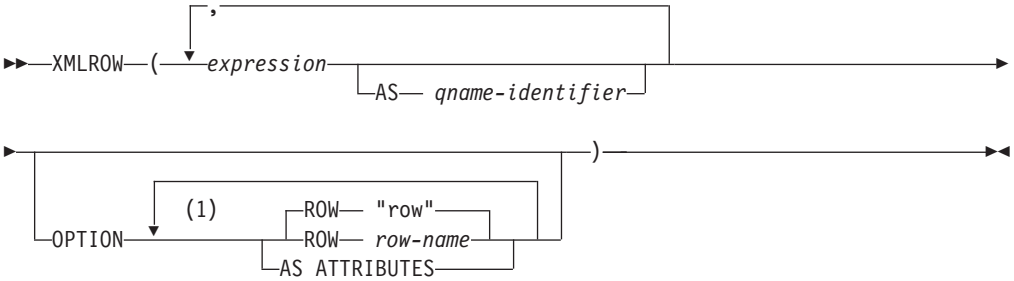
XQuery 式の評価の結果がエラーになる場合、XMLQUERY 関数は XQuery エラーを戻します (SQLSTATE クラス '10')。

注:

1. **XMLQUERY の使用上の制限**: XMLQUERY 関数は、下記のものにはできません。
   - JOIN 演算子または MERGE ステートメントと関連した ON 節の一部 (SQLSTATE 42972)
   - CREATE INDEX EXTENSION ステートメントの GENERATE KEY USING または RANGE THROUGH 節の一部 (SQLSTATE 428E3)
   - CREATE FUNCTION (外部スカラー) ステートメント内の FILTER USING 節の一部、または CREATE INDEX EXTENSION ステートメント内の FILTER USING 節の一部 (SQLSTATE 428E4)
   - チェック制約の一部、または列生成式の一部 (SQLSTATE 42621)
   - group-by 節の一部 (SQLSTATE 42822)
   - 列関数の引数の一部 (SQLSTATE 42607)
2. **副照会としての XMLQUERY**: 副照会として動作する XMLQUERY 式は、副照会を制限するステートメントにより制限される可能性があります。
3. **複数のデータベース・パーティションを持つデータベースでのサポート**: XMLQUERY はサポートされません (SQLSTATE 42997)。

XMLROW

XMLROW 関数は、XQuery 文書ノードを 1 つ持つ XML 値を戻します。これには最上位エレメント・ノードが 1 つ含まれています。



注:

1 同じ節を複数回指定することはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

expression
生成される各 XML エレメント・ノードの内容を式によって指定します。式のデータ・タイプを、構造化タイプとすることはできません (SQLSTATE 42884)。式には任意の SQL 式を指定できます。式が単純な列参照でない場合、エレメント名を指定する必要があります。

AS *qname-identifier*
SQL ID として XML エレメント名または属性名を指定します。 *qname-identifier* は、XML 修飾名の形式であるかまたは QName でなければなりません (SQLSTATE 42634)。有効な名前の詳細は、W3C の XML 名前空間仕様を参照してください。名前が修飾される場合は、名前空間の接頭部をその有効範囲内で宣言する必要があります (SQLSTATE 42635)。 *qname-identifier* が指定されない場合、 *expression* は列名でなければなりません (SQLSTATE 42703、SQLCODE -206)。エレメント名または属性名は、列名から QName への完全にエスケープしたマッピングを使用する列名から作成されます。

OPTION
XML 値を構成するための追加オプションを指定します。 OPTION 節を指定しない場合、デフォルトの動作が適用されます。

AS ATTRIBUTES
各式を列名または *qname-identifier* (属性名としての役割を果たす) を使用して属性値にマップすることを指定します。

ROW *row-name*
各行のマップ先のエレメントの名前を指定します。オプションが指定されない場合のデフォルトのエレメント名は "row" です。

注

デフォルトで、結果セットの各行は次のように XML 値にマップされます。

- 各行は "row" という名前を持つ XML エレメントに変換され、各列はネストされたエレメントに変換されます。その際、エレメント名として列名が使用されます。
- ヌル処理の動作は NULL ON NULL です。列の値が NULL の場合、そのマップ先のサブエレメントは空になります。すべての列の値が NULL の場合、関数によって NULL 値が戻されます。
- BLOB および FOR BIT DATA データ・タイプのバイナリー・コード化スキームは base64Binary エンコードです。
- XML の結果を、単一ルートを持つ整形式 XML 文書とするために、文書ノードが暗黙的に行エレメントに追加されます。

例

列 C1 および C2 を持つ次のような表 T1 があるとします。列にはリレーショナル形式で格納された数値データが入っています。

```
C1          C2
----------- -----------
          1           2
          -           2
          1           -
          -           -

  4 record(s) selected.
```

- 以下の例は、XMLRow 照会とデフォルト動作による出力断片が示されています。表を表すために一連の行エレメントがその中で使用されています。 :

```
SELECT XMLROW(C1, C2) FROM T1
```

```
<row><C1>1</C1><C2>2</C2></row>
<row><C2>2</C2></row>
<row><C1>1</C1></row>

  4 record(s) selected.
```

- 以下の例は、XMLRow 照会と属性を中心としたマッピングによる出力断片を示しています。リレーショナル・データは前例のようにネストされたエレメントとして現れておらず、エレメント属性にマップされています。

```
SELECT XMLROW(C1, C2 OPTION AS ATTRIBUTES) FROM T1
```

```
<row C1="1" C2="2"/>
<row C2="2"/>
<row C1="1"/>

  4 record(s) selected.
```

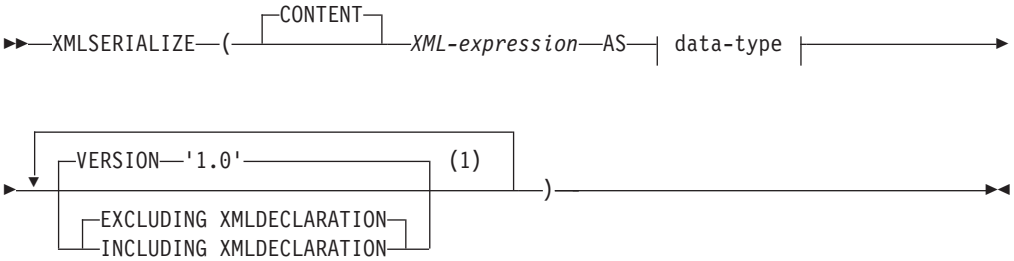
- 以下の例は、XMLRow 照会とデフォルトの <row> エレメントが <entry> によって置き換えられた出力断片を示しています。列 C1 と C2 が <column1> と <column2> エレメントで返され、C1 と C2 の合計が <total> エレメント内に返されます。

```
SELECT XMLROW(
   C1 AS "column1", C2 AS "column2",
   C1+C2 AS "total" OPTION ROW "entry")
FROM T1
```

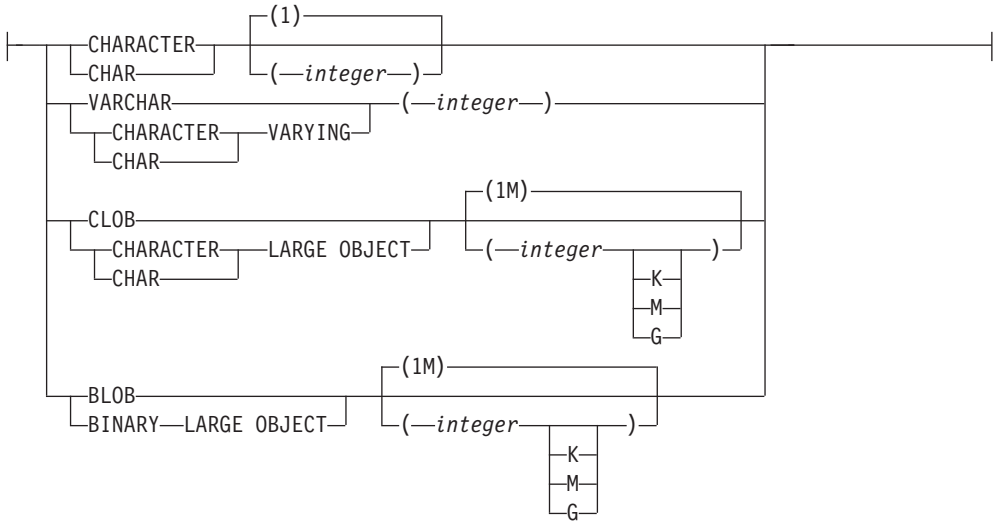
```
<entry><column1>1</column1><column2>2</column2><total>3</total></entry>
<entry><column2>2</column2></entry>
<entry><column1>1</column1></entry>

  4 record(s) selected.
```

XMLSERIALIZE



data-type:



注:

1 同じ節を複数回指定することはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLSERIALIZE 関数は、*XML-expression* 引数から生成された、指定されたデータ・タイプのシリアライズされた XML 値を戻します。

CONTENT
任意の XML 値を指定でき、シリアライゼーションの結果はこの入力値に基づくことを指定します。

XML-expression
データ・タイプ XML の値を戻す式を指定します。 XML シーケンス値に、属性ノードである項目を含めることはできません (SQLSTATE 2200W)。これはシリアライゼーション・プロセスへの入力です。

AS *data-type*
結果タイプを指定します。指定された結果のデータ・タイプの暗黙のまたは明示的な長さ属性は、シリアライズされた出力を収める十分な大きさがなければなりません (SQLSTATE 22001)。

VERSION '1.0'
シリアライズされた値の XML バージョンを指定します。サポートされている唯一のバージョンは '1.0' であり、これはストリング定数として指定する必要があります (SQLSTATE 42815)。

EXCLUDING XMLDECLARATION または INCLUDING XMLDECLARATION
XML 宣言を結果に含めるかどうかを指定します。デフォルトは EXCLUDING XMLDECLARATION です。

EXCLUDING XMLDECLARATION
XML 宣言を結果に含めないように指定します。

INCLUDING XMLDECLARATION
XML 宣言を結果に含めるように指定します。 XML 宣言はストリング '<?xml version="1.0" encoding="UTF-8"?>' です。

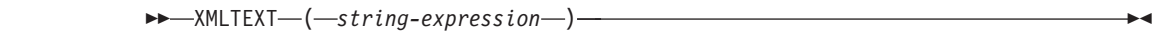
結果は、ユーザーが指定したデータ・タイプになります。 XML シーケンスは効率的に変換され、結果の XML ノードをシリアライズする前に XMLDOCUMENT を *XML-expression* に適用することで、単一の文書ノードを持つようになります。 *XML-expression* の結果が NULL になる可能性がある場合、結果も NULL になる可能性があります。 *XML-expression* の結果が NULL であれば、結果も NULL 値になります。

注:

1. **複数のデータベース・パーティションを持つデータベースでのサポート**: 関数はバージョン 8 での場合と同じようにサポートされるだけです。CONTENT キーワードを指定する必要があり、BLOB データ・タイプは指定できず、XMLDECLARATION オプションは指定できません (SQLSTATE 42997)。
2. **シリアライズされた結果のエンコード**: シリアライズされた結果は、UTF-8 でエンコードされます。 XMLSERIALIZE が文字データ・タイプで使用され、INCLUDING XMLDECLARATION 節が指定されている場合、シリアライズされた XML を含む結果の文字ストリングは、文字ストリングのコード・ページと一致しない XML エンコード宣言を持つ場合があります。 UTF-8 エンコードを使用するシリアライゼーションに続き、サーバーからクライアントに戻される文字ストリングは、クライアントのコード・ページに変換されますが、そのコード・ページは UTF-8 とは異なる場合があります。

   したがって、アプリケーションは文字ストリング・タイプを戻す XMLSERIALIZE INCLUDING XMLDECLARATION の直接使用を避け、XML 値を取り出して直接ホスト変数に入れ、外部コード・ページと XML 宣言のエンコードとの間の一致を維持する必要があります。 XMLSERIALIZE をこの状況で使用する必要がある場合、BLOB タイプを指定してコード・ページの変換を避ける必要があります。
3. **代替構文**: XMLCLOB(*XML-expression*) を XMLSERIALIZE(*XML-expression* AS CLOB(2G)) の代わりに指定することができます。これは以前の DB2 のリリースとの互換性のためにのみサポートされています。

XMLTEXT



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLTEXT 関数は、入力引数を内容として持つ、単一の XQuery テキスト・ノードがある XML 値を戻します。

string-expression
値が文字ストリング・タイプ CHAR、VARCHAR、または CLOB を持つ式。

結果のデータ・タイプは XML です。 *string-expression* の結果が NULL になる可能性がある場合、結果も NULL になる可能性があります。入力値が NULL であれば、結果も NULL 値になります。 *string-expression* の結果が空ストリングであれば、結果の値は空テキスト・ノードです。

注:

1. **非 Unicode データベースおよび複数のデータベース・パーティションを持つデータベースでのサポート**: XMLTEXT はサポートされていません (SQLSTATE 42997)。

例:

- 単純な XMLTEXT 照会を作成します。

```
VALUES(
  XMLTEXT(
    'The stock symbol for Johnson&Johnson is JNJ.'
  )
)
```

この照会は、以下のシリアライズされた結果を生成します。

```
1
--------------------------------------------------
The stock symbol for Johnson&amp;Johnson is JNJ.
```

「&」記号は、テキスト・ノードがシリアライズされるときには「&」にマップされることに注意してください。

- XMLTEXT を XMLAGG と共に使用して、混合の内容を構成します。表 T の内容が以下のようであるとします。

```
seqno  plaintext                                                  emphtext
------ ---------------------------------------------------------- -------------
1      This query shows how to construct                          mixed content
2       using XMLAGG and XMLTEXT. Without                         XMLTEXT
3      XMLAGG will not have text nodes to group with other nodes, mixed content
       therefore, cannot generate
```

```
SELECT XMLELEMENT(
  NAME "para", XMLAGG(
    XMLCONCAT(
      XMLTEXT(
        PLAINTEXT
      ),
      XMLELEMENT(
        NAME "emphasis", EMPHTEXT
      )
```

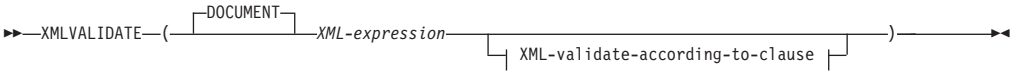
```
      )
      ORDER BY SEQNO
    ), '.'
  ) AS "result"
  FROM T
```

この照会は、次のような結果を生成します。

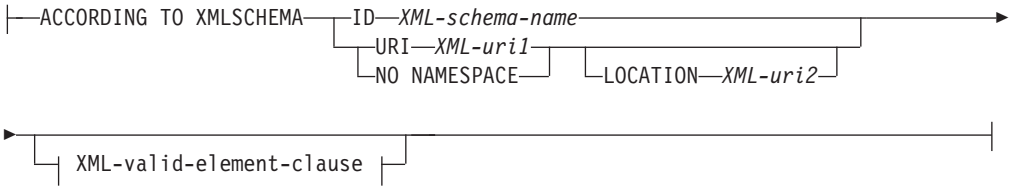
結果

```
------------------------------------------------------------------------------
<para>This query shows how to construct <emphasis>mixed content</emphasis>
using XMLAGG and XMLTEXT. Without <emphasis>XMLTEXT</emphasis> , XMLAGG
will not have text nodes to group with other nodes, therefore, cannot generate
<emphasis>mixed content</emphasis>.</para>
```

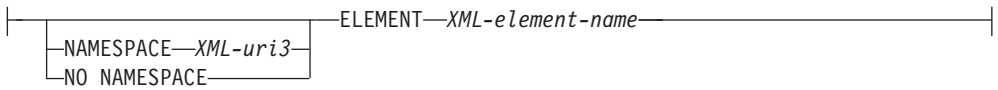
XMLVALIDATE



XML-validate-according-to-clause:



XML-valid-element-clause:



スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

XMLVALIDATE 関数は、デフォルトの値およびタイプのアノテーションを含む、XML スキーマ妥当性検査から入手した情報が加えられた入力 XML 値のコピーを戻します。

DOCUMENT
XML-expression の結果の XML 値が、XML バージョン 1.0 に準拠する整形式 XML 文書でなければならないことを指定します (SQLSTATE 2200M)。

XML-expression
データ・タイプ XML の値を戻す式。 *XML-expression* が、XML ホスト変数、または暗黙的あるいは明示的な型付きパラメーター・マーカーである場合、関数は、無視できる空白文字を除去する妥当性検査のための構文解析を実行し、CURRENT IMPLICIT XMLPARSE OPTION 設定は考慮されません。

XML-validate-according-to-clause
入力 XML 値の妥当性検査時に使用する情報を指定します。

ACCORDING TO XMLSCHEMA
妥当性検査用の XML スキーマ情報を明示的に指定することを示します。この節が組み込まれない場合、XML スキーマ情報は *XML-expression* 値の内容で提供される必要があります。

ID *XML-schema-name*
妥当性検査に使用される XML スキーマの SQL ID を指定します。この名前 (暗黙的または明示的 SQL スキーマ修飾子を含む) は、現行のサーバーで XML スキーマ・リポジトリー内の既存の XML スキーマを固有に識別しなければなりません。暗黙的または明示的に指定した SQL スキーマにこの名前の XML スキーマが存在しない場合は、エラー (SQLSTATE 42704) が戻されます。

URI *XML-uri1*
妥当性検査に使用される XML スキーマのターゲット・ネーム・スペース URI を指定します。 *XML-uri1* の値は、URI を空でない文字ストリング定数として指定します。 URI は、登録済み XML スキーマのターゲット・ネーム・スペースでなければならず (SQLSTATE 4274A)、LOCATION 節を指定しない場合は、登録済み XML スキーマを固有に識別する必要があります (SQLSTATE 4274B)。

NO NAMESPACE
妥当性検査用の XML スキーマはターゲット・ネーム・スペースを持たないことを指定します。ターゲット・ネーム・スペース URI は、明示的なターゲット・ネーム・スペース URI として指定できない空の文字ストリングと同等です。

LOCATION *XML-uri2*
妥当性検査に使用される XML スキーマの XML スキーマ・ロケーション URI を指定します。 *XML-uri2* の値は、URI を空でない文字ストリング定数として指定します。 XML スキーマ・ロケーション URI は、ターゲット・ネーム・スペース URI と結合されて登録済み XML スキーマを識別する必要があり (SQLSTATE 4274A)、登録済みのそのような XML スキーマは 1 つだけでなければなりません (SQLSTATE 4274B)。

XML-valid-element-clause
XML-expression 内の XML 値が、指定されたエレメント名を、XML 文書のルート・エレメントとして持つ必要があることを指定します。

NAMESPACE *XML-uri3* **or NO NAMESPACE**
妥当性検査されるエレメントのターゲット・ネーム・スペースを指定します。どちらの節も指定されない場合、指定されたエレメントは、妥当性検査に使用される登録済み XML スキーマのターゲット・ネーム・スペースと同じネーム・スペース内にあると想定されます。

NAMESPACE *XML-uri3*
妥当性検査されるエレメントのネーム・スペース URI を指定します。 *XML-uri3* の値は、URI を空でない文字ストリング定数として指定します。これは、妥当性検査に使用される登録済み XML スキーマが複数のネーム・スペースを持つ場合に使用することができます。

NO NAMESPACE
妥当性検査用のエレメントはターゲット・ネーム・スペースを持たないことを指定します。ターゲット・ネーム・スペース URI は、明示的なターゲット・ネーム・スペース URI として指定できない空の文字ストリングと同等です。

ELEMENT *xml-element-name*
妥当性検査に使用される XML スキーマ内のグローバル・エレメントの名前を指定します。暗黙のまたは明示的ネーム・スペースを持つ、指定されるエレメントは、*XML-expression* の値のルート・エレメントと一致しなければなりません (SQLSTATE 22535 または 22536)。

結果のデータ・タイプは XML です。 *XML-expression* の値が NULL になる可能性がある場合、結果も NULL になる可能性があります。*XML-expression* の値が NULL の場合、結果も NULL 値です。

XML 妥当性検査プロセスは、シリアライズされた XML 値に対して実行されます。 XMLVALIDATE は XML タイプの引数で呼び出されるので、この値は、妥当性検査プロセスに先だって自動的にシリアライズされます。ただし以下の 2 つの例外があります。

- XMLVALIDATE への引数が XML ホスト変数であるか、または暗黙的あるいは明示的な型付きパラメーター・マーカーである場合、妥当性検査のための構文解析操作が入力値に対して実行されます (暗黙の妥当性検査以外の構文解析は実行されず、CURRENT IMPLICIT XMLPARSE OPTION 設定は考慮されません)。
- XMLVALIDATE への引数が、オプション PRESERVE WHITESPACE を使用した XMLPARSE 呼び出しである場合、文書の XML 構文解析および XML 妥当性検査は組み合わせて、単一の妥当性検査のための構文解析操作にすることができます。

XML 値が以前に妥当性検査されている場合、以前の妥当性検査のアノテーションが付けられたタイプ情報は、シリアライゼーション・プロセスにより除去されます。ただし、以前の妥当性検査のデフォルト値およびエンティティー拡張は変更されません。妥当性検査が成功した場合、すべての無視できる空白文字は結果から除去されます。

ルート・エレメントにネーム・スペースがない文書を妥当性検査するには、xsi:noNamespaceSchemaLocation 属性がルート・エレメント上に存在していなければなりません。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**: XMLVALIDATE はサポートされません (SQLSTATE 42997)。
2. **XML スキーマの決定**: XML スキーマは、XMLVALIDATE 呼び出しの一部として ACCORDING TO XMLSCHEMA 節を使って明示的に指定するか、または入力 XML 値内の XML スキーマ・ロケーション情報から暗黙的に決定することができます。明示的または暗黙的な XML スキーマ情報は、XML スキーマ・リポジトリーに登録されている XML スキーマを識別する必要があり (SQLSTATE 42704、4274A、または 22532)、登録済みのそのような XML スキーマは 1 つだけでなければなりません (SQLSTATE 4274B または 22533)。

   ACCORDING TO XMLSCHEMA 節を使って妥当性検査用の XML スキーマが明示的に指定される場合、入力 XML 値で指定されるスキーマ・ロケーション情報は無視されます。

   ACCORDING TO XMLSCHEMA 節を使って XML スキーマ情報が指定されない場合、入力 XML 値に XML スキーマ・ロケーション情報が含まれていなければなりません (SQLSTATE 2200M)。入力 XML 値内のスキーマ・ロケーション情報、名前空間名、およびスキーマ・ロケーションは、妥当性検査に使用される XML スキーマ・リポジトリー内の XML スキーマ文書を指定します。

3. **XML スキーマ許可**: 妥当性検査に使用される XML スキーマは、使用前に XML スキーマ・リポジトリーに登録されなければなりません。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。
   - 妥当性検査時に使用される XML スキーマに対する USAGE 特権
   - SYSADM または DBADM 権限
4. **XML スキーマでの 5000 を超える maxOccurs 属性値の使用:** DB2 バージョン 9.5 フィックスパック 5 以降では、DB2 XSR に登録されている XML スキーマが 5000 を超える値を持つ maxOccurs 属性を使用する場合、その maxOccurs 属性値は「unbounded」と指定される場合と同じように扱われます。 5000 を超える maxOccurs 属性値を持つ文書エレメントは「unbounded」を指定したかのように処理されるため、XMLVALIDATE 関数を使用すると、文書の妥当性検査に使用した XML スキーマに基づく最大値をエレメントの出現回数が超える場合でも、XML 文書は妥当性検査をパスする可能性があります。

   5000 を超える maxOccurs 属性値を持つエレメントを定義する XML スキーマを使用しているときに、5000 を超える maxOccurs 属性値を持つ XML 文書をリジェクトする場合は、その条件を検査するトリガーまたはプロシージャーを定義することができます。そのトリガーまたはプロシージャーで、XPath 式を使用してエレメントの出現回数を数え、エレメントの数が maxOccurs 属性値を超える場合にエラーを返すようにします。

   例えば、以下のトリガーは、文書の電話 (phone) エレメントの出現回数が 6500 回を超えないようにします。

```
CREATE TRIGGER CUST_INSERT
  AFTER INSERT ON CUSTOMER
  REFERENCING NEW AS NEWROW
  FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
  SELECT CASE WHEN X <= 6500 THEN 'OK - Do Nothing'
              ELSE RAISE_ERROR('75000', 'TooManyPhones')  END
  FROM (
    SELECT XMLCAST(XMLQUERY('$INFO/customerinfo/count(phone)') AS INTEGER) AS X
    FROM CUSTOMER
    WHERE CUSTOMER.CID = NEWROW.CID );
END
```

例:

- XML インスタンス文書内で XML スキーマのヒントによって識別される XML スキーマを使用して妥当性検査します。

```
INSERT INTO T1(XMLCOL)
  VALUES (XMLVALIDATE(?))
```

入力パラメーター・マーカーは、XML スキーマ情報を含む XML 値にバインドされると想定します。

```
<po:order
        xmlns:po="http://my.world.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://my.world.com http://my.world.com/world.xsd" >
...
</po:order>
```

さらに、ターゲット・ネーム・スペース「http://my.world.com」と関連し、schemaLocation ヒント「http://my.world.com/world.xsd」による XML スキーマが、XML スキーマ・リポジトリー内にあると想定します。

これらの前提事項に基づき、入力 XML 値は妥当性検査され、その XML スキーマに応じてタイプのアノテーションが付けられます。

- SQL 名 PODOCS.WORLDPO. によって識別される XML スキーマを使用して妥当性検査を行います。

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA ID PODOCS.WORLDPO
    )
  )
```

SQL 名 FOO.WORLDPO に関連した XML スキーマが XML リポジトリー内にあると想定して、入力 XML 値は妥当性検査され、その XML スキーマに応じてタイプのアノテーションが付けられます。

- XML 値の指定されたエレメントを妥当性検査します。

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA ID FOO.WORLDPO
      NAMESPACE 'http://my.world.com/Mary'
      ELEMENT "po"
    )
  )
```

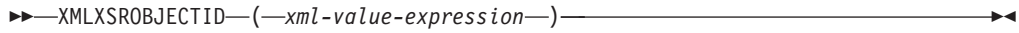
SQL 名 FOO.WORLDPO に関連した XML スキーマが XML リポジトリー内にあると想定して、XML スキーマは、ネーム・スペースが「http://my.world.com/Mary」であるエレメント「po」に対して妥当性検査されます。

- XML スキーマは、ターゲット・ネーム・スペースおよびスキーマ・ロケーションにより識別されます。

```
INSERT INTO T1(XMLCOL)
  VALUES (
    XMLVALIDATE(
      ? ACCORDING TO XMLSCHEMA URI 'http://my.world.com'
      LOCATION 'http://my.world.com/world.xsd'
    )
  )
```

ターゲット・ネーム・スペース「http://my.world.com」と関連し、schemaLocation ヒント「http://my.world.com/world.xsd」による XML スキーマが XML スキーマ・リポジトリー内にあると想定して、入力 XML 値は妥当性検査され、そのスキーマに応じてタイプのアノテーションが付けられます。

XMLXSROBJECTID



スキーマは SYSIBM です。

XMLXSROBJECTID 関数は、引数で指定された XML 文書の妥当性検査に使用された XML スキーマの XSR オブジェクト ID を戻します。 XSR オブジェクト ID は BIGINT 値として戻され、これを SYSCAT.XSROBJECTS の単一行のキーとして使用できます。

xml-value-expression
結果が XML データ・タイプの値になる式を指定します。結果の XML 値は、XML 文書または NULL 値の単一項目から成る XML シーケンスでなければなりません (SQLSTATE 42815)。引数が NULL の場合、この関数は NULL を戻します。 *xml-value-expression* が妥当性検査済み XML 文書を指定していない場合、この関数は 0 を戻します。

注:

1. **複数のデータベース・パーティション・データベースでのサポート**:
   XMLXSROBJECTID はサポートされません (SQLSTATE 42997)。
2. XML スキーマは、それを使用して妥当性検査が行われた XML 値に影響を与えることなくドロップできるため、この関数によって戻された XSR オブジェクト ID に対応する XML スキーマは、もはや存在しない場合があります。したがって、照会で XSR オブジェクト ID を使用してカタログ・ビューから XML スキーマの詳細情報をフェッチしようとした場合、空の結果セットが戻される場合があります。
3. アプリケーションは、XSR オブジェクト ID を使用して、XML スキーマに関する追加情報を取り出すことができます。例えば、XSR オブジェクト ID を使用することによって、登録済み XML スキーマを構成する個々の XML スキーマ文書を SYSCAT.SYSXSROBJECTCOMPONENTS から戻したり、XML スキーマにおける XML スキーマ文書の階層を SYSCAT.XSROBJECTHIERARCHIES から戻したりできます。

例:

- 表 MYTABLE に保管されている XML 文書 XMLDOC の XML スキーマ ID を取り出します。

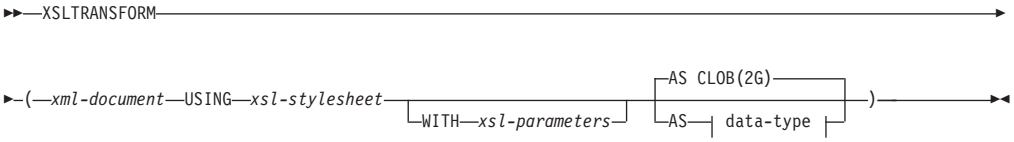
```
SELECT XMLXSROBJECTID(XMLDOC) FROM MYTABLE
```

- 表 MYTABLE 内の特定の ID (この場合は DOCKEY=1) を持つ XML 文書に関連付けられた XML スキーマ文書を、その XML スキーマを構成する XML スキーマ文書の階層を含めて取り出します。

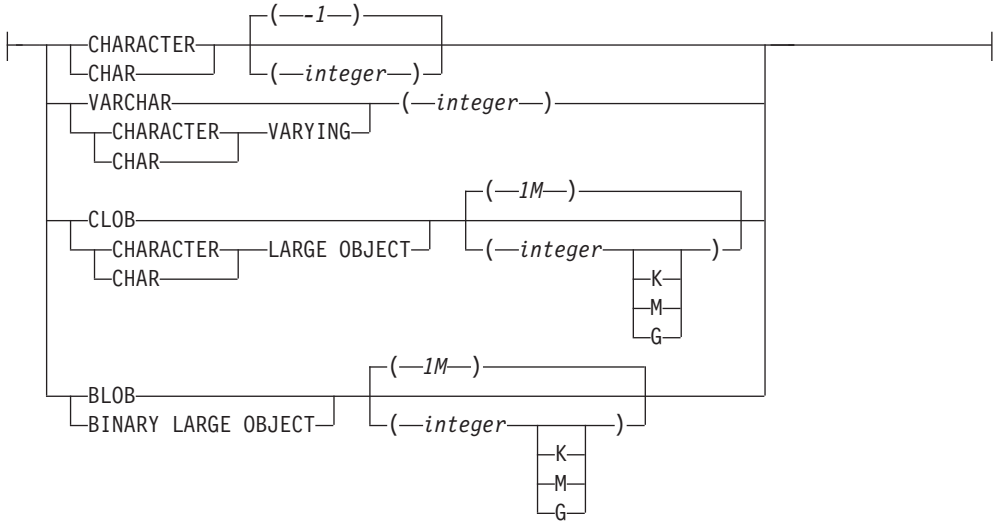
```
SELECT H.HTYPE, C.TARGETNAMESPACE, C.COMPONENT
FROM SYSCAT.XSROBJECTCOMPONENTS C, SYSCAT.XSROBJECTHIERARCHIES H
WHERE C.OBJECTID =
  (SELECT XMLXSROBJECTID(XMLDOC) FROM MYTABLE
   WHERE DOCKEY = 1)
AND C.OBJECTID = H.OBJECTID
```

XSLTRANSFORM

XSLTRANSFORM を使用して XML データを他の形式に変換します。これには、1 つの XML スキーマに準拠する XML 文書を別のスキーマに準拠する文書に変換することも含まれます。



data-type:



スキーマは SYSIBM です。この関数を修飾名で指定することはできません。

XSLTRANSFORM 関数は XML 文書を別のデータ形式に変換します。データは XSLT プロセッサーで可能なあらゆる形式、例えば、XML、HTML、またはプレーン・テキスト (ただし必ずしもこれらに限定されない) などに変換できます。

XSLTRANSFORM で使用されるパスはすべて、データベース・システム内部のパスです。現在のところ、このコマンドを外部ファイル・システムにあるファイルやスタイルシートで直接使用することができません。

xml-document
データ・タイプ XML、CHAR、VARCHAR、CLOB、または BLOB の整形式 XML 文書を戻す式。これは、*xsl-stylesheet* で指定された XSL スタイルシートを使用して変換される文書です。

注:

XML 文書は、少なくとも 1 つのルートを持つ整形式文書でなければなりません。

xsl-stylesheet
データ・タイプ XML、CHAR、VARCHAR、CLOB、または BLOB の整形式

XML 文書を戻す式。文書は W3C XSLT バージョン 1.0 勧告に準拠した XSL スタイルシートです。 XQUERY ステートメントまたは xsl:include 宣言を取り込むスタイルシートはサポートされていません。 このスタイルシートは、*xml-document* で指定された値を変換するために適用されます。

xsl-parameters
データ・タイプ XML、CHAR、VARCHAR、CLOB、または BLOB の整形式 XML 文書またはヌルを戻す式。これは、*xsl-stylesheet* で指定された XSL スタイルシートにパラメーター値を提供する文書です。パラメーターの値は、属性またはテキスト・ノードとして指定できます。

パラメーター文書の構文は次のとおりです。

```
<params xmlns="http://www.ibm.com/XSLTransformParameters">
<param name="..." value="..."/>
<param name="...">enter value here</param>
  ...
</params>
```

注:

スタイルシート文書には xsl:param エレメントが含まれている必要があり、パラメーター文書で指定されたものと一致する名前属性値がなければなりません。

AS *data-type*
結果のデータ・タイプを指定します。指定された結果のデータ・タイプの暗黙的または明示的な長さ属性には、変換された出力を収める十分な大きさがなければなりません (SQLSTATE 22001)。デフォルトの結果のデータ・タイプは CLOB(2G) です。

注:

xml-document 引数または *xsl-stylesheet* 引数のいずれかがヌルの場合、結果はヌルになります。

上記文書のいずれかを CHAR、VARCHAR、または CLOB 列に格納する際にコード・ページ変換が発生することがあります。その場合、結果として文字が失われることがあります。

例

この例では、XSLT をフォーマット・エンジンとして使用する方法を示します。セットアップのために、まず以下の 2 つの例の文書をデータベースに挿入します。

```
INSERT INTO XML_TAB VALUES
(1,
         '<?xml version="1.0"?>
<students xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "/home/steffen/xsd/xslt.xsd">
<student studentID="1" firstName="Steffen" lastName="Siegmund"
    age=â€23â€ university=â€Rostockâ€/>
</students>',
    '<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:param name="headline"/>
<xsl:param name="showUniversity"/>
<xsl:template match="students">
                <html>
```

```
                        <head/>
                           <body>
                                  <h1><xsl:value-of select="$headline"/></h1>
                                  <table border="1">
                                          <th>
                                              <tr>
                                    <td width="80">StudentID</td>
                                    <td width="200">First Name</td>
                                    <td width="200">Last Name</td>
                                    <td width="50">Age</td>
                                    <xsl:choose>
        <xsl:when test="$showUniversity ='true'">
                                                  <td width="200">University</td>
                                        </xsl:when>
        </xsl:choose>
                                 </tr>
                                    </th>
                                    <xsl:apply-templates/>
                                  </table>
                           </body>
                    </html>
                   </xsl:template>
                        <xsl:template match="student">
                        <tr>
                      <td><xsl:value-of select="@studentID"/></td>
                      <td><xsl:value-of select="@firstName"/></td>
                      <td><xsl:value-of select="@lastName"/></td>
                      <td><xsl:value-of select="@age"/></td>
                      <xsl:choose>
                          <xsl:when test="$showUniversity = 'true' ">
                            <td><xsl:value-of select="@university"/></td>
                          </xsl:when>
                      </xsl:choose>
                             </tr>
                        </xsl:template>
</xsl:stylesheet>'
);
```

次に、XSLTRANSFORM 関数を呼び出して XML データを HTML に変換し、表示します。

```
SELECT XSLTRANSFORM (XML_DOC USING XSL_DOC AS CLOB(1M)) FROM XML_TAB;
```

結果は次の文書になります。

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<h1></h1>
<table border="1">
<th>
<tr>
<td width="80">StudentID</td>
<td width="200">First Name</td>
<td width="200">Last Name</td>
<td width="50">Age</td>
</tr>
</th>
 <tr>
<td>1</td>
<td>Steffen</td><td>Siegmund</td>
<td>23</td>
```

```
</tr>
                </table>
</body>
</html>
```

この例では、出力は HTML で、各パラメーターによってどのような HTML が生成されるか、およびどのようなデータがパラメーターにもたらされるかのみが影響を受けます。このため、ここでは XSLT のエンド・ユーザーの出力用のフォーマット・エンジンとしての使用例を示しています。

使用上の注意:

XML 文書をトランスフォームする方法は多数あります。例えば、XSLTRANSFORM 関数、XQuery 更新式、および外部アプリケーション・サーバーによる XSLT 処理を使用する方法などです。 DB2 XML 列に保管される文書のトランスフォーメーションについては、多くの場合、XSLT よりも XQuery 更新式を使用した方が効率よく実行できます。XSLT では常に、トランスフォームされる XML 文書の構文解析が必要になるからです。 XSLT を使って XML 文書をトランスフォームする場合は、文書のトランスフォームをデータベースで行うか、アプリケーション・サーバーで行うかに関して、注意深く決定する必要があります。

YEAR

スキーマは SYSIBM です。

YEAR 関数は、指定された値の年の部分を戻します。

引数は、日付、タイム・スタンプ、日付期間、タイム・スタンプ期間であるか、または CLOB でも LONG VARCHAR でもない日付あるいはタイム・スタンプの有効な文字ストリング表記でなければなりません。 Unicode データベースでは、指定した引数が GRAPHIC ストリングであると、まず文字ストリングに変換されてから、関数が実行されます。

この関数の結果は長精度整数 (large integer) です。引数が NULL になる可能性がある場合、結果も NULL になる可能性があります。引数が NULL であれば、結果は NULL 値です。

その他の規則は、指定した引数のデータ・タイプに応じて以下のように異なります。

- 引数が日付、タイム・スタンプ、または日付やタイム・スタンプの有効なストリング表記の場合
  - 結果は、指定した値の年の部分 (1 から 9999 の整数) になります。
- 引数が日付期間またはタイム・スタンプ期間の場合
  - 結果は、指定した値の年の部分 (-9999 から 9999 の整数) になります。 ゼロ以外の結果の符号は、引数と同じになります。

例:

- PROJECT 表から、同一暦年内に開始 (PRSTDATE) および終了 (PRENDATE) が予定されているプロジェクトを全選択します。

```
SELECT * FROM PROJECT
  WHERE YEAR(PRSTDATE) = YEAR(PRENDATE)
```

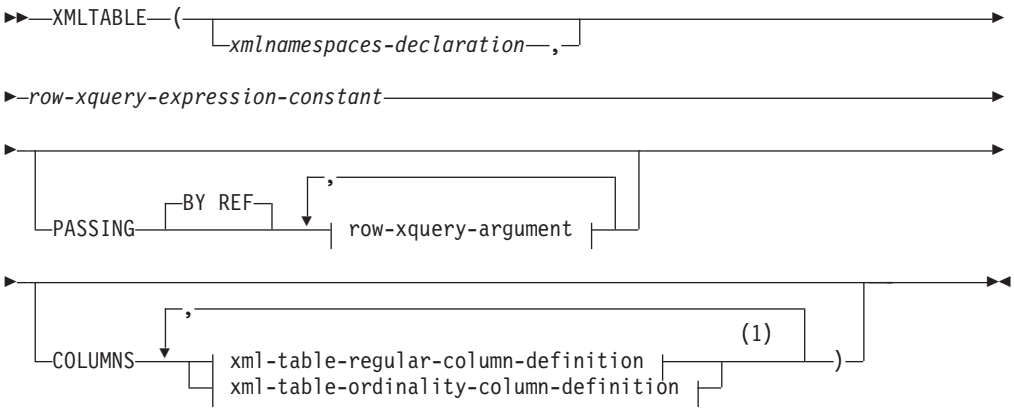
- PROJECT 表から、1 年未満での完了が予定されているプロジェクトを全選択します。

```
SELECT * FROM PROJECT
  WHERE YEAR(PRENDATE - PRSTDATE) < 1
```

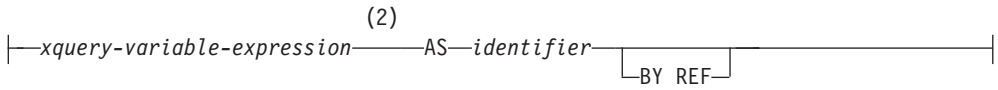
表関数

表関数は、ステートメントの FROM 節でしか使用できません。表関数は、表の列を戻します。これは、単純な CREATE TABLE ステートメントが作成する表に似ています。表関数はスキーマ名で修飾することができます。

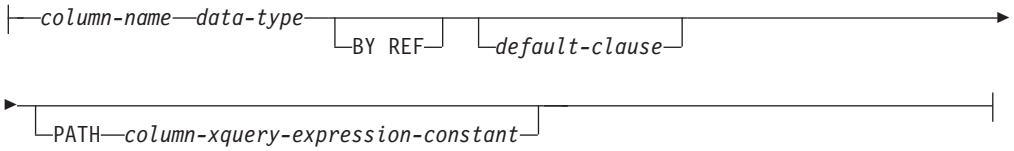
XMLTABLE



row-xquery-argument:



xml-table-regular-column-definition:



xml-table-ordinality-column-definition:

注:

1 xml-table-ordinality-column-definition 節を複数回指定することはできません (SQLSTATE 42614、SQLCODE -637)。

2 式のデータ・タイプを DECFLOAT にすることはできません。

スキーマは SYSIBM です。関数名を修飾名で指定することはできません。

場合によっては指定した入力引数を XQuery 変数として使用して、XMLTABLE 関数は結果表を XQuery 式の評価から戻します。行 XQuery 式の結果シーケンス内の各シーケンス項目は、結果表の行を表しています。

xmlnamespaces-declaration

row-xquery-expression-constant および *column-xquery-expression-constant* の静的コンテキストの一部になる、1 つ以上の XML ネーム・スペース宣言を指定します。 XMLTABLE の引数である XQuery 式の静的に認識されるネーム・スペースのセットは、事前設定された静的に認識されるネーム・スペースのセットと、

この節で指定されたネーム・スペース宣言を組み合わせたものです。 XQuery 式内の XQuery プロローグは、これらのネーム・スペースをオーバーライドする場合があります。

xmlnamespaces-declaration を指定しない場合、事前設定された静的に認識されるネーム・スペースのセットだけが XQuery 式に適用されます。

row-xquery-expression-constant
サポートされる XQuery 言語構文を使用して、XQuery 式として解釈される SQL 文字ストリング定数を指定します。定数ストリングは、データベース・コード・ページまたはセクション・コード・ページに変換されることなく、UTF-8 に直接変換されます。 XQuery 式は、オプション・セットの入力 XML 値を使用して実行し、シーケンス内の各項目についての行が生成される出力 XQuery シーケンスを戻します。 *row-xquery-expression-constant* の値は、空ストリングまたはすべてがブランクのストリングにすることはできません (SQLSTATE 10505)。

PASSING
入力値、およびそれらの値を *row-xquery-expression-constant* で指定された XQuery 式に渡す方法を指定します。デフォルトでは、関数が呼び出された有効範囲内にあるすべての固有の列名が、列の名前を変数名として使用して XQuery 式に暗黙的に渡されます。指定の row-xquery-argument 内の *identifier* が有効範囲内の列名と一致する場合、明示的な row-xquery-argument はその暗黙的な列をオーバーライドして XQuery 式に渡されます。

BY REF
デフォルトでは XML 入力引数を参照により渡すことを指定します。 XML 値を参照で渡す場合、XQuery の評価は、入力ノード・ツリーがあればそれを使用します。その場合は指定された入力式から直接、元のノードの ID および文書順序を含めすべてのプロパティーを保持したまま使用します。 2 つの引数が同じ XML 値を渡す場合、その 2 つの入力引数の間に含まれている何らかのノードに関係するノード ID 比較および文書順序比較は、同じ XML ノード・ツリー内のノードを参照する場合があります。

この節は、非 XML 値の受け渡しには影響を与えません。非 XML 値は、XML へのキャスト中に値の新規コピーを作成します。

row-xquery-argument
row-xquery-expression-constant により指定された XQuery 式に渡される引数を指定します。引数は、値およびその値が渡される方法を指定します。引数には、結果を XQuery 式に渡す前に評価される SQL 式が組み込まれます。

- 結果の値は、XML 型である場合、*input-xml-value* になります。 NULL の XML 値は、XML の空シーケンスに変換されます。
- 結果の値は、XML 型でない場合、XML データ・タイプにキャスト可能でなければなりません。 NULL 値は、XML の空シーケンスに変換されます。変換される値は、*input-xml-value* になります。

row-xquery-expression-constant が評価されるとき、XQuery 変数は *input-xml-value* と等しい値、および AS 節により指定された名前で示されます。

xquery-variable-expression
実行中に *row-xquery-expression-constant* により指定された XQuery 式が

使用できる値を持つ SQL 式を指定します。式には、NEXT VALUE 式、PREVIOUS VALUE 式 (SQLSTATE 428F9)、または OLAP 関数 (SQLSTATE 42903) を含めることはできません。式のデータ・タイプを DECFLOAT にすることはできません。

AS *identifier*
xquery-variable-expression により生成された値が、*row-xquery-expression-constant* に XQuery 変数として渡されることを指定します。変数名は *identifier* になります。 XQuery 言語の変数名に先行する先頭のドル記号 ($) は、*identifier* には含められません。identifier は有効な XQuery 変数名でなければならず、XML NCName に制限されます。 identifier は、長さが 128 バイトを超えてはなりません。同じ PASSING 節内の 2 つの引数が同じ identifier を使用することはできません (SQLSTATE 42711)。

BY REF
XML 入力値が参照により渡されるように指示します。 XML 値を参照で渡す場合、XQuery の評価は、入力ノード・ツリーがあればそれを使用します。その場合は指定された入力式から直接、元のノードの ID および文書順序を含めすべてのプロパティーを保持したまま使用します。2 つの引数が同じ XML 値を渡す場合、その 2 つの入力引数の間に含まれている何らかのノードに関係するノード ID 比較および文書順序比較は、同じ XML ノード・ツリー内のノードを参照する場合があります。 BY REF が *xquery-expression-variable* に続いて指定されない場合、XML 引数は、PASSING キーワードに続く構文により提供されるデフォルトの受け渡しメカニズムによって渡されます。このオプションは、非 XML 値に指定することはできません (SQLSTATE 42636)。非 XML 値が渡される場合、値は XML に変換されます。このプロセスによりコピーが作成されます。

COLUMNS
結果表の出力列を指定します。. この節が指定されない場合、*row-xquery-expression-constant* 内の XQuery 式を評価して得られたシーケンス項目に基づく値が指定されて、データ・タイプ XML の単一の無名列が参照によって戻されます (PATH '.' を指定した場合と同じ結果になります)。結果列を参照するには、関数に続く *correlation-clause* に *column-name* が指定されている必要があります。

xml-table-regular-column-definition
結果表の出力列を指定します。これには列名、データ・タイプ、XML 受け渡しメカニズム、および行のシーケンス項目から値を抽出する XQuery 式が含まれます。

column-name
結果表の列の名前を指定します。名前を修飾したり、表の複数の列に対して同じ名前を使用することはできません (SQLSTATE 42711)。

data-type
列のデータ・タイプを指定します。使用可能な型の構文および説明については、CREATE TABLE を参照してください。 *data-type* は、XML データ・タイプから、指定された *data-type* へのサポートされる XMLCAST がある場合に、XMLTable で使用できます。

BY REF
XML 値を、データ・タイプ XML の列の参照により戻すことを指定します。デフォルトでは、XML 値は BY REF により戻されます。 XML 値を参照で戻す場合、XML 値は、入力ノード・ツリーがあればそれを組み込みます。その場合は結果の値から直接、元のノードの ID および文書順序を含めすべてのプロパティーを保持したまま組み込みます。このオプションは、非 XML 列に指定することはできません (SQLSTATE 42636)。非 XML 列が処理される場合、値は XML から変換されます。このプロセスによりコピーが作成されます。

default-clause
列のデフォルト値を指定します。 *default-clause* の構文および説明については、CREATE TABLE を参照してください。 XMLTABLE 結果列の場合、*column-xquery-expression-constant* に含まれる XQuery 式の処理が空のシーケンスを戻す場合は、デフォルトが適用されます。

PATH *column-xquery-expression-constant*
サポートされる XQuery 言語構文を使用して、XQuery 式として解釈される SQL 文字ストリング定数を指定します。定数ストリングは、データベース・コード・ページまたはセクション・コード・ページに変換されることなく、UTF-8 に直接変換されます。 *column-xquery-expression-constant* は XQuery 式を指定しますが、これは *row-xquery-expression-constant* 内の XQuery 式の評価の結果である項目に関連して列値を決定します。外部で提供されたコンテキスト項目として *row-xquery-expression-constant* の処理の結果による項目がある場合、*column-xquery-expression-constant* が評価され、出力シーケンスが戻されます。列値は、以下のようにこの出力シーケンスに基づいて決定されます。

- 出力シーケンスに含まれている項目がゼロの場合、*default-clause* は列の値を提供します。
- 空のシーケンスが戻され、*default-clause* が指定されていない場合、NULL 値が列に割り当てられます。
- 空でないシーケンスが戻される場合、値は列に指定された *data-type* に対する XMLCAST です。この XMLCAST の処理によりエラーが戻される場合があります。

column-xquery-expression-constant の値は、空ストリングまたはすべてがブランクのストリングにすることはできません (SQLSTATE 10505)。この節が指定されない場合、デフォルトの XQuery 式は単に *column-name* になります。

xml-table-ordinality-column-definition
結果表の順序を示す列を指定します。

column-name
結果表の列の名前を指定します。名前を修飾したり、表の複数の列に対して同じ名前を使用することはできません (SQLSTATE 42711)。

FOR ORDINALITY
column-name が結果表の順序を示す列になるように指定します。この列のデータ・タイプは BIGINT です。 結果表のこの列の値は、

row-xquery-expression-constant 内の XQuery 式を評価した結果シーケンスにおける行の項目の順序番号です。

いずれかの XQuery 式の評価の結果がエラーになる場合、XMLTABLE 関数は XQuery エラーを戻します (SQLSTATE クラス '10')。

注:

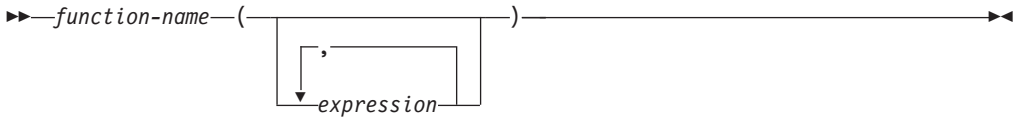
1. **複数のデータベース・パーティションを持つデータベースでのサポート**: XMLTABLE はサポートされません (SQLSTATE 42997)。

例:

- 以下は、注文の購入注文項目で状況が「NEW」の結果である表のリストです。

```
SELECT U."PO ID", U."Part #", U."Product Name",
     U."Quantity", U."Price", U."Order Date"
  FROM PURCHASEORDER P,
     XMLTABLE('$po/PurchaseOrder/item' PASSING P.PORDER AS "po"
              COLUMNS "PO ID"         INTEGER       PATH '../@PoNum',
                      "Part #"        CHAR(10)      PATH 'partid',
                      "Product Name"  VARCHAR(50)   PATH 'name',
                      "Quantity"      INTEGER       PATH 'quantity',
                      "Price"         DECIMAL(9,2)  PATH 'price',
                      "Order Date"    DATE          PATH '../@OrderDate'
             ) AS U
   WHERE P.STATUS = 'Unshipped'
```

ユーザー定義関数



ユーザー定義関数 (UDF) は、SQL 言語の既存の組み込み関数に対する拡張または追加です。ユーザー定義関数は、呼び出されるたびに単一値を戻すスカラー関数、互いに似通った一連の値を渡されてその中から単一値を戻す列関数、1 行を戻す行関数、または表を戻す表関数のいずれでかです。

SYSFUN および SYSPROC スキーマでは、多数のユーザー定義関数が提供されています。

UDF が既存の列関数をソースとして派生される場合にのみ、 UDF は列関数になります。 UDF は、修飾または無修飾の関数名およびその後に括弧で囲んだその関数の引数を指定することによって参照します。データベースに登録されたユーザー定義の列関数またはスカラー関数は、組み込み関数を使用できるのと同じコンテキストで参照することができます。ユーザー定義の行関数は、ユーザー定義タイプのトランスフォーム関数として登録しておく場合に限り、暗黙的に参照できます。データベースに登録されたユーザー定義の表関数は、 SELECT ステートメントの FROM 節でのみ参照することができます。

関数の引数の数および位置は、データベースに登録された時点のユーザー定義関数に指定されたパラメーターと対応していなければなりません。さらに、引数は、対応する定義済みパラメーターのデータ・タイプにプロモート可能なデータ・タイプでなければなりません。

関数の結果は、RETURNS 節に指定されます。 RETURNS 節 (UDF の登録時に定義される) は、関数が表関数かどうかを決定します。関数登録時に RETURNS NULL ON NULL INPUT 節が指定されていると (あるいはデフォルトでそうなっていると)、引数のいずれかが NULL 値の場合には、結果は NULL 値になります。表関数の場合、これは、戻される表は行を備えていない (つまり、空の表) という意味に解釈されます。

以下に、ユーザー定義関数例をいくつか示します。

- ADDRESS というスカラー UDF は、スクリプト・フォーマットで保管されているレジュームからホーム・アドレスを抽出します。この ADDRESS 関数には、CLOB 引数を指定し、VARCHAR(4000) の値が戻されます。

  ```
  SELECT EMPNO, ADDRESS(RESUME) FROM EMP_RESUME
    WHERE RESUME_FORMAT = 'SCRIPT'
  ```

- 表 T2 には、数値列 A があります。前の例の ADDRESS というスカラー UDF を以下のように呼び出すと、

  ```
  SELECT ADDRESS(A) FROM T2
  ```

  名前が一致して引数からプロモート可能なパラメーターを持つ関数がないので、エラー (SQLSTATE 42884) が生じます。

- WHO という表 UDF は、ステートメントの実行時にアクティブであった、サーバー・マシン上のセッションに関する情報を戻します。 WHO 関数は、キーワード TABLE および必須相関変数からなる FROM 節内で呼び出されます。 WHO() 表の列名は、CREATE FUNCTION ステートメントで定義されます。

  ```
  SELECT ID, START_DATE, ORIG_MACHINE
    FROM TABLE( WHO() ) AS QQ
    WHERE START_DATE LIKE 'MAY%'
  ```

第 4 章 プロシージャー

プロシージャーの概要

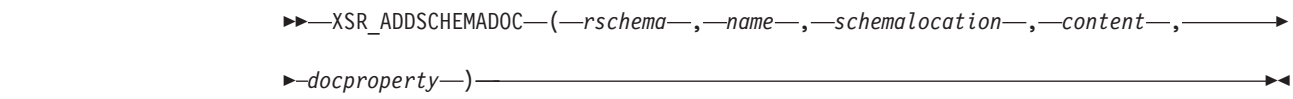
プロシージャーとは、 SQL CALL ステートメントを使って呼び出すことのできるアプリケーション・プログラムです。プロシージャーはプロシージャー名で指定され、プロシージャー名の後には括弧で囲まれた引数が付くこともあります。

プロシージャーの引数 (複数の場合あり) は個別のスカラー値で、タイプや意味が異なっていることもあります。引数を使って、プロシージャーに値を渡したり、プロシージャーから戻り値を受け取ったりします (またはその両方を行います)。

ユーザー定義プロシージャーとは、 CREATE PROCEDURE ステートメントを使って SYSCAT.ROUTINES のデータベースに登録されるプロシージャーです。このような関数のセットは、1 つは SYSFUN という名前のスキーマで、もう 1 つは SYSPROC というスキーマで、データベース・マネージャーに付属しています。

プロシージャーは、スキーマ名で修飾することができます。

XSR_ADDSCHEMADOC プロシージャー



スキーマは SYSPROC です。

XML スキーマ・リポジトリー (XSR) 内の各 XML スキーマは 1 つ以上の XML スキーマ文書で構成可能です。 XML スキーマが複数の文書で構成されている場合、XSR_ADDSCHEMADOC ストアード・プロシージャーを使用して、1 次 XML スキーマ文書以外のすべての XML スキーマを追加します。

許可

このプロシージャーの呼び出し元の許可 ID は、カタログ・ビュー SYSCAT.XSROBJECTS に記録されているような XSR オブジェクトの所有者でなければなりません。

rschema
XML スキーマのための SQL スキーマを指定する、タイプ VARCHAR (128) の入力引数。 SQL スキーマは XSR 内でこの XML スキーマの識別に使用される SQL ID の一部です。これは、完了状態に移されます。 (SQL ID のもう 1 つの部分は name 引数によって与えられます。) この引数には、NULL 値を入れることができます。このことは、CURRENT SCHEMA 特殊レジスターで定義されるように、デフォルトの SQL スキーマが使用されていることを示しています。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。 XSR オブジェクトは、XSR の外に存在するデータベ

© Copyright IBM Corp. 1993, 2010

ース・オブジェクトとの間で名前の衝突を起こすことはありません。XSR オブジェクトは、XML スキーマ・リポジトリー の外にあるオブジェクトとは違う名前空間で発生するからです。

name
XML スキーマの名前を指定する、タイプ VARCHAR (128) の入力引数。 XML スキーマの完全 SQL ID は、*rschema.name* です。 XML スキーマ名は XSR_REGISTER ストアード・プロシージャーの呼び出しの結果として既に存在していなければなりません。また、XML スキーマ登録はまだ完了することができません。この引数に NULL 値を入れることはできません。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

schemalocation
タイプ VARCHAR (1000) の入力引数 (NULL 値を入れることができる)。1 次 XML スキーマ文書を追加するこの XML スキーマ文書のスキーマ位置を示します。この引数は XML スキーマの外部名です。この 1 次文書は XML インスタンス文書内で xsi:schemaLocation 属性を指定して識別することができます。

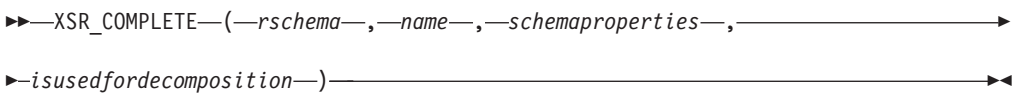
content
追加する XML スキーマ文書の内容を含むタイプ BLOB (30M) の入力パラメーター。この引数に NULL 値を入れることはできません。XML スキーマ文書を提供する必要があります。

docproperty
追加する XML スキーマ文書のプロパティーを示すタイプ BLOB (5M) の入力パラメーター。このパラメーターには NULL 値を入れることができます。そうでない場合、この値は XML 文書です。

以下に例を示します。

```
CALL SYSPROC.XSR_ADDSCHEMADOC(
  'user1',
  'POschema',
  'http://myPOschema/address.xsd',
  :content_host_var,
  0)
```

XSR_COMPLETE プロシージャー



スキーマは SYSPROC です。

XSR_COMPLETE プロシージャーは、XML スキーマ登録プロセスの一部として呼び出される最終ストアード・プロシージャーです。このプロシージャーは XML スキーマ・リポジトリー (XSR) で XML スキーマを登録します。 XML スキーマは、このストアード・プロシージャーの呼び出しを介してスキーマ登録が完了するまで妥当性検査には使用できません。

許可:

このプロシージャーの呼び出し元の許可 ID は、カタログ・ビュー SYSCAT.XSROBJECTS に記録されているような XSR オブジェクトの所有者でなければなりません。

rschema
XML スキーマのための SQL スキーマを指定する、タイプ VARCHAR (128) の入力引数。 SQL スキーマは XSR 内でこの XML スキーマの識別に使用される SQL ID の一部です。これは、完了状態に移されます。 (SQL ID のもう 1 つの部分は name 引数によって与えられます。) この引数には、NULL 値を入れることができます。このことは、CURRENT SCHEMA 特殊レジスターで定義されるように、デフォルトの SQL スキーマが使用されていることを示しています。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。 XSR オブジェクトは、XSR の外に存在するデータベース・オブジェクトとの間で名前の衝突を起こすことはありません。XSR オブジェクトは、XML スキーマ・リポジトリー の外にあるオブジェクトとは違う名前空間で発生するからです。

name
XML スキーマの名前を指定する、タイプ VARCHAR (128) の入力引数。 XML スキーマの完全 SQL ID は、*rschema.name* で、この ID に対して完了チェックが実行されます。 XML スキーマ名は XSR_REGISTER ストアード・プロシージャーの呼び出しの結果として既に存在していなければなりません。また、XML スキーマ登録はまだ完了することができません。この引数に NULL 値を入れることはできません。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

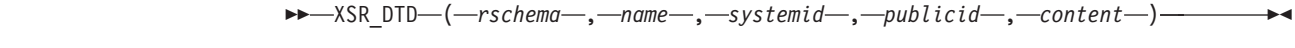
schemaproperties
XML スキーマに関連している場合、プロパティーを指定する、タイプ BLOB (5M) の入力引数。この引数の値は、NULL (関連プロパティーがない場合)、または XML 文書 (XML スキーマのプロパティーを表す) のいずれかです。

isusedfordecomposition
XML スキーマが分解に使用されるかどうかを示す integer タイプの入力パラメーター。 XML スキーマが分解に使用される場合、この値は 1 に設定してください。それ以外の場合はゼロに設定してください。

以下に例を示します。

```
CALL SYSPROC.XSR_COMPLETE(
  'user1',
  'POschema',
  :schemaproperty_host_var,
  0)
```

XSR_DTD プロシージャー



スキーマは SYSPROC です。

XSR_DTD プロシージャーは、文書タイプ宣言 (DTD) を XML スキーマ・リポジトリー (XSR) に登録します。

許可

このプロシージャーの呼び出し元の許可 ID には、少なくとも次のいずれかが必要です。

- SYSADM または DBADM 権限
- IMPLICIT_SCHEMA データベース権限 (SQL スキーマが存在しない場合)。
- CREATEIN 特権 (SQL スキーマが存在する場合)。

rschema
DTD のための SQL スキーマを指定するタイプ VARCHAR (128) の入出力引数。 SQL スキーマは XSR 内でこの DTD の識別に使用される SQL ID の一部です。 (SQL ID のもう 1 つの部分は *name* 引数によって与えられます。) この引数には、NULL 値を入れることができます。このことは、CURRENT SCHEMA 特殊レジスターで定義されるように、デフォルトの SQL スキーマが使用されていることを示しています。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。ストリング SYS で始まるリレーショナル・スキーマをこの値に使用しないでください。 XSR オブジェクトは、XSR の外に存在するデータベース・オブジェクトとの間で名前の衝突を起こすことはありません。XSR オブジェクトは、XML スキーマ・リポジトリー の外にあるオブジェクトとは違う名前空間で発生するからです。

name
DTD の名前を指定する、タイプ VARCHAR (128) の入力および出力引数。 DTD の完全 SQL ID は、*rschema.name* で、XSR にあるすべてのオブジェクト間で固有でなければなりません。この引数は NULL 値を受け入れます。この引数に NULL 値が提供される場合、固有な値が生成され、XSR 内に保存されます。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

systemid
DTD のシステム ID を指定する、タイプ VARCHAR (1000) の入力パラメーター。 DTD のシステム ID は、XML インスタンス文書の DOCTYPE 宣言または ENTITY 宣言 (使用されている場合は SYSTEM キーワードが接頭部になる) 中の DTD の URI と一致している必要があります。この引数に NULL 値を入れることはできません。システム ID と公開 ID を一緒に指定できます。

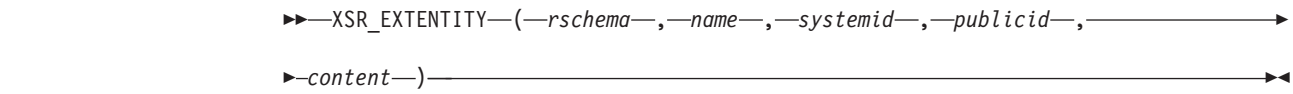
publicid
DTD の公開 ID を指定する、タイプ VARCHAR (1000) の入力パラメーター。 DTD の公開 ID は、XML インスタンス文書の DOCTYPE 宣言または ENTITY 宣言 (使用されている場合は PUBLIC キーワードが接頭部になる) 中の DTD の URI と一致している必要があります。この引数は、NULL 値を受け入れ、XML インスタンス文書の DOCTYPE 宣言または ENTITY 宣言中でも指定されている場合のみ使用する必要があります。

content
DTD 文書の内容を含むタイプ BLOB (30M) の入力パラメーター。この引数に NULL 値を入れることはできません。

例: システム ID *http://www.test.com/person.dtd* および公開 ID *http://www.test.com/person* によって識別される DTD を登録します。

```
CALL SYSPROC.XSR_DTD ( 'MYDEPT' ,
   'PERSONDTD' ,
   'http://www.test.com/person.dtd' ,
   'http://www.test.com/person',
   :content_host_variable
   )
```

XSR_EXTENTITY プロシージャー



スキーマは SYSPROC です。

XSR_EXTENTITY プロシージャーは、外部エンティティーを XML スキーマ・リポジトリー (XSR) に登録します。

許可

このプロシージャーの呼び出し元の許可 ID には、少なくとも次のいずれかが必要です。

- SYSADM または DBADM 権限
- IMPLICIT_SCHEMA データベース権限 (SQL スキーマが存在しない場合)。
- CREATEIN 特権 (SQL スキーマが存在する場合)。

rschema
外部エンティティーのための SQL スキーマを指定するタイプ VARCHAR (128) の入出力引数。 SQL スキーマは XSR 内でこの外部エンティティーの識別に使用される SQL ID の一部です。 (SQL ID のもう 1 つの部分は *name* 引数によって与えられます。) この引数には、NULL 値を入れることができます。このことは、CURRENT SCHEMA 特殊レジスターで定義されるように、デフォルトの SQL スキーマが使用されていることを示しています。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。ストリング SYS で始まるリレーショナル・スキーマをこの値に使用しないでください。XSR オブジェクトは、XSR の外に存在するデータベース・オブジェクトとの間で名前の衝突を起こすことはありません。XSR オブジェクトは、XML スキーマ・リポジトリー の外にあるオブジェクトとは違う名前空間で発生するからです。

name
外部エンティティーの名前を指定する、タイプ VARCHAR (128) の入出力引数。外部エンティティーの完全 SQL ID は、*rschema.name* で、XSR にあるすべてのオブジェクト間で固有でなければなりません。この引数は NULL 値を受け入れます。この引数に NULL 値が提供される場合、固有な値が生成され、XSR 内に保存されます。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

systemid
外部エンティティーのためのシステム ID を指定する、タイプ VARCHAR (1000) の入力パラメーター。外部エンティティーのシステム ID は、ENTITY 宣言 (使用されている場合は SYSTEM キーワードが接頭部になる) 中の外部エンティティーの URI と一致している必要があります。この引数に NULL 値を入れることはできません。システム ID と公開 ID を一緒に指定できます。

publicid
外部エンティティーのための公開 ID を指定する、タイプ VARCHAR (1000) の入力パラメーター。外部エンティティーの公開 ID は、ENTITY 宣言 (使用されている場合は PUBLIC キーワードが接頭部になる) 中の外部エンティティーの URI と一致している必要があります。この引数は、NULL 値を受け入れ、XML インスタンス文書の DOCTYPE 宣言または ENTITY 宣言中でも指定されている場合のみ使用する必要があります。

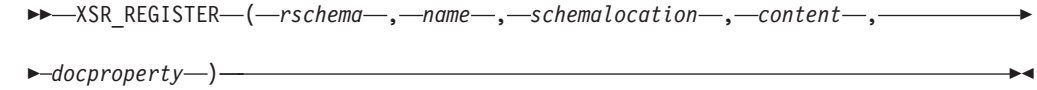
content
外部エンティティー文書の内容を含むタイプ BLOB (30M) の入力パラメーター。この引数に NULL 値を入れることはできません。

例: システム ID *http://www.test.com/food/chocolate.txt* および *http://www.test.com/food/cookie.txt* で識別される外部エンティティーを登録します。

```
CALL SYSPROC.XSR_EXTENTITY ( 'FOOD' ,
   'CHOCLATE' ,
   'http://www.test.com/food/chocolate.txt' ,
   NULL ,
   :content_of_chocolate.txt_as_a_host_variable
   )

CALL SYSPROC.XSR_EXTENTITY ( 'FOOD' ,
   'COOKIE' ,
   'http://www.test.com/food/cookie.txt' ,
   NULL ,
   :content_of_cookie.txt_as_a_host_variable
   )
```

XSR_REGISTER プロシージャー



スキーマは SYSPROC です。

XSR_REGISTER プロシージャーは、XML スキーマ登録プロセスの一部として呼び出される最初のストアード・プロシージャーです。このプロシージャーは XML スキーマ・リポジトリー (XSR) で XML スキーマを登録します。

許可

このプロシージャーの呼び出し元の許可 ID には、少なくとも次のいずれかが必要です。

- SYSADM または DBADM 権限

- IMPLICIT_SCHEMA データベース権限 (SQL スキーマが存在しない場合)。
- CREATEIN 特権 (SQL スキーマが存在する場合)。

rschema

XML スキーマのための SQL スキーマを指定するタイプ VARCHAR (128) の入出力引数。 SQL スキーマは XSR 内でこの XML スキーマの識別に使用される SQL ID の一部です。 (SQL ID のもう 1 つの部分は name 引数によって与えられます。) この引数には、NULL 値を入れることができます。このことは、CURRENT SCHEMA 特殊レジスターで定義されるように、デフォルトの SQL スキーマが使用されていることを示しています。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。ストリング SYS で始まるリレーショナル・スキーマをこの値に使用しないでください。XSR オブジェクトは、XSR の外に存在するデータベース・オブジェクトとの間で名前の衝突を起こすことはありません。XSR オブジェクトは、XML スキーマ・リポジトリー の外にあるオブジェクトとは違う名前空間で発生するからです。

name

XML スキーマの名前を指定する、タイプ VARCHAR (128) の入力および出力引数。 XML スキーマの完全 SQL ID は、*rschema.name* で、XSR にあるすべてのオブジェクト間で固有でなければなりません。この引数は NULL 値を受け入れます。この引数に NULL 値が提供される場合、固有な値が生成され、XSR 内に保存されます。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

schemalocation

タイプ VARCHAR (1000) の入力引数 (NULL 値を入れることができる)。1 次 XML スキーマ文書のスキーマ位置を示します。この引数は XML スキーマの外部名です。この 1 次文書は XML インスタンス文書内で xsi:schemaLocation 属性を指定して識別することができます。

content

1 次 XML スキーマ文書の内容を含むタイプ BLOB (30M) の入力パラメーター。この引数に NULL 値を入れることはできません。XML スキーマ文書を提供する必要があります。

docproperty

1 次 XML スキーマ文書のプロパティーを示すタイプ BLOB (5M) の入力パラメーター。このパラメーターには NULL 値を入れることができます。そうでない場合、この値は XML 文書です。

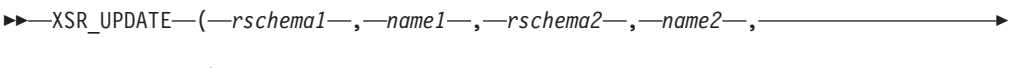
以下に例を示します。次の例は、コマンド行から XSR_REGISTER プロシージャーを呼び出す方法を示しています。

```
CALL SYSPROC.XSR_REGISTER(
  'user1',
  'POschema',
  'http://myPOschema/PO.xsd',
  :content_host_var,
  :docproperty_host_var)
```

例: 次の例は、Java アプリケーション・プログラムから XSR_REGISTER プロシージャーを呼び出す方法を示しています。

```
stmt = con.prepareCall("CALL SYSPROC.XSR_REGISTER (?, ?, ?, ?, ?)");
String xsrObjectName = "myschema1";
String xmlSchemaLocation = "po.xsd";
stmt.setNull(1, java.sql.Types.VARCHAR);
stmt.setString(2, xsrObjectName);
stmt.setString(3, xmlSchemaLocation);
stmt.setBinaryStream(4, buffer, (int)length);
stmt.setNull(5, java.sql.Types.BLOB);
stmt.registerOutParameter(1, java.sql.Types.VARCHAR);
stmt.registerOutParameter(2, java.sql.Types.VARCHAR);
stmt.execute();
```

XSR_UPDATE プロシージャー



スキーマは SYSPROC です。

XSR_UPDATE ストアード・プロシージャーは、XML スキーマ・リポジトリー (XSR) 内の既存の XML スキーマを発展させるために使用されます。これを使用すると、既存の XML 文書と新しく挿入された XML 文書の両方を妥当性検査できるように、既存の XML スキーマを変更または拡張できます。

XSR_UPDATE の引数として指定された元の XML スキーマと新しい XML スキーマは、プロシージャーが呼び出される前に XSR に登録され、かつ完成している必要があります。これらの XML スキーマは互換性がなければなりません。互換性要件の詳細については、『*XML スキーマを発展させるための互換性要件*』を参照してください。

許可

プロシージャーの呼び出し元の許可 ID が持つ特権には、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。

- SQL スキーマ *rschema1* およびオブジェクト名 *name1* によって指定された XML スキーマの OWNER
- SYSADM または DBADM 権限
- *rschema1* 引数で指定された SQL スキーマに対する ALTERIN 特権に加え、*dropnewschema* 引数がゼロに等しくない場合には、*rschema2* 引数で指定された SQL スキーマに対する DROPIN 特権。

rschema1
更新対象となる元の XML スキーマのための SQL スキーマを指定する、タイプ VARCHAR (128) の入力引数。 SQL スキーマは XSR 内でこの XML スキーマの識別に使用される SQL ID の一部です。 (SQL ID のもう 1 つの部分は *name1* 引数によって与えられます。) この引数に NULL 値を入れることはできません。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

name1
更新対象となる元の XML スキーマの名前を指定する、タイプ VARCHAR

(128) の入力引数。 XML スキーマの完全 SQL ID は、*rschema1.name1* です。この XML スキーマは、XSR に既に登録され、かつ完成している必要があります。この引数に NULL 値を入れることはできません。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

rschema2
元の XML スキーマを更新するために使用される新しい XML スキーマのための SQL スキーマを指定する、タイプ VARCHAR (128) の入力引数。 SQL スキーマは XSR 内でこの XML スキーマの識別に使用される SQL ID の一部です。 (SQL ID のもう 1 つの部分は *name2* 引数によって与えられます。) この引数に NULL 値を入れることはできません。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

name2
元の XML スキーマを更新するために使用される新しい XML スキーマの名前を指定する、タイプ VARCHAR (128) の入力引数。 XML スキーマの完全 SQL ID は、*rschema2.name2* です。この XML スキーマは、XSR に既に登録され、かつ完成している必要があります。この引数に NULL 値を入れることはできません。すべての SQL ID に適用される有効な文字と区切り文字の規則は、この引数にも適用されます。

dropnewschema
元の XML スキーマを更新するために新しい XML スキーマを使用した後にそのスキーマがドロップされるかどうかを示す integer タイプの入力パラメーター。このパラメーターをゼロ以外のいずれかの値に設定した場合、新しい XML スキーマはドロップされます。この引数に NULL 値を入れることはできません。

例:

```
CALL SYSPROC.XSR_UPDATE(
  'STORE',
  'PROD',
  'STORE',
  'NEWPROD',
  1)
```

XML スキーマ STORE.PROD の内容は STORE.NEWPROD の内容で更新され、XML スキーマ STORE.NEWPROD はドロップされます。

第 5 章 SQL 照会

照会 は結果表を指定します。照会は、いくつかの特定の SQL ステートメントのコンポーネントです。照会には、次の 3 つの形式があります。

- 副選択
- 全選択
- 選択ステートメント

許可

照会で参照される表、ビュー、またはニックネームのそれぞれに対して、ステートメントの許可 ID は少なくとも以下の 1 つを持っている必要があります。

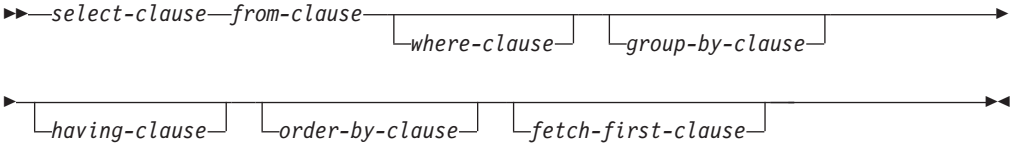
- SYSADM または DBADM 権限
- CONTROL 特権
- SELECT 特権

PUBLIC を除き、静的 SQL ステートメントに入っている照会のグループ特権はチェックされません。

ニックネームの場合、ニックネームが参照するオブジェクトでのデータ・ソースの許可要件は、照会が処理される時に適用されます。ステートメントの許可 ID は、データ・ソースに存在する別の許可 ID にマップすることができます。

© Copyright IBM Corp. 1993, 2010

副選択



副選択 は、全選択のコンポーネントの 1 つです。

副選択は、FROM 節で指定される表、ビュー、またはニックネームから派生する結果表を指定します。この派生の方法は、各操作の結果が次の演算の入力になるような、一連の操作として記述することができます。 (これは、副選択を記述する 1 つの方法にすぎません。派生操作を実行するために使用されるメソッドは、この記述とはまったく異なる場合があります。副選択の中に、正しい結果を得るのに厳密には実行する必要がないという部分があれば、その部分は実行されることもされないこともあります。)

副選択の節は以下の順序で処理されます。

1. FROM 節
2. WHERE 節
3. GROUP BY 節
4. HAVING 節
5. SELECT 節
6. ORDER BY 節
7. FETCH FIRST 節

ORDER BY または FETCH FIRST 節を備えた副選択は、以下では指定できません。

- ビューの最外部の全選択。
- マテリアライズ照会表。
- 副選択が括弧で囲まれていない場合。

例えば、以下は無効です (SQLSTATE 428FJ)。

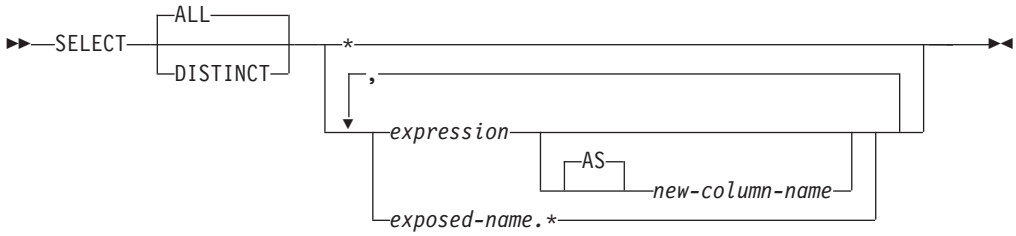
```
SELECT * FROM T1
   ORDER BY C1
UNION
SELECT * FROM T2
   ORDER BY C1
```

以下の例は有効です。

```
(SELECT * FROM T1
   ORDER BY C1)
UNION
(SELECT * FROM T2
   ORDER BY C1)
```

注: 副選択内の ORDER BY 節は、照会によって戻される行の順序には影響を与えません。 ORDER BY 節が影響を与えるのは、最外部の全選択で指定された場合に戻される行の ORDER BY だけです。

select-clause



SELECT 節は、最終結果表 R の列を指定します。列値は、*選択リスト* を R に適用することによって作成されます。選択リストとは、SELECT 節に指定された名前または式であり、 R は副選択のうち直前の操作の結果です。例えば、指定されている節が SELECT、FROM、および WHERE だけの場合、 R は WHERE 節の結果になります。

ALL
最終結果表の行すべてをそのまま保持し、冗長な重複を削除しません。これはデフォルトです。

DISTINCT
最終結果表の中に重複行があれば、その中の 1 つを除き、それ以外のすべてを削除します。 DISTINCT を使用した場合、結果表のストリング列を LONG VARCHAR、LONG VARGRAPHIC、LOB といったタイプにすることはできず、それらのタイプのいずれかを基にした特殊タイプ、または構造化タイプにすることもできません。 DISTINCT は、1 つの副選択で複数回使用することができます。これには、SELECT DISTINCT、選択リストまたは HAVING 節の列関数での DISTINCT の使用、および副選択の副照会などがあります。

2 つの行が互いに重複していると言えるのは、最初の行の各値が 2 番目の行の対応する値に等しい場合だけです。重複を判別する場合、双方が NULL 値であれば等しいものと見なされ、同じ数値であれば 10 進浮動小数点表記が異なっていても等しいものと見なされます。例えば、-0 は +0 と等しく、2.0 は 2.00 と等しいです。 10 進浮動小数点特殊値もそれぞれ等しいものと見なされます。-NAN は -NAN と等しく、-SNAN は -SNAN と等しく、-INFINITY は -INFINITY と等しく、INFINITY は INFINITY と等しく、SNAN は SNAN と等しく、NAN は NAN と等しいです。

列のデータ・タイプが 10 進浮動小数点で、同じ数値の複数の表記が列に存在する場合、SELECT DISTINCT に対して戻される特定の値は、列内の表記のうち任意のいずれかになります。詳しくは、123 ページの『数値比較』を参照してください。

他の SQL インプリメンテーションとの互換性のため、DISTINCT の同義語として UNIQUE を指定できます。

選択リストの表記法

* 表 R の列を識別する名前のリストを表します。ただし、IMPLICITLY HIDDEN として定義されている列は除外されます。リスト内の最初の名前は R の最初の列、2 番目の名前は R の 2 番目の列、というようになります。

名前のリストは、その SELECT 節の入ったプログラムのバインド時に確立されます。したがって、* (アスタリスク) では表参照の入ったステートメントのバインド後に表に追加された列は識別されません。

expression
結果列の値を指定します。有効な SQL 言語エレメントである任意の式にできますが、普通は列名を入れます。選択リストで使用される各列名は、 R の列をあいまいなところなく識別するものでなければなりません。

new-column-name **または AS** *new-column-name*
列名の名前を指定または変更します。この名前は修飾してはならず、ユニークである必要もありません。列名の後の使用方法は、次のように限定されています。

- AS 節に指定された新しい列名 (new-column-name) は、その名前がユニークであれば、ORDER BY 節で使用することができます。
- 選択リストの AS 節に指定された新しい列名を、副選択の他の節 (WHERE 節、GROUP BY 節、または HAVING 節) で使用することはできません。
- AS 節に指定された新しい列名を、UPDATE 節で使用することはできません。
- AS 節に指定された新しい列名は、ネストした表式、共通表式、および CREATE VIEW の全選択の外部で認識されます。

*name.**
結果表の列を指定する名前のリストを表します。この名前は *exposed-name* によって示されます。ただし、IMPLICITLY HIDDEN として定義されている列は除外されます。 *exposed-name* は、表名、ビュー名、ニックネーム、または相関名のいずれかにすることができ、FROM 節で指定された表、ビュー、またはニックネームを指定するものでなければなりません。リスト内の最初の名前は表、ビュー、あるいはニックネームの最初の列、 2 番目の名前は表、ビュー、またはニックネームの 2 番目の列を識別する、というようになります。

名前のリストは、その SELECT 節の入ったステートメントのバインド時に確立されます。したがって、ステートメントのバインド後に表に追加された列は、 * によって識別されません。

SELECT の結果の列の数は、命令形式の選択リスト (つまり、ステートメントの準備時に設定されたリスト) の式の数と同じであり、 500 (4K ページ・サイズの場合) または 1012 (8K、16K、 32K ページ・サイズの場合) を超えることはできません。

ストリング列に関する制限

選択リストの制限については、『可変長文字ストリング使用上の制限』 を参照してください。

選択リストの適用

選択リストを R に適用した結果は、 GROUP BY または HAVING が使用されているかどうかによって異なる場合があります。その結果について、次の 2 つのリストで説明します。

GROUP BY または HAVING が使用されている場合

- 選択リストで使用される式 *X* (列関数ではない) には、以下を指定した GROUP BY 節が必要です。
  - 各式または列名が列 R を明確に識別する *grouping-expression* (639 ページの『group-by-clause』を参照) または
  - 個別の*グループ化式* として *X* で参照される R の各列
- 選択リストは R のそれぞれのグループに対して適用され、その結果には、R にあるグループと同数の行が入ります。選択リストが R の 1 つのグループに適用されるとき、そのグループは、選択リストの列関数の引数のソースになります。

GROUP BY または HAVING のどちらも使用されていない場合

- 選択リストに列関数が入っていないか、または選択リスト内の各 *column-name* が列関数の中に指定されているか、あるいは相関列参照であるかのいずれかでなければなりません。
- 選択リストが列関数を備えていない場合、選択リストは R のそれぞれの行に対して適用され、その結果には R にある行と同数の行が示されます。
- 選択リストが列関数のリストである場合、関数の引数は R から与えられ、選択リストを適用した結果は 1 行となります。

どちらの場合も、結果の *n* 番目の列には、命令形式の選択リストにある *n* 番目の式を適用することによって指定された値が入ります。

結果列の NULL 属性

結果列は、以下から得られた場合には、NULL 値を使用できません。

- NULL 値が許されない列
- 定数
- COUNT または COUNT_BIG 関数
- 標識変数を持たないホスト変数
- NULL を使用できるオペランドが含まれていないスカラー関数または式

結果列が以下から得られた場合は、NULL 値を使用できます。

- COUNT または COUNT_BIG 以外の集約関数
- NULL 値が可能な列
- NULL が可能なオペランドが組み込まれたスカラー関数または式
- 等しい値を引数とする NULLIF 関数
- 標識変数、 SQL パラメーター、SQL 変数、またはグローバル変数を持つホスト変数
- 選択リスト内の対応項目の少なくとも 1 つが NULL 可能な場合のセット演算の結果
- 演算式から得られた演算式またはビューの列で、そのデータベースが DFT_SQLMATHWARN を Yes に設定して構成されているもの
- スカラー副選択
- 間接参照操作
- A GROUPING SETS *grouping-expression*

結果列の名前

- AS 節が指定されている場合、結果列の名前は、AS 節に指定された名前になります。
- AS 節の指定がなく、相関節に列リストが指定されている場合、結果列の名前は、相関列リスト内の対応する名前になります。
- AS 節も列リストも相関節に指定されておらず、結果列が単一の列からのみ派生している (関数も演算子も関係していない) 場合、結果列名はその列の非修飾名になります。
- AS 節も列リストも相関節に指定されておらず、結果列が単一の SQL 変数または SQL パラメーターからのみ派生している (関数も演算子も関係していない) 場合、結果列名はその SQL 変数または SQL パラメーターの非修飾名になります。
- AS 節も列リストも相関節に指定されておらず、結果列が間接参照操作を使用して派生している場合、結果列名は間接参照操作のターゲット列の非修飾名になります。
- それ以外の結果列には、名前が付けられません。システムは、これらの列に対して一時的な数字を (文字ストリングとして) 割り当てます。

結果列のデータ・タイプ

SELECT の結果の各列は、その列の派生元の式のデータ・タイプとなります。

式	結果列のデータ・タイプ
数値列の名前	列のデータ・タイプと同じ。DECIMAL 列の場合は精度と位取りも同じ。 DECFLOAT 列の場合は精度が同じ。
整数定数	INTEGER。
10 進定数	定数の精度および位取りを持つ DECIMAL。
浮動小数点定数	DOUBLE。
10 進浮動小数点定数	DECFLOAT(34)
任意の数値変数の名前	変数のデータ・タイプと同じで、DECIMAL 変数については精度と位取りも同じ。DECFLOAT 変数については精度が同じ。
n バイトを表す 16 進定数	VARCHAR(n); コード・ページはデータベース・コード・ページになります。
ストリング列の名前	列のデータ・タイプと同じで、長さ属性も同じ。
ストリング変数の名前	変数のデータ・タイプと同じ (同じ長さ属性); 変数のデータ・タイプが SQL データ・タイプと同じではない場合 (例えば、 C において NUL 文字で終了するストリング)、結果列は可変長ストリングになります。
長さ *n* の文字ストリング定数	VARCHAR(*n*)。
長さ *n* の GRAPHIC ストリング定数	VARGRAPHIC(*n*)。
日付/時刻の列の名前	列のデータ・タイプと同じ。
ユーザー定義タイプ列の名前	列のデータ・タイプと同じ。

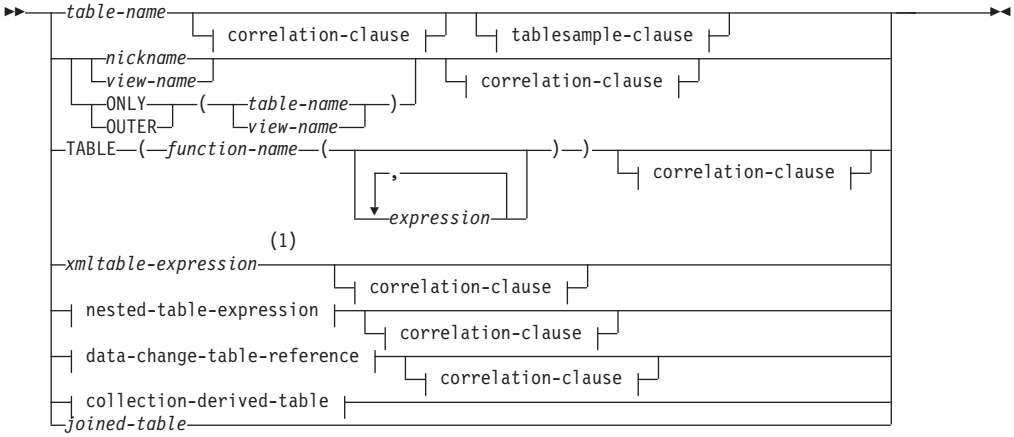
式	結果列のデータ・タイプ
参照タイプ列の名前	列のデータ・タイプと同じ。

from-clause

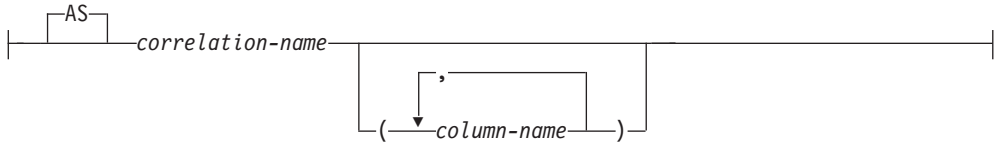
FROM 節は、中間結果表を指定します。

table-reference が 1 つだけ指定されている場合、中間結果表は単に、その *table-reference* の結果です。複数の *table-reference* が指定されている場合、中間結果表は、指定された *table-reference* の行の可能なすべての組み合わせ (デカルト積) からなります。結果の各行は、最初の *table-reference* の行を 2 番目の *table-reference* の行に連結し、それを 3 番目の *table-reference* の行に連結し、以下同様の手順で連結した行です。結果の行数は、すべての表参照の行数の積です。*表参照* については、『table-reference』 を参照してください。

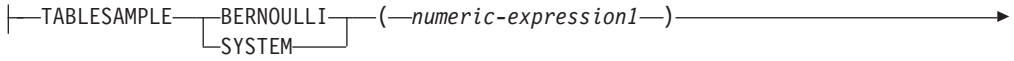
table-reference

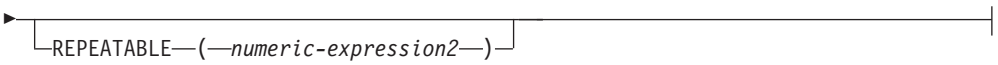


correlation-clause:

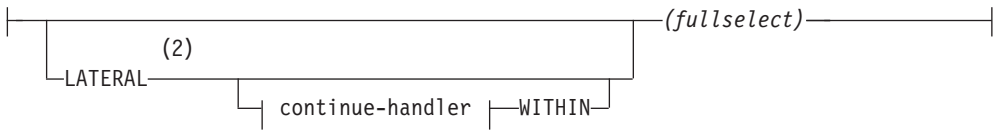


tablesample-clause:

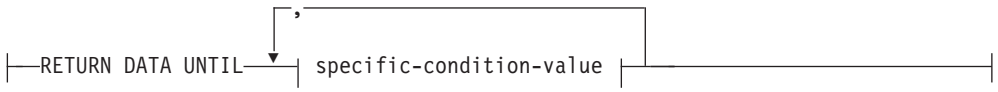




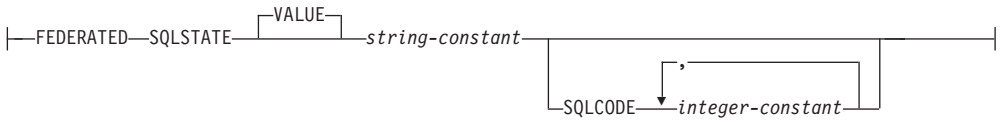
nested-table-expression:



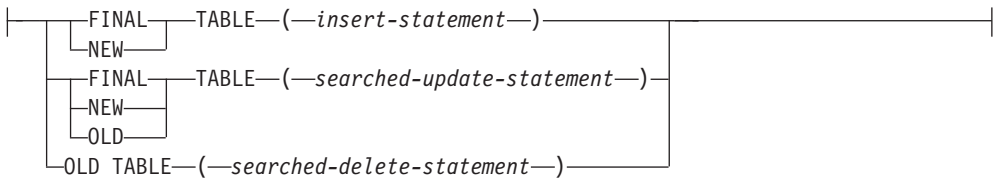
continue-handler:



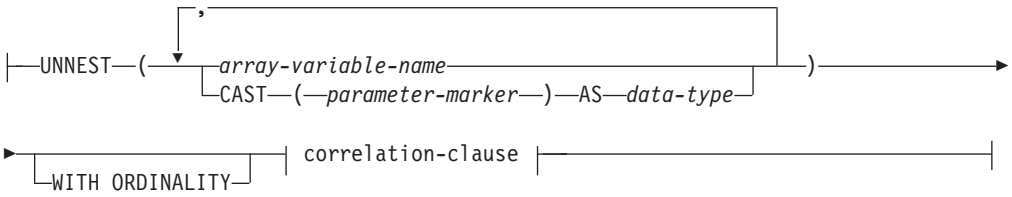
specific-condition-value:



data-change-table-reference:



collection-derived-table:



注:

1 XMLTABLE 式は、表参照の一部とすることができます。その場合、XMLTABLE 式内の副次式は、FROM 節内の前の範囲変数のスコープ内に入ります。詳しくは、『XMLTABLE』 についての記述を参照してください。

2 LATERAL の代わりに TABLE を指定できます。

表参照として指定する各 *table-name* (表名)、*view-name* (ビュー名)、または *nickname* (ニックネーム) は、アプリケーション・サーバーの既存の表、ビュー、またはニックネーム、あるいは表参照の入った全選択の前に定義された共通表式の *table-name* を指定するものでなければなりません。 *table-name* (表名) が型付き表を参照する場合、その名前はその表およびその表の副表 (*table-name* の列のみ) の UNION ALL (全合併) を表します。同様に、*view-name* (ビュー名) が型付きビューを参照する場合、その名前はそのビューおよびそのビューのサブビュー (*view-name* の列のみ) の UNION ALL を表します。

ONLY(*table-name*) または ONLY(*view-name*) を使用した場合は、適正な副表またはサブビューの行は組み込まれません。 ONLY に指定した *table-name* に副表がない場合、 ONLY(*table-name*) は *table-name* を指定することと同じになります。 ONLY に指定した *view-name* にサブビューがない場合、 ONLY(*view-name*) は *view-name* を指定することと同じになります。

OUTER(*table-name*) または OUTER(*view-name*) を指定した場合、それは仮想表を表します。 OUTER に指定した *table-name* または *view-name* に副表またはサブビューがない場合は、 OUTER を指定してもしなくても同じです。 OUTER(*table-name*) は、次のように *table-name* から派生します。

- 列には、*table-name* の列に続いて、副表によって導入された追加列 (副表がある場合) が組み込まれます。副表階層を階層深度の大きい順にスキャンし、追加列は右側に追加されます。共通の親を持つ副表の場合は、タイプの作成順にスキャンします。
- 行には、*table-name* のすべての行、およびその表の副表のすべての行が組み込まれます。その行の副表にない列には、NULL が戻されます。

上記の点は OUTER(*view-name*) にも当てはまります。その場合、*table-name* を *view-name* に、副表をサブビューに読み替えてください。

ONLY または OUTER を使用するときには、 *table-name* の副表または *view-name* のサブビューごとに、 SELECT 権限が必要です。

表参照として指定された各 *function-name* (関数名) およびその引数のタイプは、アプリケーション・サーバーの既存の表関数に解決されなければなりません。

括弧内の全選択 (fullselect) は、*ネストされた表式* と呼ばれます。

joined-table (結合表) は、 1 つまたは複数の結合演算の結果である中間結果セットを指定します。詳しくは、 637 ページの『joined-table』 を参照してください。

すべての表参照の直接的な名前はユニークでなければなりません。直接的な名前とは、以下の名前です。

- *correlation-name* (相関名)
- *後に相関名の付いていない表名*
- *後に相関名の付いていないビュー名*
- *後に相関名の付いていないニックネーム*
- *後に相関名の付いていない別名*

correlation-clause (相関節) が、*関数名*参照、*xmltable-expression* (XML 表式)、ネストされた表の式、または *data-change-table-reference* (データ変更表参照) の後に付かない場合、その表参照には直接的な名前はありません。

各*相関名*は、直前の*表名*、ビュー名、ニックネーム、*関数名*の参照、*XML 表式*、ネストされた表式、またはデータ*変更表参照*の指定子と定義されます。列に対する修飾参照では、直接的な名前を使用しなければなりません。同じ表名、ビュー名、またはニックネームを 2 回指定する場合は、その少なくとも 1 回の指定の後には*相関名* を付ける必要があります。*相関名* は、表、ビュー、またはニックネームの列に対する参照を修飾するのに使用されます。*相関名*が指定されている場合、表参照の列に名前を指定するために、*列名*を指定することもできます。*相関節*に*列名*が含まれない場合、直接的な列名は以下のようにして決定されます。

- *table-reference* (表参照) が*表名*、ビュー名、ニックネーム、または*別名*である場合は、参照されている表、ビュー、またはニックネームの列名
- *表参照*が*関数名*参照である場合は、CREATE FUNCTION ステートメントの RETURNS 節で指定されている列名
- *表参照*が *XML 表式*である場合は、*XML 表式*の COLUMNS 節で指定されている列名
- *表参照*がネストされた*表の式*である場合は、全選択により公開される列名
- *表参照*がデータ*変更表参照*である場合は、任意の定義済み INCLUDE 列に加えて、データ変更ステートメントのターゲット表にある列名

通常、コレクション型導出表、表関数、およびネストされた表の式は、FROM 節にのみ指定することができます。表関数、ネストされた表の式、またはコレクション型導出表にある列は、選択リストの中および残りの副選択で、相関名を使用して参照することができます。この相関名の有効範囲は、FROM 節の他の表名、ビュー名、またはニックネームの相関名と同じです。ネストされた表式は、次の場合に使用することができます。

- ビューの代わりに使用して、ビューが作成されないようにする場合 (ビューを一般的に使用する必要がない場合)
- 必要な結果表がホスト変数に基づいたものである場合

コレクション型導出表を使用すると、配列のエレメントを行にネスト解除できます。複数の配列が指定されている場合、最初の配列は結果表に最初の列を提供し、2 番目の配列は 2 番目の列を提供します。以下同様です。 WITH ORDINALITY を指定すると、INTEGER タイプの追加の列が付加されます。この列には、配列のエレメントの部分が入ります。配列のカーディナリティーが同一でない場合、結果として生成される表のカーディナリティーは、最大のカーディナリティーを持つ配列と同じになります。行の副索引の値が対応する配列のカーディナリティーより大きい場合は常に、表の列値は NULL 値に設定されます。つまり、各配列が 2 つの列 (1 つは副索引用、もう 1 つはデータ用) を持つ表として表示される場合、UNNEST は配列の間で、副索引に対する結合述部として等価を使用して OUTER JOIN を実行します。

UNNEST は SQL プロシージャー内でのみ指定可能です (SQLSTATE 42887)。

全選択内にあるデータ変更ステートメントで参照されているか、またはターゲットとされているネストされた表の式の選択リストにある式は、以下が入っていない場合に限り、有効です。

- SQL データの読み取りまたは変更を行う関数
- 非決定性の関数
- 外部アクションを指定する関数
- OLAP 関数

ビューが、FROM 節のデータ変更ステートメントで直接参照されている場合や、このステートメント内のネストされた表の式のターゲットとして参照されている場合、ビューは、シンメトリック (WITH CHECK OPTION が指定されている) であるか、 WITH CHECK OPTION ビューの制限を満たしている必要があります。

FROM 節のデータ変更ステートメントのターゲットがネストされた表の式である場合、変更された行は再修飾されず、WHERE 節の述部は再評価されません。また、ORDER BY 操作や FETCH FIRST 操作は再実行されません。

オプションの tablesample-clause を使用すると、指定された *table-name* のすべての内容ではなく、その *table-name* の行のランダム・サブセット (サンプル) をこの照会のために取得できます。このサンプリングは、 *where-clause* で指定されたすべての述部に加えて行われます。オプションの REPEATABLE 節が指定されていない限り、照会を実行するたびに、通常、異なるサンプルが生成されます。ただし変性の場合、サンプル・サイズに比べて表が小さすぎるので、すべてのサンプルは同じ行を戻します。サンプル・サイズは括弧内の *numeric-expression1* で制御されます。これは、表の中で戻すべきおおよそのパーセント (P) を示します。サンプルを取得する方法は TABLESAMPLE キーワードの後に指定され、 BERNOULLI (ベルヌーイ) または SYSTEM のいずれかが可能です。どちらの方法を使用しても、実際のサンプルの正確な行数は照会を実行するたびに異なることがあります。しかし平均では、述部によって行数がさらに削減される前の段階で、表のおよそ P パーセントになります。

table-name はすでに保管されている表でなければなりません。マテリアライズ照会表 (MQT) の名前も指定できますが、 MQT の定義に関連している副選択または表式は指定できません。これは、データベース・マネージャーがその副選択に対応する MQT にアクセス・パスを指定するとは限らないためです。

セマンティクス的には、表のサンプリングは、述部の適用または結合の実行などの、他のすべての照会処理よりも前に実行されます。 1 つの照会の実行中にサンプリングされた同じ表に繰り返しアクセスする場合 (例えば、ネストされたループ結合や相関副照会の場合)、常に同じサンプルが戻されます。 1 つの照会で複数の表をサンプリングすることができます。

BERNOULLI (ベルヌーイ) サンプリングではそれぞれの行が個別に考慮されます。サンプル内の各行は、他の行とは関係なく、P/100 の確率 (P は *numeric-expression1* の値) で組み込まれて、1 - P/100 の確率で除外されます。したがって、*numeric-expression1* の値が 10 (つまり 10 % のサンプル) と評価された場合、各行は 0.1 の確率で組み込まれて、0.9 の確率で除外されます。

SYSTEM サンプリングの場合、サンプリングを実行する最も効率的な方法をデータベース・マネージャーに判断させます。ほとんどの場合、*table-name* に SYSTEM サンプリングを適用すると、 *table-name* の各ページがサンプルに組み込まれる確率は P/100、除外される確率は 1 - P/100 です。組み込まれた各ページ内のすべての行がサンプルの対象となります。 *table-name* に SYSTEM サンプリングを適用する方が、検索対象のデータ・ページがより少なくなるため、ほとんどの場合 BERNOULLI サンプリングよりもかなり速く実行されます。ただし、集約関数 (例えば SUM(SALES)) の見積精度が低くなる可能性が高くなります (特に、*table-name* の行が、その照会で参照されるいずれかの列にクラスター化されている場合)。場合によっては、 SYSTEM サンプリングを BERNOULLI サンプリングであるかのように実行した方がより効率的だとオプティマイザーが判断するかもしれません。例えば、*table-name* の述部が索引によって適用され、サンプリング率 P よりもかなり限定的になる場合です。

numeric-expression1 は、 *table-name* から得られるサンプルのサイズを指定します (パーセントとして表されます)。これは定数式でなければならず、列、パラメーター・マーカー、またはホスト変数を中で使用することはできません。式は 100 以下の正数に評価される必要があります (1 と 0 の間の値でも差し支えありません)。例えば、値 0.01 は 1 % の 100 分の 1 を表し、平均して 10000 行につき 1 行がサンプルに組み入れられることを意味します。 *numeric-expression1* が値 100 に評価される場合、 tablesample-clause が指定されていないかのように扱われます。*numeric-expression1* が NULL 値に評価されたり、 100 を超えたり 0 を下回ったりする値に評価される場合には、エラーが戻されます (SQLSTATE 2202H)。

照会の実行ごとにサンプリングを繰り返すのが望ましい場合があります。例えば、レグレッション・テスト中や照会の「デバッグ」中などが該当します。これを行うには、REPEATABLE 節を指定します。 REPEATABLE 節では、括弧の中に *numeric-expression2* を指定する必要があります。この数式は、乱数発生ルーチンのシードと同じ役割を果たします。 *table-name* の tablesample-clause に REPEATABLE 節を追加すれば、 (*numeric-expression2* に同じ値を使用して) その照会を繰り返し実行した場合、同じサンプルが戻されるようになります (もちろん、データそのものが更新、再編成、または再パーティションされないことが前提です)。複数の照会の間で *table-name* の同じサンプルが使われるようにするには、グローバル一時表の使用をお勧めします。他の方法として、複数の照会を 1 つの照会に結合することもできます。その場合、WITH 節を使って定義された 1 つのサンプルへの複数の参照があります。

以下は、いくつかの例です。

例 1: 監査のために、 Sales 表から 10 % のベルヌーイ・サンプルを抽出します。

```
SELECT * FROM Sales
  TABLESAMPLE BERNOULLI(10)
```

例 2: 北東部 (Northeast) 地域での、それぞれの商品カテゴリーごとの売上総額 (Sales.Revenue) を計算します。その際、Sales 表のランダム 1 % SYSTEM サンプリングを使用します。セマンティクス的に、SUM はサンプルそのものに対する処理です。したがって、Sales 表全体の売上を推定するには、照会においてその SUM をサンプリング率 (0.01) で除算する必要があります。

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

例 3: REPEATABLE 節を使って、上記の照会を変更し、照会が実行されるたびに同じ (ランダムな) 結果が得られるようにします。 (括弧で囲まれた定数は任意の値です。)

```
SELECT SUM(Sales.Revenue) / (0.01)
  FROM Sales TABLESAMPLE SYSTEM(1) REPEATABLE(3578231)
  WHERE Sales.RegionName = 'Northeast'
  GROUP BY Sales.ProductCategory
```

表関数参照

一般的に、表関数とその引数値は、表やビューとまったく同じ方法で SELECT の FROM 節で参照することができます。ただし、特殊な考慮事項が適用されます。

- 表関数の列名

  相関名 の後に代替列名を指定する場合を除いて、表関数の列名は、 CREATE FUNCTION ステートメントの RETURNS 節に指定された列名になります。これは、 CREATE TABLE ステートメントに定義されている表の列名に類似したものです。

- 表関数解決

  表関数参照に指定された引数は関数名と共に、*関数解決* と呼ばれるアルゴリズムによって、使用する正確な関数を判別するのに用いられます。これは、ステートメントで使用される他の関数 (例えば、スカラー関数) の場合に行われるのと同じです。

- 表関数の引数

  スカラー関数の引数の場合と同じように、通常、表関数の引数は有効な SQL 式です。次の例は正しい構文です。

  ```
  例 1:       SELECT c1
              FROM TABLE( tf1('Zachary') ) AS z
              WHERE c2 = 'FLORIDA';

  例 2:       SELECT c1
              FROM TABLE( tf2 (:hostvar1, CURRENT DATE) ) AS z;

  例 3:       SELECT c1
              FROM t
              WHERE c2 IN
                       (SELECT c3 FROM
                        TABLE( tf5(t.c4) ) AS z  -- 直前の FROM 節への
                       )                         -- 相関参照
  ```

- SQL データを変更する表関数

  MODIFIES SQL DATA で指定された表関数は、SET ステートメントの副選択、SELECT INTO、または *row-fullselect* である *select-statement*、*common-table-expression*、または RETURN ステートメントで、最後の表参照としてのみ使用できます。 1 つの FROM 節中で使用できる表関数は 1 つだけであ

り、表関数の引数は、副選択内の他のすべての表参照と相関していなければなりません (SQLSTATE 429BL)。以下の例は、MODIFIES SQL DATA プロパティーをもった表関数の有効な構文です。

```
例 1:       SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

例 2:       SELECT c1
            FROM t1, t2, TABLE( tfmod(t1.c1, t2.c1) ) AS z

例 3:       SET var =
            (SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

例 4:       RETURN SELECT c1
            FROM TABLE( tfmod('Jones') ) AS z

例 5:       WITH v1(c1) AS
            (SELECT c1
            FROM TABLE( tfmod(:hostvar1) ) AS z)
            SELECT c1
            FROM v1, t1 WHERE v1.c1 = t1.c1
```

エラーを許容できるネストされた表の式

nested-table-expression で発生するある種のエラーは許容でき、エラーを戻す代わりに、照会は続行した上で結果を戻します。

RETURN DATA UNTIL 節を指定すると、指定の条件が満たされる前の全選択から戻された行によって、全選択の結果セットが作られることになります。これは、全選択の結果セットが完全なものでなくても (たとえ空の結果セットであっても)、*nested-table-expression* の結果として許容されるということです。

FEDERATED キーワードは、リモート・データ・ソースで発生したエラーのみに対処するよう条件を制限します。

条件は SQLSTATE 値として、長さ 5 の *string-constant* によって指定できます。オプションで、指定された個々の SQLSTATE 値に対応して、SQLCODE 値を指定することもできます。移植可能なアプリケーションの場合は、可能な限り SQLSTATE 値を指定するようにします。SQLCODE 値は通常、プラットフォーム間での移植ができず、SQL 標準にも含まれていないからです。

許容されるのは一部の条件だけです。照会の他の部分の実行ができなくなるようなエラーは許容できないので、そのようなエラーが発生した場合は、照会全体についてエラーが戻されます。 *specific-condition-value* では、データベース・マネージャーによって実際に許容されない条件を指定することもあります。特定の SQLSTATE または SQLCODE 値が指定されていても、その条件に当てはまればエラーが戻されます。

以下の SQLSTATE 値および SQLCODE 値を指定した場合は、データベース・マネージャーによって許容される可能性があります。

- SQLSTATE 08001; SQLCODE -1336、-30080、-30081、-30082
- SQLSTATE 08004
- SQLSTATE 42501
- SQLSTATE 42704; SQLCODE -204

- SQLSTATE 42720
- SQLSTATE 28000

エラーを許容できる *nested-table-expression* が含まれる照会やビューは読み取り専用です。

エラーを許容できる *nested-table-expression* の全選択が、マテリアライズ照会表を使用して最適化されることはありません。

表参照における相関参照

相関参照は、ネストされた表の式や、表関数の引数として使用することができます。両方の場合に適用される基本的な規則は、相関参照は、副照会の階層のより高いレベルにある*表参照* から行う必要があるということです。この階層には、FROM 節の左から右への処理ですでに解決されている表参照が入っています。ネストされた表の式の場合、LATERAL キーワードが全選択の前に指定されなければなりません。したがって、次の例は正しい構文です。

```
例 1:       SELECT t.c1, z.c5
            FROM t, TABLE( tf3(t.c2) ) AS z      -- FROM において
            WHERE t.c3 = z.c4;                   -- t が tf3 に先行するので、
                                                 -- t.c2 は既知

例 2:       SELECT t.c1, z.c5
            FROM t, TABLE( tf4(2 * t.c2) ) AS z  -- FROM において
            WHERE t.c3 = z.c4;                   -- t が tf4 に先行するので、
                                                 -- t.c2 は既知

例 3:       SELECT d.deptno, d.deptname,
                   empinfo.avgsal, empinfo.empcount
            FROM department d,
                 LATERAL (SELECT AVG(e.salary) AS avgsal,
                                 COUNT(*) AS empcount
                          FROM employee e           -- department が先行し、
                          WHERE e.workdept=d.deptno -- TABLE が指定されて
                         ) AS empinfo;              -- いるので、
                                                    -- d.deptno は既知
```

しかし、以下は正しくない例です。

```
例 4:       SELECT t.c1, z.c5
            FROM TABLE( tf6(t.c2) ) AS z, t  -- t.c2 の t を解決できない!
            WHERE t.c3 = z.c4;               -- 上記例 1 と比較

例 5:       SELECT a.c1, b.c5
            FROM TABLE( tf7a(b.c2) ) AS a, TABLE( tf7b(a.c6) ) AS b
            WHERE a.c3 = b.c4;               -- b.c2 の b を解決できない!

例 6:       SELECT d.deptno, d.deptname,
                   empinfo.avgsal, empinfo.empcount
            FROM department d,
                 (SELECT AVG(e.salary) AS avgsal,
                                 COUNT(*) AS empcount
                          FROM employee e           -- department が先行し、
                          WHERE e.workdept=d.deptno -- TABLE が指定されて
                         ) AS empinfo;              -- いないので、
                                                    -- d.deptno は不明
```

データ変更表参照

data-change-table-reference 節は、中間結果表を指定します。この表は、節に入っている検索済み UPDATE、検索済み DELETE、または INSERT ステートメントで直接変更された行に基づいています。 *data-change-table-reference* は、*select-statement*、 SELECT INTO ステートメント、または共通表式で使用される外部全選択の FROM 節で、唯一の *table-reference* として指定できます。また、割り当てステートメントの唯一の全選択として *data-change-table-reference* を指定することもできます (SQLSTATE 428FL)。データ変更ステートメントのターゲットになる表やビューは、照会で参照される表やビューとして見なされるため、照会の許可 ID には、ターゲットとなる表やビューでの SELECT 特権が必要になります。*data-change-table-reference* 節は、ビュー定義、マテリアライズ照会表定義、あるいは FOR ステートメントで指定することはできません (SQLSTATE 428FL)。

共通表式で定義される一時ビューは、 UPDATE、DELETE、または INSERT ステートメントのターゲットにできません (SQLSTATE 42807)。

FINAL TABLE
中間結果表の行がデータ変更ステートメントの完了時に表示されるとき、これらの行が、SQL データ変更ステートメントによって変更された一連の行を表すことを示します。 AFTER トリガーや参照制約があり、その結果として SQL データ変更ステートメントのターゲットになっている表に対して追加の操作が発生する場合は、エラーが戻されます (SQLSTATE 57058、SQLSTATE 560C6)。SQL データ変更ステートメントのターゲットがデータ変更タイプの INSTEAD OF トリガーで定義されたビューであった場合は、エラーが戻されます (SQLSTATE 428G3)。

NEW TABLE
中間結果表の行が、参照制約や AFTER トリガーのアプリケーションより前に SQL データ変更ステートメントによって変更された一連の行を表していることを示します。参照制約や AFTER トリガーに対する追加の処理のため、ステートメントの完了時にターゲット表にあるデータは、中間結果表のデータと一致しないことがあります。

OLD TABLE
中間結果表の行が、データ変更ステートメントのアプリケーションよりも前に存在していたために SQL データ変更ステートメントによって変更された一連の行を表すことを示します。

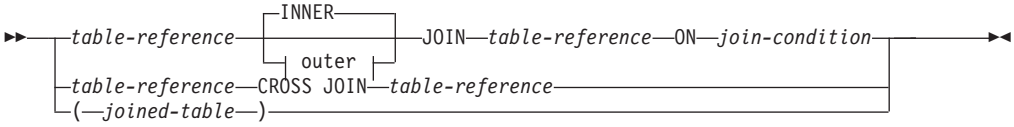
(searched-update-statement)
検索済み UPDATE ステートメントを指定します。 UPDATE ステートメントの WHERE 節や SET 節に、 UPDATE ステートメント外の列への相関参照を入れることはできません。

(searched-delete-statement)
検索済み DELETE ステートメントを指定します。 DELETE ステートメントの WHERE 節に、 DELETE ステートメント外の列への相関参照を入れることはできません。

(insert-statement)
INSERT ステートメントを指定します。 INSERT ステートメントの全選択内で、 INSERT ステートメントの全選択に入っていない列への相関参照を使用することはできません。

data-change-table-reference の中間結果表の内容は、カーソルが開かれたときに決定されます。中間結果表には、指定されたターゲット表またはビューのすべての列を含め、処理されたすべての行が示されます。 SQL データ変更ステートメントのターゲット表またはビューのすべての列は、ターゲット表またはビューから列名を使用してアクセスできます。データ変更ステートメントで INCLUDE 節が指定されている場合、中間結果表には、これらの追加の列が示されます。

joined-table



outer:

結合表 (joined table) は、内部結合または外部結合のいずれかの結果である中間結果の表を指定します。この表は、結合演算子 CROSS、INNER、LEFT OUTER、RIGHT OUTER、または FULL OUTER のいずれかをそのオペランドに適用して得ることができます。

クロス結合は、表のクロス積を表し、左側の表の各行は右側の表の各行と結合されます。内部結合は、表のクロス積と見なすことができ、結合条件が真である行のみを保持します。結果表では、結合された表の一方または両方からの行が欠落している場合があります。外部結合には、内部結合が組み込まれて、このような欠落行を保持します。外部結合には、次の 3 種類のものがあります。

- ***左外部結合***。内部結合から欠落している左側の表の行が入っています。
- ***右外部結合***。内部結合から欠落している右側の表の行が入っています。
- ***全外部結合***。内部結合から欠落している左側および右側の表の行が入っています。

結合演算子の指定がない場合、INNER が暗黙の指定になります。複数の結合が行われる順序は、結果に影響を与えます。結合は、他の結合内にネストすることができます。結合の処理順序は、通常、左から右方向ですが、必要な結合条件の位置に基づきます。ネストされた結合の順序を読みやすくするために、括弧の使用をお勧めします。以下に例を示します。

```
TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1
  RIGHT JOIN TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1
    ON TB1.C1=TB3.C1
```

これは、以下と同じです。

```
(TB1 LEFT JOIN TB2 ON TB1.C1=TB2.C1)
  RIGHT JOIN (TB3 LEFT JOIN TB4 ON TB3.C1=TB4.C1)
    ON TB1.C1=TB3.C1
```

結合表は、SELECT ステートメントの形式のいずれかが使用されるコンテキストであれば、どのようなコンテキストでも使用することができます。その SELECT ステートメントに結合表が入っている場合には、ビューまたはカーソルは読み取り専用です。

join-condition (結合条件) は *search-condition* (検索条件) です。ただし、以下の点で異なっています。

- 副照会 (スカラーなど) を入れることはできない
- 参照値がオブジェクト ID 列以外の場合、間接参照操作または DEREF 関数を組み込むことはできない。
- SQL 関数を組み入れることはできない
- *結合条件* の式で参照される列は、関連する結合のオペランド表のいずれかの列でなければならない (同じ結合表の節の範囲内)。
- 全外部結合の *結合条件* の式で参照される関数は、決定的なものでなければならず、外部アクションは持たない。
- XMLQUERY または XMLEXISTS 式を組み入れることはできない

結合条件が上記の規則にしたがっていない場合、エラーが生じます (SQLSTATE 42972)。

列参照は、列名の修飾子を解決するための規則を使用して解決されます。述部に適用されるのと同じ規則が、結合条件にも適用されます。

結合操作

結合条件 は、T1 と T2 のペアを指定します。ここで、T1 および T2 は、*結合条件* の JOIN 演算子の左と右のオペランド表です。 T1 および T2 の行のすべての組み合わせについて、*結合条件* が真であれば、T1 の行は T2 の行とペアになります。 T1 の行が T2 の行に結合する場合、結果の行は、T1 の行の値が T2 の行の値と連結された値で構成されます。この実行には、NULL 行の生成が関与する場合があります。表の NULL 行は、列が NULL 値を許すか否かに関係なく、表の各列の NULL 値で構成されます。

以下に、結合演算の結果を要約します。

- T1 CROSS JOIN T2 の結果は、これらの表の行のペア化の候補となるものすべてで構成されます。
- T1 INNER JOIN T2 の結果は、結合条件が真であるペアの行で構成されます。
- T1 LEFT OUTER JOIN T2 の結果は、結合条件が真であるペアの行、およびペアになっていない T1 の行ごとに、その行を T2 の NULL 行に連結したもので構成されます。 T2 から得られるすべての列には NULL 値を使用することができます。

- T1 RIGHT OUTER JOIN T2 の結果は、結合条件が真であるペアの行、およびペアになっていない T2 の行ごとに、その行を T1 の NULL 行に連結したもので構成されます。 T1 から得られるすべての列には NULL 値を使用することができます。
- T1 FULL OUTER JOIN T2 の結果は、ペアの行、およびペアになっていない T2 の行ごとにその行を T1 の NULL 行に連結したもの、およびペアになっていない T1 の行ごとにその行を T2 の NULL 行に連結したもので構成されます。 T1 および T2 から得られるすべての列には NULL 値を使用することができます。

where-clause

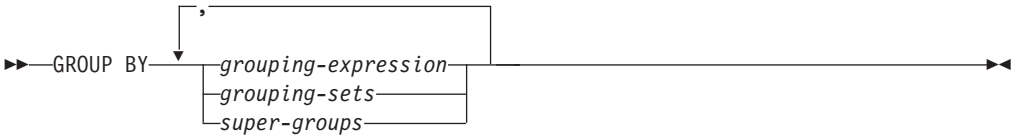
WHERE 節は、 *search-condition (検索条件)* が真である R の行で構成される中間結果表を指定します。 R は、その副選択の FROM 節の結果です。

search-condition は、以下の規則に適合していなければなりません。

- 各 *列名* は、R の列をあいまいなところなく指定するか、あるいは相関参照でなければなりません。副選択外の*表参照* の列を識別している*列名* は、相関参照となります。
- WHERE 節が HAVING 節の副照会に指定されていて、関数の引数がグループに対する相関参照であるのでない限り、列関数を指定することはできません。

search-condition 内の副照会は、R の各行に対して実際に実行され、*search-condition* を R のその行に適用するときにその結果が使用されます。副照会が実際に R の各行に対して実行されるのは、その中に相関参照が組み込まれている場合だけです。実際、相関参照のない副照会は 1 回しか実行されませんが、相関参照のある副照会は、各行ごとに 1 回ずつ実行されます。

group-by-clause



GROUP BY 節は、R の行のグループ化で構成される中間結果表を指定します。R は、副選択のそれ以前の節の結果です。

最も単純な形式では、 GROUP BY 節に *grouping expression* (グループ化式) が入っています。グループ化式とは、R のグループ化の定義で使用される *expression* のことです。グループ化式に入っている各式または *column name* は、R の列を明確に識別していなければなりません (SQLSTATE 42702 または 42703)。グループ化式には、スカラー全選択または、XMLQUERY あるいは XMLEXISTS 式 (SQLSTATE 42822)、または可変の式または関数、あるいは外部処理を伴う式または関数 (SQLSTATE 42845) を組み入れることはできません。

注: 明示的列参照を含んでいない以下の式を *grouping-expression* で使用して、R の列を識別することができます。

- ROW CHANGE TIMESTAMP FOR *table-designator*
- ROW CHANGE TOKEN FOR *table-designator*
- RID_BIT または RID スカラー関数

複雑な形式の GROUP BY 節には、 *grouping-sets* (グループ化集合) および *super-groups* (スーパー・グループ) が組み入れられます。これらの形式の説明については、それぞれ 641 ページの『grouping-sets』 と 641 ページの『super-groups』を参照してください。

GROUP BY の結果は、いくつかの行グループの集まりです。この結果の各行は、*グループ化式* が等しい行の集合を表します。グループ化では、*グループ化式* の NULL 値はすべて等しいものと見なされます。

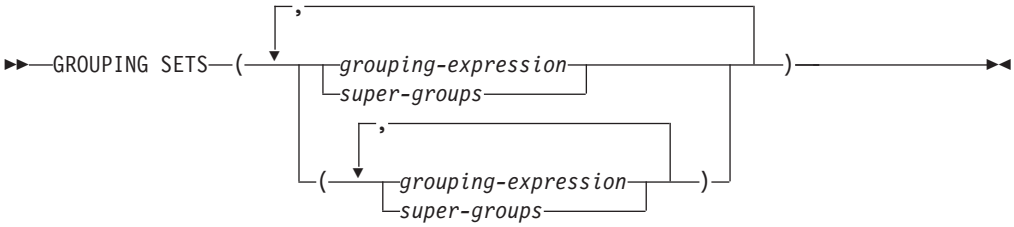
*グループ化式*に 10 進浮動小数点列が含まれ、同じ数値の複数の表記がこれらの列に存在する場合、戻される数値は、数値の表記のうち任意のいずれかになります。

グループ化式 は、HAVING 節の検索条件、SELECT 節の式、または ORDER BY 節の *sort-key-expression* (ソート・キー式) (詳細については、 646 ページの『order-by-clause』 を参照) で使用することができます。いずれの場合も、その参照は各グループの 1 つの値だけを指定します。例えば、*grouping-expression* が *col1+col2* である場合、選択リストで使用できる式は、*col1+col2+3* になります。式の関連性規則では、類似した式 *3+col1+col2* を使用することを許可しません。ただし、対応する式を同じ順序で評価するために括弧を使用する場合を除きます。したがって、選択リストでは *3+(col1+col2)* も使用することができます。連結演算子を使用する場合、*グループ化式* は、選択リストで指定された式とまったく同じように使用しなければなりません。

グループ化式 に末尾のブランクの付いた可変長ストリングが入っている場合、そのグループの値は末尾のブランクの数が異なる場合があり、必ずしも同じ長さになるとはかぎりません。そのような場合でも、*グループ化式* への参照は各グループに 1 つの値だけを指定しますが、グループの値は使用可能な値の集合の中から任意に選択されます。したがって、結果値の実際の長さは予測できません。

前述のように、GROUP BY 節が、 SELECT 節で指定された列を式 (スカラー全選択、可変または外部処理関数) として直接参照することができない場合があります。そのような式を使用してグループ化を行うには、ネストした表式または共通表式を使用して、まず結果の列が式となる結果表を指定します。 ネストした表式の使用例については、例 A9 を参照してください。

grouping-sets



grouping-sets (グループ化集合) の指定により、複数のグループ化節を単一ステートメントで指定することができます。これは、行の複数のグループを単一の結果セットにまとめたものと見なすことができます。これは、1 つのグループ化集合に対応する各副選択ごとに GROUP BY 節を持つ複数の副選択を合併したものと論理的に等しくなります。グループ化集合は、単一のエレメント、もしくは括弧によって区切られる複数のエレメントのリストになります。この場合、エレメントはグループ化式、またはスーパー・グループのいずれかです。*グループ化集合* を使用して、基本表の単一パスを使用してグループを計算することができます。

グループ化集合 の指定により、単純な*グループ化式* を使用するか、またはより複雑な形式の*スーパー・グループ* を使用することができます。*スーパー・グループ* については、 『super-groups』 を参照してください。

グループ化集合は、GROUP BY 演算の基礎的な構築ブロックであることに注意してください。単純な列を使用する単純な GROUP BY は、 1 つのエレメントを持つグループ化集合と見なすことができます。以下に例を示します。

```
GROUP BY a
```

これは、以下と同じです。

```
GROUP BY GROUPING SETS((a))
```

および

```
GROUP BY a,b,c
```

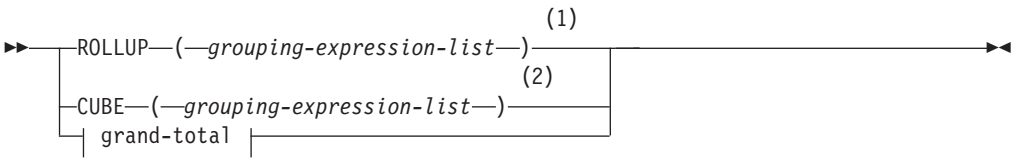
これは、以下と同じです。

```
GROUP BY GROUPING SETS((a,b,c))
```

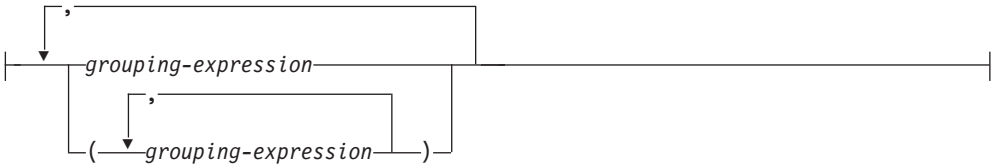
グループ化集合から除外される副選択の選択リストの非集約の列は、そのグループ化集合用に生成される各行の列に NULL 値を戻します。これは、集約が列の値を考慮せずに行われたことを表しています。

例 C2 から 例 C7 は、グループ化集合の使用法を示しています。

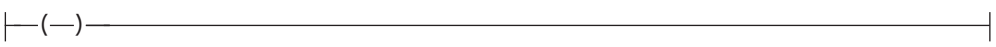
super-groups



grouping-expression-list:



grand-total:



注:

1 GROUP BY 節で単独で使用される代替仕様は、 grouping-expression-list WITH ROLLUP です。

2 GROUP BY 節で単独で使用される代替仕様は、 grouping-expression-list WITH CUBE です。

ROLLUP (*grouping-expression-list* **)**

ROLLUP grouping (ROLLUP グループ化) は GROUP BY 節の拡張であり、『通常の』グループ化された行に加えて*小計* 行の入った結果セットを生成します。*小計* 行は、グループ化行を入手するのに使用されたのと同じ列関数を適用して値が得られる集合体が入っている『スーパー集約』行です。これらの行は小計が最も一般的な用途なので小計行と呼ばれますが、集約には任意の列関数を使用することができます。例えば、例 C8 では MAX および AVG が使用されます。

ROLLUP グループ化は、一連の*グループ化集合* です。 *n* 個のエレメントを持つ ROLLUP の一般的な仕様は、次のとおりです。

```
GROUP BY ROLLUP(C1,C2,...,Cn-1,Cn)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((C1,C2,...,Cn-1,Cn)
                       (C1,C2,...,Cn-1)
                       ...
                       (C1,C2)
                       (C1)
                       () )
```

ROLLUP の *n* のエレメントは、 *n* +1 のグループ化集合に変換される点に注意してください。*グループ化式* が指定されている順序が、 ROLLUP にとって重要である点も注意してください。以下に例を示します。

```
GROUP BY ROLLUP(a,b)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b)
                       (a)
                       () )
```

while

```
GROUP BY ROLLUP(b,a)
```

これは、以下と同じです。

```
GROUP BY GROUPING SETS((b,a)
                       (b)
                       () )
```

ORDER BY 節は、結果セットの行の ORDER BY を確保する唯一の方法です。 例 C3 は ROLLUP の使用法を示しています。

CUBE (*grouping-expression-list*)

CUBE grouping (CUBE グループ化) は GROUP BY 節の拡張であり、すべての ROLLUP 集約行に加えて、「クロス集計」行の入った結果セットを生成します。*クロス集計* 行は、小計による集約の一部ではない「スーパー集約行」です。

ROLLUP と同じように、CUBE グループ化も、一連の*グループ化集合* と見なすことができます。 CUBE の場合、*グループ化式リスト* のすべての順列が総計とともに計算されます。したがって、CUBE の *n* 個のエレメントは、 2**n (2 の *n* 乗) の*グループ化集合* に変換されます。例えば、以下のように指定する場合を考えてみます。

```
GROUP BY CUBE(a,b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (b,c)
                       (a)
                       (b)
                       (c)
                       () )
```

CUBE の 3 個のエレメントは、8 個のグループ化集合に変換されることに注意してください。

エレメントの指定の順序は、CUBE の場合、重要ではありません。 CUBE (DayOfYear, Sales_Person)' と 'CUBE (Sales_Person, DayOfYear)' は同じ結果セットを生成します。「同じ」とは、結果セットの順序ではなく、結果セットの内容を指します。 ORDER BY 節は、結果セットの行の ORDER BY を確保する唯一の方法です。 例 C4 は CUBE の使用法を示しています。

grouping-expression-list

grouping-expression-list (グループ化式リスト) は、 CUBE または ROLLUP 演算のエレメント数を定義するために、 CUBE または ROLLUP 節で使用されます。これは、複数の*グループ化式* を持つエレメントを区切るために括弧を使用して制御されます。

グループ化式 の規則については、 639 ページの『group-by-clause』 で説明しています。例えば、照会が、County ではなく Province 内の City の ROLLUP について合計費用を戻すものと想定します。しかし、以下の節の場合、

```
GROUP BY ROLLUP(Province, County, City)
```

County についての不要な小計行が生じます。以下の節では、

```
GROUP BY ROLLUP(Province, (County, City))
```

複合 (County, City) が ROLLUP の 1 つのエレメントを形成するので、この節を使用する照会は、必要な結果を生じます。つまり、次のように 2 つのエレメントからなる ROLLUP の場合、

```
GROUP BY ROLLUP(Province, (County, City))
```

生成:

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province)
                       () )
```

さらに、3 つのエレメントから成る ROLLUP の場合、次が生成されます。

```
GROUP BY GROUPING SETS((Province, County, City)
                       (Province, County)
                       (Province)
                       () )
```

例 C2 でも、複合列値が使用されています。

grand-total
CUBE および ROLLUP は、全体の集約 (総計) である行を戻します。これは、GROUPING SET 節に空の括弧を使用して別々に指定することができます。また、GROUP BY 節に直接指定することもできます。これは照会の結果には影響しません。 例 C4 では、総計構文を使用しています。

グループ化集合の結合

これは、任意のタイプの GROUP BY 節を組み合わせるのに使用することができます。単純な*グループ化式* のフィールドを他のグループと組み合わせる場合、結果の*グループ化集合* の始めに「追加」されます。 ROLLUP 式または CUBE 式を組み合わせる場合、それらの式は、残りの式では「乗数」のように動作し、 ROLLUP または CUBE の定義に応じて追加のグループ化集合項目を生成します。

例えば、*グループ化式* のエレメントを組み合わせると、次のようになります。

```
GROUP BY a, ROLLUP(b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a) )
```

もしくは

```
GROUP BY a, b, ROLLUP(c,d)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b) )
```

ROLLUP エレメントを組み合わせると、次のようになります。

```
GROUP BY ROLLUP(a), ROLLUP(b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a)
                       (b,c)
                       (b)
                       () )
```

同様に、

```
GROUP BY ROLLUP(a), CUBE(b,c)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c)
                       (a,b)
                       (a,c)
                       (a)
                       (b,c)
                       (b)
                       (c)
                       () )
```

CUBE と *ROLLUP* のエレメントの組み合わせは次のようになります。

```
GROUP BY CUBE(a,b), ROLLUP(c,d)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b,c,d)
                       (a,b,c)
                       (a,b)
                       (a,c,d)
                       (a,c)
                       (a)
                       (b,c,d)
                       (b,c)
                       (b)
                       (c,d)
                       (c)
                       () )
```

単純な*グループ化式* の場合のように、グループ化集合を組み合わせると、各グループ化集合内で重複したものが除去されます。例えば、次のようになります。

```
GROUP BY a, ROLLUP(a,b)
```

これは、以下と同じ意味になります。

```
GROUP BY GROUPING SETS((a,b)
                       (a) )
```

グループ化集合を組み合わせる場合の詳しい例は、完全な CUBE 集約について戻される特定行を除去する結果セットを構成する場合です。

例えば、以下の GROUP BY 節について考えてみます。

```
GROUP BY Region,
         ROLLUP(Sales_Person, WEEK(Sales_Date)),
         CUBE(YEAR(Sales_Date), MONTH (Sales_Date))
```

GROUP BY のすぐ右側にリストされている列は単純にグループ化され、 ROLLUP の後の括弧内の列がロールアップされ、 CUBE の後の括弧内の列は 3 乗されます。したがって、上記の節の結果は、YEAR 内の MONTH のキューブが生成されてから、 REGION 内の Sales_Person 内の WEEK の集約内でロールアップが行わ

れます。この結果は、Region、 Sales_Person または WEEK(Sales_Date) の総計行にもクロス集計行にもならないため、生成される行は、以下の節より少なくなります。

```
GROUP BY ROLLUP (Region, Sales_Person, WEEK(Sales_Date),
                 YEAR(Sales_Date), MONTH(Sales_Date) )
```

having-clause

HAVING 節は、 *search-condition (検索条件)* が真である R のグループで構成される中間結果表を指定します。 R は、副選択のそれ以前の節の結果です。その節が GROUP BY ではない場合、 R はグループ化列のない単一のグループと見なされます。

検索条件内の各*列名* は、以下のいずれかを満たすものであることが必要です。

- R のグループ化列を明確に識別すること。
- 列関数内で指定されていること。
- 相関参照であること。副選択外の*表参照* の列を識別している*列名* は、相関参照となります。

検索条件が適用される R のグループは、検索条件内の各列関数 (引数が相関参照である関数を除く) の引数を提供するものとなります。

検索条件に副照会が入っている場合、その副照会は、検索条件が R のグループに適用されるたびに実行され、その結果は検索条件の適用において使用されるものと見なすことができます。実際には、副照会が各グループごとに実行されるのは、その中に相関参照が入っている場合だけです。この違いについては、例 A6 と 例 A7 を参照してください。

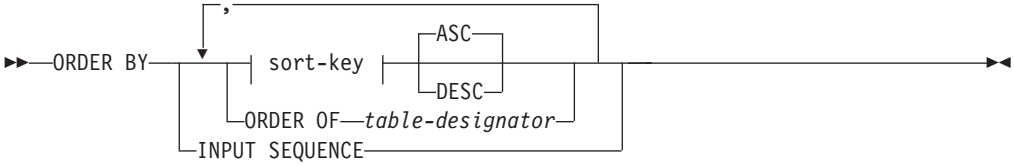
R のグループに対する相関参照は、グループ化列を指定するものであるか、あるいは列関数に入っているものでなければなりません。

HAVING が GROUP BY なしで使用される場合、選択リストに含められるものは、列名 (それが集約関数の引数となっている場合)、相関列参照、グローバル変数、ホスト変数、リテラル、特殊レジスター、SQL 変数、または SQL パラメーターに限られます。

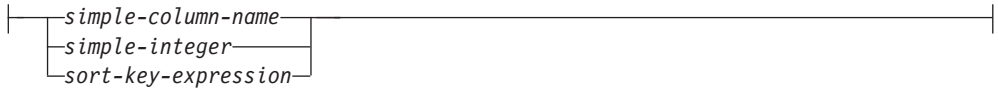
注: 次の式が集約関数内に含まれている場合、HAVING 節でのみ指定可能です (SQLSTATE 42803)。

- ROW CHANGE TIMESTAMP FOR *table-designator*
- ROW CHANGE TOKEN FOR *table-designator*
- RID_BIT または RID スカラー関数

order-by-clause



sort-key:



ORDER BY 節は、結果表の行の ORDER BY を指定します。単一のソート指定 (方向が関連した 1 つの *sort-key* (ソート・キー)) が指定された場合、行は、そのソート指定の値によって順序付けられます。複数のソート指定を指定すると、行の順序は、最初に指定されたソート指定の値によってソートされ、次に 2 番目に指定されたソート指定の値によってソートされます。それぞれの *sort-key* のデータ・タイプは、LONG VARCHAR、CLOB、 LONG VARGRAPHIC、XML、DBCLOB、BLOB にすることはできません。また、これらのタイプの特殊タイプや構造化タイプにすることもできません (SQLSTATE 42907)。

選択リストに指定された列は、 *simple-integer* または *simple-column-name* であるソート・キー によって識別することができます。選択リストに指定されていない列は、*simple-integer*、もしくは場合によっては、選択リストの式と一致する *sort-key-expression* (*sort-key-expression* の詳細を参照) によって識別されなければなりません。列は、AS 節の指定がなく、しかもその列が定数、演算子の入った式、または関数から派生した列の場合には無名です。

順序付けは、比較規則に従って行われます。 ORDER BY 節に 10 進浮動小数点列が含まれ、同じ数値の複数の表記がこれらの列に存在する場合、同じ数値の複数の表記の順序は指定されません。 NULL 値は、他のどのような値よりも高位として扱われます。 ORDER BY 節で行が完全に順序付けされない場合、指定されたすべての列の値が重複する複数の行は、任意の ORDER BY で表示されます。

simple-column-name
通常、結果表の列を識別します。この場合、*simple-column-name (単純列名)* は、選択リストに指定された列の列名でなければなりません。

また、照会が副選択である場合、*simple-column-name* として、 FROM 節で識別される表、ビュー、またはネストされた表の列名も指定することができます。これには暗黙的に隠された列として定義された列も含まれます。副選択が以下の場合は、エラーが生じます。

- SELECT 節に DISTINCT を指定する場合 (SQLSTATE 42822)
- グループ化された結果を生成する場合に、*単純列名* が*グループ化式* ではない場合 (SQLSTATE 42803)

結果の順序付けにどの列を使用するかの決定については、以下の『ソート・キーの列名』で説明されています。

simple-integer
0 より大きく、結果表の列の数以下でなければなりません (SQLSTATE 42805)。整数 *n* は、結果表の *n* 番目の列を指定します。

sort-key-expression
単なる列名または符号なし整数定数ではない式。順序付けが適用される照会は、この形式のソート・キーを使用するためには*副選択* でなければなりません。*sort-key-expression* には、相関スカラー全選択 (SQLSTATE 42703)、または外部処理を伴う関数 (SQLSTATE 42845) を組み入れることはできません。

sort-key-expression 内の列名は、以下の『ソート・キーの列名』で説明されている規則に従っていなければなりません。

指定可能な式にさらに制約が加わる特殊な場合があります。

- DISTINCT が、副選択の SELECT 節に指定されている (SQLSTATE 42822)。

  ソート・キー式は、副選択の選択リスト内の式と完全に一致しなければなりません (スカラー全選択は一致しません)。
- 副選択がグループ化されている (SQLSTATE 42803)。

  ソート・キー式は以下が可能です。
  - 副選択の選択リスト内の式である。
  - 副選択の GROUP BY 節の*グループ化式* が組み込まれる。
  - 列関数、定数、またはホスト変数が組み込まれる。

ASC
列の値を昇順に使用します。これはデフォルトです。

DESC
列の値を降順に使用します。

ORDER OF *table-designator*
表指定子 で使用されているのと同じ順序付けを、副選択の結果表にも適用することを指定します。この節を指定する副選択の FROM 節内には、*表指定子* に一致する表参照がなければなりません (SQLSTATE 42703)。指定された *表指定子* に対応する副選択 (または全選択) には、データに依存する ORDER BY 節が入っていなければなりません (SQLSTATE 428FI)。適用される ORDER BY は、ネストされた副選択 (または全選択) 内の ORDER BY 節の列が外部副選択 (または全選択) に入っていた場合、およびそれらの列が ORDER OF 節の代わりに指定された場合と同じです。

この形式は、全選択 (全選択の変性形式を除く) では許可されていないので注意してください。例えば、以下は無効です。

```
(SELECT C1 FROM T1
   ORDER BY C1)
UNION
SELECT C1 FROM T2
   ORDER BY ORDER OF T1
```

以下の例は有効です。

```
SELECT C1 FROM
   (SELECT C1 FROM T1
      UNION
    SELECT C1 FROM T2
    ORDER BY C1 ) AS UTABLE
ORDER BY ORDER OF UTABLE
```

INPUT SEQUENCE
INSERT ステートメントの場合に、結果表が配列済みデータ行の入力配列を反映していることを示します。 INPUT SEQUENCE 配列は、 FROM 節で INSERT ステートメントが指定されている場合にのみ使用できます (SQLSTATE 428G4)。 627 ページの『table-reference』を参照してください。 INPUT SEQUENCE が指定されていても、入力データが配列されていない場合は、INPUT SEQUENCE 節は無視されます。

注

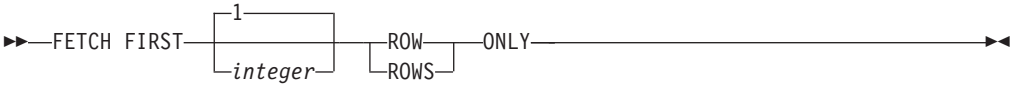
- **ソート・キーの列名**
  - 列名が修飾されている場合

    照会は*副選択* でなければなりません (SQLSTATE 42877)。列名は、副選択の FROM 節の表、ビュー、またはネストされた表の列を明確に識別する必要があります (SQLSTATE 42702)。列の値は、ソート指定の値を計算するのに使用されます。
  - 列名が無修飾の場合
    - 照会は副選択です。

      列名が結果表の複数の列の名前と同一である場合、この列名は、順序付け副選択の FROM 節内の表、ビュー、またはネストされた表の列を明確に識別する必要があります (SQLSTATE 42702)。列名が 1 つの列と同一である場合、その列は、ソート指定の値を計算するのに使用されます。列名が結果表の列と同一でない場合、この列名は、選択ステートメントの全選択の FROM 節の表、ビュー、またはネストされた表の列を明確に識別する必要があります (SQLSTATE 42702)。
    - 照会は、副選択ではありません (UNION、EXCEPT、または INTERSECT などの SET 演算が組み入れられます)。

      列名は、結果表の複数の列の名前と同一にすることはできません (SQLSTATE 42702)。列名は、結果表のちょうど 1 つの列と同一でなければなりません (SQLSTATE 42707)。この列は、ソート指定の値を計算するのに使用されます。
- **制限**: ソート・キー式 またはその列が選択リストにない*単純列名* を使用すると、ソートに使用される一時表にその列または式が追加される場合があります。これにより、表の列の数の制限、または表の行のサイズの制限に到達する場合があります。ソート操作を行うのに一時表が必要になる場合、このような制限を超えると、エラーが生じます。

fetch-first-clause



FETCH FIRST 節 は、検索できる最大行数を設定します。この節は、アプリケーションが最大 *integer* 行数までしか検索しないことを、データベース・マネージャーに認識させます。これは、この節が指定されていない場合に、結果表にある行数に影響されません。 *integer* 行より多くの行を取り出そうとすると、通常のデータの終わりと同じように処理されます (SQLSTATE 02000)。 *integer* の値は、正の整数 (ゼロを除く) でなければなりません。

fetch-first-clause を使用すると、副選択または全選択の照会最適化は、多くても *integer* 個の行が取り出されるという事実に基づいて、影響を受けます。最外部の全選択に fetch-first-clause が指定され、選択ステートメントに optimize-for-clause が指定されている場合、データベース・マネージャーは optimize-for-clause からの *integer* を使用して、最外部の全選択の照会最適化に影響を与えます。

結果表を最初の *integer* 行に限定することにより、パフォーマンスが向上します。データベース・マネージャーは、一度最初の *integer* 行を判別すると、照会処理を停止します。 *fetch-first-clause* と *optimize-for-clause* の両方が指定されている場合には、これらの節の *integer* 値のうちの小さい方を使用して通信バッファー・サイズが決定されます。

全選択に FROM 節の SQL データ変更ステートメントが入っている場合は、フェッチされる行の数の限度に関係なく、すべての行が変更されます。

副選択の例

例 A1: EMPLOYEE 表の列および行を全選択します。

```
SELECT * FROM EMPLOYEE
```

例 A2: EMP_ACT 表および EMPLOYEE 表を結合し、 EMP_ACT 表からすべての列を選択して、 EMPLOYEE 表の従業員の姓 (LASTNAME) を結果の各行に追加します。

```
SELECT EMP_ACT.*, LASTNAME
  FROM EMP_ACT, EMPLOYEE
  WHERE EMP_ACT.EMPNO = EMPLOYEE.EMPNO
```

例 A3: EMPLOYEE 表と DEPARTMENT 表を結合し、1930 年よりも前に生まれた (BIRTHDATE) 従業員すべての従業員番号 (EMPNO)、従業員の姓 (LASTNAME)、部門番号 (EMPLOYEE 表の WORKDEPT と DEPARTMENT 表の DEPTNO)、および部門名 (DEPTNAME) を選択します。

```
SELECT EMPNO, LASTNAME, WORKDEPT, DEPTNAME
   FROM EMPLOYEE, DEPARTMENT
   WHERE WORKDEPT = DEPTNO
   AND YEAR(BIRTHDATE) < 1930
```

例 A4: EMPLOYEE 表の中で同じジョブ・コードをもつ行のグループごとに、ジョブ (JOB) と給与 (SALARY) の最低額と最高額を選択します。ただし、グループの中でも、複数の行を備えていて、しかも給与の最高額が 27000 以上のグループについてのみ選択を行います。

```
SELECT JOB, MIN(SALARY), MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY JOB
  HAVING COUNT(*) > 1
  AND MAX(SALARY) >= 27000
```

例 A5: EMP_ACT 表の中から、部門 (WORKDEPT) 'E11' の従業員 (EMPNO) についてのすべての行を選択します。(従業員部門番号は、EMPLOYEE 表に示されています。)

```
SELECT *
  FROM EMP_ACT
  WHERE EMPNO IN
          (SELECT EMPNO
              FROM EMPLOYEE
              WHERE WORKDEPT = 'E11')
```

例 A6: EMPLOYEE 表から、給与の最高額が従業員全体の平均給与に満たないすべての部門について、部門番号 (WORKDEPT) と部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
  FROM EMPLOYEE
  GROUP BY WORKDEPT
  HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                          FROM EMPLOYEE)
```

この例では、HAVING 節の副照会は一度だけ実行されることになります。

例 A7: EMPLOYEE 表を使用して、部門別給与の最高額が他のすべての部門の平均給与より少ない部門の部門番号 (WORKDEPT) とその部門別給与 (SALARY) の最高額を選択します。

```
SELECT WORKDEPT, MAX(SALARY)
   FROM EMPLOYEE EMP_COR
    GROUP BY WORKDEPT
    HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                            FROM EMPLOYEE
                            WHERE NOT WORKDEPT = EMP_COR.WORKDEPT)
```

例 A6 とは反対に、HAVING 節の副照会は、各グループごとに実行する必要があります。

例 A8: 営業担当員の従業員番号と給与、およびその部門の給与平均額と人数とを調べます。

この照会では、まずネストされた表式 (DINFO) を作成して、AVGSALARY 列と EMPCOUNT 列、また WHERE 節で使用される DEPTNO 列を入手する必要があります。

```
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY, DINFO.AVGSALARY, DINFO.EMPCOUNT
 FROM EMPLOYEE THIS_EMP,
       (SELECT OTHERS.WORKDEPT AS DEPTNO,
                          AVG(OTHERS.SALARY) AS AVGSALARY,
                          COUNT(*) AS EMPCOUNT
         FROM EMPLOYEE OTHERS
```

```
        GROUP BY OTHERS.WORKDEPT
       ) AS DINFO
WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

このような場合には、ネストした表式を使用することによって、 DINFO ビューを正規のビューとして作成することによりオーバーヘッドを軽減することができます。ステートメントの準備中に、ビューのカタログにはアクセスされません。これは、照会の残りの部分のコンテキストにより、ビューによって考慮する必要があるのはセールス担当の部門の行だけだからです。

例 A9: 5 つの従業員のランダム・グループについて平均的な教育レベルと給与を表示します。

この照会では、各従業員のランダム値を GROUP BY 節で使用できるようにするために、ネストした表式を使用してこのランダム値を設定する必要があります。

```
SELECT RANDID , AVG(EDLEVEL), AVG(SALARY)
  FROM ( SELECT EDLEVEL, SALARY, INTEGER(RAND()*5) AS RANDID
           FROM EMPLOYEE
       ) AS EMPRAND
  GROUP BY RANDID
```

例 A10: EMP_ACT 表を照会して、その中の従業員の給与が全従業員の中で上位 10 人に入るプロジェクト番号を戻します。

```
SELECT EMP_ACT.EMPNO,PROJNO
   FROM EMP_ACT
   WHERE EMP_ACT.EMPNO IN
       (SELECT EMPLOYEE.EMPNO
        FROM EMPLOYEE
        ORDER BY SALARY DESC
        FETCH FIRST 10 ROWS ONLY)
```

例 A11: PHONES と IDS は同じカーディナリティーの配列値を持つ 2 つの SQL 変数であると想定して、これらの配列を各配列エレメントごとに 3 つの列 (各配列用ごとに 1 つずつ、もう 1 つは位置用) と 1 つの行を持つ表に変換します。

```
SELECT T.PHONE, T.ID, T.INDEX FROM UNNEST(PHONES, IDS)
  WITH ORDINALITY AS T(PHONE, ID, INDEX)
  ORDER BY T.INDEX
```

結合の例

例 B1: この例では、表 J1 および J2 を使用した種々の結合の結果を示しています。これらの表には、以下の行が入っています。

```
SELECT * FROM J1

W   X
--- ------
A       11
B       12
C       13

SELECT * FROM J2

Y   Z
--- ------
A       21
C       22
D       23
```

以下の照会では、両方の表の最初の列が一致する J1 および J2 の内部結合を行っています。

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
```

この内部結合の例では、J1 の列 W='C' の行および J2 の列 Y='D' の行が、結果に入っていません。これは、これらの行がもう一方の表に一致するものがないからです。次のような代替形式の内部結合照会で同じ結果が生成されることに注意してください。

```
SELECT * FROM J1, J2 WHERE W=Y
```

以下の左外部結合では、J2 の列が NULL 値である J1 の欠落行を戻します。 J1 のすべての行が組み入れられます。

```
SELECT * FROM J1 LEFT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
B       12 -        -
C       13 C       22
```

以下の右外部結合では、J1 の列が NULL 値である J2 の欠落行を戻します。 J2 のすべての行が組み入れられます。

```
SELECT * FROM J1 RIGHT OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
```

以下の全外部結合では、NULL 値である J1 と J2 の両方の欠落行を戻します。 J1 と J2 の両方のすべての行が組み入れられます。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y

W   X      Y   Z
--- ------ --- ------
A       11 A       21
C       13 C       22
-        - D       23
B       12 -        -
```

例 B2: 上記の例の表 J1 および J2 を使用して、述部が検索条件に追加されるとどうなるかを調べます。

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

条件を追加すると、内部結合は、例 B1 の内部結合と比較して 1 行のみを選択します。

全外部結合に対するこの影響に注意してください。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=13

W   X      Y   Z
--- ------ --- ------
-        - A       21
C       13 C       22
-        - D       23
A       11 -        -
B       12 -        -
```

内部結合には 1 行のみがあり、両方の表のすべての行を戻す必要があるため、結果は 5 行になります (追加の述部がない場合の 4 行と比較して)。

以下の照会では、同じ述部を WHERE 節に追加することにより、まったく異なる結果を生成される場合を示しています。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=13

W   X      Y   Z
--- ------ --- ------
C       13 C       22
```

WHERE 節は、全外部結合の中間結果の後に適用されます。この中間結果は、例 B1 の全外部結合照会の結果と同じになります。WHERE 節は、この中間結果に適用され、X=13 の行を除くすべての行を除去します。外部結合を行う場合に、述部の位置の選択によって、結果に大きな影響を与える可能性があります。述部が X=13 ではなく X=12 であるとどうなるかを考えてみます。以下の内部結合は行を戻しません。

```
SELECT * FROM J1 INNER JOIN J2 ON W=Y AND X=12
```

したがって、全外部結合は 6 行を返します。つまり、J2 の列が NULL 値である J1 の 3 行と、J1 の列が NULL 値である J2 の 3 行です。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y AND X=12

W   X      Y   Z
--- ------ --- ------
-        - A       21
-        - C       22
-        - D       23
A       11 -        -
B       12 -        -
C       13 -        -
```

追加の述部が WHERE 節にある場合には、1 行が戻されます。

```
SELECT * FROM J1 FULL OUTER JOIN J2 ON W=Y
   WHERE X=12

W   X      Y   Z
--- ------ --- ------
B       12 -        -
```

例 B3: 管理者のいない部門も含めて、すべての部門を従業員番号と管理者の姓と共にリストします。

```
SELECT DEPTNO, DEPTNAME, EMPNO, LASTNAME
  FROM DEPARTMENT LEFT OUTER JOIN EMPLOYEE
       ON MGRNO = EMPNO
```

例 B4: 管理者のいない従業員も含めて、すべての従業員の番号と姓を管理者の従業員番号と姓と共にリストします。

```
SELECT E.EMPNO, E.LASTNAME, M.EMPNO, M.LASTNAME
  FROM EMPLOYEE E LEFT OUTER JOIN
                                        DEPARTMENT INNER JOIN EMPLOYEE M
       ON MGRNO = M.EMPNO
       ON E.WORKDEPT = DEPTNO
```

内部結合は、DEPARTMENT 表で識別されるすべての管理者の姓を判別し、左外部結合によって、対応する部門が DEPARTMENT にない場合であっても各従業員を必ずリストすることができます。

グループ化集合、CUBE、および ROLLUP の例

例 C1 から 例 C4 の照会では、述部 'WEEK(SALES_DATE) = 13' に基づいて SALES 表の行のサブセットを使用しています。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SALES AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
```

これは次の結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 LUCCHESSI                 3
         13           6 LUCCHESSI                 1
         13           6 LEE                       2
         13           6 LEE                       2
         13           6 LEE                       3
         13           6 LEE                       5
         13           6 GOUNOT                    3
         13           6 GOUNOT                    1
         13           6 GOUNOT                    7
         13           7 LUCCHESSI                 1
         13           7 LUCCHESSI                 2
         13           7 LUCCHESSI                 1
         13           7 LEE                       7
         13           7 LEE                       3
         13           7 LEE                       7
         13           7 LEE                       4
         13           7 GOUNOT                    2
         13           7 GOUNOT                   18
         13           7 GOUNOT                    1
```

例 C1: これは、3 つの列に対して基本の GROUP BY 節を使用している照会です。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
```

```
          13           7 GOUNOT                   21
          13           7 LEE                      21
          13           7 LUCCHESSI                 4
```

例 C2: SALES 表の行の 2 つのグループ化集合に基づいて結果を生成します。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY GROUPING SETS ( (WEEK(SALES_DATE), SALES_PERSON),
                         (DAYOFWEEK(SALES_DATE), SALES_PERSON))
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
          13           - GOUNOT                   32
          13           - LEE                      33
          13           - LUCCHESSI                 8
           -           6 GOUNOT                   11
           -           6 LEE                      12
           -           6 LUCCHESSI                 4
           -           7 GOUNOT                   21
           -           7 LEE                      21
           -           7 LUCCHESSI                 4
```

WEEK 13 の行は、最初のグループ化集合のものであり、それ以外の行は 2 番目のグループ化集合のものです。

例 C3: 例 C2 のグループ化集合に関与した 3 つの特殊な列を使用して、ROLLUP を実行する場合、(WEEK、DAY_WEEK、SALES_PERSON)、(WEEK、DAY_WEEK)、(WEEK) および総計のグループ化集合が表示されます。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
          13           6 GOUNOT                   11
          13           6 LEE                      12
          13           6 LUCCHESSI                 4
          13           6 -                        27
          13           7 GOUNOT                   21
          13           7 LEE                      21
          13           7 LUCCHESSI                 4
          13           7 -                        46
          13           - -                        73
           -           - -                        73
```

例 C4: 例 C3 と同じ照会を実行して ROLLUP を CUBE に置き換えるだけの場合、結果に (WEEK、SALES_PERSON)、 (DAY_WEEK、SALES_PERSON)、(DAY_WEEK)、(SALES_PERSON) の追加のグループ化集合が表示されます。

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SALES_PERSON, SUM(SALES) AS UNITS_SOLD
FROM SALES
WHERE WEEK(SALES_DATE) = 13
GROUP BY CUBE ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE), SALES_PERSON )
ORDER BY WEEK, DAY_WEEK, SALES_PERSON
```

これは次のような結果になります。

```
WEEK        DAY_WEEK    SALES_PERSON    UNITS_SOLD
----------- ----------- --------------- -----------
         13           6 GOUNOT                   11
         13           6 LEE                      12
         13           6 LUCCHESSI                 4
         13           6 -                        27
         13           7 GOUNOT                   21
         13           7 LEE                      21
         13           7 LUCCHESSI                 4
         13           7 -                        46
         13           - GOUNOT                   32
         13           - LEE                      33
         13           - LUCCHESSI                 8
         13           - -                        73
          -           6 GOUNOT                   11
          -           6 LEE                      12
          -           6 LUCCHESSI                 4
          -           6 -                        27
          -           7 GOUNOT                   21
          -           7 LEE                      21
          -           7 LUCCHESSI                 4
          -           7 -                        46
          -           - GOUNOT                   32
          -           - LEE                      33
          -           - LUCCHESSI                 8
          -           - -                        73
```

例 C5: SALES 表から選択された行の総計と、 SALES_PERSON および MONTH によって集計された行のグループの入った結果セットを入手します。

```
SELECT SALES_PERSON,
       MONTH(SALES_DATE) AS MONTH,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY GROUPING SETS ( (SALES_PERSON, MONTH(SALES_DATE)),
                         ()
                       )
ORDER BY SALES_PERSON, MONTH
```

これは次のような結果になります。

```
SALES_PERSON    MONTH       UNITS_SOLD
--------------- ----------- -----------
GOUNOT                    3          35
GOUNOT                    4          14
GOUNOT                   12           1
LEE                       3          60
LEE                       4          25
LEE                      12           6
LUCCHESSI                 3           9
LUCCHESSI                 4           4
LUCCHESSI                12           1
-                         -         155
```

例 C6: この例では、2 つの単純な ROLLUP 照会を示し、その後に、2 つの ROLLUP を単一結果セットのグループ化集合として扱い、グループ化集合に入っている列ごとに行の順序を指定する照会を示しています。

例 C6-1:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

結果は次のようになります。

```
WEEK        DAY_WEEK    UNITS_SOLD
----------- ----------- -----------
         13           6          27
         13           7          46
         13           -          73
         14           1          31
         14           2          43
         14           -          74
         53           1           8
         53           -           8
          -           -         155
```

例 C6-2:

```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP ( MONTH(SALES_DATE), REGION );
ORDER BY MONTH, REGION
```

結果は次のようになります。

```
MONTH       REGION          UNITS_SOLD
----------- --------------- -----------
          3 Manitoba                22
          3 Ontario-North            8
          3 Ontario-South           34
          3 Quebec                  40
          3 -                      104
          4 Manitoba                17
          4 Ontario-North            1
          4 Ontario-South           14
          4 Quebec                  11
          4 -                       43
         12 Manitoba                 2
         12 Ontario-South            4
         12 Quebec                   2
         12 -                        8
          - -                      155
```

例 C6-3:

```
SELECT WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD
```

```
  FROM SALES
  GROUP BY GROUPING SETS ( ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) ),
                           ROLLUP( MONTH(SALES_DATE), REGION )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

結果は次のようになります。

```
WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
----------- ----------- ----------- --------------- -----------
         13           6           - -                        27
         13           7           - -                        46
         13           -           - -                        73
         14           1           - -                        31
         14           2           - -                        43
         14           -           - -                        74
         53           1           - -                         8
         53           -           - -                         8
          -           -           3 Manitoba                 22
          -           -           3 Ontario-North             8
          -           -           3 Ontario-South            34
          -           -           3 Quebec                   40
          -           -           3 -                       104
          -           -           4 Manitoba                 17
          -           -           4 Ontario-North             1
          -           -           4 Ontario-South            14
          -           -           4 Quebec                   11
          -           -           4 -                        43
          -           -          12 Manitoba                  2
          -           -          12 Ontario-South             4
          -           -          12 Quebec                    2
          -           -          12 -                         8
          -           -           - -                       155
          -           -           - -                       155
```

2 つの ROLLUP をグループ化集合として使用すると、結果に重複した行が組み入れられます。総計行も 2 つになります。

ORDER BY を使用すると結果にどのような影響があるかを調べてみます。

- 最初のグループ化集合では、week 53 が最後に位置変更されています。
- 2 番目のグループ化集合では、month 12 が最後に位置付けられ、地域がアルファベット順になっています。
- NULL 値は上位にソートされます。

例 C7: 単一のパスで複数の ROLLUP を実行する照会 (例えば 例 C6-3) では、各行を生成したのがどのグループ化集合であるかを示すことができます。以下のステップは、結果セット内の各行の発生点を示す列 (GROUP と呼ばれます) を提供する方法を示しています。結果セットの行を生成したのが、2 つのグループ化集合のいずれであるかということです。

ステップ 1: VALUES 節から選択する照会 (代替形式の全選択) を使用して、新しいデータ値を「生成する」方法を導入します。この照会は、2 つの列 "R1" と "R2"、および 1 行のデータがある、 "X" と呼ばれる表を得る方法を示しています。

```
  SELECT R1,R2
  FROM (VALUES('GROUP 1','GROUP 2')) AS X(R1,R2);
```

結果は次のようになります。

```
R1      R2
------- -------
GROUP 1 GROUP 2
```

ステップ *2:* SALES 表を使ってこの表 "X" のクロス積を生成します。これにより、各行に列 "R1" および "R2" が追加されます。

```
SELECT R1, R2, WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION,
       SALES AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
```

これにより、各行に列 "R1" および "R2" が追加されます。

ステップ *3:* これで、これらの列をグループ化集合と組み合わせて、 ROLLUP 分析にこれらの列を組み入れることができるようになりました。

```
SELECT R1, R2,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                          DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY WEEK, DAY_WEEK, MONTH, REGION
```

結果は次のようになります。

```
R1      R2      WEEK     DAY_WEEK  MONTH     REGION          UNITS_SOLD
------- ------- -------- --------- --------- --------------- -----------
GROUP 1 -             13         6         - -                          27
GROUP 1 -             13         7         - -                          46
GROUP 1 -             13         -         - -                          73
GROUP 1 -             14         1         - -                          31
GROUP 1 -             14         2         - -                          43
GROUP 1 -             14         -         - -                          74
GROUP 1 -             53         1         - -                           8
GROUP 1 -             53         -         - -                           8
-       GROUP 2        -         -         3 Manitoba                   22
-       GROUP 2        -         -         3 Ontario-North               8
-       GROUP 2        -         -         3 Ontario-South              34
-       GROUP 2        -         -         3 Quebec                     40
-       GROUP 2        -         -         3 -                         104
-       GROUP 2        -         -         4 Manitoba                   17
-       GROUP 2        -         -         4 Ontario-North               1
-       GROUP 2        -         -         4 Ontario-South              14
-       GROUP 2        -         -         4 Quebec                     11
-       GROUP 2        -         -         4 -                          43
-       GROUP 2        -         -        12 Manitoba                    2
-       GROUP 2        -         -        12 Ontario-South               4
-       GROUP 2        -         -        12 Quebec                      2
-       GROUP 2        -         -        12 -                           8
-       GROUP 2        -         -         - -                         155
GROUP 1 -              -         -         - -                         155
```

ステップ *4:* R1 および R2 が異なるグループ化集合で使用されるため、 R1 の結果が非 NULL である場合は常に R2 は NULL であり、 R2 の結果が非 NULL である場合は常に R1 は NULL になることに注意してください。つまり、

COALESCE 関数を使用すれば、これらの列を単一列に統合できるということです。また、ORDER BY 節でこの列を使用すれば、 2 つのグループ化集合の結果をまとめることもできます。

```
SELECT COALESCE(R1,R2) AS GROUP,
       WEEK(SALES_DATE) AS WEEK,
       DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
       MONTH(SALES_DATE) AS MONTH,
       REGION, SUM(SALES) AS UNITS_SOLD
FROM SALES,(VALUES('GROUP 1','GROUP 2')) AS X(R1,R2)
GROUP BY GROUPING SETS ((R1, ROLLUP(WEEK(SALES_DATE),
                          DAYOFWEEK(SALES_DATE))),
                       (R2,ROLLUP( MONTH(SALES_DATE), REGION ) )  )
ORDER BY GROUP, WEEK, DAY_WEEK, MONTH, REGION;
```

結果は次のようになります。

```
GROUP   WEEK        DAY_WEEK    MONTH       REGION          UNITS_SOLD
------- ----------- ----------- ----------- --------------- -----------
GROUP 1          13           6           - -                          27
GROUP 1          13           7           - -                          46
GROUP 1          13           -           - -                          73
GROUP 1          14           1           - -                          31
GROUP 1          14           2           - -                          43
GROUP 1          14           -           - -                          74
GROUP 1          53           1           - -                           8
GROUP 1          53           -           - -                           8
GROUP 1           -           -           - -                         155
GROUP 2           -           -           3 Manitoba                   22
GROUP 2           -           -           3 Ontario-North               8
GROUP 2           -           -           3 Ontario-South              34
GROUP 2           -           -           3 Quebec                     40
GROUP 2           -           -           3 -                         104
GROUP 2           -           -           4 Manitoba                   17
GROUP 2           -           -           4 Ontario-North               1
GROUP 2           -           -           4 Ontario-South              14
GROUP 2           -           -           4 Quebec                     11
GROUP 2           -           -           4 -                          43
GROUP 2           -           -          12 Manitoba                    2
GROUP 2           -           -          12 Ontario-South               4
GROUP 2           -           -          12 Quebec                      2
GROUP 2           -           -          12 -                           8
GROUP 2           -           -           - -                         155
```

例 C8: 以下の例は、CUBE を実行する場合の種々の列関数の使用例を示しています。また、この例は、cast 関数および round 関数を利用して、妥当な精度と位取りで 10 進数の結果を生成します。

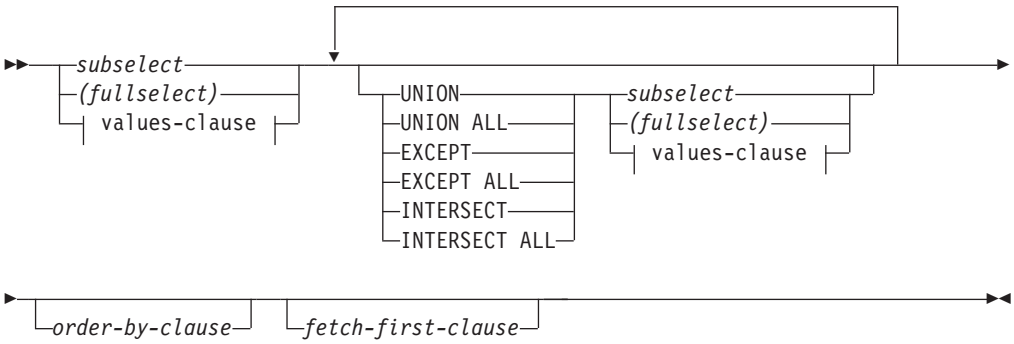
```
SELECT MONTH(SALES_DATE) AS MONTH,
       REGION,
       SUM(SALES) AS UNITS_SOLD,
       MAX(SALES) AS BEST_SALE,
       CAST(ROUND(AVG(DECIMAL(SALES)),2) AS DECIMAL(5,2)) AS AVG_UNITS_SOLD
FROM SALES
GROUP BY CUBE(MONTH(SALES_DATE),REGION)
ORDER BY MONTH, REGION
```

これは次のような結果になります。

```
MONTH       REGION          UNITS_SOLD  BEST_SALE   AVG_UNITS_SOLD
----------- --------------- ----------- ----------- --------------
          3 Manitoba                 22           7           3.14
          3 Ontario-North             8           3           2.67
          3 Ontario-South            34          14           4.25
          3 Quebec                   40          18           5.00
          3 -                       104          18           4.00
```

```
 4 Manitoba                17           9          5.67
 4 Ontario-North            1           1          1.00
 4 Ontario-South           14           8          4.67
 4 Quebec                  11           8          5.50
 4 -                       43           9          4.78
12 Manitoba                 2           2          2.00
12 Ontario-South            4           3          2.00
12 Quebec                   2           1          1.00
12 -                        8           3          1.60
 - Manitoba                41           9          3.73
 - Ontario-North            9           3          2.25
 - Ontario-South           52          14          4.00
 - Quebec                  53          18          4.42
 - -                      155          18          3.87
```

全選択



values-clause:

values-row:

全選択 (fullselect) は、select ステートメント、 INSERT ステートメント、および CREATE VIEW ステートメントのコンポーネントの 1 つです。また、これはステートメントのコンポーネントである、特定の述部のコンポーネントともなります。述部のコンポーネントである全選択は、*副照会 (subquery)* と呼ばれ、括弧で囲んだ全選択 (fullselect) も、副照会と呼ばれることがあります。

セット演算子である UNION、EXCEPT、および INTERSECT は、関係演算子の合併、差、積に対応しています。

全選択は結果表を指定します。セット演算子を使用しない全選択の結果は、指定した副選択または VALUES 節の結果になります。

values-clause
結果表の行の各列ごとに式を使用して実際の値を指定することによって、結果表を派生させます。複数の行を指定することができます。

NULL は、複数の *Values-row* でのみ使用することができ、同一列の少なくとも 1 行は NULL 以外でなければなりません (SQLSTATE 42826)。

Values-row は以下によって指定されます。

- 結果表の単一の列についての単一の式

- コンマで区切った *n* 個の式 (または NULL) を括弧で囲んだもの (*n* は結果表の列の数)

複数行からなる Values-clause には、各 *Values-row* に同数の式が必要です (SQLSTATE 42826)。

以下に、Values-clause の例とその意味を示します。

```
VALUES (1),(2),(3)          - 3 rows of 1 column
VALUES 1, 2, 3              - 3 rows of 1 column
VALUES (1, 2, 3)            - 1 row of 3 columns
VALUES (1,21),(2,22),(3,23) - 3 rows of 2 columns
```

Values-clause は、 *n* 個の指定の *Values-row* RE_1 から RE_n (*n* は 2 以上) で構成され、以下と同等です。

```
RE1 UNION ALL RE2 ... UNION ALL REn
```

これは、各 *Values-row* に対応する式は比較可能でなければなりません (SQLSTATE 42825)。

UNION または UNION ALL
2 つの結果表 (R1 と R2) を組み合わせて、新たな結果表を導きます。 UNION ALL を指定すると、結果は R1 と R2 のすべての行から構成されるものになります。 ALL オプションなしで UNION を指定すると、結果は R1 または R2 のいずれかの行すべての集合から、重複行を除去したものになります。しかしいずれにしても、UNION 表の各行は R1 か R2 のどちらかから取られた行です。

EXCEPT または EXCEPT ALL
2 つの結果表 (R1 と R2) を組み合わせて、新たな結果表を導きます。
EXCEPT ALL を指定すると、結果は、重複行の数を勘定に入れつつ、 R2 の中に対応する行のないすべての行で構成されるものになります。 ALL オプションなしで EXCEPT を指定すると、結果は、それぞれの重複行を除去してから R1 にのみ存在する行を取り出したもので構成されます。

他の SQL インプリメンテーションとの互換性のため、EXCEPT の同義語として MINUS を指定できます。

INTERSECT または INTERSECT ALL
2 つの結果表 (R1 と R2) を組み合わせて、新たな結果表を導きます。
INTERSECT ALL を指定すると、結果は R1 と R2 の両方に入っている行すべてで構成されるものになります。 ALL オプションなしで INTERSECT を指定すると、結果は、R1 と R2 の両方にある行すべての集合から重複行を除去したものになります。

order-by-clause
ORDER BY または FETCH FIRST 節の入った全選択は、以下では指定できません。

- マテリアライズ照会表
- ビューの最外部の全選択 (SQLSTATE 428FJ)。

注: 全選択内の ORDER BY 節は、照会によって戻される行の順序には影響を与えません。 ORDER BY 節が影響を与えるのは、最外部の全選択で指定された場合に戻される行の ORDER BY だけです。

結果表 R1 の中の列の数と R2 の中の列の数は、同じでなければなりません (SQLSTATE 42826)。 ALL キーワードを指定していない場合、 LONG VARCHAR、CLOB、LONG VARGRAPHIC、DBCLOB、 BLOB、の各データ・タイプ、これらの型の特殊タイプ、または構造化タイプをもつ列を R1 および R2 に組み入れることはできません (SQLSTATE 42907)。

結果の列の名前は、次のようになります。

- R1 の *n* 番目の列と R2 の *n* 番目の列の結果列の名前が同じ場合、 R の *n* 番目の列が結果列の名前になります。
- R1 の *n* 番目の列と R2 の *n* 番目の列の結果列の名前が異なる場合は、名前が生成されます。この名前を、ORDER BY 節または UPDATE 節の列名として使用することはできません。

生成された名前を調べるには、SQL ステートメントの DESCRIBE を実行して、SQLNAME フィールドを参照します。

重複した行: 2 つの行が重複していると言えるのは、最初の行の各値が 2 番目の行の対応する値に等しい場合です。重複を判別する場合、2 つの NULL 値は等しいものと見なされ、同じ数値の 2 つの 10 進浮動小数点表記は等しいものと見なされます。例えば、2.00 と 2.0 とは同じ値を持っています (2.00 と 2.0 は等しいものとして比較される) が、異なる指数を持っていて、2.00 と 2.0 の両方の表記が可能になります。したがって、例えば、UNION 演算の結果表に 10 進浮動小数点列と、同じ数値の複数の表記が存在する場合、戻されるもの (例えば 2.00 か 2.0 か) は予測不能です。詳しくは、 123 ページの『数値比較』を参照してください。

複数の演算を 1 つの式の中に結合した場合は、括弧内の演算が先に実行されます。括弧がない場合、演算は左から右に実行されますが、例外として、すべての INTERSECT 演算は UNION または EXCEPT の演算の前に実行されます。

次の例では、表 R1 と R2 の値を左端に示しています。他にリストされている見出しは、R1 と R2 の種々のセット演算の結果の値を示しています。

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					

R1	R2	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTER-SECT ALL	INTER-SECT
		3					
		4					
		4					
		4					
		5					

全選択の例

例 1: EMPLOYEE 表からすべての列と行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2: EMPLOYEE 表の従業員で、その部門番号 (WORKDEPT) が E で始まる部門に属しているか、**または**プロジェクト番号 (PROJNO) が MA2100、MA2110、または MA2112 である EMP_ACT 表のプロジェクトに割り当てられている従業員すべての従業員番号 (EMPNO) をリストします。

```
SELECT EMPNO
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 3: 例 2 と同じ照会を行い、さらに EMPLOYEE 表の行には 'emp'、EMP_ACT 表の行には 'emp_act' という“タグ”を付けます。例 2 の結果とは異なり、この照会では、同じ EMPNO が複数回戻され、付加される “タグ” によりどの表からとられたかが示されます。

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHERE WORKDEPT LIKE 'E%'
UNION
SELECT EMPNO, 'emp_act' FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 4: 例 2 と同じ照会を行いますが、重複行が除去されないように UNION ALL を使用します。

```
SELECT EMPNO
    FROM EMPLOYEE
    WHERE WORKDEPT LIKE 'E%'
UNION ALL
SELECT EMPNO
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
```

例 5: 例 3 と同じ照会を行いますが、現在どの表にもない 2 人の従業員を追加して、それらの行に "new" というタグを付けます。

```
SELECT EMPNO, 'emp'
   FROM EMPLOYEE
   WHEREWORKDEPTLIKE 'E%'
UNION
```

```
SELECT EMPNO, 'emp_act'
   FROM EMP_ACT
   WHERE PROJNO IN('MA2100', 'MA2110', 'MA2112')
UNION
   VALUES ('NEWAAA', 'new'), ('NEWBBB', 'new')
```

例 6: この EXCEPT の例は、T1 に存在し、T2 に存在しない行をすべて生成します。

```
(SELECT * FROM T1)
EXCEPT ALL
(SELECT * FROM T2)
```

NULL 値が関与していない場合、この例は次の例と同じ結果を戻します。

```
SELECT ALL *
   FROM T1
   WHERE NOT EXISTS (SELECT * FROM T2
                     WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

例 7: この INTERSECT の例は、表 T1 と T2 の両方にあるすべての行を生成し、重複した行を除去します。

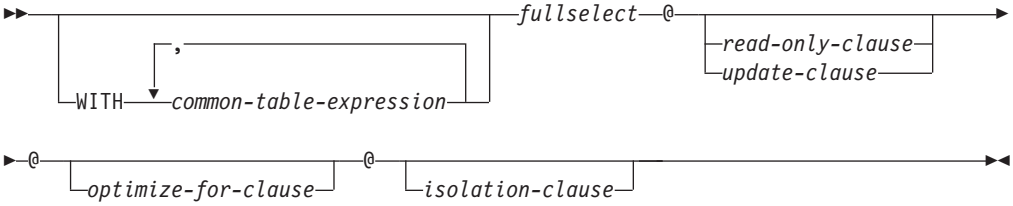
```
(SELECT * FROM T1)
INTERSECT
(SELECT * FROM T2)
```

NULL 値が関与していない場合、この例は次の例と同じ結果を戻します。

```
SELECT DISTINCT * FROM T1
   WHERE EXISTS (SELECT * FROM T2
                 WHERE T1.C1 = T2.C1 AND T1.C2 = T2.C2 AND...)
```

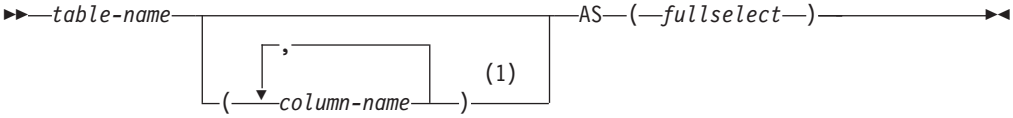
ここで、C1、C2、などは T1 と T2 の列を表します。

Select-statement



select-statement は、DECLARE CURSOR ステートメントに直接指定することや、準備した後で DECLARE CURSOR ステートメントで参照することができる形式の照会です。また、選択ステートメントはコマンド行プロセッサー (または同種のツール) を使用する動的 SQL ステートメントを使って発行することができ、それにより、結果表を画面に表示することもできます。いずれの場合も、*select-statement* によって指定される表は、全選択 (fullselect) の結果です。

common-table-expression



注:

1 共通表式 (common table expression) が再帰的である場合、あるいは全選択の結果として列名が重複する場合は、列名を指定する必要があります。

*共通表式*を使用すると、結果表を *table-name (表名)* によって定義して、それをその後に続く全選択の任意の FROM 節に指定できるようにすることができます。単一の WITH キーワードの後に、複数の共通表式を指定することができます。指定する各共通表式は、それ以降の共通表式の FROM 節の中でも名前によって参照することができます。

列のリストを指定する場合、その中の列の名前の数は、全選択の結果表内の列数と同じ数でなければなりません。各 *column-name* (列名) は、ユニークで、しかも非修飾でなければなりません。これらの列名を指定しない場合、共通表式の定義に使用された全選択の選択リストから名前が得られます。

共通表式の *table-name* は、同じステートメントの他の共通表式の *table-name* すべてと異なるものでなければなりません (SQLSTATE 42726)。共通表式が INSERT ステートメントに指定されている場合、*table-name* を、その挿入の対象である表またはビューの名前にすることはできません (SQLSTATE 42726)。共通表式の *table-name* は、その全選択を通じて、どの FROM 節の中でも表名として指定することができます。共通表式の *table-name* は、(カタログの中で) 同じ修飾名の既存の表、ビュー、または別名をオーバーライドするものとなります。

同じステートメントの中に複数の共通表式が定義されている場合、共通表式相互間の循環参照があってはなりません (SQLSTATE 42835)。*循環参照* が生じるのは、 2 つの共通表式 *dt1* と *dt2* が作成された場合に、 *dt1* が *dt2* を参照し、 *dt2* が *dt1* を参照するようになる場合です。

共通表式の全選択の FROM 節に *data-change-table-reference* が入っている場合は、その共通表式がデータを変更するように指示を受けます。データを変更する共通表式は、その共通表式がステートメントの別の箇所で使用されているかどうかに関係なく、ステートメントが処理されるときに常に評価されます。データの読み取りまたは変更を行う共通表式が 1 つでもある場合は、すべての共通表式が発生した順に処理されます。そして、データの読み取りまたは変更を行う各共通表式は、制約やトリガーもすべて含めて完全に実行されます。 1 つの共通表式が完全に実行されるまで、次の共通表式は実行されません。

共通表式は、 CREATE VIEW および INSERT の全選択 (fullselect) の前でもオプションとして使用できます。

共通表式は、以下のような場合に使用することができます。

- ビューの代わりに使用して、ビューが作成されないようにするため (ビューを一般的に使用する必要がなく、定位置の更新や削除を使わない場合)
- スカラー副選択や可変の関数または外部処理を伴う関数から得られる列によりグループ化できるようにする場合
- 必要な結果表がホスト変数に基づいたものである場合
- 同じ結果表を全選択で共用する必要がある場合
- 結果表を再帰的に派生させる必要がある場合
- 照会の中で複数の SQL データ変更ステートメントを処理する必要がある場合

共通表式の全選択の FROM 節の中にそれ自体への参照が入っている場合、その共通表式は、*再帰的共通表式* です。再帰処理を使用した照会は、部品表 (BOM)、予約システム、およびネットワーク・プランなどのアプリケーションをサポートする上で役立ちます。

再帰的共通表式では、以下のことが成り立っていなければなりません。

- 再帰サイクルの一部をなす各全選択は、 SELECT または SELECT ALL で始まっていなければなりません。 SELECT DISTINCT は使用できません (SQLSTATE 42925)。また、集合の和を求める場合には UNION ALL を使用する必要があります (SQLSTATE 42925)。
- 共通表式の *table-name (表名)* の後には、必ず列名を指定する必要があります (SQLSTATE 42908)。
- 最初の UNION の最初の全選択 (初期化全選択) には、どの FROM 節の共通表式のどの列に対する参照も入っていてはなりません (SQLSTATE 42836)。
- 共通表式の列名が反復全選択において参照される場合、その列のデータ・タイプ、長さ、およびコード・ページは、初期化全選択に基づいて決められます。反復全選択の中の対応する列のデータ・タイプと長さは、初期化全選択に基づいて決められたデータ・タイプと長さと同じでなければならず、コード・ページは一致していなければなりません (SQLSTATE 42825)。ただし、文字ストリング・タ

イプの場合は、 2 つのデータ・タイプの長さが違っても構いません。この場合、反復全選択の列の長さは、初期化全選択から決められた長さに常に割り当て可能な長さでなければなりません。

- 再帰サイクルの一部をなす各全選択には、列関数、GROUP BY 節、または HAVING 節が入っていてはなりません (SQLSTATE 42836)。

  これらの全選択の FROM 節には、再帰サイクルの一部である共通表式に対する参照を多くても 1 つまで組み入れることができます (SQLSTATE 42836)。
- 反復全選択および全体再帰的全選択には、 ORDER BY 節を組み入れることはできません (SQLSTATE 42836)。
- 副照会 (スカラーまたは定量化されたもの) が再帰サイクルの一部であってはなりません (SQLSTATE 42836)。

再帰的共通表式を開発するときには、無限再帰サイクル (ループ) が作成される恐れについて常に注意してください。再帰サイクルは、必ず終了するようにしてください。これは、関係しているデータが循環している場合に特に重要です。再帰的共通表式には、無限ループを防止する述部を組み入れるようにしてください。再帰的共通表式には、以下のものを組み入れるようにしてください。

- 反復全選択の中に、定数ずつ増分される整数列。
- "counter_col < constant" または "counter _col < :hostvar" の形式の反復全選択の WHERE 節の述部。

この構文が再帰的共通表式に見つからないなら、警告が出されます (SQLSTATE 01605)。

再帰の例: 部品表

部品表 (BOM) のアプリケーションは、多くの業務環境において一般的に必要になります。 BOM アプリケーションの再帰的共通表式の機能を示すため、部品とそれに関連する副部品、そして各部品に必要な副部品の数量を示す表について考えてみます。この例では、以下のようにして表を作成します。

```
CREATE TABLE PARTLIST
               (PART VARCHAR(8),
                SUBPART VARCHAR(8),
                QUANTITY INTEGER);
```

この例の照会結果を示すために、PARTLIST 表には次のようなデータが入っているものとします。

```
PART     SUBPART  QUANTITY
-------- -------- -----------
00       01                 5
00       05                 3
01       02                 2
01       03                 3
01       04                 4
01       06                 3
02       05                 7
02       06                 6
03       07                 6
04       08                10
04       09                11
05       10                10
05       11                10
```

```
  06       12                10
  06       13                10
  07       14                 8
  07       12                 8
```

例 1: 単一レベルの展開

最初の例は、単一レベルの展開と呼ばれるものです。これは「'01' で示されている部品を作成するにはどの部品が必要か」という質問に答えるものです。 このリストには、直接の副部品、副部品の副部品などが入ります。しかし、ある部品が何回も使用される場合でも、その副部品は 1 回しかリストに示されません。

```
WITH RPL (PART, SUBPART, QUANTITY) AS
     (  SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
        FROM PARTLIST ROOT
        WHERE ROOT.PART = '01'
      UNION ALL
        SELECT CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
        FROM RPL PARENT, PARTLIST CHILD
        WHERE  PARENT.SUBPART = CHILD.PART
     )
SELECT DISTINCT PART, SUBPART, QUANTITY
 FROM RPL
  ORDER BY PART, SUBPART, QUANTITY;
```

上記の照会では、*RPL* という名前で指定されている共通表式が組み込まれており、それによってこの照会の再帰的な部分が表されています。この照会には、再帰的共通表式の基本的なエレメントが示されています。

UNION の第 1 オペランド (全選択) は*初期化全選択* と呼ばれるもので、それによって部品 '01' の直接の子が求まります。この全選択の FROM 節ではソース表が参照されていますが、それ自身 (この場合は *RPL*) を参照することはありません。この最初の全選択の結果が、共通表式 *RPL* (再帰的 PARTLIST) の中に入れられることになります。この例の場合、UNION は常に UNION ALL でなければなりません。

UNION の第 2 オペランド (全選択) は、 *RPL* を使って、副部品の副部品を計算しています。これは、FROM 節で共通表式 *RPL* とソース表 (CHILD: 子) の部品を、*RPL* (PARENT: 親) に入っている現行の結果の副部品に結び付けることによります。この結果は、再度 *RPL* に入れられます。このようにして、UNION の第 2 オペランドは、子が存在しなくなるまで繰り返し使用されます。

この照会の主要な全選択の SELECT DISTINCT では、同じ部品/副部品が 2 回以上リストに現れることがないようにしています。

照会結果は、次のようになります。

```
  PART     SUBPART  QUANTITY
  -------- -------- -----------
  01       02                 2
  01       03                 3
  01       04                 4
  01       06                 3
  02       05                 7
  02       06                 6
  03       07                 6
  04       08                10
  04       09                11
  05       10                10
```

```
05       11               10
06       12               10
06       13               10
07       12                8
07       14                8
```

この結果では、部品 '01' から '02' へ、そこからさらに '06' へと進むようになっています。さらに、部品 '06' へは、'01' から直接に 1 回、 '02' から 1 回の計 2 回達することに注意してください。しかしこの出力では、そのサブコンポーネントが 1 回しかリストに現れないようになっています (これは SELECT DISTINCT を使用した結果です)。

再帰的共通表式では、*無限ループ* になる可能性を必ず考慮に入れてください。この例で、親表と子表を結合する第 2 オペランドの検索条件を以下のようにコーディングしたとすると、無限ループが作成されることになります。

```
PARENT.SUBPART = CHILD.SUBPART
```

無限ループが発生するこの例は、意図したとおりにコーディングされていない場合であることは明らかです。しかし、再帰サイクルが必ず終了するようにコーディングすることにも注意を払ってください。

この例の照会によって得られる結果は、再帰的共通表式を使用しなくても、アプリケーション・プログラム内で作成することができます。しかし、そのような方法では、すべての再帰レベルごとに新しい照会を開始する必要があります。さらに、すべての結果をデータベースに入れ、その結果を並べ替えることを、アプリケーションで行う必要があります。そのような方法では、アプリケーションのロジックが複雑になり、パフォーマンスはよくありません。要約正展開やインデント正展開の照会など、その他の部品表の照会では、アプリケーションのロジックがさらに複雑で効率の悪いものとなってしまいます。

例 2: 要約正展開

2 番目の例は、要約正展開の例です。ここでの質問は、「部品 '01' の作成には各部品が合計どれくらい必要か」というものです。単一レベル正展開と異なる主な点は、数量を集計する必要があるということです。例 1 は、部品が必要になったときにそれに必要な副部品の数量を示すものです。部品 '01' を作成するのに、副部品が結局どれだけ必要かは示されていません。

```
WITH RPL (PART, SUBPART, QUANTITY) AS
   (
      SELECT ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
       FROM PARTLIST ROOT
       WHERE ROOT.PART = '01'
    UNION ALL
      SELECT PARENT.PART, CHILD.SUBPART, PARENT.QUANTITY*CHILD.QUANTITY
       FROM RPL PARENT, PARTLIST CHILD
       WHERE PARENT.SUBPART = CHILD.PART
   )
SELECT PART, SUBPART, SUM(QUANTITY) AS "Total QTY Used"
 FROM RPL
  GROUP BY PART, SUBPART
  ORDER BY PART, SUBPART;
```

上記の照会では、 *RPL* という名前で指定されている再帰的共通表式の中の UNION の第 2 オペランドの選択リストによって、数量の合計が示されています。副部品の使用量を求めるには、親の数量に、親 1 個当たりの子の数量を乗算します。 1 つ

の部品が異なる複数のロケーションで何回も使用される場合は、もう 1 つ最終的な集計が必要になります。これは、共通表式 *RPL* をグループ化し、主要全選択の選択リストの中で SUM 列関数を使用することによって行います。

照会結果は、次のようになります。

```
PART     SUBPART  Total Qty Used
-------- -------- --------------
01       02                    2
01       03                    3
01       04                    4
01       05                   14
01       06                   15
01       07                   18
01       08                   40
01       09                   44
01       10                  140
01       11                  140
01       12                  294
01       13                  150
01       14                  144
```

この出力のうち、副部品が '06' の行に注目してください。合計使用量の値 15 は、部品 '01' のための直接の数 3 と、部品 '02' のための数 (6) に部品 '01' の数 (2) を掛けたものとを加えた数です。

例 3: 深さの制御

この表の中に存在する部品のレベルが、とりあえず照会で必要なレベルより深い場合はどうなるのでしょうか。つまり、「'01' で指定される部品を作成するために必要な部品の最初の 2 つのレベルはどんなものか」という質問に答えるためには、どんな照会を作成したらよいでしょうか。例をわかりやすくするため、レベル番号を結果に組み入れることにします。

```
WITH RPL (LEVEL, PART, SUBPART, QUANTITY) AS
      (
         SELECT 1,               ROOT.PART, ROOT.SUBPART, ROOT.QUANTITY
          FROM PARTLIST ROOT
          WHERE ROOT.PART = '01'
       UNION ALL
         SELECT PARENT.LEVEL+1, CHILD.PART, CHILD.SUBPART, CHILD.QUANTITY
          FROM RPL PARENT, PARTLIST CHILD
          WHERE PARENT.SUBPART = CHILD.PART
            AND PARENT.LEVEL < 2
      )
 SELECT PART, LEVEL, SUBPART, QUANTITY
   FROM RPL;
```

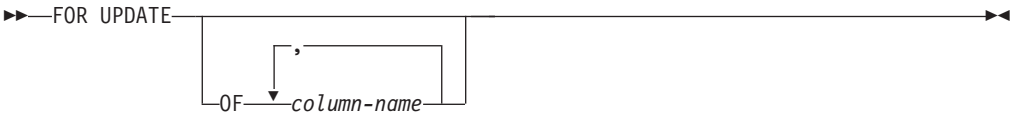
この照会は例 1 に似ています。元の部品からのレベルを示すために、列 *LEVEL* を使っています。初期化全選択では、*LEVEL* 列の値を 1 に初期化しています。それ以降の全選択では、親のレベルに 1 ずつ加算します。次に、結果のレベル数を制御するため、2 番目の全選択に、親のレベルが 2 未満でなければならないという条件が組み入れられています。これによって、2 番目の全選択では、子の処理が 2 次レベルまでしか行われないことになります。

照会結果は、次のようになります。

```
PART     LEVEL       SUBPART  QUANTITY
-------- ----------- -------- -----------
01                 1 02                 2
01                 1 03                 3
```

```
01              1 04              4
01              1 06              3
02              2 05              7
02              2 06              6
03              2 07              6
04              2 08             10
04              2 09             11
06              2 12             10
06              2 13             10
```

update-clause



FOR UPDATE 節は、その後実行される UPDATE ステートメントで更新可能となる列を指定します。各 *column-name (列名)* は非修飾でなければならず、全選択の最初の FROM 節で指定された表またはビューの列を指定するものでなければなりません。列名のない FOR UPDATE 節を指定すると、全選択の最初の FROM 節に指定された表またはビューの列のうち更新可能な列すべてが組み込まれます。

以下のいずれかに該当する場合、FOR UPDATE 節は使用できません。

- 選択ステートメントに関連付けられているカーソルが削除不能である。
- 選択された列のいずれかがカタログ表の更新不能な列であり、 FOR UPDATE 節を使ってその列が除外されていない。

read-only-clause

FOR READ ONLY 節は、結果表が読み取り専用であり、カーソルは UPDATE ステートメントおよび DELETE ステートメントで参照されません。 FOR FETCH ONLY も同じ意味です。

結果表の中には、最初から読み取り専用のものがあります。 (読み取り専用ビューに基づく表など。) FOR READ ONLY は、このような表にも指定できますが、指定しても効果はありません。

更新と削除ができない結果表の場合、FOR READ ONLY (または FOR FETCH ONLY) を指定すると、データベース・マネージャーが、ブロッキングを行うことができるため、 FETCH 操作のパフォーマンスが向上する可能性があります。たとえば、FOR READ ONLY 節または ORDER BY 節のない動的 SQL ステートメントの入ったプログラムでは、 FOR UPDATE 節が指定されたかのようにして、データベース・マネージャーがカーソルをオープンする場合があります。したがって、UPDATE または DELETE ステートメントで照会を使用する場合以外は、パフォーマンスを向上させるために FOR READ ONLY 節を使用するようにしてください。

読み取り専用結果表は、それが最初から読み取り専用であるか、それとも FOR READ ONLY (FOR FETCH ONLY) として指定されたのかには関係なく、定位置

UPDATE または DELETE ステートメントで参照することはできません。

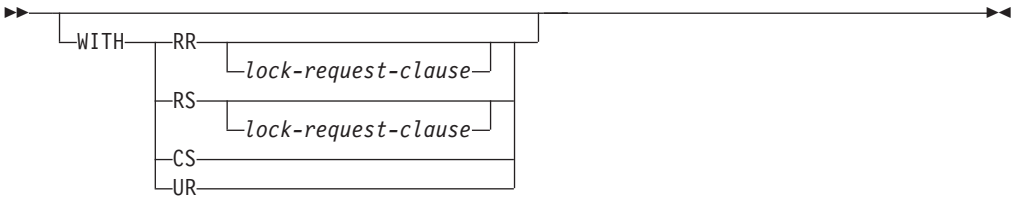
optimize-for-clause

OPTIMIZE FOR 節は、*選択ステートメント* の特殊な処理を要求します。この節が省略されると、結果表のすべての行が検索されることが想定されます。指定されている場合には、検索される行数はおそらく *n* を超えないことを前提としています。ここで、*n* は *integer* の値です。 *n* の値は、正の整数でなければなりません。OPTIMIZE FOR 節を使用すると、 *n* 個の行が検索されることを前提とする照会の最適化に影響を与えます。さらに、ブロックされているカーソルの場合、この節は、各ブロックで戻される行の数に影響を与えます (すなわち、各ブロックで戻される行の数は *n* 行以下になります)。 *fetch-first-clause* と *optimize-for-clause* の両方が指定されている場合には、これらの節の integer 値のうちの小さい方を使用して通信バッファー・サイズが決定されます。これらの値は最適化処理専用です。

この節を指定しても、取り出される行の数が制限されることはなく、パフォーマンス以外ではどんな点でも結果に影響を与えることはありません。 OPTIMIZE FOR *n* ROWS を使用した場合、 *n* 個以下の行を取り出す場合にはパフォーマンスが向上することがありますが、 *n* 個を超える行を取り出す場合にはパフォーマンスが低下する可能性があります。

n の値に行のサイズを乗算した値が、通信バッファーのサイズを超える場合、OPTIMIZE FOR 節はデータ・バッファーに影響を与えません。通信バッファーのサイズは、**rqrioblk** または **aslheapsz** 構成パラメーターによって定義されます。

isolation-clause



オプションの *isolation-clause* は、ステートメントが実行される分離レベルを指定し、さらに特定のタイプのロックをかけるかどうかを指定します。

- RR - 反復可能読み取り
- RS - 読み取り固定
- CS - カーソル固定
- UR - 非コミット読み取り

ステートメントのデフォルト分離レベルは、ステートメントがバインドされているパッケージの分離レベルです。ニックネームが *select-statement* の中で、DB2 ファミリーや Microsoft SQL Server のデータ・ソースのデータへのアクセスに使用される場合、*isolation-clause* をステートメントの分離レベルの指定のためにステートメントに組み込むことができます。 *isolation-clause* が他のデータ・ソースにアクセス

するステートメントに組み込まれた場合、指定の分離レベルは無視されます。フェデレーテッド・サーバーの現行の分離レベルは、データ・ソースへの各接続上のデータ・ソースの対応する分離レベルにマップされます。データ・ソースへの接続が確立された後は、その接続が継続する限り、分離レベルを変更することはできません。

lock-request-clause

オプションの *lock-request-clause* は、データベース・マネージャーがかけたままにするロックのタイプを指定します。

SHARE
並行処理プロセスが、データに対する SHARE または UPDATE ロックをかけることができます。

UPDATE
並行処理プロセスが、データに SHARE ロックをかけることができますが、並行処理中のいずれのプロセスも UPDATE または EXCLUSIVE ロックをかけることはできません。

EXCLUSIVE
並行処理プロセスは、データにロックをかけることはできません。

lock-request-clause を適用される対象は、副照会、SQL 関数、および SQL メソッド内のものも含め、照会に必要なすべての基本表と索引スキャンです。これは、プロシージャー、外部関数、または外部メソッドでかけられたロックに対しては効力をもちません。このステートメントで呼び出される (直接または間接に) SQL 関数または SQL メソッドはすべて、 INHERIT ISOLATION LEVEL WITH LOCK REQUEST で作成されたものでなければなりません(SQLSTATE 42601)。トリガーを呼び出す可能性があるか、または参照保全検査を必要とする修正照会と一緒に *lock-request-clause* を使用することはできません (SQLSTATE 42601)。

SELECT ステートメントの例

例 1: EMPLOYEE 表からすべての列と行を選択します。

```
SELECT * FROM EMPLOYEE
```

例 2: PROJECT 表からプロジェクト名 (PROJNAME)、開始日 (PRSTDATE)、および終了日 (PRENDATE) を選択します。その日付が最新の終了日から順に結果表を配列します。

```
SELECT PROJNAME, PRSTDATE, PRENDATE
 FROM PROJECT
 ORDER BY PRENDATE DESC
```

例 3: EMPLOYEE 表のすべての部門の部門番号 (WORKDEPT) と部門別給与 (SALARY) の平均額を選択します。結果表は、部門別給与の平均額の昇順に配列します。

```
SELECT WORKDEPT, AVG(SALARY)
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   ORDER BY 2
```

例 4: C プログラムで使用する UP_CUR という名前のカーソルを宣言して、PROJECT 表の開始日 (PRSTDATE) と終了日 (PRENDATE) の列を更新します。プログラムは、各行のこれらの 2 つの値と、プロジェクト番号 (PROJNO) とを受け取る必要があります。

```
EXEC SQL  DECLARE UP_CUR CURSOR FOR
            SELECT PROJNO, PRSTDATE, PRENDATE
              FROM PROJECT
              FOR UPDATE OF PRSTDATE, PRENDATE;
```

例 5: この例では、SAL+BONUS+COMM に TOTAL_PAY という名前を付けます。

```
SELECT SALARY+BONUS+COMM AS TOTAL_PAY
  FROM EMPLOYEE
  ORDER BY TOTAL_PAY
```

例 6: 営業担当員の従業員番号と給与、およびその部門の給与平均額と人数とを調べます。また、部門別給与平均額と、平均額の最高値も調べます。

ここでは、共通表式を使用することによって、 DINFO ビューを正規のビューとして作成したときのオーバーヘッドを軽減します。ステートメントの作成中に、ビューのカタログにはアクセスされません。これは、全選択の残りの部分のコンテキストにより、ビューによって考慮する必要があるのはセールス担当の部門の行だけだからです。

```
WITH
   DINFO (DEPTNO, AVGSALARY, EMPCOUNT) AS
       (SELECT OTHERS.WORKDEPT, AVG(OTHERS.SALARY), COUNT(*)
         FROM EMPLOYEE OTHERS
         GROUP BY OTHERS.WORKDEPT
       ),
   DINFOMAX AS
       (SELECT MAX(AVGSALARY) AS AVGMAX FROM DINFO)
SELECT THIS_EMP.EMPNO, THIS_EMP.SALARY,
       DINFO.AVGSALARY, DINFO.EMPCOUNT, DINFOMAX.AVGMAX
 FROM EMPLOYEE THIS_EMP, DINFO, DINFOMAX
 WHERE THIS_EMP.JOB = 'SALESREP'
 AND THIS_EMP.WORKDEPT = DINFO.DEPTNO
```

例 7: EMPLOYEE と PROJECT という 2 つの表があるとして、従業員 SALLY を新しい従業員 GEORGE に置き換え、 SALLY が指揮していたプロジェクトをすべて GEORGE に割り当て、更新されたプロジェクトの名前を戻します。

```
WITH
  NEWEMP AS (SELECT EMPNO FROM NEW TABLE
               (INSERT INTO EMPLOYEE(EMPNO, FIRSTNME)
                   VALUES(NEXT VALUE FOR EMPNO_SEQ, 'GEORGE'))),
  OLDEMP AS (SELECT EMPNO FROM EMPLOYEE WHERE FIRSTNME = 'SALLY'),
  UPPROJ AS (SELECT PROJNAME FROM NEW TABLE
               (UPDATE PROJECT
                   SET RESPEMP = (SELECT EMPNO FROM NEWEMP)
                   WHERE RESPEMP = (SELECT EMPNO FROM OLDEMP))),
  DELEMP AS (SELECT EMPNO FROM OLD TABLE
               (DELETE FROM EMPLOYEE
                   WHERE EMPNO = (SELECT EMPNO FROM OLDEMP)))
SELECT PROJNAME FROM UPPROJ;
```

例 8: DEPT 表からデータを取り出します。次にそのデータは、取り出された更新内容に合わせて更新されますが、照会の実行時にはロックされていなければなりません。

```
SELECT DEPTNO, DEPTNAME, MGRNO
  FROM DEPT
  WHERE ADMRDEPT ='A00'
  FOR READ ONLY WITH RS USE AND KEEP EXCLUSIVE LOCKS
```

付録 A. SQL と XML の制限

以下の表は、SQL と XML の具体的な制限値を示しています。最も制限が厳しい場合に準拠することによって、容易に移植できるアプリケーション・プログラムを設計することができます。

表 42 は、制限値をバイト単位でリストしています。 ID の作成時に、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・ページに変換された後に、これらの制限が課されます。また、データベースからの ID の検索時に、データベース・コード・ページからアプリケーション・コード・ページに変換された後にも、これらの制限が課されます。これらのいずれかのプロセスの中で ID 長さ限界を超えた場合には、切り捨てが生じるか、またはエラーが戻されます。

文字の長さ制限は、データベースのコード・ページとアプリケーションのコード・ページに応じて変わります。例えば、UTF-8 文字の幅は 1 から 4 バイトの範囲にわたるため、限界が 128 バイトの Unicode 表における ID の文字の長さ制限は、どんな文字が使用されるかによって 32 から 128 文字の範囲になります。名前の長さが、データベース・コード・ページへの変換後にこの表の限界を超えるような ID を作成しようとした場合には、エラーが戻されます。

ID 名を保管するアプリケーションは、コード・ページ変換が生じた後に ID のサイズが大きくなる可能性に対処できなければなりません。 ID がカタログから検索されるとき、それらはアプリケーション・コード・ページに変換されます。データベース・コード・ページからアプリケーション・コード・ページに変換されると、結果として ID は、表のバイト限界よりも長くなってしまう可能性があります。アプリケーションによって宣言されるホスト変数が、コード・ページ変換後に ID 全体を格納できない場合、それは切り捨てられます。それが受け入れられない場合には、ID 名全体を受け入れられるように、ホスト変数のサイズを大きくすることができます。

DB2 ユーティリティーのデータ検索、およびユーザー指定コード・ページへのデータの変換にも、同じ規則が適用されます。エクスポートなどの DB2 ユーティリティーがデータを検索し、(エクスポートの CODEPAGE 修飾子または **DB2CODEPAGE** レジストリー変数を使用して) ユーザー指定コード・ページへの変換を課す場合に、コード・ページ変換のために以下の表に明記されている限界を超えて ID が拡張すると、エラーが戻されるか、または ID が切り捨てられる可能性があります。

表 42. ID の長さの制限値

説明	バイト単位の最大値
別名	128
属性名	128
監査ポリシー名	128
許可名 (1 バイト文字のみ可)	128
バッファー・プール名	18
列名[2]	128

© Copyright IBM Corp. 1993, 2010

表 42. ID の長さの制限値 (続き)

説明	バイト単位の最大値
制約名	128
相関名	128
カーソル名	128
データ・パーティション名	128
データ・ソース列名	255
データ・ソース索引名	128
データ・ソース名	128
データ・ソース表名 (リモート表名)	128
データベース・パーティション・グループ名	128
データベース・パーティション名	128
イベント・モニター名	128
外部プログラム名	128
関数マッピング名	128
グループ名	128
ホスト ID[1]	255
データ・ソースのユーザー (リモート許可名) の ID	128
SQL プロシージャー中の ID (条件名、FOR ループ ID、ラベル、結果セット・ロケーター、ステートメント名、変数名)	128
索引名	128
索引拡張名	18
SPECIFICATION ONLY 指定の索引名	128
ラベル名	128
ネーム・スペースの URI (Uniform Resource Identifier)	1000
ニックネーム	128
パッケージ名	128
パッケージ・バージョン ID	64
パラメーター名	128
データ・ソースにアクセスするパスワード	32
プロシージャー名	128
ロール名	128
セーブポイント名	128
スキーマ名[2]	128
セキュリティー・ラベル・コンポーネント名	128
セキュリティー・ラベル名	128
セキュリティー・ポリシー名	128
シーケンス名	128
サーバー名 (データベース別名)	8
特定名	128
SQL 条件名	128

表 42. ID の長さの制限値 (続き)

説明	バイト単位の最大値
SQL 変数名	128
ステートメント名	128
表名	128
表スペース名	18
トランスフォーム・グループ名	18
トリガー名	128
トラステッド・コンテキストの名前	128
タイプ・マッピング名	18
ユーザー定義関数名	128
ユーザー定義メソッド名	128
ユーザー定義タイプ名[2]	128
ビュー名	128
ラッパー名	128
XML エレメント名、属性名、接頭部名	1000
XML スキーマ・ロケーションの URI (Uniform Resource Identifier)	1000

注:

1. ホスト言語コンパイラーによっては、変数名に関してより厳しい制限がある場合があります。
2. SQLDA 構造が 30 バイトの列名を格納するように制限されている場合には、 18 バイトのユーザー定義タイプ名、およびユーザー定義タイプのための 8 バイトのスキーマ名。DESCRIBE ステートメントでは SQLDA が使用されるので、DESCRIBE ステートメントを使用して列またはユーザー定義タイプ名情報を取得する組み込み SQL アプリケーションは、これらの制限に従う必要があります。

表 43. 数値の制限値

説明	制限値
SMALLINT (短精度整数) の最小値	-32 768
SMALLINT (短精度整数) の最大値	+32 767
INTEGER (整数) の最小値	-2 147 483 648
INTEGER (整数) の最大値	+2 147 483 647
BIGINT (64 ビット整数) の最小値	-9 223 372 036 854 775 808
BIGINT (64 ビット整数) の最大値	+9 223 372 036 854 775 807
10 進数の精度の最大値	31
REAL 値の最大指数 (E_{max})	38
REAL (実数) の最小値	-3.402E+38
REAL (実数) の最大値	+3.402E+38
REAL 値の最小指数 (E_{min})	-37

表 43. 数値の制限値 (続き)

説明	制限値
REAL (実数) の正の最大値	+1.175E-37
REAL (実数) の負の最大値	-1.175E-37
DOUBLE 値の最大指数 (E_{max})	308
DOUBLE (倍精度) の最小値	-1.79769E+308
DOUBLE (倍精度) の最大値	+1.79769E+308
DOUBLE 値の最小指数 (E_{min})	-307
DOUBLE (倍精度) の正の最小値	+2.225E-307
DOUBLE (倍精度) の負の最大値	-2.225E-307
DECFLOAT(16) 値の最大指数 (E_{max})	384
最小 DECFLOAT(16) 値[1]	-9.999999999999999E+384
最大 DECFLOAT(16) 値	9.999999999999999E+384
DECFLOAT(16) 値の最小指数 (E_{min})	-383
最小の正 DECFLOAT(16) 値	1.000000000000000E-383
最大の負 DECFLOAT(16) 値	-1.000000000000000E-383
DECFLOAT(34) 値の最大指数 (E_{max})	6144
最小 DECFLOAT(34) 値[1]	-9.999999999999999999999999999999999E+6144
最大 DECFLOAT(34) 値	9.999999999999999999999999999999999E+6144
DECFLOAT(34) 値の最小指数 (E_{min})	-6143
最小の正 DECFLOAT(34) 値	1.000000000000000000000000000000000E-6143
最大の負 DECFLOAT(34) 値	-1.000000000000000000000000000000000E-6143

注:

1. 正規の 10 進浮動小数点数には限界があります。有効な 10 進浮動小数点値には、特殊値 NAN、-NAN、SNAN、-SNAN、INFINITY、および -INFINITY が含まれます。加えて、有効な値には非正規数が含まれます。

   非正規数は、調整された指数が E_{min} よりも小さいゼロ以外の数値です。非正規数では、指数の最小値は E_{min} - (*precision*-1) で、これを E_{tiny} といいます。ここで、*precision* は処理精度 (16 または 34) です。すなわち、非正規数では数値の範囲が、DECFLOAT(16) では 15 桁、DECFLOAT(34) では 33 桁、ゼロに近い側にそれぞれ広がります。非正規数は、その最大桁数が処理精度 (16 または 34) よりも小さいという点で正規の数と異なっています。 10 進浮動小数点数は、正規の数を表現する際と同じほど正確には非正規数を表現することができません。 DECFLOAT(34) の正の最小非正規数は $1x10^{-6176}$ で、これは 1 桁しかありません。一方、DECFLOAT(34) の正の最小正規数は $1.000000000000000000000000000000000x10^{-6143}$ であり、これは 34 桁あります。DECFLOAT(16) の正の最小非正規数は $1x10^{-398}$ です。

表 44. ストリングの制限値

説明	制限値
CHAR の最大長 (バイト単位)	254
VARCHAR の最大長 (バイト単位)	32 672
LONG VARCHAR の最大長 (バイト単位)[1]	32 700
CLOB の最大長 (バイト単位)	2 147 483 647
シリアライズ XML の最大長 (バイト単位)	2 147 483 647
GRAPHIC の最大長 (2 バイト文字単位)	127
VARGRAPHIC の最大長 (2 バイト文字単位)	16 336
LONG VARGRAPHIC の最大長 (2 バイト文字単位)[1]	16 350
DBCLOB の最大長 (2 バイト文字単位)	1 073 741 823
BLOB の最大長 (バイト単位)	2 147 483 647
文字定数の最大長	32 672
GRAPHIC 定数の最大長	16 336
連結後の文字ストリングの最大長	2 147 483 647
連結後の GRAPHIC ストリングの最大長	1 073 741 823
連結後のバイナリー・ストリングの最大長	2 147 483 647
16 進定数の最大桁数	32 672
実行時の構造化タイプ列オブジェクトの最大インスタンス (ギガバイト単位)	1
カタログ・コメントの最大サイズ (バイト単位)	254
注: 1. データ・タイプ LONG VARCHAR と LONG VARGRAPHIC は使用すべきでないものであり、推奨されません。これは将来のリリースで除去される可能性があります。	

表 45. XML 制限

説明	制限値
XML 文書の最大の深さ (レベル数)	125
XML スキーマ文書の最大サイズ (バイト単位)	31 457 280

表 46. 日付/時刻の制限値

説明	制限値
DATE (日付) の最小値	0001-01-01
DATE (日付) の最大値	9999-12-31
TIME (時刻) の最小値	00:00:00
TIME (時刻) の最大値	24:00:00
TIMESTAMP (タイム・スタンプ) の最小値	0001-01-01-00.00.00.000000
TIMESTAMP (タイム・スタンプ) の最大値	9999-12-31-24.00.00.000000

表 47. データベース・マネージャーの制限値

説明	制限値
アプリケーション	

表 47. データベース・マネージャーの制限値 (続き)

説明	制限値
プリコンパイル済みプログラム中のホスト変数宣言の最大数[3]	ストレージ
ホスト変数値の最大長 (バイト単位)	2 147 483 647
プログラムで宣言できるカーソルの最大数	ストレージ
作業単位で変更される行の最大数	ストレージ
一度にオープンできるカーソルの最大数	ストレージ
DB2 クライアント内のプロセス当たりの接続の最大数	512
トランザクション内の同時にオープンできる LOB ロケーターの最大数	4 194 304
SQLDA の最大サイズ (バイト単位)	ストレージ
準備されるステートメントの最大数	ストレージ
バッファー・プール	
32 ビット・リリースでのバッファー・プールの最大 NPAGES	1 048 576
64 ビット・リリースでのバッファー・プールの最大 NPAGES	2 147 483 647
すべてのバッファー・プール・スロットの最大合計サイズ (4K)	2 147 483 646
並行性	
サーバーの同時ユーザーの最大数[4]	64 000
インスタンス当たりの並行ユーザーの最大数	64 000
データベース当たりの並行アプリケーションの最大数	60 000
インスタンス当たりの並行使用可能なデータベースの最大数	256
制約	
表に対する制約の最大数	ストレージ
ユニーク (UNIQUE) 制約の列の最大数 (ユニーク索引によってサポートされる)	64
ユニーク (UNIQUE) 制約の列の結合後の最大長 (ユニーク索引によってサポートされる) (バイト単位)[9]	8192
外部キーで参照される列の最大数	64
外部キーで参照される列の結合後の最大長 (バイト単位)[9]	8192
チェック制約の指定の最大長 (バイト単位)	65 535
データベース	
最大データベース・パーティション番号	999
索引	
表の索引の最大数	32 767 またはストレージのサイズによる
索引キーの列の最大数	64
すべてのオーバーヘッドを含む索引キーの最大長[7] [9]	*indexpagesize*/4
変数の索引キー部分の最大長 (バイト単位)[8]	1022 またはストレージのサイズによる

表 47. データベース・マネージャーの制限値 (続き)

説明	制限値
SMS 表スペース内のデータベース・パーティション当たりの索引の最大サイズ (ギガバイト単位)[7]	16 384
REGULAR DMS 表スペース内のデータベース・パーティション当たりの索引の最大サイズ (ギガバイト単位)[7]	512
LARGE DMS 表スペース内のデータベース・パーティション当たりの索引の最大サイズ (ギガバイト単位)[7]	16 384
XML データに対する索引 に対する索引の変数の索引キー部分の最大長 (バイト単位)[7]	*pagesize*/4 - 207
ログ・レコード	
最大ログ・シーケンス番号	0xFFFF 0000 0000
モニター	
同時に起動できるイベント・モニターの最大数	32
ルーチン	
LANGUAGE SQL を指定したプロシージャー中のパラメーターの最大数	32 767
PROGRAM TYPE MAIN を指定した外部プロシージャー中のパラメーターの最大数	32 767
PROGRAM TYPE SUB を指定した外部プロシージャー中のパラメーターの最大数	90
ユーザー定義関数のパラメーターの最大数	90
ルーチンのネスト・レベルの最大数	64
SQL パス内のスキーマの最大数	64
SQL パスの最大長 (バイト単位)	2048
セキュリティー	
タイプ・セットまたはツリーのセキュリティー・ラベル・コンポーネント内のエレメントの最大数	64
タイプ配列のセキュリティー・ラベル・コンポーネント内のエレメントの最大数	65 535
セキュリティー・ポリシー内のセキュリティー・ラベル・コンポーネントの最大数	16
SQL	
SQL ステートメントの最大全長 (バイト単位)	2 097 152
SQL ステートメントまたはビューで参照される表の最大数	ストレージ
SQL ステートメント中のホスト変数参照の最大数	32 767
ステートメント中の定数の最大数	ストレージのサイズによる
選択リスト内のエレメントの最大数[7]	1012
WHERE または HAVING 節の述部の最大数	ストレージ
GROUP BY 節中の列の最大数[7]	1012
GROUP BY 節中の列の合計長の最大値 (バイト単位)[7]	32 677
ORDER BY 節中の列の最大数[7]	1012
ORDER BY 節中の列の合計長の最大値 (バイト単位)[7]	32 677
副照会ネストの最大レベル	ストレージ

表 47. データベース・マネージャーの制限値 (続き)

説明	制限値
単一のステートメント中の副照会の最大数	ストレージ
挿入操作内の値の最大数[7]	1012
単一の更新操作中の SET 節の最大数[7]	1012
表とビュー	
表の列の最大数 [7]	1012
ビューの列の最大数[1]	5000
ニックネームによって参照されるデータ・ソース表またはビューにある列の最大数	5000
分散キーの列の最大数[5]	500
すべてのオーバーヘッドを含む行の最大長[2] [7]	32 677
非パーティション表の、データベース・パーティション当たりの行の最大数	128 x 10^{10}
データ・パーティションの、データベース・パーティション当たりの行の最大数	128 x 10^{10}
REGULAR 表スペース内のデータベース・パーティション当たりの表の最大サイズ (ギガバイト単位)[3] [7]	512
LARGE DMS 表スペース内のデータベース・パーティション当たりの表の最大サイズ (ギガバイト単位)[7]	16 384
1 つの表のデータ・パーティションの最大数	32 767
表パーティション列の最大数	16
表スペース	
LOB オブジェクトの最大サイズ (テラバイト単位)	4
LF オブジェクトの最大サイズ (テラバイト単位)	2
データベース内の表スペースの最大数	32 768
SMS 表スペース内の表の最大数	65 532
REGULAR DMS 表スペースの最大サイズ (ギガバイト単位)[3] [7]	512
LARGE DMS 表スペースの最大サイズ (テラバイト単位)[3] [7]	16
一時 DMS 表スペースの最大サイズ (テラバイト単位) [37]	16
DMS 表スペース内の表オブジェクトの最大数[6]	687 ページの表 48を参照してください。
自動ストレージ・データベース内のストレージ・パスの最大数	128
自動ストレージ・データベースに関連付けられているストレージ・パスの最大長 (バイト単位)	175
トリガー	
トリガーのカスケード実行時の最大の深さ	16
ユーザー定義タイプ	
構造化タイプ内の属性の最大数	4082

表 47. データベース・マネージャーの制限値 (続き)

説明	制限値
注: 1. この最大値は、 CREATE VIEW ステートメントで結合を使うことによって達成できます。そのようなビューからの選択は、選択リスト内のエレメントの最大数の制限値によって制限されます。 2. BLOB、CLOB、LONG VARCHAR、DBCLOB、および LONG VARGRAPHIC の列の実際のデータは、このカウントには含まれません。しかし、そのデータの格納場所についての情報のために、行内に一定のスペースが確保されます。 3. 示されている数値は、アーキテクチャー上の制限値であり、近似値です。実際の制限値はもっと小さい場合があります。 4. 実際の値はデータベース・マネージャーの構成パラメーター **max_connections** および **max_coordagents** によって制御されます。 5. これはアーキテクチャー上の制限値です。実際上の制限値としては、索引キーの列の最大数に関する制限値を使用する必要があります。 6. 表 48 の脚注 1 を参照してください。 7. ページ・サイズ固有の値については、表 48を参照してください。 8. これには、オーバーヘッドがすべて組み入れられており、最も長い索引キーによってのみ制限されます (バイト単位)。索引キー部分の数が増えるにつれて、各キー部分の最大長も増加します。 9. 索引オプションによっては、最大値がそれより小さな値になることもあります。	

表 48. データベース・マネージャーのページ・サイズ固有の制限値

説明	4K ページ・サイズの制限値	8K ページ・サイズの制限値	16K ページ・サイズの制限値	32K ページ・サイズの制限値
DMS 表スペース内の表オブジェクトの最大数[1]	51 971[2] 53 212[3]	53 299	53 747	54 264
表の列の最大数	500	1012	1012	1012
すべてのオーバーヘッドを含む行の最大長	4005	8101	16 293	32 677
REGULAR 表スペース内のデータベース・パーティション当たりの表の最大サイズ (ギガバイト単位)	64	128	256	512
LARGE DMS 表スペース内のデータベース・パーティション当たりの表の最大サイズ (ギガバイト単位)	2048	4096	8192	16,384
すべてのオーバーヘッドを含む索引キーの最大長 (バイト単位)	1024	2048	4096	8192
SMS 表スペース内のデータベース・パーティション当たりの索引の最大サイズ (ギガバイト単位)	2048	4096	8192	16,384

表 48. データベース・マネージャーのページ・サイズ固有の制限値 (続き)

説明	4K ページ・サイズの制限値	8K ページ・サイズの制限値	16K ページ・サイズの制限値	32K ページ・サイズの制限値
REGULAR DMS 表スペース内のデータベース・パーティション当たりの索引の最大サイズ (ギガバイト単位)	64	128	256	512
LARGE DMS 表スペース内のデータベース・パーティション当たりの索引の最大サイズ (ギガバイト単位)	2048	4096	8192	16 384
通常 DMS 表スペース中の最大サイズ (ギガバイト単位)	64	128	256	512
LARGE DMS 表スペースの最大サイズ (ギガバイト単位)	2048	4096	8192	16 384
一時 DMS 表スペースの最大サイズ (テラバイト単位)	2	4	8	16
選択リスト内のエレメントの最大数	500	1012	1012	1012
GROUP BY 節中の列の最大数	500	1012	1012	1012
GROUP BY 節中の列の合計長の最大値 (バイト単位)	4005	8101	16 293	32 677
ORDER BY 節中の列の最大数	500	1012	1012	1012
ORDER BY 節中の列の合計長の最大値 (バイト単位)	4005	8101	16 293	32 677
挿入操作内の値の最大数	500	1012	1012	1012
単一の更新操作中の SET 節の最大数	500	1012	1012	1012
REGULAR 表スペースのページあたりの最大レコード数	251	253	254	253
LARGE 表スペースのページあたりの最大レコード数	287	580	1165	2335

注:

1. 表オブジェクトには、表データ、索引、LONG VARCHAR 列、VARGRAPHIC 列、および LOB 列が組み入れられます。 表データと同じ表スペース中にある表オブジェクトは、制限値に対して余分のオブジェクトとしてカウントされません。しかし、表データとは異なる表スペースにある各表オブジェクトは、表オブジェクトがある表スペース中の表当たりの表オブジェクト・タイプの制限値に対して、実際に 1 個のオブジェクトとしてカウントされます。
2. エクステント・サイズが 2 ページの時点。
3. エクステント・サイズが 2 ページ以外のサイズの時点。

付録 B. SQLCA (SQL 連絡域)

SQLCA は、すべての SQL ステートメントのそれぞれ実行の終了時に更新される変数の集まりです。実行可能な SQL ステートメントを収めていて、オプション LANGLEVEL SAA1 (デフォルト) または MIA を指定してプリコンパイルされたプログラムは、 1 つだけの SQLCA を用意する必要があります。ただし、マルチスレッドのアプリケーションでは、スレッドごとに 1 つの SQLCA を用意し、その結果 SQLCA が複数になることがあります。

オプション LANGLEVEL SQL92E を指定してプログラムをプリコンパイルした場合、 SQLCODE 変数または SQLSTATE 変数を SQL 宣言セクションで宣言することができ、また SQLCODE 変数をプログラムで宣言することができます。

LANGLEVEL SQL92E を使用すると、SQLCA は用意されません。 REXX 以外のすべての言語では、 SQL INCLUDE ステートメントを使用して SQLCA を宣言することができます。 REXX では、自動的に SQLCA が用意されます。

コマンド行プロセッサーでそれぞれのコマンドを実行した後に SQLCA を表示するには、コマンド db2 -a を実行します。これにより、SQLCA は後続のコマンドの出力の一部として表示されます。同時に SQLCA は db2diag.log ファイルにダンプされます。

SQLCA フィールド記述

表 49. SQLCA のフィールド： この表によって示されているフィールド名は、 INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlcaid	CHAR(8)	SQLCA の入ったストレージ・ダンプの「目印」。 SQL プロシージャー本体の解析から行番号情報が戻される場合、6 番目のバイトは L になります。
sqlcabc	INTEGER	SQLCA の長さ (136) が入ります。
sqlcode	INTEGER	SQL 戻りコードが入ります。 **コード　意味** **0**　正常に実行された (1 つ以上の SQLWARN 標識が設定される場合があります)。 **正の値**　正常に実行されたが、警告状態である。 **負の値**　エラー状態。
sqlerrml	SMALLINT	*sqlerrmc* の長さ標識 (0 から 70 の範囲)。 0 は、*sqlerrmc* の値が関係ないことを示します。

© Copyright IBM Corp. 1993, 2010

表 49. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlerrmc	VARCHAR (70)	エラー条件の説明の中の変数を置き換える 1 つまたは複数のトークンが入り、各トークンは X'FF' で区切られます。 このフィールドは、正常に接続が完了した場合にも使用されます。 NOT ATOMIC コンパウンド SQL ステートメントが発行されると、最大 7 つのエラーに関する情報を入れることができます。 最後のトークンの後に X'FF' が続くことがあります。*sqlerrml* の値には、後続の X'FF' が含まれます。
sqlerrp	CHAR(8)	製品を示す 3 文字の ID の後に、製品のバージョン、リリース、および修正レベルを示す 5 つの英数字が続きます。文字 A から Z までは 9 より高い修正レベルを示します。A は修正レベル 10 を示し、B は修正レベル 11 を示し、以後同様です。例えば、SQL0905C は、DB2 バージョン 9、リリース 5、修正レベル 12 を意味します。 SQLCODE がエラー条件を示している場合、そのエラーを戻したモジュールがこのフィールドに示されます。 このフィールドは、正常に接続が完了した場合にも使用されます。
sqlerrd	ARRAY	診断情報の提供に使用される 6 個の INTEGER 変数。これらの値は、エラーがない場合は通常は空ですが、パーティション・データベースの sqlerrd(6) は例外です。
sqlerrd(1)	INTEGER	接続が正常に起動された場合、アプリケーションのコード・ページからデータベースのコード・ページに変換する際に予想される混合文字データ (CHAR データ・タイプ) の長さの差の最大値が入ります。 0 または 1 の値は、長さが変わらないことを示し、1 より大きい値は長さが長くなることを、負の値は切り捨てが生じることを示します。 SQL プロシージャーから正常に戻された場合、その SQL プロシージャーからの戻り状況値が入ります。
sqlerrd(2)	INTEGER	接続が正常に起動された場合、データベースのコード・ページからアプリケーションのコード・ページに変換する際に予想される混合文字データ (CHAR データ・タイプ) の長さの差の最大値が入ります。 0 または 1 の値は、長さが変わらないことを示し、1 より大きい値は長さが長くなることを、負の値は切り捨てが生じることを示します。 SQLCA がエラーの発生した NOT ATOMIC コンパウンド SQL ステートメントの結果である場合、この値はエラーの発生したステートメントの番号に設定されます。

表 49. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlerrd(3)	INTEGER	PREPARE を呼び出して正常に実行した場合は、戻される行の数の見積もりが入ります。 INSERT、UPDATE、DELETE、または MERGE を実行した後は、実際にその操作のために修飾された行の数になります。コンパウンド SQL を呼び出した場合は、すべてのサブステートメント行の累計が入ります。 CONNECT が呼び出され、データベースが更新可能な場合は 1、データベースが読み取り専用の場合は 2 が入ります。 OPEN ステートメントが呼び出され、カーソルに SQL データ変更ステートメントが入っている場合、このフィールドには、組み込まれた挿入、更新、削除、またはマージ操作のために修飾された行の合計が入ります。 SQL プロシージャーで CREATE PROCEDURE を呼び出したものの、 SQL プロシージャー本体の解析でエラーが発生した場合、エラーが発生した行番号が入ります。この場合、有効な行番号を表すために、sqlcaid の 6 バイト目は必ず 'L' になります。
sqlerrd(4)	INTEGER	PREPARE が正常に呼び出された場合、ステートメントを処理するのに必要なリソースの相対コスト見積もりが入ります。コンパウンド SQL を呼び出した場合は、正常に実行されたサブステートメントの数のカウントが入ります。 CONNECT を呼び出した場合は、下位レベルのクライアントからの 1 フェーズ・コミットなら 0、 1 フェーズ・コミットなら 1、1 フェーズの読み取り専用コミットなら 2、 2 フェーズ・コミットなら 3 が入ります。
sqlerrd(5)	INTEGER	以下の両方の結果として削除、挿入、または更新された行の合計数が入ります。 • 削除操作が正常に実行された後の制約の実施 • 起動したトリガーから引き起こされた SQL ステートメントの処理 コンパウンド SQL を呼び出した場合は、すべてのサブステートメントの上記のような行の累計が入ります。場合によっては、エラーが発生した場合に、内部エラー・ポインターである負の値が入ります。 CONNECT が呼び出された場合に入る値は、サーバー認証の場合は 0、クライアント認証の場合は 1、 DB2 Connect を使用した認証の場合は 2、SERVER_ENCRYPT 認証の場合は 4、暗号化を指定された DB2 Connect を使った認証では 5、 KERBEROS 認証の場合は 7、 GSSPLUGIN 認証の場合は 9、 DATA_ENCRYPT 認証の場合は 11、認証が指定されていなければ 255 です。

表 49. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlerrd(6)	INTEGER	パーティション・データベースの場合、エラーまたは警告が出されたデータベース・パーティションのパーティション番号が入ります。エラーまたは警告が出されなかった場合は、このフィールドにはコーディネーター・パーティションのパーティション番号が入ります。このフィールドに入る番号は、 db2nodes.cfg ファイルでデータベース・パーティションに対して指定されたものと同じです。
sqlwarn	配列	一連の警告標識で、それぞれの標識はブランクか W です。コンパウンド SQL を呼び出した場合は、すべてのサブステートメントについての警告標識の累積が入ります。
sqlwarn0	CHAR(1)	他の標識がすべてのブランクの場合はブランク、 1 つでもブランクでない標識があれば「W」。
sqlwarn1	CHAR(1)	ホスト変数への割り当て時にストリング列の値が切り捨てられた場合は「W」になります。 NULL 終止符が切り捨てられた場合は「N」になります。 CONNECT または ATTACH が正常に実行され、その接続の許可名が 8 バイトを超える場合は「A」になります。 sqlerrd(4) 内に保管されている PREPARE ステートメントの相対コスト見積もりが、INTEGER 内に保管できたであろう値を超える場合や、1 より小さかった場合に、 CURRENT EXPLAIN MODE または CURRENT EXPLAIN SNAPSHOT 特殊レジスターが NO 以外の値に設定されていると、「P」になります。
sqlwarn2	CHAR(1)	列関数の引数から NULL 値が削除された場合、「W」が入ります。[a] CONNECT が呼び出されて正常に実行された場合に、データベースが静止状態であれば「D」が入り、インスタンスが静止状態であれば「I」が入ります。
sqlwarn3	CHAR(1)	列の数とホスト変数の数が等しくない場合、「W」が入ります。 ASSOCIATE LOCATORS ステートメントに指定されている結果セット・ロケーターが、プロシージャーから戻された結果セットの数より少ない場合、「Z」が入ります。
sqlwarn4	CHAR(1)	準備された UPDATE または DELETE ステートメントに WHERE 節が指定されていない場合に、「W」が入ります。
sqlwarn5	CHAR(1)	SQL ステートメントの実行中にエラーが許容された場合、「E」が入ります。
sqlwarn6	CHAR(1)	存在しない日付を避けるために日付演算の結果が調整された場合に「W」が入ります。
sqlwarn7	CHAR(1)	将来の使用のために予約済み。 CONNECT が呼び出されて正常に実行された場合に、**dyn_query_mgmt** データベース構成パラメーターが使用可能であれば、'E' が入ります。

表 49. SQLCA のフィールド (続き): この表によって示されているフィールド名は、INCLUDE ステートメントによって得られる SQLCA のものです。

名前	データ・タイプ	フィールドの値
sqlwarn8	CHAR(1)	変換できなかった文字が置換文字に置き換えられた場合に「W」になります。トラステッド接続の確立の試行に失敗した場合に「Y」になります。
sqlwarn9	CHAR(1)	エラーのある算術式が列関数の処理時に無視された場合に「W」になります。
sqlwarn10	CHAR(1)	SQLCA 内のフィールドのいずれかで、文字データ値の変換時に変換エラーが起きた場合、「W」が入ります。
sqlstate	CHAR(5)	直前に実行された SQL ステートメントの結果を示す戻りコード。

[a] 一部の関数では、NULL 値が除去されても SQLWARN2 が W にならないことがあります。これは、結果が NULL 値の除去によるものでない場合に生じます。

エラー報告

エラー報告の順序は、以下のとおりです。

1. 重大エラー条件は必ず報告されます。重大エラーが報告される場合、SQLCA に追加されるものはありません。
2. 重大エラーが発生しなかった場合、デッドロック・エラーはその他のエラーよりも優先します。
3. その他のエラーの場合、最初の負の SQL コードの SQLCA が戻されます。
4. 負の SQL コードが検出されない場合、最初の警告の SQLCA (つまり正の SQL コード) が戻されます。

   パーティション・データベース・システムで、あるデータベース・パーティションでは空で、他のデータベース・パーティションにはデータがある表に対してデータ操作コマンドが呼び出される場合は、この規則の例外です。すべてのデータベース・パーティションで表が空であるため、または UPDATE ステートメントの WHERE 節の条件を満たす行がもうないために、すべてのデータベース・パーティションのエージェントが SQL0100W を返した場合のみ、アプリケーションに SQLCODE +100 が返されます。

パーティション・データベース・システムでの SQLCA の使用法

パーティション・データベース・システムでは、異なるデータベース・パーティションにある多くのエージェントが 1 つの SQL ステートメントを実行する場合があり、それぞれのエージェントが異なるエラーまたは警告についての異なる SQLCA を戻す場合があります。また、コーディネーターのエージェントには独自の SQLCA があります。

アプリケーションについての表示に一貫性を持たせるために、 SQLCA の値はすべて 1 つの構造の中に組み合わされ、 SQLCA のフィールドは以下のようにグローバルなカウントを表示します。

- エラーと警告のすべてについて、*sqlwarn* フィールドにはすべてのエージェントから受け取った警告標識が入ります。
- 行カウントを表示する *sqlerrd* フィールドの値は、すべてのエージェントからの累計です。

トリガーにより実行された SQL ステートメントの処理中に発生したエラーのケースによっては、 SQLSTATE 09000 が戻されないことがあるので注意してください。

付録 C. SQLDA (SQL 記述子域)

SQLDA は、SQL DESCRIBE ステートメントの実行に必要な変数の集まりです。 SQLDA 変数は、PREPARE、OPEN、FETCH、および EXECUTE ステートメントでは、オプションとして使用できます。 SQLDA は、動的 SQL との間で情報を伝えます。これは、DESCRIBE ステートメントで使用され、ホスト変数のアドレスで変更して、 FETCH または EXECUTE ステートメントで再び使用することができます。

SQLDA は、すべての言語でサポートされていますが、事前定義の宣言を使用できるのは、 C、REXX、FORTRAN、および COBOL に対してだけです。

SQLDA の情報の意味は、その使用法によって異なります。 PREPARE と DESCRIBE の場合、 SQLDA は準備されるステートメントに関する情報をアプリケーション・プログラムに提供します。 OPEN、EXECUTE、および FETCH の場合、SQLDA はホスト変数を記述します。

DESCRIBE および PREPARE において、記述する列のいずれかが LOB タイプ (LOB ロケーターおよびファイル参照変数では、 2 倍の SQLDA は必要ありません)、参照タイプ、またはユーザー定義タイプの場合、 SQLDA 全体の SQLVAR 項目の数を 2 倍にする必要があります。以下に例を示します。

- 3 つの VARCHAR の列と 1 つの INTEGER の列の入っている表を記述する場合、 SQLVAR 項目は 4 つになります。
- 2 つの VARCHAR の列と 1 つの CLOB の列、および 1 つの INTEGER の列の入っている表を記述する場合には、 SQLVAR 項目は 8 つになります。

EXECUTE、FETCH、および OPEN において、記述する変数のいずれかが LOB タイプ (LOB ロケーターおよびファイル参照変数では、 2 倍の SQLDA は必要ありません) または構造化タイプの場合、 SQLDA 全体の SQLVAR 項目の数を 2 倍にする必要があります。特殊タイプと参照タイプは、これらの場合には関係ありません。これは、それらのタイプの場合、データベースが 2 倍の数の項目の追加情報を必要とはしないためです。)

SQLDA フィールド記述

SQLDA は、4 つの変数と、その後に SQLVAR と総称して呼ばれる変数の任意の数のオカレンスによって構成されています。 OPEN、FETCH、および EXECUTE では、 SQLVAR の各オカレンスによってホスト変数が記述されます。 DESCRIBE と PREPARE では、 SQLVAR の各オカレンスによって結果表またはパラメーター・マーカーの列が記述されます。 SQLVAR の項目には、以下の 2 つのタイプがあります。

- **基本 SQLVAR:** これらの項目は常に存在します。これらの項目には、データ・タイプのコード、長さ属性、列名、ホスト変数のアドレス、および標識変数アドレスなどの列、パラメーター・マーカー、またはホスト変数に関する基本的な情報が入れられます。

© Copyright IBM Corp. 1993, 2010

- **2次 SQLVAR:** これらの項目は、上記で概説した規則に従って SQLVAR 項目の数が 2 倍になった場合にのみ存在します。ユーザー定義タイプ (特殊または構造) の場合は、ユーザー定義タイプ名が入ります。参照タイプの場合は、参照のターゲット・タイプが入れられます。 LOB の場合は、ホスト変数の長さ属性と、実際の長さの入っているバッファーを指すポインターが入れられます。 (特殊タイプと LOB の情報が重なり合うことはないので、 DESCRIBE においては、SQLVAR 項目を 3 倍にしなくても、LOB に基づく特殊タイプを使用できます。) ロケーターまたはファイル参照変数を使用して LOB を示す場合、これらの項目は必要ありません。

SQLDA に上記 2 つのタイプの項目が両方とも入っている場合、基本 SQLVAR は 2次 SQLVAR のブロックの前のブロックに入れられます。それぞれの場合において、項目の数は SQLD の値で示されます (2次 SQLVAR 項目の多くは使用されない場合があります)。

DESCRIBE によって SQLVAR 項目が設定される環境については、 700 ページの『SQLDA に対する DESCRIBE の効果』に示されています。

SQLDA ヘッダーのフィールド

表 50. SQLDA ヘッダーのフィールド

C での名前	SQL データ・タイプ	DESCRIBE および PREPARE で使用する場合 (SQLN を除き、データベース・マネージャーで設定)	FETCH、OPEN、および EXECUTE で使用する場合 (ステートメントの実行前にアプリケーションで設定)
sqldaid	CHAR(8)	このフィールドの 7 番目のバイトは、SQLDOUBLED という名前のフラグ・バイトです。データベース・マネージャーは、それぞれの列に対して 2 つの SQLVAR 項目が作成された場合は SQLDOUBLED を文字 '2' に設定し、その他の場合はブランク (ASCII では X'20'、 EBCDIC では X'40' に設定します)。 SQLDOUBLED がいつ設定されるかについては、 700 ページの『SQLDA に対する DESCRIBE の効果』を参照してください。	このフィールドの 7 番目のバイトは、SQLVAR の数が 2 倍になった場合に使用されます。これは SQLDOUBLED という名前のフィールドです。記述されるホスト変数が構造化タイプ、BLOB、CLOB、または DBCLOB の場合、この 7 番目のバイトは文字 '2' に設定され、それ以外の場合は任意の文字に設定できます (ブランクの使用をお勧めします)。
sqldabc	INTEGER	32 ビットの場合、SQLDA の長さ = SQLN*44+16。 16。 64 ビットの場合、SQLDA の長さ = SQLN*56+16。	32 ビットの場合、SQLDA の長さ >= SQLN*44+16。 64 ビットの場合、SQLDA の長さ >= SQLN*56+16。
sqln	SMALLINT	データベース・マネージャーはこれを変更しません。 DESCRIBE ステートメントを実行する前に、ゼロまたはゼロより大きい値を設定する必要があります。 これは、SQLVAR のオカレンスの合計数を示します。	SQLDA の SQLVAR のオカレンスの合計。 SQLN には、ゼロまたはゼロより大きい値を設定する必要があります。
sqld	SMALLINT	データベース・マネージャーによって、結果表の列の数またはパラメーター・マーカーの数に設定されます。	SQLVAR のオカレンスにより記述されるホスト変数の数

基本 SQLVAR のオカレンスのフィールド

表 51. 基本 SQLVAR のフィールド

名前	データ・タイプ	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
sqltype	SMALLINT	列のデータ・タイプと、その列またはパラメーター・マーカーが NULL 可能能かどうかを示します。(パラメーター・マーカーは常に NULL 可能と見なされます。) 702 ページの表 53 は、許される値とその意味をリストしています。 特殊タイプまたは参照タイプの場合は、その基本タイプのデータ・タイプがこのフィールドに入れられます。構造化タイプの場合は、そのタイプの変換グループ (CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターに基づく) の FROM SQL 変換関数が入れられます。基本 SQLVAR には、それがユーザー定義タイプまたは参照タイプの記述の一部であるかどうかを示す標識はありません。	ホスト変数の場合と同じ。日付/時刻の値のホスト変数は、文字ストリング変数でなければなりません。 FETCH の場合、日付/時刻のタイプ・コードは、固定長文字ストリングを意味します。 sqltype が偶数値の場合、sqlind フィールドは無視されます。
sqllen	SMALLINT	列またはパラメーター・マーカーの長さ属性。日付/時刻の列またはパラメーター・マーカーの場合は、値のストリング表記の長さ。 702 ページの表 53 を参照してください。 ラージ・オブジェクト・ストリングの場合、この値は 0 に設定されます。その長さ属性が 2 バイト整数に入る小さいものであっても、設定値はやはり 0 になります。	ホスト変数の長さ属性。 702 ページの表 53 を参照してください。 CLOB、DBCLOB、および BLOB の列の場合、データベース・マネージャーはこの値を無視します。代わりに、2次 SQLVAR の len.sqllonglen フィールドが使用されます。
sqldata	ポインター	ストリング SQLVARS の場合、 sqldata にはコード・ページが組み入れられます。文字ストリング SQLVAR の場合、 FOR BIT DATA 属性で列を定義すると、sqldata は 0 になります。ほかの文字ストリング SQLVARS の場合、 sqldata には、 SBCS データの SBCS コード・ページか、 MBCS データの複合 MBCS コード・ページに関連付けられた SBCS コード・ページのいずれかが入っています。日本語の EUC、中国語 (繁体字) の EUC、および Unicode の UTF-8 文字ストリング SQLVARS では、 sqldata にそれぞれ 954、964、および 1208 が組み入れられます。 他のすべてのタイプの列の場合、sqldata は未定義です。	ホスト変数のアドレスが入っています (取り出したデータを保管するロケーション)。

表 51. 基本 SQLVAR のフィールド (続き)

名前	データ・タイプ	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
sqlind	ポインター	文字ストリング SQLVARS の場合、sqlind は 0 になります。ただし、sqlind が複合 DBCS コード・ページに関連付けられた DBCS コード・ページの場合の、 MBCS データは例外です。 他のすべてのタイプの場合、sqlind は未定義です。	関連する標識変数があれば、そのアドレスが入ります。それ以外の場合は、使用されません。 sqltype が偶数値の場合、sqlind フィールドは無視されます。
sqlname	VARCHAR (30)	非修飾の列名またはパラメーター・マーカー名。 システム生成の名前を持つ列およびパラメーター・マーカーの場合、 30 番目のバイトが X'FF' に設定されます。 AS 節によって列名が指定された場合は、このバイトは X'00' になります。	ホスト・データベースに接続する場合、FOR BIT DATA ストリングを指定するには、sqlname を以下のように設定します。 • SQLDA ヘッダー内の SQLDAID の 6 番目のバイトを '+' に設定します。 • sqlname の長さを 8 にします。 • sqlname の最初の 2 バイトを X'0000' にします。 • sqlname の 3 番目と 4 番目のバイトを X'0000' にします。 • sqlname の残りの 4 バイトは予約済みであり、X'00000000' に設定します。 XML データを処理する場合、XML サブタイプを指定するには、sqlname を以下のように設定します。 • sqlname の長さを 8 にします。 • sqlname の最初の 2 バイトを X'0000' にします。 • sqlname の 3 番目と 4 番目のバイトを X'0000' にします。 • sqlname の 5 番目のバイトを X'01' にします。 • sqlname の残りの 3 バイトは予約済みであり、X'000000' に設定します。

2次 SQLVAR のオカレンスのフィールド

表 52. 2次 SQLVAR のフィールド

名前	データ・タイプ	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
len.sqllonglen	INTEGER	BLOB、CLOB、または DBCLOB の列またはパラメーター・マーカーの長さ属性。	BLOB、CLOB、または DBCLOB ホスト変数の長さ属性。データベース・マネージャーは、それらのデータ・タイプに対しては基本 SQLVAR の SQLLEN フィールドを無視します。長さ属性は、BLOB または CLOB ではバイト数、DBCLOB では 2 バイト文字の数になります。
reserve2	32 ビットの場合は CHAR(3)、64 ビットの場合は CHAR(11)。	使用されません。	使用されません。
sqlflag4	CHAR(1)	SQLVAR の表している参照タイプが sqldatatype_name に指定されたターゲット・タイプに関連付けられたものである場合、この値は X'01' になります。SQLVAR の表している構造化タイプで、sqldatatype_name にユーザー定義タイプ名が指定されている場合、値は X'12' になります。それ以外の場合は、値は X'00' です。	SQLVAR の表している参照タイプが sqldatatype_name に指定されたターゲット・タイプに関連付けられたものである場合、 X'01' に設定されます。SQLVAR の表している構造化タイプで、 sqldatatype_name にユーザー定義タイプ名が指定されている場合、 X'12' に設定されます。それ以外の場合は、値は X'00' です。
sqldatalen	ポインター	使用されません。	BLOB、CLOB、および DBCLOB ホスト変数でのみ使用されます。 このフィールドが NULL の場合は、データの直前に実際の長さ (2 バイト文字単位) を 4 バイトで保管し、SQLDATA はフィールド長の最初のバイトを指すようにする必要があります。 このフィールドが NULL (NULL 値) でない場合は、対応する基本 SQLVAR 内の SQLDATA フィールドの指すバッファー内のデータの実際の長さ (バイト単位、 DBCLOB の場合も含む) の入っている 4 バイト長のバッファーを指すポインターが入れられます。 このフィールドを使用するか否かに関係なく、 len.sqllonglen フィールドは設定する必要があります。

表 52. 2次 SQLVAR のフィールド (続き)

名前	データ・タイプ	DESCRIBE および PREPARE で使用する場合	FETCH、OPEN、および EXECUTE で使用する場合
sqldatatype_name	VARCHAR(27)	ユーザー定義タイプの場合、データベース・マネージャーはこれを完全修飾ユーザー定義タイプ名に設定します。[注 1] 参照タイプの場合は、データベース・マネージャーはこれを参照のターゲット・タイプの完全修飾タイプ名に設定します。	構造化タイプの場合、表の注[1] で示されているフォーマットの完全修飾ユーザー定義タイプ名に設定されます。
予約済み	CHAR(3)	使用されません。	使用されません。

[1] 最初の 8 バイトには、タイプのスキーマ名が入れられます (必要に応じて右側にスペースが入れられます)。バイト 9 はドット文字 (.) です。. 10 から 27 バイトには、タイプ名のうちの下位部分が入れられます。それは、右側にスペースを入れて拡張することはできません。

このフィールドの基本的な目的は、ユーザー定義型の名前を入れることですが、IBM の定義済みデータ・タイプ用に設定することもできます。この場合、スキーマ名は SYSIBM、名前の下位部分は DATATYPES カタログ・ビューの TYPENAME 列に保管されている名前になります。以下に例を示します。

```
type name         length   sqldatatype_name
---------         ------   ----------------
A.B               10       A       .B
INTEGER           16       SYSIBM  .INTEGER
"Frank's".SMINT   13       Frank's .SMINT
MY."type  "       15       MY      .type
```

SQLDA に対する DESCRIBE の効果

DESCRIBE OUTPUT または PREPARE OUTPUT INTO ステートメントの場合、データベース・マネージャーは、常に SQLD を結果セットの列の数、または出力パラメーター・マーカーの数に設定します。 DESCRIBE INPUT または PREPARE INPUT INTO ステートメントの場合、データベース・マネージャーは、常に SQLD をステートメント内の入力パラメーター・マーカーの数に設定します。 CALL ステートメント内の INOUT パラメーターに対応するパラメーター・マーカーは、入力記述子と出力記述子の両方で記述されるので注意してください。

SQLDA の SQLVAR は、以下の場合に設定されます。

- SQLN >= SQLD で、しかも LOB、ユーザー定義タイプ、または参照タイプの項目がない

  最初の SQLD SQLVAR 項目が設定され、SQLDOUBLED はブランクに設定されます。

- SQLN >= 2*SQLD で、しかも少なくとも 1 つの項目が LOB、ユーザー定義タイプ、または参照タイプである

  2 倍の数の SQLD SQLVAR 項目が設定され、SQLDOUBLED は '2' に設定されます。

- SQLD <= SQLN < 2*SQLD で、しかも少なくとも 1 つの項目が特殊タイプまたは参照タイプで、 LOB の項目または構造化タイプの項目はない

最初の SQLD SQLVAR 項目が設定され、SQLDOUBLED はブランクに設定されます。 SQLWARN BIND オプションが YES の場合は、警告 SQLCODE +237 (SQLSTATE 01594) が出されます。

SQLDA の SQLVAR は、以下の場合には設定されません (さらに多くのスペースの割り振りと別の DESCRIBE が必要)。

- SQLN < SQLD で、しかも LOB、ユーザー定義タイプ、または参照タイプの項目がない

  SQLVAR 項目は設定されず、SQLDOUBLED はブランクに設定されます。 SQLWARN BIND オプションが YES の場合は、警告 SQLCODE +236 (SQLSTATE 01005) が出されます。

  DESCRIBE が正常に実行される場合には、SQLD 個の SQLVAR が割り振られます。
- SQLN < SQLD で、しかも少なくとも 1 つの項目が特殊タイプまたは参照タイプで、 LOB の項目または構造化タイプの項目はない

  SQLVAR 項目は設定されず、SQLDOUBLED はブランクに設定されます。 SQLWARN BIND オプションが YES の場合は、警告 SQLCODE +239 (SQLSTATE 01005) が出されます。

  特殊タイプ名および参照タイプのターゲット・タイプを組み込まれた DESCRIBE が正常に実行されると、 2*SQLD 個の SQLVAR が割り振られます。
- SQLN < 2*SQLD で、しかも少なくとも 1 つの項目が LOB または構造化タイプである

  SQLVAR 項目は設定されず、SQLDOUBLED はブランクに設定されます。 (SQLWARN BIND オプションの設定に関係なく) 警告 SQLCODE +238 (SQLSTATE 01005) が出されます。

  DESCRIBE が正常に実行される場合には、2*SQLD 個の SQLVAR が割り振られます。

上記リストでの「LOB 項目」には、ソース・タイプが LOB タイプである特殊タイプの項目も入ります。

DESCRIBE (または PREPARE INTO) から警告 SQLCODE +236、+237、+239 を戻すかどうかを制御するには、 BIND または PREP コマンドの SQLWARN オプションを使用します。 使用するアプリケーション・コードでは、これらの SQLCODE がいつ戻されてもよいようにしておいてください。選択リストに LOB または構造化タイプの項目が入っていて、 SQLDA の中の SQLVAR が不足している場合には、常に警告 SQLCODE +238 が戻されます。これは、結果セット内に LOB または構造化タイプの項目があるために SQLVAR 数を 2 倍にする必要があることをアプリケーションに認識させる唯一の方法です。

構造化タイプの項目を記述しようとしているものの、FROM SQL トランスフォームが定義されていない場合 (CURRENT DEFAULT TRANSFORM GROUP 特殊レジスターを使用した TRANSFORM GROUP の指定が行われていない (SQLSTATE 42741) か、またはその名前グループが FROM SQL トランスフォーム関数を定義し

ていない (SQLSTATE 42744) ため))、DESCRIBE はエラーを戻します。このエラーは、構造化タイプの項目がある表の DESCRIBE で戻されるエラーと同じです。

データベース・マネージャーが SQLDA に保管できるよりも長い ID を戻す場合、ID は切り捨てられ、警告が戻されます (SQLSTATE 01665)。しかし、構造化タイプの名前が切り捨てられるときには、エラーが戻されます (SQLSTATE 42622)。ID の長さの制限については、『SQL と XQuery の制限値』を参照してください。

SQLTYPE と SQLLEN

表 53 に、 SQLDA の SQLTYPE フィールドと SQLLEN フィールドに現れる値を示します。 DESCRIBE と PREPARE INTO においては、 SQLTYPE の値が偶数ならその列では NULL 値が使えないこと、また奇数ならその列で NULL 値が可能であることを意味しています。 FETCH、OPEN、および EXECUTE において、SQLTYPE の値が偶数の場合には標識変数がないこと、奇数の場合には SQLIND に標識変数のアドレスが入れられていることを意味しています。

表 53. SQLTYPE と SQLLEN の値 (DESCRIBE、FETCH、OPEN、および EXECUTE の場合)

	DESCRIBE および PREPARE INTO の場合		**FETCH、OPEN、および EXECUTE の場合**	
SQLTYPE	**列のデータ・タイプ**	**SQLLEN**	**ホスト変数のデータ・タイプ**	**SQLLEN**
384/385	日付	10	日付の固定長文字ストリング表示	ホスト変数の長さ属性
388/389	時刻	8	時刻の固定長文字ストリング表示	ホスト変数の長さ属性
392/393	タイム・スタンプ	26	タイム・スタンプの固定長文字ストリング表示	ホスト変数の長さ属性
400/401	N/A	N/A	NULL 終了 GRAPHIC ストリング	ホスト変数の長さ属性
404/405	BLOB	0 *	BLOB	使用されません。 *
408/409	CLOB	0 *	CLOB	使用されません。 *
412/413	DBCLOB	0 *	DBCLOB	使用されません。 *
448/449	可変長文字ストリング	列の長さ属性	可変長文字ストリング	ホスト変数の長さ属性
452/453	固定長文字ストリング	列の長さ属性	固定長文字ストリング	ホスト変数の長さ属性
456/457	長形式可変長文字ストリング	列の長さ属性	長形式可変長文字ストリング	ホスト変数の長さ属性
460/461	該当なし	該当なし	NULL 終了文字ストリング	ホスト変数の長さ属性
464/465	可変長 GRAPHIC ストリング	列の長さ属性	可変長 GRAPHIC ストリング	ホスト変数の長さ属性
468/469	固定長 GRAPHIC ストリング	列の長さ属性	固定長 GRAPHIC ストリング	ホスト変数の長さ属性
472/473	長形式可変長 GRAPHIC ストリング	列の長さ属性	ロング GRAPHIC ストリング	ホスト変数の長さ属性
480/481	浮動小数点	倍精度の場合は 8、単精度の場合は 4	浮動小数点	倍精度の場合は 8、単精度の場合は 4

表 53. SQLTYPE と SQLLEN の値 (DESCRIBE、FETCH、OPEN、および EXECUTE の場合) (続き)

	DESCRIBE および PREPARE INTO の場合		FETCH、OPEN、および EXECUTE の場合	
SQLTYPE	列のデータ・タイプ	SQLLEN	ホスト変数のデータ・タイプ	SQLLEN
484/485	パック 10 進数	バイト 1 は精度、バイト 2 は位取り	パック 10 進数	バイト 1 は精度、バイト 2 は位取り
492/493	64 ビット整数	8	64 ビット整数	8
496/497	長精度整数	4	長精度整数	4
500/501	短精度整数	2	短精度整数	2
916/917	該当なし	該当なし	BLOB ファイル参照変数	267
920/921	該当なし	該当なし	CLOB ファイル参照変数	267
924/925	該当なし	該当なし	DBCLOB ファイル参照変数	267
960/961	該当なし	該当なし	BLOB ロケーター	4
964/965	該当なし	該当なし	CLOB ロケーター	4
968/969	該当なし	該当なし	DBCLOB ロケーター	4
988/989	XML	0	該当なし。代わりに、XML AS <ストリングまたはバイナリーの LOB タイプ> ホスト変数を使用します。	使用されません。
996	10 進浮動小数点数	DECFLOAT(16) の場合 8、DECFLOAT(34) の場合 16	10 進浮動小数点数	DECFLOAT(16) の場合 8、DECFLOAT(34) の場合 16

注:

- 2次 SQLVAR の len.sqllonglen フィールドに、列の長さ属性が入れられます。
- SQLTYPE は、DB2 での移植性のために旧バージョンから変更されました。旧バージョンの値 (旧バージョンの SQL リファレンスを参照) は、引き続きサポートされています。

認識されない非サポート SQLTYPE

SQLDA の SQLTYPE フィールドに表示される値は、データの送信側および受信側で使用可能なデータ・タイプ・サポートのレベルによって異なります。 これは、新しいデータ・タイプが製品に追加される場合に特に重要です。

新しいデータ・タイプは、データの送信側または受信側にサポートされることもあれば、サポートされないこともあり、データの送信側や受信側に認識されないことさえあります。 状況に応じて、新しいデータ・タイプが戻されたり、送信側と受信側の両方が認めた互換データ・タイプが戻されたり、あるいは結果としてエラーが発生したりします。

送信側と受信局が互換データ・タイプの使用に同意する場合、以下に示すマッピングが実行されます。このマッピングは、送信側または受信側の少なくともどちらか

が指定データ・タイプをサポートしない場合に実行されます。 非サポート・データ・タイプは、アプリケーションまたはデータベース・マネージャーのどちらかによって提供されます。

データ・タイプ	互換データ・タイプ
BIGINT	DECIMAL(19, 0)
ROWID[1]	VARCHAR(40) FOR BIT DATA

[1] ROWID は、DB2 Universal Database for z/OS バージョン 8 によってサポートされています。

SQLDA では、データ・タイプが置換されたことは示されないので注意してください。

パック 10 進数

パック 10 進数は、一種のバイナリー・コードによる 10 進数 (BCD) 表記で保管されます。 BCD においては、1 ニブル (4 ビット) で 10 進数の 1 桁が表されます。たとえば、0001 0111 1001 は 179 を表します。したがって、パック 10 進数の値はニブルごとに読む必要があります。 値の保管はバイト単位で行い、16 進数表記としてそれらのバイトを読み、それを 10 進数に戻します。たとえば、0001 0111 1001 は、バイナリー表記では 00000001 01111001 となります。 この数値を 16 進数として読むと、0179 になります。

小数点は、位取りによって決まります。 たとえば、DEC(12,5) の列の場合、小数点より右側に 5 桁あることになります。

符号は、桁数を表すニブルの右側のニブルで示します。 正符号または負符号は、以下のように示します。

表 54. パック 10 進数の符号標識の値

	表記		
符号	**バイナリー**	**10 進**	**16 進**
正符号 (+)	1100	12	C
負符号 (-)	1101	13	D

まとめ:

- 値を保管するためには、*p*/2+1 バイトを割り振ります。*p* は精度です。
- 値を表すために、ニブルを左から右へ割り当てます。 数値の精度が偶数の場合は、最初にゼロのニブルを追加します。この割り当てには、先行 (無効な) ゼロと後続 (有効な) ゼロの桁が入ります。
- 符号ニブルは、最後のバイトの第 2 ニブルになります。

以下に例を示します。

列	値	バイトごとにグループにした 16 進のニブル
DEC(8,3)	6574.23	00 65 74 23 0C
DEC(6,2)	-334.02	00 33 40 2D
DEC(7,5)	5.2323	05 23 23 0C

列	値 バイトごとにグループにした 16 進のニブル
DEC(5,2)	-23.5 02 35 0D

10 進数の SQLLEN フィールド

SQLLEN フィールドには、 10 進数の列の精度 (第 1 バイト) と位取り (第 2 バイト) が入れられます。アプリケーションを移植可能にするには、精度のバイトと位取りのバイトを短精度整数として一度に設定するのではなく、個々に設定するようにしてください。これによって、整数のバイト反転の問題が回避されます。

たとえば、C の場合には以下のようにします。

```
((char *)&(sqlda->sqlvar[i].sqllen))[0] = precision;
((char *)&(sqlda->sqlvar[i].sqllen))[1] = scale;
```

付録 D. システム・カタログ・ビュー

データベース・マネージャーは、基本システム・カタログ表の上に定義される 2 組のシステム・カタログのビューを作成し、保守しています。

- SYSCAT ビューは、SYSCAT スキーマに存在する読み取り専用のカタログ・ビューです。これらのビューに対する SELECT 特権は、デフォルトで PUBLIC に付与されています。
- SYSSTAT ビューは、SYSSTAT スキーマに存在する更新可能なカタログ・ビューです。更新可能なビューには、オプティマイザーで使用される統計情報が入れられます。これらのビューのいくつかの列内の値を変更して、パフォーマンスをテストすることができます。 (統計を変更する前に、RUNSTATS コマンドを呼び出してすべての統計が現行状態を反映するようにすることをお勧めします。)

アプリケーションを作成する際は、基本のカタログ表ではなく、SYSCAT および SYSSTAT ビューを対象とする必要があります。

すべてのシステム・カタログ・ビューは、データベースの作成時に作成されます。カタログ・ビューは、明示的に作成またはドロップすることはできません。Unicode データベースでは、カタログ・ビューは IDENTITY 照合で作成されます。非 Unicode データベースでは、カタログ・ビューはデータベース照合で作成されます。ビューは、SQL データ定義ステートメント、環境ルーチン、および特定のユーティリティーに対応する通常の操作の過程で更新されます。システム・カタログ・ビューのデータは、通常の SQL 照会機能によって使用可能です。システム・カタログ・ビューは (一部の更新可能なカタログ・ビューを除いて)、通常の SQL データ操作コマンドを使用して変更することはできません。

ユーザーの更新可能カタログ・ビューにオブジェクト (表、列、関数、または索引) が現れるのは、そのユーザーがそれを作成した場合か、そのユーザーにそのオブジェクトに対する CONTROL 特権、または明示的な DBADM 権限が与えられている場合だけです。このことは、SYSADM 権限を持ち、暗黙的に DBADM 権限を持つユーザーにも適用されます。 SYSADM ユーザーが自身の所有しないオブジェクトに関するすべての SYSSTAT ビュー内のレコードを表示するには、そのユーザーは該当オブジェクトに対する CONTROL 特権を明示的に付与されるか、DBADM 権限を明示的に付与される必要があります。

ビュー内での列の順序はリリースによって変更されることがあります。これによりプログラミング・ロジックが影響を受けないようにするためには、選択リスト内で列を明示的に指定し、`SELECT *` は使用しないようにします。列は、記述するオブジェクトのタイプに基づいて、一貫性のある名前を持ちます。

記述されるオブジェクト	列名
表	TABSCHEMA, TABNAME
索引	INDSCHEMA, INDNAME
ビュー	VIEWSCHEMA, VIEWNAME

© Copyright IBM Corp. 1993, 2010

制約 CONSTSCHEMA, CONSTNAME

トリガー
TRIGSCHEMA, TRIGNAME

パッケージ
PKGSCHEMA, PKGNAME

タイプ TYPESCHEMA, TYPENAME, TYPEID

関数 ROUTINESCHEMA, ROUTINENAME, ROUTINEID

メソッド
ROUTINESCHEMA, ROUTINENAME, ROUTINEID

プロシージャー
ROUTINESCHEMA, ROUTINENAME, ROUTINEID

列 COLNAME

スキーマ
SCHEMANAME

表スペース
TBSPACE

データベース・パーティション・グループ
DPGNAME

バッファー・プール
BPNAME

イベント・モニター
EVMONNAME

タイム・スタンプの作成
CREATE_TIME

カタログ・ビューのロードマップ

表 55. 読み取り専用のカタログ・ビューのロードマップ

説明	カタログ・ビュー
構造化データ・タイプの属性	714 ページの『SYSCAT.ATTRIBUTES』
監査ポリシー	716 ページの『SYSCAT.AUDITPOLICIES』
	718 ページの『SYSCAT.AUDITUSE』
データベースに対する権限	747 ページの『SYSCAT.DBAUTH』
データベース・パーティション・グループのバッファー・プールの構成	720 ページの『SYSCAT.BUFFERPOOLS』
データベース・パーティションのバッファー・プールのサイズ	719 ページの『SYSCAT.BUFFERPOOLDBPARTITIONS』
cast 関数	721 ページの『SYSCAT.CASTFUNCTIONS』
チェック制約	722 ページの『SYSCAT.CHECKS』
列の特権	723 ページの『SYSCAT.COLAUTH』
列	732 ページの『SYSCAT.COLUMNS』
チェック制約によって参照される列	724 ページの『SYSCAT.COLCHECKS』
ディメンションで使用される列	737 ページの『SYSCAT.COLUSE』
キーで使用される列	779 ページの『SYSCAT.KEYCOLUSE』
制約の従属関係	738 ページの『SYSCAT.CONSTDEP』
データベース・パーティション・グループのデータベース・パーティション	749 ページの『SYSCAT.DBPARTITIONGROUPDEF』
データベース・パーティション・グループの定義	750 ページの『SYSCAT.DBPARTITIONGROUPS』
データ・パーティション	741 ページの『SYSCAT.DATAPARTITIONEXPRESSION』
	742 ページの『SYSCAT.DATAPARTITIONS』
データ・タイプ	744 ページの『SYSCAT.DATATYPES』
列グループ統計値の詳細	726 ページの『SYSCAT.COLGROUPCOLS』
	727 ページの『SYSCAT.COLGROUPDIST』
	728 ページの『SYSCAT.COLGROUPDISTCOUNTS』
	729 ページの『SYSCAT.COLGROUPS』
列オプションの詳細	731 ページの『SYSCAT.COLOPTIONS』
列統計値の詳細	725 ページの『SYSCAT.COLDIST』
分散マップ	794 ページの『SYSCAT.PARTITIONMAPS』
イベント・モニターの定義	751 ページの『SYSCAT.EVENTMONITORS』
現在モニター中のイベント	753 ページの『SYSCAT.EVENTS』
	754 ページの『SYSCAT.EVENTTABLES』
関数の従属関係[1]	802 ページの『SYSCAT.ROUTINEDEP』
関数マッピング	758 ページの『SYSCAT.FUNCMAPPINGS』
関数マッピングのオプション	756 ページの『SYSCAT.FUNCMAPOPTIONS』
関数パラメーター・マッピングのオプション	757 ページの『SYSCAT.FUNCMAPPARMOPTIONS』
関数パラメーター[1]	805 ページの『SYSCAT.ROUTINEPARMS』
関数[1]	807 ページの『SYSCAT.ROUTINES』

表 55. 読み取り専用のカタログ・ビューのロードマップ (続き)

説明	カタログ・ビュー
グローバル変数	863 ページの『SYSCAT.VARIABLEAUTH』
	864 ページの『SYSCAT.VARIABLEDEP』
	865 ページの『SYSCAT.VARIABLES』
階層 (タイプ、表、ビュー)	759 ページの『SYSCAT.HIERARCHIES』
	755 ページの『SYSCAT.FULLHIERARCHIES』
ID 列	730 ページの『SYSCAT.COLIDENTATTRIBUTES』
索引列	764 ページの『SYSCAT.INDEXCOLUSE』
索引の従属関係	765 ページの『SYSCAT.INDEXDEP』
索引活用	772 ページの『SYSCAT.INDEXEXPLOITRULES』
索引拡張従属関係	773 ページの『SYSCAT.INDEXEXTENSIONDEP』
索引拡張パラメーター	775 ページの『SYSCAT.INDEXEXTENSIONPARMS』
索引拡張検索メソッド	774 ページの『SYSCAT.INDEXEXTENSIONMETHODS』
索引拡張	776 ページの『SYSCAT.INDEXEXTENSIONS』
索引オプション	777 ページの『SYSCAT.INDEXOPTIONS』
索引特権	763 ページの『SYSCAT.INDEXAUTH』
索引	766 ページの『SYSCAT.INDEXES』
メソッドの従属関係[1]	802 ページの『SYSCAT.ROUTINEDEP』
メソッド・パラメーター[1]	807 ページの『SYSCAT.ROUTINES』
メソッド[1]	807 ページの『SYSCAT.ROUTINES』
ニックネーム	781 ページの『SYSCAT.NICKNAMES』
オブジェクト・マッピング	780 ページの『SYSCAT.NAMEMAPPINGS』
パッケージの従属関係	785 ページの『SYSCAT.PACKAGEDEP』
パッケージ特権	784 ページの『SYSCAT.PACKAGEAUTH』
パッケージ	787 ページの『SYSCAT.PACKAGES』
パーティション表	841 ページの『SYSCAT.TABDETACHEDDEP』
パススルー特権	795 ページの『SYSCAT.PASSTHRUAUTH』
述部指定	796 ページの『SYSCAT.PREDICATESPECS』
プロシージャーのオプション	803 ページの『SYSCAT.ROUTINEOPTIONS』
プロシージャーのパラメーター・オプション	804 ページの『SYSCAT.ROUTINEPARMOPTIONS』
プロシージャー・パラメーター[1]	805 ページの『SYSCAT.ROUTINEPARMS』
プロシージャー[1]	807 ページの『SYSCAT.ROUTINES』

表 55. 読み取り専用のカタログ・ビューのロードマップ (続き)

説明	カタログ・ビュー
保護表	820 ページの『SYSCAT.SECURITYLABELACCESS』
	821 ページの『SYSCAT.SECURITYLABELCOMPONENTELEMENTS』
	822 ページの『SYSCAT.SECURITYLABELCOMPONENTS』
	823 ページの『SYSCAT.SECURITYLABELS』
	824 ページの『SYSCAT.SECURITYPOLICIES』
	825 ページの『SYSCAT.SECURITYPOLICYCOMPONENTRULES』
	826 ページの『SYSCAT.SECURITYPOLICYEXEMPTIONS』
	835 ページの『SYSCAT.SURROGATEAUTHIDS』
DB2 for z/OS の互換性を提供する	886 ページの『SYSIBM.SYSDUMMY1』
参照制約	797 ページの『SYSCAT.REFERENCES』
リモート表オプション	851 ページの『SYSCAT.TABOPTIONS』
ロール	798 ページの『SYSCAT.ROLEAUTH』
	799 ページの『SYSCAT.ROLES』
ルーチンの従属関係	802 ページの『SYSCAT.ROUTINEDEP』
ルーチン・パラメーター[1]	805 ページの『SYSCAT.ROUTINEPARMS』
ルーチン特権	800 ページの『SYSCAT.ROUTINEAUTH』
ルーチン[1]	807 ページの『SYSCAT.ROUTINES』
	816 ページの『SYSCAT.ROUTINESFEDERATED』
スキーマ特権	818 ページの『SYSCAT.SCHEMAAUTH』
スキーマ	819 ページの『SYSCAT.SCHEMATA』
シーケンス特権	827 ページの『SYSCAT.SEQUENCEAUTH』
シーケンス	828 ページの『SYSCAT.SEQUENCES』
サーバー・オプション	830 ページの『SYSCAT.SERVEROPTIONS』
サーバー固有のユーザー・オプション	862 ページの『SYSCAT.USEROPTIONS』
パッケージ中のステートメント	834 ページの『SYSCAT.STATEMENTS』
ストアード・プロシージャー	807 ページの『SYSCAT.ROUTINES』
システム・サーバー	831 ページの『SYSCAT.SERVERS』
表の制約	838 ページの『SYSCAT.TABCONST』
表の従属関係	839 ページの『SYSCAT.TABDEP』
表特権	836 ページの『SYSCAT.TABAUTH』
表スペースの使用特権	852 ページの『SYSCAT.TBSPACEAUTH』
表スペース	849 ページの『SYSCAT.TABLESPACES』
表	842 ページの『SYSCAT.TABLES』
トランスフォーム	855 ページの『SYSCAT.TRANSFORMS』
トリガーの従属関係	856 ページの『SYSCAT.TRIGDEP』
トリガー	857 ページの『SYSCAT.TRIGGERS』

表 55. 読み取り専用のカタログ・ビューのロードマップ (続き)

説明	カタログ・ビュー
トラステッド・コンテキスト	739 ページの『SYSCAT.CONTEXTATTRIBUTES』
	740 ページの『SYSCAT.CONTEXTS』
タイプ・マッピング	859 ページの『SYSCAT.TYPEMAPPINGS』
ユーザー定義関数	807 ページの『SYSCAT.ROUTINES』
ビューの従属関係	839 ページの『SYSCAT.TABDEP』
ビュー	842 ページの『SYSCAT.TABLES』
	866 ページの『SYSCAT.VIEWS』
ワークロード管理	760 ページの『SYSCAT.HISTOGRAMTEMPLATEBINS』
	761 ページの『SYSCAT.HISTOGRAMTEMPLATES』
	762 ページの『SYSCAT.HISTOGRAMTEMPLATEUSE』
	832 ページの『SYSCAT.SERVICECLASSES』
	853 ページの『SYSCAT.THRESHOLDS』
	867 ページの『SYSCAT.WORKACTIONS』
	870 ページの『SYSCAT.WORKACTIONSETS』
	871 ページの『SYSCAT.WORKCLASSES』
	872 ページの『SYSCAT.WORKCLASSSETS』
	873 ページの『SYSCAT.WORKLOADAUTH』
	874 ページの『SYSCAT.WORKLOADCONNATTR』
	875 ページの『SYSCAT.WORKLOADS』
ラッパー・オプション	877 ページの『SYSCAT.WRAPOPTIONS』
ラッパー	878 ページの『SYSCAT.WRAPPERS』
XML 値の索引	778 ページの『SYSCAT.INDEXXMLPATTERNS』
XSR オブジェクト	879 ページの『SYSCAT.XDBMAPGRAPHS』
	880 ページの『SYSCAT.XDBMAPSHREDTREES』
	881 ページの『SYSCAT.XSROBJECTAUTH』
	882 ページの『SYSCAT.XSROBJECTCOMPONENTS』
	883 ページの『SYSCAT.XSROBJECTDEP』
	884 ページの『SYSCAT.XSROBJECTHIERARCHIES』
	885 ページの『SYSCAT.XSROBJECTS』

注 [1] DB2 バージョン 7.1 以前で定義されていた、以下に示す関数、メソッド、およびプロシージャーのカタログ・ビューは、引き続き有効です。

```
Functions:  SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Methods:    SYSCAT.FUNCTIONS, SYSCAT.FUNCDEP, SYSCAT.FUNCPARMS
Procedures: SYSCAT.PROCEDURES, SYSCAT.PROCPARMS
```

しかしこれらのビューは、DB2 バージョン 7.1 以後更新されていません。代わりに、SYSCAT.ROUTINES、SYSCAT.ROUTINEDEP、または SYSCAT.ROUTINEPARMS カタログ・ビューを使用してください。

表 56. 更新可能なカタログ・ビューのロードマップ

説明	カタログ・ビュー
列	891 ページの『SYSSTAT.COLUMNS』

表 56. 更新可能なカタログ・ビューのロードマップ (続き)

説明	カタログ・ビュー
列グループ統計値の詳細	888 ページの『SYSSTAT.COLGROUPDIST』
	889 ページの『SYSSTAT.COLGROUPDISTCOUNTS』
	890 ページの『SYSSTAT.COLGROUPS』
列統計値の詳細	887 ページの『SYSSTAT.COLDIST』
索引	892 ページの『SYSSTAT.INDEXES』
ルーチン[1]	896 ページの『SYSSTAT.ROUTINES』
表	897 ページの『SYSSTAT.TABLES』

注 [1] 関数およびメソッドの統計を更新するための、 SYSSTAT.FUNCTIONS カタログ・ビューがまだ存在します。しかし、このビューは DB2 バージョン 7.1 からの変更を反映していません。

SYSCAT.ATTRIBUTES

各行は、ユーザー定義の構造化データ・タイプに定義された属性を表します。サブタイプの継承された属性も含まれます。

表 57. SYSCAT.ATTRIBUTES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TYPESCHEMA	VARCHAR(128)		属性の入った構造化データ・タイプのスキーマ名。
TYPENAME	VARCHAR(128)		属性の入った構造化データ・タイプの非修飾名。
ATTR_NAME	VARCHAR(128)		属性名。
ATTR_TYPESCHEMA	VARCHAR(128)		属性のデータ・タイプのスキーマ名。
ATTR_TYPENAME	VARCHAR(128)		属性のデータ・タイプの非修飾名。
TARGET_TYPESCHEMA	VARCHAR(128)	Y	ターゲット行タイプのスキーマ名。参照タイプにのみ適用されます。それ以外の場合は NULL 値。
TARGET_TYPENAME	VARCHAR(128)	Y	ターゲット行タイプの非修飾名。参照タイプにのみ適用されます。それ以外の場合は NULL 値。
SOURCE_TYPESCHEMA	VARCHAR(128)		継承された属性の場合、属性を最初に定義した時のデータ・タイプのスキーマ名。非継承属性の場合、この列は TYPESCHEMA と同じです。
SOURCE_TYPENAME	VARCHAR(128)		継承された属性の場合、属性を最初に定義した時のデータ・タイプの非修飾名。非継承属性の場合、この列は TYPENAME と同じです。
ORDINAL	SMALLINT		構造化データ・タイプの定義における属性の位置 (0 から開始する)。
LENGTH	INTEGER		ストリング・タイプの場合、最大長が入ります。DECIMAL 型の場合、精度 (桁数) が入ります。その他の場合は、0。
SCALE	SMALLINT		DECIMAL 型の場合、位取り (小数点の右側の桁数) が入ります。その他の場合は、0。
CODEPAGE	SMALLINT		ストリング・タイプの場合、コード・ページを示します。0 は FOR BIT DATA を示します。ストリング・タイプでない場合は 0。
COLLATIONSCHEMA	VARCHAR (128)	Y	ストリング・タイプの場合は、属性の照合のスキーマ名。それ以外の場合は NULL 値。
COLLATIONNAME	VARCHAR(128)	Y	ストリング・タイプの場合は、属性の照合の非修飾名。それ以外の場合は NULL 値。
LOGGED	CHAR(1)		LOB 型のみに適用されます。その他の場合はブランク。 • N = 変更のログ記録を取らない • Y = 変更のログ記録を取る

表 57. SYSCAT.ATTRIBUTES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COMPACT	CHAR(1)		LOB 型のみに適用されます。その他の場合はブランク。 • N = 非短縮形式で保管される • Y = 短縮形式で保管される
DL_FEATURES	CHAR(10)		この列は使用されなくなりました。将来のリリースで除去されます。
JAVA_FIELDNAME	VARCHAR(256)	Y	将来の使用のために予約済み。

SYSCAT.AUDITPOLICIES

各行は、監査ポリシーを表します。

表 58. SYSCAT.AUDITPOLICIES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
AUDITPOLICYNAME	VARCHAR(128)		監査ポリシーの名前。
AUDITPOLICYID	INTEGER		監査ポリシーの ID。
CREATE_TIME	TIMESTAMP		監査ポリシーが作成された時刻。
ALTER_TIME	TIMESTAMP		監査ポリシーが最後に変更された時刻。
AUDITSTATUS	CHAR(1)		AUDIT カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
CONTEXTSTATUS	CHAR(1)		CONTEXT カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
VALIDATESTATUS	CHAR(1)		VALIDATE カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
CHECKINGSTATUS	CHAR(1)		CHECKING カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
SECMAINTSTATUS	CHAR(1)		SECMAINT カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
OBJMAINTSTATUS	CHAR(1)		OBJMAINT カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功

表 58. SYSCAT.AUDITPOLICIES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SYSADMINSTATUS	CHAR(1)		SYSADMIN カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
EXECUTESTATUS	CHAR(1)		EXECUTE カテゴリーの状況。 • B = 両方 • F = 失敗 • N = なし • S = 成功
EXECUTEWITHDATA	CHAR(1)		EXECUTE カテゴリーに記録されるホスト変数およびパラメーター・マーカー。 • N = No • Y = Yes
ERRORTYPE	CHAR(1)		監査エラー・タイプ。 • A = 監査 • N = 正常
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.AUDITUSE

各行は、USER、GROUP、または権限 (SYSADM、SYSCTRL、SYSMAINT) などの非データベース・オブジェクトと関連付けられている監査ポリシーを表します。

表 59. SYSCAT.AUDITUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
AUDITPOLICYNAME	VARCHAR(128)		監査ポリシーの名前。
AUDITPOLICYID	INTEGER		監査ポリシーの ID。
OBJECTTYPE	CHAR(1)		この監査ポリシーが関連付けられるオブジェクトのタイプ。 • S = MQT • T = 表 • g = 権限 • i = 許可 ID • x = トラステッド・コンテキスト • ブランク = データベース
SUBOBJECTTYPE	CHAR(1)		OBJECTTYPE が 'i' の場合、これは許可 ID が表すタイプです。 • G = グループ • R = ロール • U = ユーザー • ブランク = 該当しない場合
OBJECTSCHEMA	VARCHAR(128)		監査ポリシーが使用されているオブジェクトのスキーマ名。スキーマが適用されないオブジェクトを OBJECTTYPE が識別する場合、OBJECTSCHEMA はヌルになります。
OBJECTNAME	VARCHAR(128)		この監査ポリシーが使用されているオブジェクトの非修飾名。

SYSCAT.BUFFERPOOLDBPARTITIONS

各行は、バッファー・プールとデータベース・パーティションの組み合わせのうち、パーティション上のバッファー・プールのサイズが、同じデータベース・パーティション・グループ内の他のパーティションにおけるバッファー・プールのデフォルト・サイズ (SYSCAT.BUFFERPOOLS で表記されている) と異なっているものを表します。

表 60. SYSCAT.BUFFERPOOLDBPARTITIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
BUFFERPOOLID	INTEGER		内部バッファー・プール ID。
DBPARTITIONNUM	SMALLINT		データベース・パーティション番号。
NPAGES	INTEGER		このデータベース・パーティションに対するバッファー・プールのページ数。

SYSCAT.BUFFERPOOLS

各行は、データベースの 1 つのデータベース・パーティション・グループ上、またはデータベースのすべてのデータベース・パーティション上のバッファー・プールの構成を表します。

表 61. SYSCAT.BUFFERPOOLS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
BPNAME	VARCHAR (128)		バッファー・プールの名前。
BUFFERPOOLID	INTEGER		バッファー・プールの ID。
DBPGNAME	VARCHAR (128)	Y	データベース・パーティション・グループの名前 (そのバッファー・プールが、データベース内のすべてのデータベース・パーティションに対して存在する場合は NULL)。
NPAGES	INTEGER		このデータベース・パーティション・グループ内のデータベース・パーティション上にある、このバッファー・プールのデフォルト・ページ数。
PAGESIZE	INTEGER		このデータベース・パーティション・グループ内のデータベース・パーティション上にある、このバッファー・プールのページ・サイズ。
ESTORE	CHAR (1)		常に 'N'。拡張ストレージは適用されなくなりました。
NUMBLOCKPAGES	INTEGER		ブロック・ベースの領域に置かれるバッファー・プールのページ数。バッファー・プールのブロック・ベースの領域は、順次プリフェッチを行うプリフェッチャーによってのみ使用されます。
BLOCKSIZE	INTEGER		ブロック内のページ数。
NGNAME[1]	VARCHAR (128)	Y	データベース・パーティション・グループの名前 (そのバッファー・プールが、データベース内のすべてのデータベース・パーティションに対して存在する場合は NULL)。

注:

1. NGNAME 列は、後方互換性のために含まれています。DBPGNAME を参照してください。

SYSCAT.CASTFUNCTIONS

各行は、cast 関数を表します (組み込み cast 関数は含まれない)。

表 62. SYSCAT.CASTFUNCTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FROM_TYPESCHEMA	VARCHAR(128)		パラメーターのデータ・タイプのスキーマ名。
FROM_TYPENAME	VARCHAR(128)		パラメーターのデータ・タイプの名前。
TO_TYPESCHEMA	VARCHAR(128)		キャスト後の結果のデータ・タイプを示すスキーマ名。
TO_TYPENAME	VARCHAR(128)		キャスト後の結果のデータ・タイプの名前。
FUNCSCHEMA	VARCHAR(128)		関数のスキーマ名。
FUNCNAME	VARCHAR(128)		関数の非修飾名。
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
ASSIGN_FUNCTION	CHAR(1)		• N = 割り当て関数ではない • Y = 暗黙的な割り当て関数

SYSCAT.CHECKS

各行は、チェック制約またはマテリアライズ照会表内の派生列を表します。表階層の場合、チェック制約はそれぞれ作成された階層レベルでのみ記録されます。

表 63. SYSCAT.CHECKS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONSTNAME	VARCHAR(128)		チェック制約の名前。
OWNER	VARCHAR(128)		そのチェック制約を作成した許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
TABSCHEMA	VARCHAR(128)		この制約が適用される表のスキーマ名。
TABNAME	VARCHAR(128)		この制約が適用される表の名前。
CREATE_TIME	TIMESTAMP		制約が定義された時刻。この制約の一部である関数の解決に使用されます。制約が定義された後に作成された関数は選択されません。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。非修飾参照を完了するために使用します。
TYPE	CHAR(1)		チェック制約のタイプ。 • C = チェック制約 • F = 関数の従属関係 • O = 制約はオブジェクト・プロパティー • S = GENERATED ALWAYS 列のシステム生成チェック制約
FUNC_PATH	CLOB(2K)		制約が定義されたときに使用されていた SQL パス。制約の一部である関数およびタイプを解決するために使用されます。
TEXT	CLOB(2M)		チェック条件のテキストまたは派生列の定義。[1]
PERCENTVALID	SMALLINT		インフォメーショナル制約が有効である行の数。全体のパーセンテージで表されます。
COLLATIONSCHEMA	VARCHAR(128)		制約の照合のスキーマ名。
COLLATIONNAME	VARCHAR(128)		制約の照合の非修飾名。
COLLATIONSCHEMA_ ORDERBY	VARCHAR(128)		制約の ORDER BY 節の照合のスキーマ名。
COLLATIONNAME_ORDERBY	VARCHAR(128)		制約の ORDER BY 節の照合の非修飾名。
DEFINER[2]	VARCHAR(128)		そのチェック制約を作成した許可 ID。

注:

1. カタログ・ビューでは、チェック条件のテキストは常にデータベース・コード・ページ内で示されますが、その中で置換文字を使用することができます。チェック制約は常にターゲット表のコード・ページ内で適用されますが、適用時には置換文字を含みません。 (チェック制約は、置換文字の入っていない可能性のあるターゲット表のコード・ページ内のオリジナル・テキストに基づいて適用されます。)
2. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.COLAUTH

各行は、列に対して 1 つ以上の特権を付与されたユーザーまたはグループを表します。

表 64. SYSCAT.COLAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
TABSCHEMA	VARCHAR(128)		特権が保有されている表またはビューのスキーマ名。
TABNAME	VARCHAR(128)		特権が保有されている表またはビューの非修飾名。
COLNAME	VARCHAR(128)		この特権が適用される列の名前。
COLNO	SMALLINT		表の中のこの列の列番号 (0 から始まる)。
PRIVTYPE	CHAR(1)		• R = 参照特権 • U = 更新特権
GRANTABLE	CHAR(1)		• G = 特権は付与可能 • N = 特権は付与不可

注:

1. 特権の付与は列ごとに行えますが、取り消しは表全体の単位で行うことしかできません。

SYSCAT.COLCHECKS

各行は、チェック制約によって、またはマテリアライズ照会表の定義によって参照される列を表します。表階層の場合、チェック制約はそれぞれ作成された階層レベルでのみ記録されます。

表 65. SYSCAT.COLCHECKS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONSTNAME	VARCHAR(128)		チェック制約の名前。
TABSCHEMA	VARCHAR(128)		参照された列の入った表のスキーマ名。
TABNAME	VARCHAR(128)		参照された列の入った表の非修飾名。
COLNAME	VARCHAR(128)		列の名前。
USAGE	CHAR(1)		• D = 列は関数の従属関係において子です。 • P = 列は関数の従属関係において親です。 • R = 列はチェック制約内で参照されます。 • S = 列はマテリアライズ照会表をサポートするシステム生成の列チェック制約のソースです。 • T = 列はマテリアライズ照会表をサポートするシステム生成の列チェック制約のターゲットです。

SYSCAT.COLDIST

各行は、列の中で *n* 番目に高い頻度の値、または列の *n* 番目の変位 (累積分布) 値を表します。実表 (ビューではない) の列にのみ適用されます。型付き表の継承列の場合、統計は記録されません。

表 66. SYSCAT.COLDIST カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		統計が適用される表のスキーマ名。
TABNAME	VARCHAR(128)		統計が適用される表の非修飾名。
COLNAME	VARCHAR(128)		統計が適用される列の名前。
TYPE	CHAR(1)		• F = 頻度 • Q = 変位値
SEQNO	SMALLINT		TYPE= 'F' の場合、この列の値 *n* は最大頻度が *n* 番目であることを示す。TYPE= 'Q' の場合、この列の値 *n* は変位値が *n* 番目であることを示す。
COLVALUE[1]	VARCHAR(254)	Y	データ値 (文字リテラルまたは NULL 値)。
VALCOUNT	BIGINT		TYPE= 'F' の場合、VALCOUNT は、その列の中の COLVALUE の出現回数。 TYPE= 'Q' の場合、VALCOUNT は、値が COLVALUE 以下の行の数。
DISTCOUNT[2]	BIGINT	Y	TYPE= 'Q' の場合、この列は COLVALUE 以下の特殊値の数 (入手不能の場合は NULL) を記録します。

注:

1. カタログ・ビュー内の COLVALUE の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。
2. DISTCOUNT は、索引の最初のキー列である列でのみ収集されます。

SYSCAT.COLGROUPCOLS

各行は、列グループを構成する列を表します。

表 67. SYSCAT.COLGROUPCOLS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
COLGROUPID	INTEGER		列グループの ID。
COLNAME	VARCHAR(128)		列グループ内の列の名前。
TABSCHEMA	VARCHAR(128)		列グループ内の列を持つ表のスキーマ名。
TABNAME	VARCHAR(128)		列グループ内の列を持つ表の非修飾名。
ORDINAL	SMALLINT		列グループ内の列の順序数。

SYSCAT.COLGROUPDIST

各行は、列グループの中で *n* 番目に高い頻度の値または *n* 番目の変位値を構成する、列グループ内の列の値を表します。

表 68. SYSCAT.COLGROUPDIST カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
COLGROUPID	INTEGER		列グループの ID。
TYPE	CHAR(1)		• F = 頻度 • Q = 変位値
ORDINAL	SMALLINT		列グループ内の列の順序数。
SEQNO	SMALLINT		TYPE= 'F' の場合、この列の値 *n* は最大頻度が *n* 番目であることを示す。TYPE= 'Q' の場合、この列の値 *n* は変位値が *n* 番目であることを示す。
COLVALUE[1]	VARCHAR(254)		データ値 (文字リテラルまたは NULL 値)。

注:

1. カタログ・ビュー内の COLVALUE の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。

SYSCAT.COLGROUPDISTCOUNTS

各行は、列グループの中で *n* 番目に高い頻度の値、または列グループの中で *n* 番目の変位値に適用される分散統計を表します。

表 69. SYSCAT.COLGROUPDISTCOUNTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
COLGROUPID	INTEGER		列グループの ID。
TYPE	CHAR(1)		• F = 頻度 • Q = 変位値
SEQNO	SMALLINT		*n* 番目の TYPE 値を表すシーケンス番号 *n*。
VALCOUNT	BIGINT		TYPE= 'F' の場合、VALCOUNT は、この SEQNO を持つ列グループの中の COLVALUE の出現回数です。 TYPE= 'Q' の場合、VALCOUNT は、値がこの SEQNO を持つ列グループの COLVALUE 以下の行の数です。
DISTCOUNT	BIGINT		TYPE= 'Q' の場合、この列はこの SQENO を持つ列グループの COLVALUE 以下の値の種類数 (入手不能の場合は NULL) を記録します。

SYSCAT.COLGROUPS

各行は、列グループ、およびその列グループ全体に適用される統計を表します。

表 70. SYSCAT.COLGROUPS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
COLGROUPSCHEMA	VARCHAR(128)		列グループのスキーマ名。
COLGROUPNAME	VARCHAR(128)		列グループの非修飾名。
COLGROUPID	INTEGER		列グループの ID。
COLGROUPCARD	BIGINT		列グループのカーディナリティー。
NUMFREQ_VALUES	SMALLINT		列グループに関して収集された頻度の数。
NUMQUANTILES	SMALLINT		列グループに関して収集された変位値の数。

SYSCAT.COLIDENTATTRIBUTES

各行は、表に定義されている ID 列を表します。

表 71. SYSCAT.COLIDENTATTRIBUTES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		この列のある表またはビューのスキーマ名。
TABNAME	VARCHAR(128)		この列のある表またはビューの非修飾名。
COLNAME	VARCHAR(128)		列の名前。
START	DECIMAL(31,0)		シーケンスの開始値。
INCREMENT	DECIMAL(31,0)		増分値。
MINVALUE	DECIMAL(31,0)		シーケンスの最小値。
MAXVALUE	DECIMAL(31,0)		シーケンスの最大値。
CYCLE	CHAR(1)		その最大値または最小値に達した後、シーケンスが値の生成を続行できるかどうかを示します。 • N = シーケンスは循環できない • Y = シーケンスは循環できる
CACHE	INTEGER		アクセスを高速化するために、メモリーに事前割り振りするシーケンス値の数。 0 は、シーケンスの値が事前割り振りされないことを示します。パーティション・データベースでは、この値はそれぞれのデータベース・パーティションに適用されます。
ORDER	CHAR(1)		要求の順序でシーケンス番号が生成されるかどうかを示します。 • N = 要求の順序でシーケンス番号を生成しません。 • Y = 要求の順序でシーケンス番号を生成します。
NEXTCACHEFIRSTVALUE	DECIMAL(31,0)	Y	次のキャッシュ・ブロックで割り当てることのできる最初の値。キャッシュしない場合は、割り当てることのできる次の値。
SEQID	INTEGER		シーケンスの ID。

SYSCAT.COLOPTIONS

各行には、列固有のオプション値が入ります。

表 72. SYSCAT.COLOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		ニックネームのスキーマ名。
TABNAME	VARCHAR(128)		オプションが設定されている列のニックネーム。
COLNAME	VARCHAR(128)		ローカル列名。
OPTION	VARCHAR(128)		列オプションの名前。
SETTING	CLOB (32K)		値。

SYSCAT.COLUMNS

各行は、表、ビュー、またはニックネームに定義された列を表します。

表 73. SYSCAT.COLUMNS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		この列のある表、ビュー、またはニックネームのスキーマ名。
TABNAME	VARCHAR(128)		この列のある表、ビュー、またはニックネームの非修飾名。
COLNAME	VARCHAR(128)		列の名前。
COLNO	SMALLINT		表の中のこの列の番号 (0 から始まる)。
TYPESCHEMA	VARCHAR(128)		列のデータ・タイプのスキーマ名。
TYPENAME	VARCHAR(128)		列のデータ・タイプの非修飾名。
LENGTH	INTEGER		データの最大長。特殊タイプの場合は 0。LENGTH 列は、DECIMAL フィールドの精度を示し、10 進浮動小数点列に必要なストレージのバイト数を示します。DECFLOAT(16) と DECFLOAT(34) ではそれぞれ 8 と 16 です。
SCALE	SMALLINT		列タイプが DECIMAL の場合は位取り。その他の場合は 0。
DEFAULT[1]	VARCHAR(254)	Y	列のデータ・タイプに適した定数、特殊レジスター、または cast 関数で表された表の列のデフォルト値。キーワード NULL の場合もあります。値は、デフォルト値として指定された値から変換されることがあります。例えば、日時の定数は ISO フォーマットで表示され、cast 関数名はスキーマ名で修飾され、ID は区切り文字で区切られます。DEFAULT 節が指定されていないか、または列がビューの列の場合は、NULL 値になります。
NULLS[2]	CHAR(1)		列の NULL 可能性属性。 • N = 列は NULL 不可。 • Y = 列は NULL 可能。 式または関数から派生したビューの列の場合、値は 'N' である可能性があります。それでも、このビューを使用するステートメントが処理され、算術計算エラーの警告を出された場合は、この列に NULL 値が入ります。
CODEPAGE	SMALLINT		この列のデータに使用されるコード・ページ。列が FOR BIT DATA として定義されている場合、またはストリング・タイプではない場合は 0。

表 73. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLATIONSCHEMA	VARCHAR(128)	Y	ストリング・タイプの場合は、列の照合のスキーマ名。それ以外の場合は NULL 値。
COLLATIONNAME	VARCHAR(128)	Y	ストリング・タイプの場合は、列の照合の非修飾名。それ以外の場合は NULL 値。
LOGGED	CHAR(1)		LOB タイプまたは LOB に基づく特殊タイプの列だけに適用されます。それ以外はブランクです。 • N = 列のログ記録を取らない。 • Y = 列のログ記録を取る。
COMPACT	CHAR(1)		LOB タイプまたは LOB に基づく特殊タイプの列だけに適用されます。それ以外はブランクです。 • N = 列を圧縮しない。 • Y = ストレージ内で列を圧縮する。
COLCARD	BIGINT		列の特殊値の数。統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
HIGH2KEY[3]	VARCHAR(254)	Y	2 番目に高いデータ値。数値データを文字リテラルに変更して表現。統計が収集されていない場合は空。継承列および階層表の列の場合は空。
LOW2KEY[3]	VARCHAR(254)	Y	2 番目に低いデータ値。数値データを文字リテラルに変更して表現。統計が収集されていない場合は空。継承列および階層表の列の場合は空。
AVGCOLLEN	INTEGER		列に必要な平均のスペース (バイト単位)。長形式フィールドまたは LOB の場合、または統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
KEYSEQ	SMALLINT	Y	表の主キー内の列の位置番号。副表および階層表の列の場合は NULL 値。
PARTKEYSEQ	SMALLINT	Y	表の分散キー内の列の位置番号。列が分散キー内にない場合は、0 または NULL 値。副表および階層表の列の場合は NULL 値。
NQUANTILES	SMALLINT		この列の SYSCAT.COLDIST に記録された変位値の数。統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
NMOSTFREQ	SMALLINT		この列の SYSCAT.COLDIST に記録された最大頻度の値の数。統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
NUMNULLS	BIGINT		列内の NULL 値の数。統計が収集されていない場合は -1。

表 73. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
TARGET_TYPESCHEMA	VARCHAR(128)	Y	この列の型が REFERENCE の場合、ターゲット行タイプのスキーマ名。その他の場合、NULL 値。
TARGET_TYPENAME	VARCHAR(128)	Y	この列の型が REFERENCE の場合、ターゲット行タイプの非修飾名。その他の場合、NULL 値。
SCOPE_TABSCHEMA	VARCHAR(128)	Y	この列の型が REFERENCE の場合、有効範囲 (ターゲット表) のスキーマ名。その他の場合、NULL 値。
SCOPE_TABNAME	VARCHAR(128)	Y	この列の型が REFERENCE の場合、有効範囲 (ターゲット表) の非修飾名。その他の場合、NULL 値。
SOURCE_TABSCHEMA	VARCHAR(128)	Y	型付き表またはビューの列の場合、列が最初に使われた表やビューのスキーマ名。非継承列の場合、これは TABSCHEMA と同じです。型なし表およびビューの列の場合は NULL 値。
SOURCE_TABNAME	VARCHAR(128)	Y	型付き表またはビューの列の場合、列が最初に使われた表やビューの非修飾名。非継承列の場合、これは TABNAME と同じです。型なし表およびビューの列の場合は NULL 値。
DL_FEATURES	CHAR(10)	Y	この列は使用されなくなりました。将来のリリースで除去されます。
SPECIAL_PROPS	CHAR (8)	Y	REFERENCE 型の列だけに適用されます。その他の場合はブランクです。各バイト位置は、次のようにして定義されます。 • 1 = オブジェクト ID (OID) 列 (yes の場合は 'Y'、no の場合は 'N') • 2 = ユーザー生成またはシステム生成 (ユーザーの場合は 'U'、システムの場合は 'S') バイト 3 から 8 は、将来の使用のために予約されています。
HIDDEN	CHAR(1)		隠し列のタイプ。 • I = 列は IMPLICITLY HIDDEN として定義される • S = システムに管理される隠し列。 • ブランク = 列は隠されていない。
INLINE_LENGTH	INTEGER		基本表に保管できる XML 文書または構造化タイプのインスタンスの内部表記の最大サイズ (バイト数)。適用されない場合には 0。
IDENTITY	CHAR(1)		• N = ID 列ではない。 • Y = ID 列。

表 73. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ROWCHANGETIMESTAMP	CHAR (1)		• N = 行変更タイム・スタンプ列ではない • Y = 行変更タイム・スタンプ列
GENERATED	CHAR (1)		生成される列の型。 • A = 列の値が常に生成される • D = 列の値がデフォルトで生成される • ブランク = 列は生成されない
TEXT	CLOB(2M)	Y	式として生成されると定義された列の場合、このフィールドにはキーワード AS で始まる、生成された列式のテキストが入ります。
COMPRESS	CHAR(1)		• O = 圧縮をオフにする • S = システム・デフォルト値を圧縮する
AVGDISTINCTPERPAGE	DOUBLE	Y	将来の利用。
PAGEVARIANCERATIO	DOUBLE	Y	将来の利用。
SUB_COUNT	SMALLINT		列のサブエレメントの平均数。文字ストリング列のみに適用されます。
SUB_DELIM_LENGTH	SMALLINT		列内の各サブエレメントを区切る、区切り文字の平均の長さ。文字ストリング列のみに適用されます。
AVGCOLLENCHAR	INTEGER		列に必要な平均文字数 (列で有効になっている照合に基づく)。長形式フィールドまたは LOB の場合、または統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
IMPLICITVALUE[4]	VARCHAR(254)	Y	表の作成後に表に追加された列の場合、列の追加時のデフォルト値を保管します。表の作成時に定義された列の場合、NULL 値を保管します。
SECLABELNAME	VARCHAR(128)	Y	保護された列の場合、その列に関連したセキュリティー・ラベルの名前。その他の場合は NULL 値。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

表 73. SYSCAT.COLUMNS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明

注:

1. バージョン 2.1.0 では cast-function 名は区切られていなかったため、 DEFAULT 列にこのように表示される場合があります。また、ビュー列にはデフォルト値が入っていたため、これも DEFAULT 列に表示されます。
2. バージョン 2 以降では、値 D (デフォルト値が NULL でないことを示す) は使用されなくなりました。代わりに、WITH DEFAULT の使用は、 DEFAULT 欄の値が NULL 以外であることによって示されます。
3. カタログ・ビューでは、HIGH2KEY と LOW2KEY の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。
4. データ・パーティションのアタッチは、以下の状況でない限り、許可されています。すなわち、ソース列とターゲット列の両方に関して特定の列の IMPLICITVALUE が NULL 以外の値であり、かつ両者の値が一致していない場合です。この場合は、ソース表をドロップしてから再作成しなければなりません。以下の条件のうちの 1 つが満たされる場合に、列は IMPLICITVALUE フィールドで NULL 以外の値を持つことができます。
   - 列が ALTER TABLE...ADD COLUMN ステートメントの結果として作成される場合。
   - IMPLICITVALUE フィールドがアタッチ中にソース表から伝搬される場合。
   - IMPLICITVALUE フィールドがデタッチ中にソース表から継承される場合。
   - IMPLICITVALUE フィールドがバージョン 8 からバージョン 9 へのマイグレーション中に設定され、その際、追加された列であると判別されるか、追加された列である可能性がある場合。列が追加されたものかどうかをデータベースが確定できない場合、列は追加されたものとして扱われます。追加された列とは、ALTER TABLE...ADD COLUMN ステートメントの結果として作成された列です。

   マイグレーション以外のシナリオでこのような不整合を避けるため、アタッチ予定の表を作成する場合には必ずすべての列を定義しておくことをお勧めします。つまり、表をアタッチする前には、決して ALTER TABLE ステートメントを使って表に列を追加しないということです。

SYSCAT.COLUSE

各行は、CREATE TABLE ステートメントの DIMENSIONS 節で参照される列を表します。

表 74. SYSCAT.COLUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		列の入った表のスキーマ名。
TABNAME	VARCHAR(128)		列の入った表の非修飾名。
COLNAME	VARCHAR(128)		列の名前。
DIMENSION	SMALLINT		DIMENSIONS 節に指定されたディメンション順序に基づくディメンション番号 (最初の位置は 0)。複合ディメンションでは、この値はディメンションのどのコンポーネントでも同じです。
COLSEQ	SMALLINT		列が属するディメンション内での列の位置番号 (最初の位置は 0)。複合ではないディメンションの単一列では、この値は 0 です。
TYPE	CHAR(1)		ディメンションのタイプ。 • C = クラスタリングまたはマルチディメンション・クラスタリング • P = パーティション

SYSCAT.CONSTDEP

各行は、他のオブジェクトに対する制約の従属関係を表します。制約は、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更は制約に影響します。

表 75. SYSCAT.CONSTDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONSTNAME	VARCHAR(128)		制約の非修飾名。
TABSCHEMA	VARCHAR(128)		制約が適用される表のスキーマ名。
TABNAME	VARCHAR(128)		制約が適用される表の非修飾名。
BTYPE	CHAR(1)		制約が依存するオブジェクトのタイプ。使用できる値は以下のとおりです。 • F = ルーチン・インスタンス • I = 索引 • R = 構造化タイプ
BSCHEMA	VARCHAR(128)		制約が従属しているオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		制約が従属しているオブジェクトの非修飾名。

SYSCAT.CONTEXTATTRIBUTES

各行は、トラステッド・コンテキスト属性を表します。

表 76. SYSCAT.CONTEXTATTRIBUTES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONTEXTNAME	VARCHAR(128)		トラステッド・コンテキストの名前。
ATTR_NAME	VARCHAR(128)		属性の名前。以下のいずれか • ADDRESS • ENCRYPTION
ATTR_VALUE	VARCHAR(128)		属性の値。
ATTR_OPTIONS	VARCHAR(128)	Y	ATTR_NAME が 'ADDRESS' の場合は、この特定アドレスで必要な暗号化のレベルを指定します。 NULL 値であれば、グローバル ENCRYPTION 属性が適用されます。

SYSCAT.CONTEXTS

各行は、トラステッド・コンテキストを表します。

表 77. SYSCAT.CONTEXTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONTEXTNAME	VARCHAR(128)		トラステッド・コンテキストの名前。
CONTEXTID	INTEGER		トラステッド・コンテキストの ID。
SYSTEMAUTHID	VARCHAR(128)		トラステッド・コンテキストに関連付けられたシステム許可 ID。
DEFAULTCONTEXTROLE	VARCHAR(128)	Y	コンテキストのデフォルトのロール。
CREATE_TIME	TIMESTAMP		トラステッド・コンテキストが作成された時刻。
ALTER_TIME	TIMESTAMP		トラステッド・コンテキストが最後に変更された時刻。
ENABLED	CHAR(1)		トラステッド・コンテキストの状態。 • N = 使用不可 • Y = 使用可能
AUDITPOLICYID	INTEGER	Y	監査ポリシーの ID。
AUDITPOLICYNAME	VARCHAR(128)	Y	監査ポリシーの名前。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.DATAPARTITIONEXPRESSION

各行は、表パーティション・キーの一部に入る式を表します。

表 78. SYSCAT.DATAPARTITIONEXPRESSION カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		パーティション化された表のスキーマ名。
TABNAME	VARCHAR(128)		パーティション化された表の非修飾名。
DATAPARTITIONKEYSEQ	INTEGER		式キー部分のシーケンス ID (1 から始まる)。
DATAPARTITIONEXPRESSION	CLOB (32K)		シーケンス内のこの項目の式 (SQL 構文)。
NULLSFIRST	CHAR(1)		• N = この式で NULL 値を比較するときには高いものとして扱う • Y = この式で NULL 値を比較するときには低いものとして扱う

SYSCAT.DATAPARTITIONS

各行はデータ・パーティションを表します。

表 79. SYSCAT.DATAPARTITIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
DATAPARTITIONNAME	VARCHAR(128)		データ・パーティションの名前。
TABSCHEMA	VARCHAR (128)		このデータ・パーティションが属する表のスキーマ名。
TABNAME	VARCHAR (128)		このデータ・パーティションが属する表の非修飾名。
DATAPARTITIONID	INTEGER		データ・パーティションの ID。
TBSPACEID	INTEGER	Y	このデータ・パーティションが保管されている表スペースの ID。STATUS が 'I' の場合、NULL。
PARTITIONOBJECTID	INTEGER	Y	表スペース内のデータ・パーティションの ID。
LONG_TBSPACEID	INTEGER	Y	長形式データが保管されている表スペースの ID。 STATUS が 'I' の場合、NULL。
ACCESS_MODE	CHAR(1)		データ・パーティションのアクセス制限の状態。これらの状態は、SET INTEGRITY によりペンディング状態にあるオブジェクト、または SET INTEGRITY ステートメントによって処理されたオブジェクトにのみ適用されます。使用できる値は以下のとおりです。 • D = データの移動不可 • F = フル・アクセス権限 • N = アクセス不可 • R = 読み取り専用アクセス
STATUS	VARCHAR(32)		• A = データ・パーティションは新規にアタッチされた • D = データ・パーティションはデタッチされた • I = 非同期の索引クリーンアップ中にのみカタログ内の項目が維持される、デタッチされたデータ・パーティション。デタッチされたパーティションを参照するすべての索引レコードの削除時に、STATUS 値が 'I' の行は除去される。 • 空ストリング = データ・パーティションは表示できる (通常の状況)。 バイト 2 から 32 は、将来の使用のために予約されています。
SEQNO	INTEGER		データ・パーティションのシーケンス番号 (0 から始まる)。

表 79. SYSCAT.DATAPARTITIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
LOWINCLUSIVE	CHAR(1)		• N = 低い方のキー値はそれ自身を含まない • Y = 低い方のキー値はそれ自身を含む
LOWVALUE	VARCHAR(512)		このデータ・パーティションの低い方のキー値 (SQL 値のストリング表記)。
HIGHINCLUSIVE	CHAR(1)		• N = 高い方のキー値はそれ自身を含まない • Y = 高い方のキー値はそれ自身を含む
HIGHVALUE	VARCHAR(512)		このデータ・パーティションの高い方のキー値 (SQL 値のストリング表記)。

SYSCAT.DATATYPES

各行は組み込みまたはユーザー定義のデータ・タイプを表します。

表 80. SYSCAT.DATATYPES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TYPESCHEMA	VARCHAR(128)		データ・タイプのスキーマ名。組み込みタイプのスキーマ名は 'SYSIBM' です。
TYPENAME	VARCHAR(128)		データ・タイプの非修飾名。
OWNER	VARCHAR(128)		タイプ作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
SOURCESCHEMA	VARCHAR(128)	Y	特殊タイプのソース・タイプのスキーマ名。ユーザー定義構造化タイプの場合、これは参照表示タイプのタイプ・スキーマになります。その他の場合は NULL 値。
SOURCENAME	VARCHAR(128)	Y	特殊タイプのソース・タイプの非修飾名。ユーザー定義構造化タイプの場合、これは参照表示タイプのタイプ名になります。その他の場合は NULL 値。
METATYPE	CHAR(1)		• A = 配列型 • R = ユーザー定義構造化タイプ • S = システムで定義済みのタイプ • T = ユーザー定義特殊タイプ
TYPEID	SMALLINT		データ・タイプの ID。
SOURCETYPEID	SMALLINT	Y	ソース・タイプの ID (組み込みタイプの場合は NULL 値)。ユーザー定義構造化タイプの場合、これは参照表示タイプの ID になります。
LENGTH	INTEGER		タイプの最大長。組み込みパラメーター化タイプ (DECIMAL や VARCHAR など) の場合は 0。ユーザー定義構造化タイプの場合、これは参照表示タイプの長さです。
SCALE	SMALLINT		組み込み DECIMAL タイプに基づく特殊タイプまたは参照表示タイプの位取り。他のすべてのタイプの場合 (DECIMAL 自体を含む) は 0。ユーザー定義構造化タイプの場合、これは参照表示タイプの長さを示します。
CODEPAGE	SMALLINT		ストリング・タイプ、ストリング・タイプに基づいた特殊タイプ、または参照表示タイプのデータベース・コード・ページ。その他の場合、0。
COLLATIONSCHEMA	VARCHAR (128)	Y	ストリング・タイプの場合は、データ・タイプの照合のスキーマ名。それ以外の場合は NULL 値。

表 80. SYSCAT.DATATYPES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLATIONNAME	VARCHAR (128)	Y	ストリング・タイプの場合は、データ・タイプの照合の非修飾名。それ以外の場合は NULL 値。
ARRAY_LENGTH	INTEGER	Y	配列の最大カーディナリティー。このタイプが配列タイプでない場合には NULL。
CREATE_TIME	TIMESTAMP		データ・タイプの作成時刻。
ATTRCOUNT	SMALLINT		データ・タイプ内の属性の数。
INSTANTIABLE	CHAR (1)		• N = タイプをインスタンス化できない。 • Y = タイプをインスタンス化できる。
WITH_FUNC_ACCESS	CHAR (1)		• N = 関数表記を使ってこのタイプ用のメソッドすべてを呼び出せない。 • Y = 関数表記を使ってこのタイプ用のメソッドすべてを呼び出せる。
FINAL	CHAR (1)		• N = ユーザー定義タイプにサブタイプを指定できる。 • Y = ユーザー定義タイプにサブタイプを指定できない。
INLINE_LENGTH	INTEGER		基本表の行で保持できる構造化タイプの最大長。その他の場合、0。
NATURAL_INLINE_LENGTH	INTEGER	Y	構造化タイプ・インスタンスのシステム生成の自然なインラインの長さ。このタイプが構造化タイプでない場合、NULL 値。
JARSCHEMA	VARCHAR (128)	Y	SQL タイプをインプリメントする Java クラスを含む Jar ファイルを識別する JAR_ID のスキーマ名。 EXTERNAL NAME 節が指定されていない場合は NULL 値。
JAR_ID	VARCHAR (128)	Y	この SQL タイプをインプリメントする Java クラスを含む Jar ファイルの ID。 EXTERNAL NAME 節が指定されていない場合は NULL 値。
CLASS	VARCHAR (384)	Y	この SQL タイプをインプリメントする Java クラス。 EXTERNAL NAME 節が指定されていない場合は NULL 値。
SQLJ_REPRESENTATION	CHAR (1)	Y	この SQL タイプをインプリメントする Java クラスの SQLJ "representation_spec"。 EXTERNAL NAME ... LANGUAGE JAVA REPRESENTATION SPEC 節が指定されていない場合は NULL 値。 • D = SQL データ • S = シリアライズ可能
ALTER_TIME	TIMESTAMP		データ・タイプが最後に変更された時刻。
DEFINER[1]	VARCHAR (128)		タイプ作成時の許可 ID。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

表 80. SYSCAT.DATATYPES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.DBAUTH

各行は、1 つ以上のデータベース・レベルの権限を付与されたユーザーまたはグループを表します。

表 81. SYSCAT.DBAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		権限の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		権限の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
BINDADDAUTH	CHAR(1)		パッケージを作成する権限。 • N = 保有しない • Y = 保有する
CONNECTAUTH	CHAR(1)		データベースに接続する権限。 • N = 保有しない • Y = 保有する
CREATETABAUTH	CHAR(1)		表を作成する権限。 • N = 保有しない • Y = 保有する
DBADMAUTH	CHAR(1)		DBADM 権限。 • N = 保有しない • Y = 保有する
EXTERNALROUTINEAUTH	CHAR(1)		外部ルーチンを作成する権限。 • N = 保有しない • Y = 保有する
IMPLSCHEMAAUTH	CHAR(1)		存在しないスキーマを指定し、オブジェクトを作成した場合に、暗黙的にスキーマを作成する権限。 • N = 保有しない • Y = 保有する
LOADAUTH	CHAR(1)		DB2 ロード・ユーティリティーを使用する権限。 • N = 保有しない • Y = 保有する
NOFENCEAUTH	CHAR(1)		fenced でないユーザー定義関数を作成する権限。 • N = 保有しない • Y = 保有する

表 81. SYSCAT.DBAUTH カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
QUIESCECONNECTAUTH	CHAR(1)		静止中のデータベースにアクセスする権限。 • N = 保有しない • Y = 保有する
LIBRARYADMAUTH	CHAR(1)		将来の使用のために予約済み。
SECURITYADMAUTH	CHAR(1)		セキュリティー管理者権限。 • N = 保有しない • Y = 保有する

SYSCAT.DBPARTITIONGROUPDEF

各行は、データベース・パーティション・グループに属するデータベース・パーティションを表します。

表 82. SYSCAT.DBPARTITIONGROUPDEF カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
DBPGNAME	VARCHAR(128)		データベース・パーティションの入ったデータベース・パーティション・グループの名前。
DBPARTITIONNUM	SMALLINT		データベース・パーティション・グループに属するデータベース・パーティションのパーティション番号。有効なパーティション番号は、0 以上 999 以下の間です。
IN_USE	CHAR(1)		データベース・パーティションの状況。 • A = 新しく追加されたデータベース・パーティションは分散マップに入っていないが、データベース・パーティション・グループ内の表スペースにコンテナーが作成された。データベース・パーティションは、REDISTRIBUTION が進行中になると分散マップに追加される • D = データベース・パーティションは、データベース・パーティション・グループ再配分操作が正常に完了したときにドロップされる。 • T = 新しく追加されたデータベース・パーティションは、分散マップに入っておらず、WITHOUT TABLESPACES 節を使用して追加された。データベース・パーティション・グループの表スペースにコンテナーを追加する必要がある。 • Y = データベース・パーティションは分散マップに入っている。

SYSCAT.DBPARTITIONGROUPS

各行はデータベース・パーティション・グループを表します。

表 83. SYSCAT.DBPARTITIONGROUPS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
DBPGNAME	VARCHAR(128)		データベース・パーティション・グループの名前。
OWNER	VARCHAR(128)		データベース・パーティション・グループ作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS カタログ・ビュー内の分散マップの ID。
REDISTRIBUTE_PMAP_ID	SMALLINT		現在再配分のために使用されている分散マップの ID。再配分が現在進行中でない場合、-1。
CREATE_TIME	TIMESTAMP		データベース・パーティション・グループの作成時刻。
DEFINER[1]	VARCHAR(128)		データベース・パーティション・グループ作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.EVENTMONITORS

各行は、イベント・モニターを表します。

表 84. SYSCAT.EVENTMONITORS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
EVMONNAME	VARCHAR(128)		イベント・モニターの名前。
OWNER	VARCHAR(128)		イベント・モニター作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
TARGET_TYPE	CHAR(1)		イベント・データの書き込み先 (ターゲット) のタイプ。 • F = ファイル • P = パイプ • T = 表
TARGET	VARCHAR (762)		ファイルまたはパイプのイベント・モニター・データの書き込み先 (ターゲット) の名前。ファイルの場合、この名前は絶対パス名または相対パス名 (データベースのデータベース・パスに対して相対。これは LIST ACTIVE DATABASES コマンドを使用して確認できる) のいずれかになります。パイプの場合、この名前は絶対パス名にできます。
MAXFILES	INTEGER	Y	このイベント・モニターの 1 つのイベント・パスで許されるイベント・ファイルの最大数。最大値がない場合、または TARGET_TYPE が 'F' (ファイル) でない場合は NULL 値。
MAXFILESIZE	INTEGER	Y	各イベント・ファイルの最大サイズ (4K ページ単位)。このサイズに達すると、イベント・モニターは新しいファイルを作成します。最大値がない場合、または TARGET_TYPE が 'F' (ファイル) でない場合は NULL 値。
BUFFERSIZE	INTEGER	Y	ファイルに出力する場合、イベント・モニターの使用するバッファー・サイズ (4K ページ単位)。それ以外の場合は NULL 値。
IO_MODE	CHAR(1)	Y	ファイル入出力 (I/O) のモード。 • B = ブロック化 • N = 非ブロック化 • Null = TARGET_TYPE が 'F' (ファイル) または 'T' (表) ではない

表 84. SYSCAT.EVENTMONITORS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
WRITE_MODE	CHAR(1)	Y	このイベント・モニターの起動時に、モニターが既存のイベント・データを処理する方法。 • A = 追加 • R = 置換 • Null = TARGET_TYPE が 'F' (ファイル) でない
AUTOSTART	CHAR(1)		データベース開始時にこのイベント・モニターが自動的に活動化されるかどうかを示す。 • N = No • Y = Yes
DBPARTITIONNUM	SMALLINT		イベント・モニターが稼働してイベントをログ記録するデータベース・パーティションの番号。
MONSCOPE	CHAR(1)		モニターの有効範囲。 • G = グローバル • L = ローカル • T = 表スペースが存在する各データベース・パーティション • ブランク = WRITE TO TABLE イベント・モニター
EVMON_ACTIVATES	INTEGER		このイベント・モニターが活動化された回数
NODENUM[1]	SMALLINT		イベント・モニターが稼働してイベントをログ記録するデータベース・パーティションの番号。
DEFINER[2]	VARCHAR(128)		イベント・モニター作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	将来の使用のために予約済み。

注:

1. NODENUM 列は、後方互換性のために含まれています。DBPARTITIONNUM を参照してください。
2. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.EVENTS

各行は、モニター対象のイベントを表します。一般に、1 つのイベント・モニターは複数のイベントをモニターします。

表 85. SYSCAT.EVENTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
EVMONNAME	VARCHAR(128)		このイベントをモニターするイベント・モニターの名前。
TYPE	VARCHAR(128)		モニターされるイベントのタイプ。使用できる値は以下のとおりです。 • ACTIVITIES • CONNECTIONS • DATABASE • DEADLOCKS • DETAILDEADLOCKS • STATEMENTS • TABLES • TABLESPACES • THRESHOLD_VIOLATIONS • TRANSACTIONS • STATISTICS
FILTER	CLOB(64K)	Y	このイベントに適用される WHERE 節のテキスト全体。

SYSCAT.EVENTTABLES

各行は、SQL 表に書き込むイベント・モニターのターゲット表を表します。

表 86. SYSCAT.EVENTTABLES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
EVMONNAME	VARCHAR(128)		イベント・モニターの名前。
LOGICAL_GROUP	VARCHAR(128)		論理データ・グループの名前。使用できる値は以下のとおりです。 • ACTIVITYHISTORY • BUFFERPOOL • CONN • CONNHEADER • CONTROL • DATAVAL • DB • DEADLOCK • DLCONN • DLLOCK • SCSTATS • STMT • STMTHIST • STMTVALS • SUBSECTION • TABLE • TABLESPACE • THRESHOLDVIOLATIONS • WCSTATS • WLSTATS • XACT
TABSCHEMA	VARCHAR(128)		ターゲット表のスキーマ名。
TABNAME	VARCHAR(128)		ターゲット表の非修飾名。
PCTDEACTIVATE	SMALLINT		イベント・モニターは、DMS 表スペースがこの比率の値を超えると、自動的に非活動化されます。 SMS 表スペースには、100 に設定します。

SYSCAT.FULLHIERARCHIES

各行は、副表とスーパー表、サブタイプとスーパータイプ、またはサブビューとスーパービューのリレーションシップを表しています。このビューには、直接のリレーションシップをはじめとする、すべての階層リレーションシップが入っています。

表 87. SYSCAT.FULLHIERARCHIES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
METATYPE	CHAR(1)		リレーションシップ・タイプ。 • R = 構造化タイプの相互関係 • U = 型付き表の相互関係 • W = 型付きビューの相互関係
SUB_SCHEMA	VARCHAR(128)		サブタイプ、副表、またはサブビューのスキーマ名。
SUB_NAME	VARCHAR(128)		サブタイプ、副表、またはサブビューの非修飾名。
SUPER_SCHEMA	VARCHAR(128)	Y	スーパータイプ、スーパー表、またはスーパービューのスキーマ名。
SUPER_NAME	VARCHAR(128)	Y	スーパータイプ、スーパー表、またはスーパービューの非修飾名。
ROOT_SCHEMA	VARCHAR(128)		階層のルートにある表、ビュー、またはタイプのスキーマ名。
ROOT_NAME	VARCHAR(128)		階層のルートにある表、ビュー、またはタイプの非修飾名。

SYSCAT.FUNCMAPOPTIONS

各行は、関数マッピングのオプション値を表します。

表 88. SYSCAT.FUNCMAPOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FUNCTION_MAPPING	VARCHAR(128)		関数マッピングの名前。
OPTION	VARCHAR(128)		関数マッピングのオプション名。
SETTING	VARCHAR (2048)		関数マッピングのオプションの値。

SYSCAT.FUNCMAPPARMOPTIONS

各行は、関数マッピングのパラメーター・オプションの値を表します。

表 89. SYSCAT.FUNCMAPPARMOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FUNCTION_MAPPING	VARCHAR(128)		関数マッピングの名前。
ORDINAL	SMALLINT		パラメーターの位置。
LOCATION	CHAR(1)		パラメーターのロケーション。 • L = ローカル・パラメーター • R = リモート・パラメーター
OPTION	VARCHAR(128)		関数マッピングのパラメーター・オプションの名前。
SETTING	VARCHAR (2048)		関数マッピングのパラメーター・オプションの値。

SYSCAT.FUNCMAPPINGS

各行は関数マッピングを表します。

表 90. SYSCAT.FUNCMAPPINGS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FUNCTION_MAPPING	VARCHAR(128)		関数マッピングの名前 (システム生成の場合もある)。
FUNCSCHEMA	VARCHAR(128)	Y	関数のスキーマ名。 NULL の場合、関数は組み込み関数であると見なされます。
FUNCNAME	VARCHAR(1024)	Y	ユーザー定義関数または組み込み関数の非修飾名。
FUNCID	INTEGER	Y	関数の ID。
SPECIFICNAME	VARCHAR(128)	Y	ルーチン・インスタンスの名前 (システム生成の場合もある)。
OWNER	VARCHAR(128)		マッピング作成時の許可 ID。 SYSIBM' は、これが組み込み関数であることを示します。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
WRAPNAME	VARCHAR(128)	Y	このマッピングが適用されるラッパー。
SERVERNAME	VARCHAR(128)	Y	データ・ソースの名前。
SERVERTYPE	VARCHAR(30)	Y	マッピングが適用されるデータ・ソースのタイプ。
SERVERVERSION	VARCHAR(18)	Y	マッピングが適用されるサーバー・タイプのバージョン。
CREATE_TIME	TIMESTAMP		マッピングが作成された時刻。
DEFINER[1]	VARCHAR(128)		マッピング作成時の許可 ID。 SYSIBM' は、これが組み込み関数であることを示します。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.HIERARCHIES

各行は、副表とすぐ上のスーパー表、サブタイプとすぐ上のスーパータイプ、またはサブビューとすぐ上のスーパービューのリレーションシップを表しています。このビューには、直接の階層リレーションシップしか入っていません。

表 91. SYSCAT.HIERARCHIES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
METATYPE	CHAR(1)		リレーションシップ・タイプ。 • R = 構造化タイプの相互関係 • U = 型付き表の相互関係 • W = 型付きビューの相互関係
SUB_SCHEMA	VARCHAR(128)		サブタイプ、副表、またはサブビューのスキーマ名。
SUB_NAME	VARCHAR(128)		サブタイプ、副表、またはサブビューの非修飾名。
SUPER_SCHEMA	VARCHAR(128)		スーパータイプ、スーパー表、またはスーパービューのスキーマ名。
SUPER_NAME	VARCHAR(128)		スーパータイプ、スーパー表、またはスーパービューの非修飾名。
ROOT_SCHEMA	VARCHAR(128)		階層のルートにある表、ビュー、またはタイプのスキーマ名。
ROOT_NAME	VARCHAR(128)		階層のルートにある表、ビュー、またはタイプの非修飾名。

SYSCAT.HISTOGRAMTEMPLATEBINS

各行は、ヒストグラム・テンプレート bin を表します。

表 92. SYSCAT.HISTOGRAMTEMPLATEBINS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATENAME	VARCHAR(128)	Y	ヒストグラム・テンプレートの名前。
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
BINID	INTEGER		ヒストグラム・テンプレート bin の ID。
BINUPPERVALUE	BIGINT		ヒストグラム・テンプレートの 1 つの bin の高い方の値。

SYSCAT.HISTOGRAMTEMPLATES

各行は、ヒストグラム・テンプレートを表します。

表 93. SYSCAT.HISTOGRAMTEMPLATES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
TEMPLATENAME	VARCHAR(128)		ヒストグラム・テンプレートの名前。
CREATE_TIME	TIMESTAMP		ヒストグラム・テンプレートが作成された時刻。
ALTER_TIME	TIMESTAMP		ヒストグラム・テンプレートが最後に変更された時刻。
NUMBINS	INTEGER		ヒストグラム・テンプレートの bin の数。これには上限値のない最後の bin も含まれます。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.HISTOGRAMTEMPLATEUSE

各行は、ヒストグラム・テンプレートを使用できるワークロード管理オブジェクトとヒストグラム・テンプレートの関係を表します。

表 94. SYSCAT.HISTOGRAMTEMPLATEUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATENAME	VARCHAR(128)	Y	ヒストグラム・テンプレートの名前。
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
HISTOGRAMTYPE	CHAR(1)		このテンプレートに基づくヒストグラムで収集される情報のタイプ。 • C = アクティビティー見積コストのヒストグラム • E = アクティビティー実行時間のヒストグラム • I = 1 つのアクティビティーが到着してから別のアクティビティーが到着するまでの時間のヒストグラム • L = アクティビティー存続時間のヒストグラム • Q = アクティビティー・キュー時間のヒストグラム • R = 要求実行時間のヒストグラム
OBJECTTYPE	CHAR(1)		WLM オブジェクトのタイプ。 • b = サービス・クラス • k = 作業アクション
OBJECTID	INTEGER		WLM オブジェクトの ID。
SERVICECLASSNAME	VARCHAR(128)	Y	サービス・クラスの名前。
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	親サービス・クラスの名前。
WORKACTIONNAME	VARCHAR(128)	Y	作業アクションの名前。
WORKACTIONSETNAME	VARCHAR(128)	Y	作業アクション・セットの名前。

SYSCAT.INDEXAUTH

各行は、索引に対して CONTROL 特権を付与されたユーザーまたはグループを表します。

表 95. SYSCAT.INDEXAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
INDSCHEMA	VARCHAR(128)		索引のスキーマ名。
INDNAME	VARCHAR(128)		索引の非修飾名。
CONTROLAUTH	CHAR(1)		CONTROL 特権。 • N = 保有しない • Y = 保有する

SYSCAT.INDEXCOLUSE

各行は、索引に関与する列を表します。

表 96. SYSCAT.INDEXCOLUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引のスキーマ名。
INDNAME	VARCHAR(128)		索引の非修飾名。
COLNAME	VARCHAR(128)		列の名前。
COLSEQ	SMALLINT		索引内の列の位置番号 (最初の位置は 1 です。)
COLORDER	CHAR(1)		この索引列の値の順序。使用できる値は以下のとおりです。 • A = 昇順 • D = 降順 • I = INCLUDE 列 (順序は無視される)
COLLATIONSCHEMA	VARCHAR(128)	Y	ストリング・タイプの場合は、列の照合のスキーマ名。それ以外の場合は NULL 値。
COLLATIONNAME	VARCHAR(128)	Y	ストリング・タイプの場合は、列の照合の非修飾名。それ以外の場合は NULL 値。

SYSCAT.INDEXDEP

各行は、他のオブジェクトに対する索引の従属関係を表します。索引は、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更は索引に影響します。

表 97. SYSCAT.INDEXDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引のスキーマ名。
INDNAME	VARCHAR(128)		索引の非修飾名。
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • B = トリガー • F = ルーチン・インスタンス • H = 階層表 • K = パッケージ • L = デタッチされた表 • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • Q = シーケンス • R = 構造化タイプ • S = マテリアライズ照会表 • T = 表 (型付きではない) • U = 型付き表 • V = ビュー (型付きではない) • W = 型付きビュー • X = 索引拡張 • Z = XSR オブジェクト • v = グローバル変数
BSCHEMA	VARCHAR(128)		従属関係があるオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		従属関係があるオブジェクトの非修飾名。ルーチン (BTYPE = 'F') の場合、これは特定名です。
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、従属索引で必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.INDEXES

各行は、索引を表します。型付き表の索引は、2 つの行で表されます。1 つは型付き表の「論理索引」用、もう 1 つは階層表の「階層索引」用です。

表 98. SYSCAT.INDEXES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引のスキーマ名。
INDNAME	VARCHAR(128)		索引の非修飾名。
OWNER	VARCHAR(128)		索引作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
TABSCHEMA	VARCHAR(128)		索引が定義されている表またはニックネームのスキーマ名。
TABNAME	VARCHAR(128)		索引が定義されている表またはニックネームの非修飾名。
COLNAMES	VARCHAR(640)		この列は使用されなくなりました。次のリリースで除去されます。詳しくは、SYSCAT.INDEXCOLUSE を参照してください。
UNIQUERULE	CHAR(1)		ユニーク規則。 • D = 重複を許可する • U = ユニーク • P = 主キーをインプリメントする
MADE_UNIQUE	CHAR(1)		• N = 索引は作成時のまま • Y = この索引は元は非ユニークだったが、ユニーク・キー制約または主キー制約をサポートするために、ユニーク索引に変換された。制約がドロップされると、この索引は非ユニークに戻る。
COLCOUNT	SMALLINT		キー内の列数と組み込み列 (存在する場合) の数の合計。
UNIQUE_COLCOUNT	SMALLINT		ユニーク・キーに必要な列の数。常に <= COLCOUNT になり、組み込み列がある場合に限って < COLCOUNT になります。索引にユニーク・キーがない (つまり索引が重複を許可する) 場合は -1 です。

表 98. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
INDEXTYPE[5]	CHAR(4)		索引のタイプ。 • BLOK = ブロック索引 • CLUS = クラスタリング索引 (新しく挿入された行の物理的配置を制御する) • DIM = ディメンション・ブロック索引 • REG = 通常の索引 • XPTH = XML パス索引 • XRGN = XML 領域索引 • XVIL = XML 列の索引 (論理) • XVIP = XML 列の索引 (物理)
ENTRYTYPE	CHAR(1)		• H = この行は階層表の索引を表す • L = この行は型付き表の論理索引を表す • ブランク = この行は非型付き表の索引を表す
PCTFREE	SMALLINT		索引を最初に作成する際に予約する各索引ページのパーセント。このスペースは、索引の作成後のデータ挿入に使用可能です。
IID	SMALLINT		索引の ID。
NLEAF	BIGINT		リーフ・ページの数。統計が収集されていない場合は -1。
NLEVELS	SMALLINT		索引レベルの数。統計が収集されていない場合は -1。
FIRSTKEYCARD	BIGINT		最初のキーの値の種類数。統計が収集されていない場合は -1。
FIRST2KEYCARD	BIGINT		索引の最初の 2 つの列を使用するキーの種類数。統計が収集されていない場合、または適用されない場合は -1。
FIRST3KEYCARD	BIGINT		索引の最初の 3 つの列を使用するキーの種類数。統計が収集されていない場合、または適用されない場合は -1。
FIRST4KEYCARD	BIGINT		索引の最初の 4 つの列を使用するキーの種類数。統計が収集されていない場合、または適用されない場合は -1。
FULLKEYCARD	BIGINT		全キー値の種類数。統計が収集されていない場合は -1。
CLUSTERRATIO[3]	SMALLINT		索引によるデータ・クラスタリングの程度。統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1 (それらの場合は CLUSTERFACTOR の方が使用されます)。
CLUSTERFACTOR[3]	DOUBLE		より高い計算精度のクラスタリング。統計を収集していない場合、あるいはニックネームに索引が定義されている場合は -1。

表 98. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SEQUENTIAL_PAGES	BIGINT		索引キー順にディスクに存在し、それらの間にほとんど（またはまったく）大きなギャップがないようなリーフ・ページの数。統計が収集されていない場合は -1。
DENSITY	INTEGER		索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0 から 100 の整数)。統計が収集されていない場合は -1。
USER_DEFINED	SMALLINT		この索引がユーザー定義であってドロップされていない場合は 1、それ以外の場合は 0。
SYSTEM_REQUIRED	SMALLINT		• 次の条件のいずれかを満たす場合は 1。 – この索引が主キー制約またはユニーク・キー制約で必要である、またはこの索引がマルチディメンション・クラスタリング (MDC) 表のディメンション・ブロック索引または複合ブロック索引である。 – これは型付き表のオブジェクト ID (OID) 列に対する索引である。 • 次の条件の両方を満たす場合は 2。 – この索引は主キー制約またはユニーク・キー制約に必要です。または、この索引は MDC 表に対するディメンション・ブロック索引またはコンポジット・ブロック索引です。 – これが型付き表の OID 列上の索引である。 • それ以外の場合は 0。
CREATE_TIME	TIMESTAMP		索引の作成された時刻。
STATS_TIME	TIMESTAMP	Y	この索引について記録されている統計値が最後に変更された時刻。統計が使用可能でない場合は、NULL 値。
PAGE_FETCH_PAIRS[3]	VARCHAR(520)		文字形式で表された整数ペアのリスト。それぞれのペアは、仮のバッファー内のページ数と、その仮のバッファーを使用した表のスキャンに必要なページ・フェッチ回数を表しています。データが利用できない場合は、長さゼロのストリング。
MINPCTUSED	SMALLINT		ゼロ以外の整数は、索引がオンライン・デフラグ可能であることを示すとともに、ページの使用スペースが最低どれだけのパーセンテージになったらページのマージを試行できるようになるかを示します。ゼロ値はページ・マージが試行されないことを示します。

表 98. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REVERSE_SCANS	CHAR(1)		• N = 索引は逆スキャンをサポートしない • Y = 索引は逆スキャンをサポートする
INTERNAL_FORMAT	SMALLINT		使用できる値は以下のとおりです。 • 1 = 索引にリバース・ポインターがない • 2 以上 = 索引にリバース・ポインターがある • 6 = 索引が複合ブロック索引
IESCHEMA	VARCHAR(128)	Y	索引拡張のスキーマ名。通常の索引の場合は NULL。
IENAME	VARCHAR(128)	Y	索引拡張の非修飾名。通常の索引の場合は NULL。
IEARGUMENTS	CLOB(64K)	Y	索引の作成時に指定されるパラメーターの外部情報。通常の索引の場合は NULL。
INDEX_OBJECTID	INTEGER		索引オブジェクトの ID。
NUMRIDS	BIGINT		索引内の行 ID (RID) またはブロック ID (BID) の合計数。不明の場合、-1。
NUMRIDS_DELETED	BIGINT		削除対象としてマークされている、索引内の行 ID (またはブロック ID) の合計数 (すべての ID が削除対象としてマークされている、リーフ・ページ上の ID は除く)。
NUM_EMPTY_LEAFS	BIGINT		すべての行 ID (またはブロック ID) が削除対象としてマークされている、索引リーフ・ページの合計数。
AVERAGE_RANDOM_FETCH_PAGES[1,2]	DOUBLE		索引を使用してフェッチする際の、順次ページ・アクセス間のランダム表ページの平均数。不明の場合は -1。
AVERAGE_RANDOM_PAGES[2]	DOUBLE		順次ページ・アクセス間のランダム表ページの平均数。不明の場合は -1。
AVERAGE_SEQUENCE_GAP[2]	DOUBLE		索引ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、索引ページ・シーケンスの間でランダムにフェッチしなければならない索引ページの平均数を表します。不明の場合は -1。
AVERAGE_SEQUENCE_FETCH_GAP[1,2]	DOUBLE		索引を使用してフェッチする際の、表ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、一連の表ページの間でランダムにフェッチしなければならない表ページの平均数を表します。不明の場合は -1。
AVERAGE_SEQUENCE_PAGES[2]	DOUBLE		順次にアクセス可能な索引ページの平均数 (つまり、順番になっているものとしてプリフェッチャーが検出する索引ページの数)。不明の場合は -1。

表 98. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
AVERAGE_SEQUENCE_FETCH_PAGES[1,2]	DOUBLE		索引を使用してフェッチする際の、順次にアクセス可能な表ページの平均数 (つまり、プリフェッチャーが順次として検出する表ページの数)。不明の場合は -1。
TBSPACEID	INTEGER		索引表スペースの ID。
LEVEL2PCTFREE	SMALLINT		索引を最初に作成する際に予約する索引レベル 2 の各ページのパーセント。このスペースは、索引の作成後に行う挿入用に使用可能です。
PAGESPLIT	CHAR(1)		索引ページ分割の振る舞い。 • H = 高 • L = 低 • S = 対称
AVGPARTITION_CLUSTERRATIO[3]	SMALLINT		単一のデータ・パーティション内でのデータ・クラスタリングの程度。表がパーティション化されていない場合、統計が収集されていない場合、または詳細な索引統計が収集されている場合 (その場合は AVGPARTITION_CLUSTERFACTOR の方が使用されます) は、-1。
AVGPARTITION_CLUSTERFACTOR[3]	DOUBLE		単一のデータ・パーティション内でのクラスタリングの程度の詳細測定値。表がパーティション化されていない場合、統計が収集されていない場合、あるいはニックネームに索引が定義されている場合は -1。
AVGPARTITION_PAGE_FETCH_PAIRS[3]	VARCHAR(520)		文字形式の整数ペアのリスト。各ペアは、潜在的なバッファー・プール・サイズと、表の単一データ・パーティションにアクセスするのに必要なページのフェッチ回数との対応を示します。データが利用できない場合、または表がパーティション化されていない場合は、長さゼロのストリング。
DATAPARTITION_CLUSTERFACTOR	DOUBLE		データ・パーティションに関しての索引キーの「クラスタリング」を測定する統計。これは 0 から 1 の間の数値で、1 は完全なクラスタリングを表し、0 はクラスタリングがないことを表します。
INDCARD	BIGINT		索引のカーディナリティー。表の行と索引項目との間に 1 対 1 の関係がない索引の場合、これは表のカーディナリティーと異なる場合があります。
OS_PTR_SIZE	INTEGER		索引作成時のプラットフォーム・ワード・サイズ。 • 32 = 32 ビット • 64 = 64 ビット

表 98. SYSCAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTSTATISTICS	CHAR(1)		索引作成時の統計収集方法を指定します。 • D = 詳細な索引統計を収集する • S = サンプル化詳細索引統計を収集する • Y = 基本索引統計を収集する • ブランク = 索引統計を収集しない
DEFINER[4]	VARCHAR(128)		索引作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DMS 表スペースの使用時には、この統計は計算されません。
2. LOAD...STATISTICS YES または CREATE INDEX...COLLECT STATISTICS の操作時や、データベース構成パラメーター *seqdetect* がオフになっているときは、プリフェッチ統計は集められません。
3. AVGPARTITION_CLUSTERRATIO、AVGPARTITION_CLUSTERFACTOR、および AVGPARTITION_PAGE_FETCH_PAIRS は、単一のデータ・パーティション内でのクラスタリング (ローカル・クラスタリング) の程度を測定します。CLUSTERRATIO、CLUSTERFACTOR、および PAGE_FETCH_PAIRS は、表全体におけるクラスタリング (グローバル・クラスタリング) の程度を測定します。表パーティション・キーが索引キーの接頭部でない場合、あるいは表パーティション・キーと索引キーが論理的に互いに独立している場合、グローバル・クラスタリングの値とローカル・クラスタリングの値の差が大きくなる場合があります。
4. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。
5. XPTH、XRGN、および XVIP 索引は索引メタデータを戻すど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も認識されません。

SYSCAT.INDEXEXPLOITRULES

各行は、索引活用規則を表します。

表 99. SYSCAT.INDEXEXPLOITRULES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FUNCID	INTEGER		関数の ID。
SPECID	SMALLINT		述部指定の番号。
IESCHEMA	VARCHAR(128)		索引拡張のスキーマ名。
IENAME	VARCHAR(128)		索引拡張の非修飾名。
RULEID	SMALLINT		活用規則の ID。
SEARCHMETHODID	SMALLINT		特定の索引拡張での検索メソッドの ID。
SEARCHKEY	VARCHAR(640)		索引を活用するのに使用するキー。
SEARCHARGUMENT	VARCHAR (2700)		索引を活用するのに使用する検索引数。
EXACT	CHAR(1)		• N = 索引検索は述部の評価の点で厳密でない • Y = 索引検索は述部の評価の点で厳密である

SYSCAT.INDEXEXTENSIONDEP

各行は、他のオブジェクトに対する索引拡張の従属関係を表します。索引拡張は、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更は索引拡張に影響します。

表 100. SYSCAT.INDEXEXTENSIONDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
IESCHEMA	VARCHAR(128)		索引拡張のスキーマ名。
IENAME	VARCHAR(128)		索引拡張の非修飾名。
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • B = トリガー • F = ルーチン・インスタンス • H = 階層表 • K = パッケージ • L = デタッチされた表 • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • Q = シーケンス • R = 構造化タイプ • S = マテリアライズ照会表 • T = 表 (型付きではない) • U = 型付き表 • V = ビュー (型付きではない) • W = 型付きビュー • X = 索引拡張 • Z = XSR オブジェクト • v = グローバル変数
BSCHEMA	VARCHAR(128)		従属関係があるオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		従属関係があるオブジェクトの非修飾名。ルーチン (BTYPE = 'F') の場合、これは特定名です。
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、従属索引拡張で必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.INDEXEXTENSIONMETHODS

各行は、検索メソッドを表します。索引拡張には、複数の検索メソッドを含めることができます。

表 101. SYSCAT.INDEXEXTENSIONMETHODS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
METHODNAME	VARCHAR(128)		検索メソッドの名前。
METHODID	SMALLINT		索引拡張内のメソッドの数。
IESCHEMA	VARCHAR(128)		このメソッドが定義されている索引拡張のスキーマ名。
IENAME	VARCHAR(128)		このメソッドが定義されている索引拡張の非修飾名。
RANGEFUNCSCHEMA	VARCHAR(128)		範囲指定関数のスキーマ名。
RANGEFUNCNAME	VARCHAR(128)		範囲指定関数の非修飾名。
RANGESPECIFICNAME	VARCHAR(128)		範囲指定関数の関数固有名。
FILTERFUNCSCHEMA	VARCHAR(128)	Y	フィルター関数のスキーマ名。
FILTERFUNCNAME	VARCHAR(128)	Y	フィルター関数の非修飾名。
FILTERSPECIFICNAME	VARCHAR(128)	Y	フィルター関数の関数固有名。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.INDEXEXTENSIONPARMS

各行は、索引拡張のインスタンス・パラメーターまたはソース・キー列を表します。

表 102. SYSCAT.INDEXEXTENSIONPARMS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
IESCHEMA	VARCHAR(128)		索引拡張のスキーマ名。
IENAME	VARCHAR(128)		索引拡張の非修飾名。
ORDINAL	SMALLINT		パラメーターまたはキー列のシーケンス番号。
PARMNAME	VARCHAR(128)		パラメーターまたはキー列の名前。
TYPESCHEMA	VARCHAR(128)		パラメーターまたはキー列のデータ・タイプのスキーマ名。
TYPENAME	VARCHAR(128)		パラメーターまたはキー列のデータ・タイプの非修飾名。
LENGTH	INTEGER		パラメーターまたはキー列のデータ・タイプの長さ。
SCALE	SMALLINT		パラメーターまたはキー列のデータ・タイプの位取り。該当しない場合は、0。
PARMTYPE	CHAR(1)		• K = ソース・キー列 • P = 索引拡張のインスタンス・パラメーター
CODEPAGE	SMALLINT		索引拡張のインスタンス・パラメーターのコード・ページ。ストリング・タイプでない場合は 0。
COLLATIONSCHEMA	VARCHAR(128)	Y	ストリング・タイプの場合は、パラメーターの照合のスキーマ名。それ以外の場合は NULL 値。
COLLATIONNAME	VARCHAR(128)	Y	ストリング・タイプの場合は、パラメーターの照合の非修飾名。それ以外の場合は NULL 値。

SYSCAT.INDEXEXTENSIONS

各行は、索引拡張を表します。

表 103. SYSCAT.INDEXEXTENSIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
IESCHEMA	VARCHAR(128)		索引拡張のスキーマ名。
IENAME	VARCHAR(128)		索引拡張の非修飾名。
OWNER	VARCHAR(128)		索引拡張の作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
CREATE_TIME	TIMESTAMP		索引拡張が定義された時刻。
KEYGENFUNCSCHEMA	VARCHAR(128)		キー生成関数のスキーマ名。
KEYGENFUNCNAME	VARCHAR(128)		キー生成関数の非修飾名。
KEYGENSPECIFICNAME	VARCHAR(128)		キー生成関数インスタンスの名前 (システム生成の場合もある)。
TEXT	CLOB(2M)		CREATE INDEX EXTENSION ステートメントのテキスト全体。
DEFINER[1]	VARCHAR(128)		索引拡張の作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.INDEXOPTIONS

各行は、索引固有のオプション値を表します。

表 104. SYSCAT.INDEXOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		索引のスキーマ名。
INDNAME	VARCHAR(128)		索引の非修飾名。
OPTION	VARCHAR(128)		索引オプションの名前。
SETTING	VARCHAR (2048)		索引オプションの値。

SYSCAT.INDEXXMLPATTERNS

各行は、XML 列に対する索引内のパターン節を表します。

表 105. SYSCAT.INDEXXMLPATTERNS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
INDSCHEMA	VARCHAR(128)		論理索引のスキーマ名。
INDNAME	VARCHAR(128)		論理索引の非修飾名。
PINDNAME	VARCHAR(128)		物理索引の非修飾名。
PINDID	SMALLINT		物理索引の ID。
TYPEMODEL	CHAR(1)		• Q = SQL DATA TYPE (無効値を無視) • R = SQL DATA TYPE (無効値をリジェクト)
DATATYPE	VARCHAR(128)		データ・タイプの名前。
HASHED	CHAR(1)		値がハッシュされるかどうかを示します。 • N = ハッシュされない • Y = ハッシュされる
LENGTH	SMALLINT		VARCHAR(*n*) の長さ。その他の場合 0。
PATTERNID	SMALLINT		パターンの ID。
PATTERN	CLOB(2M)	Y	パターンの定義。

注:

1. XML 列に対する索引の作成時に、XML パターン情報を使用する論理索引が作成され、物理 B ツリー索引が、その論理索引をサポートするための DB2 生成のキー列と共に作成されます。物理索引は、CREATE INDEX ステートメントの xmltype 節で指定されるデータ・タイプをサポートするために作成されます。

SYSCAT.KEYCOLUSE

各行は、ユニーク制約、主キー制約、または外部キー制約で定義されたキーに関与する列を表します。

表 106. SYSCAT.KEYCOLUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONSTNAME	VARCHAR(128)		制約の名前。
TABSCHEMA	VARCHAR(128)		列の入った表のスキーマ名。
TABNAME	VARCHAR(128)		列の入った表の非修飾名。
COLNAME	VARCHAR(128)		列の名前。
COLSEQ	SMALLINT		制約のキー内の列の位置番号 (最初の位置は 1 です)。制約に既存の索引が使用される場合、この値は索引の列の位置番号です。

SYSCAT.NAMEMAPPINGS

各行は、「論理」オブジェクト (型付き表または型付きビュー、およびその列と索引 (継承列を含む)) と、その論理オブジェクトを実装するための「実装」オブジェクト (階層表または階層ビュー、およびその列と索引) との間のマッピングを表します。

表 107. SYSCAT.NAMEMAPPINGS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TYPE	CHAR(1)		• C = 列 • I = 索引 • U = 型付き表
LOGICAL_SCHEMA	VARCHAR(128)		論理オブジェクトのスキーマ名。
LOGICAL_NAME	VARCHAR(128)		論理オブジェクトの非修飾名。
LOGICAL_COLNAME	VARCHAR(128)	Y	TYPE = 'C' の場合、論理列の名前。その他の場合は NULL 値。
IMPL_SCHEMA	VARCHAR(128)		論理オブジェクトを実装する実装オブジェクトのスキーマ名。
IMPL_NAME	VARCHAR(128)		論理オブジェクトを実装する実装オブジェクトの非修飾名。
IMPL_COLNAME	VARCHAR(128)	Y	TYPE = 'C' の場合、実装列の名前。その他の場合は NULL 値。

SYSCAT.NICKNAMES

各行はニックネームを表します。

表 108. SYSCAT.NICKNAMES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		ニックネームのスキーマ名。
TABNAME	VARCHAR(128)		ニックネームの非修飾名。
OWNER	VARCHAR(128)		ニックネームを作成したユーザー。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
STATUS	CHAR(1)		オブジェクトの状況。 • C = SET INTEGRITY ペンディング • N = 正常 • X = 作動不能
CREATE_TIME	TIMESTAMP		オブジェクトが作成された時刻。
STATS_TIME	TIMESTAMP	Y	このオブジェクトについて記録されている統計値が最後に変更された時刻。統計が収集されていない場合は、NULL 値。
COLCOUNT	SMALLINT		列の数。継承された列がある場合は、それも含む。
TABLEID	SMALLINT		内部論理オブジェクト ID。
TBSPACEID	SMALLINT		このオブジェクトの PRIMARY 表スペースの内部論理 ID。
CARD	BIGINT		行の総数。統計が収集されていない場合は -1。
NPAGES	BIGINT		ニックネームの行が存在しているページの総数。統計が収集されていない場合は -1。
FPAGES	BIGINT		ページの総数。統計が収集されていない場合は -1。
OVERFLOW	BIGINT		オーバーフロー・レコードの総数。統計が収集されていない場合は -1。
PARENTS	SMALLINT	Y	このオブジェクトの親表の数。つまり、このオブジェクトが従属オブジェクトになっている参照制約の数。
CHILDREN	SMALLINT	Y	このオブジェクトに従属している表の数。つまり、このオブジェクトが親になっている参照制約の数。
SELFREFS	SMALLINT	Y	このオブジェクトに対する自己参照になっている参照制約の数。つまり、このオブジェクトが親と従属の両方を兼ねる参照制約の数。
KEYCOLUMNS	SMALLINT	Y	主キーを構成する列の数。
KEYINDEXID	SMALLINT	Y	主キー索引の索引 ID。主キーがない場合は、0 または NULL 値。

表 108. SYSCAT.NICKNAMES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
KEYUNIQUE	SMALLINT		このオブジェクトに定義されたユニーク・キー制約 (主キー制約を除く) の数。
CHECKCOUNT	SMALLINT		このオブジェクトに定義されたチェック制約の数。
DATACAPTURE	CHAR(1)		• L = ニックネームはデータ複製 (LONG VARCHAR および LONG VARGRAPHIC 列の複製を含む) に関与している • N = ニックネームはデータ複製に関与していない • Y = ニックネームはデータ複製に関与している
CONST_CHECKED	CHAR(32)		• バイト 1 は、外部キー制約を表します。 • バイト 2 は、チェック制約を表します。 • バイト 5 は、マテリアライズ照会表を表します。 • バイト 6 は生成される列を表します。 • バイト 7 は、ステージング表を表します。 • バイト 8 は、データ・パーティション制約を表します。 • 他のバイトは、将来の使用のために予約されています。 使用できる値は以下のとおりです。 • F = バイト 5 では、マテリアライズ照会表をインクリメンタル更新できない。バイト 7 では、ステージング表の内容は不完全で、関連したマテリアライズ照会表のインクリメンタル更新に使用することができない。 • N = チェックなし • U = ユーザーによるチェック • W = 表が SET INTEGRITY ペンディング状態になったときに 'U' の状態だった • Y = システムによるチェック
PARTITION_MODE	CHAR(1)		将来の使用のために予約済み。
STATISTICS_PROFILE	CLOB(10M)	Y	オブジェクトの統計プロファイルの登録に使用された RUNSTATS コマンド。

表 108. SYSCAT.NICKNAMES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACCESS_MODE	CHAR(1)		オブジェクトのアクセス制限の状態。これらの状態は、SET INTEGRITY によりペンディング状態にあるオブジェクト、または SET INTEGRITY ステートメントによって処理されたオブジェクトにのみ適用されます。使用できる値は以下のとおりです。 • D = データの移動不可 • F = フル・アクセス権限 • N = アクセス不可 • R = 読み取り専用アクセス
CODEPAGE	SMALLINT		オブジェクトのコード・ページ。これが、すべての文字の列、トリガー、チェック制約、および式で生成された列のデフォルトのコード・ページです。
REMOTE_TABLE	VARCHAR(128)	Y	ニックネーム作成の対象になった、特定のデータ・ソース・オブジェクト (表またはビューなど) の非修飾名。
REMOTE_SCHEMA	VARCHAR(128)	Y	ニックネーム作成の対象になった、特定のデータ・ソース・オブジェクト (表またはビューなど) のスキーマ名。
SERVERNAME	VARCHAR(128)	Y	ニックネーム作成の対象となった、表またはビューを含むデータ・ソースの名前。
REMOTE_TYPE	CHAR(1)	Y	データ・ソースにあるオブジェクトのタイプ。 • A = 別名 • N = ニックネーム • S = マテリアライズ照会表 • T = 表 (非型付き) • V = ビュー (非型付き)
CACHINGALLOWED	VARCHAR(1)		• N = キャッシングは許可されない • Y = キャッシングは許可される
DEFINER[1]	VARCHAR(128)		表、ビュー、別名、またはニックネーム作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.PACKAGEAUTH

各行は、パッケージに対して 1 つ以上の特権を付与されたユーザーまたはグループを表します。

表 109. SYSCAT.PACKAGEAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
PKGSCHEMA	VARCHAR(128)		パッケージのスキーマ名。
PKGNAME	VARCHAR(128)		パッケージの非修飾名。
CONTROLAUTH	CHAR(1)		CONTROL 特権。 • N = 保有しない • Y = 保有する
BINDAUTH	CHAR(1)		パッケージをバインドする特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
EXECUTEAUTH	CHAR(1)		パッケージを実行する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

SYSCAT.PACKAGEDEP

各行は、他のオブジェクトに対するパッケージの従属関係を表します。パッケージは、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更はパッケージに影響します。

表 110. SYSCAT.PACKAGEDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
PKGSCHEMA	VARCHAR(128)		パッケージのスキーマ名。
PKGNAME	VARCHAR(128)		パッケージの非修飾名。
BINDER	VARCHAR(128)		パッケージをバインドしたユーザー。
BINDERTYPE	CHAR(1)		• U = バインドしたのは個々のユーザー
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • D = サーバー定義 • F = ルーチン・インスタンス • I = 索引 • M = 関数マッピング • N = ニックネーム • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • P = ページ・サイズ • Q = シーケンス・オブジェクト • R = ユーザー定義のデータ・タイプ • S = マテリアライズ照会表 • T = 表 (非型付き) • U = 型付き表 • V = ビュー (非型付き) • W = 型付きビュー • Z = XSR オブジェクト • n = データベース・パーティション・グループ • v = グローバル変数
BSCHEMA	VARCHAR(128)		パッケージが従属しているオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		パッケージが従属しているオブジェクトの非修飾名。
TABAUTH	SMALLINT	Y	BTYPE が 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、このパッケージで必要な特権 (SELECT、INSERT、UPDATE、または DELETE) をエンコードします。

表 110. SYSCAT.PACKAGEDEP カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
UNIQUE_ID	CHAR(8) FOR BIT DATA		同じ名前を持つ複数のパッケージが存在している場合の、特定のパッケージの ID。
PKGVERSION	VARCHAR(64)	Y	パッケージのバージョン ID。

注:

1. 従属関係のある関数インスタンスがドロップされると、パッケージは「作動不能」状態になり、明示的に再バインドする必要があります。従属関係のあるほかのオブジェクトがドロップされると、パッケージは「無効」状態になり、最初の参照時に、システムによってパッケージの再バインドが自動的に試みられます。

SYSCAT.PACKAGES

各行は、アプリケーション・プログラムをバインドして作成されたパッケージを表します。

表 111. SYSCAT.PACKAGES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
PKGSCHEMA	VARCHAR(128)		パッケージのスキーマ名。
PKGNAME	VARCHAR(128)		パッケージの非修飾名。
BOUNDBY	VARCHAR(128)		パッケージをバインドしたユーザーの許可 ID。
BOUNDBYTYPE	CHAR(1)		• U = バインドしたのは個々のユーザー
OWNER	VARCHAR(128)		パッケージのバインド時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
DEFAULT_SCHEMA	VARCHAR (128)		静的 SQL ステートメントの非修飾名に使用されるデフォルト・スキーマの名前。
VALID[1]	CHAR (1)		• N = 再バインドが必要 • V = 実行時に妥当性検査 • X = パッケージが従属している関数インスタンスがドロップされたので、パッケージは作動不能。明示的再バインドが必要。 • Y = 有効
UNIQUE_ID	CHAR (8) FOR BIT DATA		同じ名前を持つ複数のパッケージが存在している場合の、特定のパッケージの ID。
TOTAL_SECT	SMALLINT		パッケージのセクションの数。
FORMAT	CHAR (1)		パッケージに関連した日付と時刻のフォーマット。 • 0 = クライアントのテリトリー・コードに関連したフォーマット • 1 = USA • 2 = EUR • 3 = ISO • 4 = JIS • 5 = ローカル
ISOLATION	CHAR (2)	Y	分離レベル。 • CS = カーソル固定 • RR = 反復可能読み取り • RS = 読み取り固定 • UR = 非コミット読み取り

表 111. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
BLOCKING	CHAR (1)	Y	カーソル・ブロッキング・オプション。 • B = すべてのカーソルをブロック化 • N = ブロッキングなし • U = 確定カーソルをブロック化
INSERT_BUF	CHAR (1)		INSERT BIND オプション (パーティション・データベース・システムに適用される) の設定。 • N = 挿入内容はバッファーに入れられない • Y = データベース・パーティション間のトラフィックを最小化するため、挿入内容は、コーディネーター・データベース・パーティションでバッファーに入れられる
LANG_LEVEL	CHAR(1)	Y	LANGLEVEL BIND オプションの設定。 • 0 = SAA1 • 1 = MIA • 2 = SQL92E
FUNC_PATH	CLOB(2K)		このパッケージの最後のバインド操作で使用された SQL パス。これは再バインド操作のデフォルトのパスとして使用されます。バージョン 2 より前のパッケージでは SYSIBM。
QUERYOPT	INTEGER		このパッケージをバインドした最適化クラス。再バインド操作に使用されます。
EXPLAIN_LEVEL	CHAR(1)		EXPLAIN または EXPLSNAP BIND オプションを使用して、 Explain が要求されたか否か。 • P = パッケージ選択レベル • ブランク = Explain が要求されていない
EXPLAIN_MODE	CHAR(1)		EXPLAIN BIND オプションの値。 • A = ALL • N = No • R = REOPT • Y = Yes
EXPLAIN_SNAPSHOT	CHAR(1)		EXPLSNAP BIND オプションの値。 • A = ALL • N = No • R = REOPT • Y = Yes

表 111. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SQLWARN	CHAR(1)		動的 SQL ステートメントの結果の正の SQLCODE がアプリケーションに戻されるかどうかを示します。 • N = No (抑制される) • Y = Yes
SQLMATHWARN	CHAR(1)		バインド時の *dft_sqlmathwarn* データベース構成パラメーターの値。可能な限り照会処理を継続させて、算術計算エラーや検索変換エラー時に警告と NULL 値 (標識 -2) を戻すかどうかを示します。 • N = No (エラーが戻される) • Y = Yes (警告が戻される)
EXPLICIT_BIND_TIME	TIMESTAMP		このパッケージを最後に明示的にバインドまたは再バインドした時刻。パッケージを暗黙に再バインドすると、この時点以降に作成された関数インスタンスは選択されません。
LAST_BIND_TIME	TIMESTAMP		このパッケージが最後に明示的にバインドまたは再バインドされた時刻。 Explain データの妥当性検査に使用されます。
CODEPAGE	SMALLINT		バインド時のアプリケーションのコード・ページ。不明の場合は -1。
COLLATIONSCHEMA	VARCHAR(128)		パッケージの照合のスキーマ名。
COLLATIONNAME	VARCHAR(128)		パッケージの照合の非修飾名。
COLLATIONSCHEMA_ ORDERBY	VARCHAR(128)		パッケージの ORDER BY 節の照合のスキーマ名。
COLLATIONNAME_ORDERBY	VARCHAR(128)		パッケージの ORDER BY 節の照合の非修飾名。
DEGREE	CHAR(5)		パッケージのバインド時に指定されたパーティション内並列処理の度合い。 • 1 = 並列処理なし • 2 から 32767 = ユーザー指定の制限 • ANY = システムによって決定される度合い (制限は指定なし)
MULTINODE_PLANS	CHAR(1)		• N = パッケージはパーティション・データベース環境でバインドされなかった • Y = パッケージはパーティション・データベース環境でバインドされた

表 111. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
INTRA_PARALLEL	CHAR(1)		パッケージ内の静的 SQL ステートメントによるパーティション内並列処理の使用。 • F = このパッケージ内の 1 つまたは複数の静的 SQL ステートメントはパーティション内並列処理を使用可能。この並列処理は、パーティション内並列処理を使用するように構成されていないシステムでは使用不可。 • N = 静的 SQL ステートメントはパーティション内並列処理を使用しない。 • Y = パッケージ内の 1 つまたは複数の静的 SQL ステートメントがパーティション内並列処理を使用する。
VALIDATE	CHAR(1)		妥当性検査が実行時まで据え置きできるかどうかを示します。 • B = すべての検査はバインド時に行う必要がある • R = バインド時に存在しない表、ビュー、および特権の妥当性検査は実行時に行われる

表 111. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
DYNAMICRULES	CHAR(1)		• B = BIND (動的 SQL ステートメントは DYNAMICRULES BIND 動作で実行される) • D = DEFINERBIND (パッケージがルーチン・コンテキスト内で実行される場合は、パッケージ内の動的 SQL ステートメントは DEFINE 動作で実行される。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントは BIND 動作で実行される) • E = DEFINERRUN (パッケージがルーチン・コンテキスト内で実行される場合は、パッケージ内の動的 SQL ステートメントは DEFINE 動作で実行される。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントは RUN 動作で実行される) • H = INVOKEBIND (パッケージがルーチン・コンテキスト内で実行される場合は、パッケージ内の動的 SQL ステートメントは INVOKE 動作で実行される。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントは BIND 動作で実行される) • I = INVOKERUN (パッケージがルーチン・コンテキスト内で実行される場合は、パッケージ内の動的 SQL ステートメントは INVOKE 動作で実行される。パッケージがルーチン・コンテキスト内で実行されない場合は、パッケージ内の動的 SQL ステートメントは RUN 動作で実行される) • R = RUN (動的 SQL ステートメントは RUN 動作で実行される。これがデフォルト)
SQLERROR	CHAR(1)		パッケージをバインドまたは再バインドした最新のサブコマンドの SQLERROR オプション。 • C = CONTINUE (SQL ステートメントのバインディング中にエラーが生じても、パッケージを作成する) • N = NOPACKAGE (エラーが生じた場合、パッケージまたはバインド・ファイルを作成しない)

表 111. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REFRESHAGE	DECIMAL(20,6)		マテリアライズ照会表 (MQT) に対する REFRESH TABLE ステートメント実行から、その MQT が基本表の代わりに使用されるまでの最大時間を示す、タイム・スタンプ期間。
FEDERATED	CHAR(1)		• N = FEDERATED bind または prep オプションがオフ • U = FEDERATED bind または prep オプションが未指定 • N = FEDERATED bind または prep オプションがオン
TRANSFORMGROUP	VARCHAR(1024)	Y	TRANSFORM GROUP bind オプションの値。トランスフォーム・グループが指定されていない場合は NULL 値。
REOPTVAR	CHAR(1)		入力可変値を使って実行時にアクセス・パスが再決定されるかどうかを示します。 • A = OPEN または EXECUTE 要求ごとにアクセス・パスは再最適化される。 • N = アクセス・パスはバインド時に決定される。 • O = 最初の OPEN または EXECUTE 要求でのみアクセス・パスは再最適化される。その後はキャッシュされる。
OS_PTR_SIZE	INTEGER		パッケージの作成場所であるプラットフォームのワード・サイズ。 • 32 = パッケージは 32 ビット・パッケージ • 64 = パッケージは 64 ビット・パッケージ
PKGVERSION	VARCHAR(64)		パッケージのバージョン ID。
PKG_CREATE_TIME	TIMESTAMP		パッケージが最初にバインドされた時刻。
STATICREADONLY	CHAR(1)		静的カーソルが READ ONLY として扱われるかどうかを示します。使用できる値は以下のとおりです。 • N = 静的カーソルは、指定のステートメント・テキストと LANGLEVEL プリコンパイル・オプションの設定に基づいて通常生成される場合と同じ属性を取ります。 • Y = FOR UPDATE または FOR READ ONLY 節を含まない静的カーソルは、READ ONLY と見なされます。

表 111. SYSCAT.PACKAGES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
FEDERATED_ASYNCHRONY	INTEGER		非同期の限度 (プラン内の ATQ の数) を、パッケージをバインドしたときの bind オプションに従って示します。 • 0 = 非同期なし • *n* = ユーザー指定の制限 (32767 が最大) • -1 = 非同期の度合いはシステムが決定する • -2 = 非同期の度合いは指定されていない 非フェデレーテッド・システムの場合の値は 0 です。
OPTPROFILESCHEMA	VARCHAR(128)	Y	OPTPROFILE BIND オプションの一部として指定された最適化プロファイル・スキーマの値。
OPTPROFILENAME	VARCHAR(128)	Y	OPTPROFILE BIND オプションの一部として指定された最適化プロファイル名の値。
DEFINER[2]	VARCHAR(128)		パッケージのバインド時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. 従属関係のある関数インスタンスがドロップされると、パッケージは「作動不能」状態になり、明示的に再バインドする必要があります。従属関係のあるほかのオブジェクトがドロップされると、パッケージは「無効」状態になり、最初の参照時に、システムによってパッケージの再バインドが自動的に試みられます。
2. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.PARTITIONMAPS

各行は、表の分散キーのハッシュに基づいて、データベース・パーティション・グループ内のパーティションの間で表の行を分散するために使用される分散マップを表します。

表 112. SYSCAT.PARTITIONMAPS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
PMAP_ID	SMALLINT		分散マップの ID。
PARTITIONMAP	BLOB (8192)		分散マップ。複数のパーティション・データベースのデータベース・パーティション・グループの場合は、4096 個の 2 バイト整数から成るベクトル。単一のパーティション・データベースのデータベース・パーティション・グループの場合は、単一パーティションのパーティション番号を示す項目が 1 つあります。

SYSCAT.PASSTHRUAUTH

各行は、データ・ソースを照会するパススルー許可を付与されたユーザーまたはグループを表します。

表 113. SYSCAT.PASSTHRUAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
SERVERNAME	VARCHAR(128)		許可が付与されているデータ・ソースの名前。

SYSCAT.PREDICATESPECS

各行は述部指定を表します。

表 114. SYSCAT.PREDICATESPECS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
FUNCSCHEMA	VARCHAR(128)		関数のスキーマ名。
FUNCNAME	VARCHAR(128)		関数の非修飾名。
SPECIFICNAME	VARCHAR(128)		関数インスタンスの名前。
FUNCID	INTEGER		関数の ID。
SPECID	SMALLINT		この述部指定の数。
CONTEXTOP	CHAR (8)		比較演算子。組み込み関係演算子 (=、<、>、>= など) のいずれか。
CONTEXTEXP	CLOB(2M)		定数、または SQL 式。
FILTERTEXT	CLOB (32K)	Y	データ・フィルター式のテキスト。

SYSCAT.REFERENCES

各行は、参照整合性 (外部キー) 制約を表します。

表 115. SYSCAT.REFERENCES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONSTNAME	VARCHAR(128)		制約の名前。
TABSCHEMA	VARCHAR(128)		従属表のスキーマ名。
TABNAME	VARCHAR(128)		従属表の非修飾名。
OWNER	VARCHAR(128)		その制約を作成した許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
REFKEYNAME	VARCHAR(128)		親キーの名前。
REFTABSCHEMA	VARCHAR(128)		親表のスキーマ名。
REFTABNAME	VARCHAR(128)		親表の非修飾名。
COLCOUNT	SMALLINT		外部キーを構成する列の数。
DELETERULE	CHAR(1)		削除規則。 • A = NO ACTION (アクションなし) • C = CASCADE (削除規則 : カスケード) • N = SET NULL (NULL 設定) • R = RESTRICT (削除規則 : 削除禁止)
UPDATERULE	CHAR(1)		更新規則。 • A = NO ACTION (アクションなし) • R = RESTRICT (削除規則 : 削除禁止)
CREATE_TIME	TIMESTAMP		制約が定義された時刻。
FK_COLNAMES	VARCHAR(640)		この列は使用されなくなりました。将来のリリースで除去されます。詳しくは、SYSCAT.KEYCOLUSE を参照してください。
PK_COLNAMES	VARCHAR(640)		この列は使用されなくなりました。将来のリリースで除去されます。詳しくは、SYSCAT.KEYCOLUSE を参照してください。
DEFINER[1]	VARCHAR(128)		その制約を作成した許可 ID。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.ROLEAUTH

各行は、ユーザー、グループ、ロール、または PUBLIC に付与されたロールを表します。

表 116. SYSCAT.ROLEAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		ロールを付与した許可 ID。
GRANTORTYPE	CHAR(1)		• U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		ロールが付与された許可 ID。
GRANTEETYPE	CHAR(1)		• G = 被認可者はグループ • R = 被認可者はロール • U = 被認可者は個人ユーザー
ROLENAME	VARCHAR(128)		ロールの名前。
ROLEID	INTEGER		ロールの ID。
ADMIN	CHAR(1)		他のユーザーに対してロールを付与したり取り消したりするための特権、またはロールに対してコメントを付けるための特権。 • N = 保有しない • Y = 保有する

SYSCAT.ROLES

各行はロールを表します。

表 117. SYSCAT.ROLES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROLENAME	VARCHAR(128)		ロールの名前。
ROLEID	INTEGER		ロールの ID。
CREATE_TIME	TIMESTAMP		ロールが作成された時刻。
AUDITPOLICYID	INTEGER	Y	監査ポリシーの ID。
AUDITPOLICYNAME	VARCHAR(128)	Y	監査ポリシーの名前。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.ROUTINEAUTH

各行は、データベース内の特定のルーチン (関数、メソッド、またはプロシージャー) に対する、または特定のスキーマのすべてのルーチンに対する EXECUTE 特権を付与されているユーザーまたはグループを表します。

表 118. SYSCAT.ROUTINEAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。特権がシステムによって付与された場合は、'SYSIBM'。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
SCHEMA	VARCHAR(128)		ルーチンのスキーマ名。
SPECIFICNAME	VARCHAR (128)	Y	ルーチンの特定名を指定します。SPECIFICNAME が NULL で ROUTINETYPE が 'M' ではない場合、特権は SCHEMA で指定されたスキーマ内の、ROUTINETYPE で指定されたタイプのすべてのルーチンに適用されます。SPECIFICNAME が NULL で ROUTINETYPE が 'M' である場合、特権は TYPESCHEMA で指定されるスキーマ内の、TYPENAME で指定されるサブジェクト・タイプのすべてのメソッドに適用されます。SPECIFICNAME が NULL、ROUTINETYPE が 'M'、かつ TYPENAME および TYPESCHEMA の両方が NULL である場合、特権はスキーマ内のすべてのタイプのすべてのメソッドに適用されます。
TYPESCHEMA	VARCHAR(128)	Y	メソッドのタイプのスキーマ名。ROUTINETYPE が 'M' でない場合は、NULL 値。
TYPENAME	VARCHAR (128)	Y	メソッドのタイプの非修飾名。ROUTINETYPE が 'M' でない場合は、NULL 値。 TYPENAME が NULL で ROUTINETYPE が 'M' の場合、特権は、SCHEMA で指定されたスキーマにある、あらゆるサブジェクト・タイプのすべてのメソッドに適用されます。

表 118. SYSCAT.ROUTINEAUTH カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ROUTINETYPE	CHAR(1)		ルーチンのタイプ • F = 関数 • M = メソッド • P = プロシージャー
EXECUTEAUTH	CHAR(1)		ルーチンを実行する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
GRANT_TIME	TIMESTAMP		特権が付与された時刻。

SYSCAT.ROUTINEDEP

各行は、他のオブジェクトに対するルーチンの従属関係を表します。ルーチンは、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更はルーチンに影響します。

表 119. SYSCAT.ROUTINEDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		別のオブジェクトに従属するルーチンのスキーマ名。
SPECIFICNAME	VARCHAR(128)		別のオブジェクトに従属するルーチンの特定名。
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • B = トリガー • F = ルーチン・インスタンス • H = 階層表 • K = パッケージ • L = デタッチされた表 • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • Q = シーケンス • R = 構造化タイプ • S = マテリアライズ照会表 • T = 表 (型付きではない) • U = 型付き表 • V = ビュー (型付きではない) • W = 型付きビュー • X = 索引拡張 • Z = XSR オブジェクト • v = グローバル変数
BSCHEMA	VARCHAR(128)		従属関係があるオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		従属関係があるオブジェクトの非修飾名。ルーチン (BTYPE = 'F') の場合、これは特定名です。
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、従属ルーチンで必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。
ROUTINENAME	VARCHAR(128)		この列は使用されなくなりました。将来のリリースで除去されます。SPECIFICNAME を参照してください。

SYSCAT.ROUTINEOPTIONS

各行は、ルーチン固有のオプション値を表します。

表 120. SYSCAT.ROUTINEOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		ルーチンのスキーマ名。
ROUTINENAME	VARCHAR(128)		ルーチンの非修飾名。
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
OPTION	VARCHAR(128)		フェデレート・ルーチン・オプションの名前。
SETTING	VARCHAR (2048)		フェデレート・ルーチン・オプションの値。

SYSCAT.ROUTINEPARMOPTIONS

各行は、ルーチン・パラメーター固有のオプション値を表します。

表 121. SYSCAT.ROUTINEPARMOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		ルーチンのスキーマ名。
ROUTINENAME	VARCHAR(128)		ルーチンの非修飾名。
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
ORDINAL	SMALLINT		ルーチン・シグニチャー内でのパラメーターの位置。
OPTION	VARCHAR(128)		フェデレート・ルーチン・オプションの名前。
SETTING	VARCHAR (2048)		フェデレート・ルーチン・オプションの値。

SYSCAT.ROUTINEPARMS

各行は、SYSCAT.ROUTINES に定義されているルーチンの、パラメーターまたは結果を表します。

表 122. SYSCAT.ROUTINEPARMS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)	Y	ルーチンのスキーマ名。
ROUTINENAME	VARCHAR(128)	Y	ルーチンの非修飾名。
SPECIFICNAME	VARCHAR(128)	Y	ルーチン・インスタンスの名前 (システム生成の場合もある)。
PARMNAME	VARCHAR(128)	Y	パラメーターまたは結果の列の名前、あるいは名前が存在しない場合は NULL 値。
ROWTYPE	CHAR(1)		• B = 入力と出力の両方のパラメーター • C = キャスト後の結果 • O = 出力パラメーター • P = 入力パラメーター • R = キャスト前の結果
ORDINAL	SMALLINT		ROWTYPE = 'B'、'O'、または 'P' である場合、1 から始まる、ルーチン・シグニチャー内でのパラメーターの位置番号。ROWTYPE = 'R' であり、ルーチンが表を戻す場合、1 から始まる、結果表内の指定された列の位置番号。その他の場合、0。
TYPESCHEMA	VARCHAR(128)	Y	データ・タイプのスキーマ名。組み込みタイプのスキーマ名は 'SYSIBM' です。
TYPENAME	VARCHAR(128)	Y	データ・タイプの非修飾名。
LOCATOR	CHAR(1)		• N = パラメーターまたは結果は、ロケーターの形式で渡されない • Y = パラメーターまたは結果は、ロケーターの形式で渡される
LENGTH[1]	INTEGER		パラメーターまたは結果の長さ。パラメーターまたは結果がユーザー定義のデータ・タイプの場合は 0。
SCALE[1]	SMALLINT		パラメーターまたは結果の位取り。パラメーターまたは結果がユーザー定義のデータ・タイプの場合は 0。
CODEPAGE	SMALLINT		このパラメーターまたは結果のコード・ページ。該当しない場合、または FOR BIT DATA 属性を指定して宣言された文字データのパラメーターか結果の場合は 0。
COLLATIONSCHEMA	VARCHAR(128)	Y	ストリング・タイプの場合は、パラメーターの照合のスキーマ名。それ以外の場合は NULL 値。

表 122. SYSCAT.ROUTINEPARMS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLATIONNAME	VARCHAR(128)	Y	ストリング・タイプの場合は、パラメーターの照合の非修飾名。それ以外の場合は NULL 値。
CAST_FUNCSCHEMA	VARCHAR(128)	Y	引数また結果のキャストに使用される関数のスキーマ名。ソース関数および外部関数に適用され、それ以外の場合は NULL 値。
CAST_FUNCSPECIFIC	VARCHAR(128)	Y	引数または結果のキャストに使用される関数の非修飾名。ソース関数および外部関数に適用され、それ以外の場合は NULL 値。
TARGET_TYPESCHEMA	VARCHAR(128)	Y	パラメーターまたは結果のタイプが REFERENCE の場合、ターゲット・タイプのスキーマ名。パラメーターまたは結果のタイプが REFERENCE でない場合は、NULL 値。
TARGET_TYPENAME	VARCHAR(128)	Y	パラメーターまたは結果のタイプが REFERENCE の場合、ターゲット・タイプの非修飾名。パラメーターまたは結果のタイプが REFERENCE でない場合は、NULL 値。
SCOPE_TABSCHEMA	VARCHAR(128)	Y	パラメーターのタイプが REFERENCE の場合、有効範囲 (ターゲット表) のスキーマ名。その他の場合、NULL 値。
SCOPE_TABNAME	VARCHAR(128)	Y	パラメーターのタイプが REFERENCE の場合、有効範囲 (ターゲット表) の非修飾名。その他の場合、NULL 値。
TRANSFORMGRPNAME	VARCHAR(128)	Y	構造化タイプのパラメーターまたは結果の場合は、トランスフォーム・グループの名前。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. ソース関数からの派生関数 (他の関数を参照して定義された関数) はソースのパラメーターの長さと位取りを継承するので、そのような関数の LENGTH と SCALE は 0 に設定されます。

SYSCAT.ROUTINES

各行はユーザー定義ルーチン (スカラー関数、表関数、ソース派生関数、メソッド、またはプロシージャー) を表します。組み込み関数は含まれません。

表 123. SYSCAT.ROUTINES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		ルーチンのスキーマ名。
ROUTINENAME	VARCHAR(128)		ルーチンの非修飾名。
ROUTINETYPE	CHAR(1)		ルーチンのタイプ • F = 関数 • M = メソッド • P = プロシージャー
OWNER	VARCHAR(128)		ルーチン作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
ROUTINEID	INTEGER		ルーチンの ID。
RETURN_TYPESCHEMA	VARCHAR(128)	Y	スカラー関数またはメソッドの場合、戻りタイプのスキーマ名。
RETURN_TYPENAME	VARCHAR(128)	Y	スカラー関数またはメソッドの場合、戻りタイプの非修飾名。
ORIGIN	CHAR(1)		• B = 組み込み • E = ユーザー定義、外部 • M = テンプレート関数 • F = フェデレーテッド・プロシージャー • Q = SQL 本体[1] • S = システム生成 • T = システム生成トランスフォーム関数 (直接呼び出し不可) • U = ユーザー定義、ソース関数に基づく
FUNCTIONTYPE	CHAR(1)		• C = 列または集約 • R = 行 • S = スカラー • T = 表 • ブランク = プロシージャー
PARM_COUNT	SMALLINT		ルーチン・パラメーター数。

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
LANGUAGE	CHAR (8)		ルーチン本体の (あるいは、この関数が別の関数を基にしたソース派生関数の場合は、ソース関数本体の) インプリメンテーション言語。使用できる値は、'C'、'CLR'、'COBOL'、'JAVA'、'OLE'、'OLEDB'、または 'SQL' です。ORIGIN が 'E' または 'Q' ではない場合、ブランク。
SOURCESCHEMA	VARCHAR(128)	Y	ORIGIN= 'U'、かつソース関数がユーザー定義関数の場合、ソース関数の特定名のスキーマ名。ORIGIN= 'U'、かつソース関数が組み込み関数の場合、値 'SYSIBM'。ORIGIN が 'U' でない場合は、NULL 値。
SOURCESPECIFIC	VARCHAR(128)	Y	ORIGIN= 'U'、かつソース関数がユーザー定義関数の場合、ソース関数の非修飾特定名。ORIGIN= 'U'、かつソース関数が組み込み関数の場合、値 'N/A for built-in'。ORIGIN が 'U' でない場合は、NULL 値。
DETERMINISTIC	CHAR(1)		• N = 結果は非決定論的 (同じパラメーターが指定されても、異なるルーチンによる呼び出しでは異なる結果を出す可能性がある) • Y = 結果は決定論的 • ブランク = ORIGIN は 'E'、'F'、または 'Q' でない
EXTERNAL_ACTION	CHAR(1)		• E = 関数に外部副次作用がある (したがって、呼び出しの数が重要) • N = 副次作用がない • ブランク = ORIGIN は 'E'、'F'、または 'Q' でない
NULLCALL	CHAR(1)		• N = RETURNS NULL ON NULL INPUT (1 つ以上のオペランドが NULL 値である場合、暗黙のうちに関数の結果は NULL 値) • Y = CALLED ON NULL INPUT • ブランク = ORIGIN は 'E' または 'Q' でない
CAST_FUNCTION	CHAR(1)		• N = cast 関数ではない • Y = cast 関数 • ブランク = ROUTINETYPE は 'F' でない
ASSIGN_FUNCTION	CHAR(1)		• N = 割り当て関数ではない • Y = 暗黙的な割り当て関数 • ブランク = ROUTINETYPE は 'F' でない

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SCRATCHPAD	CHAR(1)		• N = ルーチンにはスクラッチパッドがない • Y = ルーチンにはスクラッチパッドがある • ブランク = ORIGIN は 'E' でないか、ROUTINETYPE が 'P' である
SCRATCHPAD_LENGTH	SMALLINT		ルーチンのスクラッチパッドのサイズ (バイト単位)。 • -1 = LANGUAGE が 'OLEDB' かつ SCRATCHPAD が 'Y' • 0 = SCRATCHPAD が 'Y' でない
FINALCALL	CHAR(1)		• N = 最終呼び出しを行わない • Y = ステートメント終了の実行時に、このルーチンに対して最終呼び出しが行われる • ブランク = ORIGIN は 'E' でないか、ROUTINETYPE が 'P' である
PARALLEL	CHAR(1)		• N = ルーチンを並列して実行できない • Y = ルーチンを並列して実行できる • ブランク = ORIGIN は 'E' でない
PARAMETER_STYLE	CHAR (8)		ルーチンの作成時に宣言されたパラメーターのスタイル。使用できる値は以下のとおりです。 • DB2DARI • DB2GENRL • DB2SQL • GENERAL • GNRLNULL • JAVA • SQL • ブランク = ORIGIN は 'E' でない
FENCED	CHAR(1)		• N = fenced でない • Y = fenced • ブランク = ORIGIN は 'E' でない

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
SQL_DATA_ACCESS	CHAR(1)		ルーチンに含まれている SQL ステートメント (もしあれば) のタイプに関して、データベース・マネージャーが想定する内容を示します。 • C = SQL が含まれている (副照会のない単純式のみ) • M = データを変更する SQL ステートメントが含まれている • N = SQL ステートメントが含まれていない • R = 読み取り専用 SQL ステートメントが含まれている • ブランク = ORIGIN は 'E'、'F'、または 'Q' でない
DBINFO	CHAR(1)		DBINFO パラメーターが外部ルーチンに受け渡されるかどうかを示します。 • N = DBINFO は渡されない • Y = DBINFO は渡される • ブランク = ORIGIN は 'E' でない
PROGRAMTYPE	CHAR(1)		外部ルーチンを呼び出す方法を示します。 • M = メイン • S = サブルーチン • ブランク = ORIGIN が 'F'
COMMIT_ON_RETURN	CHAR(1)		正常にこのプロシージャーが戻ったときにトランザクションがコミットされるかどうかを示します。 • N = 作業単位はコミットされない • Y = 作業単位はコミットされる • ブランク = ROUTINETYPE は 'P' でない
RESULT_SETS	SMALLINT		結果セットの最大数の見積もり。
SPEC_REG	CHAR(1)		特殊レジスターが呼び出し元から継承されるか、またはルーチンが呼び出されたときにデフォルト値に再初期化されるかを示します。 • D = デフォルトの特殊レジスター • I = 継承された特殊レジスター • ブランク = PARAMETER_STYLE が 'DB2DARI' であるか、ORIGIN が 'E' または 'Q' でない

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
FEDERATED	CHAR(1)		ルーチンからフェデレーテッド・オブジェクトにアクセスできるかどうかを示します。 • N = フェデレーテッド・オブジェクトにアクセスできない • Y = フェデレーテッド・オブジェクトにアクセスできる • ブランク = ORIGIN は 'E'、'F'、または 'Q' でない
THREADSAFE	CHAR(1)		ルーチンを他のルーチンと同じプロセスで実行できるかどうかを示します。 • N = ルーチンはスレッド・セーフでない • Y = ルーチンはスレッド・セーフである • ブランク = ORIGIN は 'E' でない
VALID	CHAR(1)		LANGUAGE = 'SQL' の場合のみ適用されます。その他の場合はブランク。 • N = ルーチンは再バインドが必要 • Y = ルーチンは有効 • X = ルーチンは作動不能で、再作成が必要
METHODIMPLEMENTED	CHAR(1)		• N = メソッド本体はインプリメントされていない • Y = メソッド本体がインプリメントされている • ブランク = ROUTINETYPE は 'M' でない
METHODEFFECT	CHAR(2)		• CN = コンストラクター・メソッド • MU = Mutator メソッド • OB = オブザーバー・メソッド • ブランク = システム・メソッドではない
TYPE_PRESERVING	CHAR (1)		• N = 戻りタイプはメソッドの宣言された戻りタイプ。 • Y = 戻りタイプは "type-preserving" パラメーターで管理される。システム生成 mutator メソッドはすべて type-preserving。 • ブランク = ROUTINETYPE は 'M' でない
WITH_FUNC_ACCESS	CHAR(1)		• N = このメソッドは関数表記を使用して呼び出すことができない • Y = このメソッドは関数表記を使用して呼び出すことができる。つまり、「WITH FUNCTION ACCESS」属性が指定されている。 • ブランク = ROUTINETYPE は 'M' でない

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
OVERRIDDEN_METHODID	INTEGER	Y	ユーザー定義のメソッドに OVERRIDING オプションが指定されている場合、オーバーライドされたメソッドの ID。ROUTINETYPE が 'M' でない場合は、NULL 値。
SUBJECT_TYPESCHEMA	VARCHAR(128)	Y	ユーザー定義メソッドのサブジェクト・タイプのスキーマ名。ROUTINETYPE が 'M' でない場合は、NULL 値。
SUBJECT_TYPENAME	VARCHAR(128)	Y	ユーザー定義メソッドのサブジェクト・タイプの非修飾名。ROUTINETYPE が 'M' でない場合は、NULL 値。
CLASS	VARCHAR (384)	Y	LANGUAGE JAVA、CLR、OLE の場合は、このルーチンをインプリメントしたクラスであり、それ以外の場合は、NULL 値です。
JAR_ID	VARCHAR(128)	Y	LANGUAGE JAVA の場合は、このルーチンの作成時に jar ファイルを指定していれば、このルーチンをインプリメントしたインストール済みの jar ファイルの JAR_ID であり、それ以外の場合は、NULL 値です。 LANGUAGE CLR の場合は、このルーチンをインプリメントしたアセンブリー・ファイルです。
JARSCHEMA	VARCHAR(128)	Y	LANGUAGE JAVA の場合は、JAR_ID が存在していれば、このルーチンをインプリメントした jar ファイルのスキーマ名であり、それ以外の場合は、NULL 値です。
JAR_SIGNATURE	VARCHAR (2048)	Y	LANGUAGE JAVA の場合は、このルーチンの指定の Java メソッドのメソッド・シグニチャーです。LANGUAGE CLR の場合は、この CLR ルーチンの参照フィールドです。それ以外の場合、NULL 値です。
CREATE_TIME	TIMESTAMP		ルーチンが作成された時刻。
ALTER_TIME	TIMESTAMP		ルーチンが最後に変更された時刻。
FUNC_PATH	CLOB (2K)	Y	ルーチンの定義された時点の SQL パス。LANGUAGE が 'SQL' でない場合は、NULL 値。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。非修飾参照を完了するために使用します。
IOS_PER_INVOC	DOUBLE		呼び出しごとの入力/出力 (I/O) の推定数。0 がデフォルト。不明の場合は -1。
INSTS_PER_INVOC	DOUBLE		呼び出しごとの命令の数の見積もり。デフォルト値は 450。不明の場合は -1。
IOS_PER_ARGBYTE	DOUBLE		入力引数 1 バイトごとの入出力回数の見積もり。デフォルト値は 0。不明の場合は -1。

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
INSTS_PER_ARGBYTE	DOUBLE		入力引数 1 バイトごとの命令数の見積もり。0 がデフォルト。不明の場合は -1。
PERCENT_ARGBYTES	SMALLINT		ルーチンが実際に読み取る入力引数バイトの平均パーセント値の見積もり。デフォルト値は 100。不明の場合は -1。
INITIAL_IOS	DOUBLE		最初のルーチン呼び出し時に実行される入出力回数の見積もり。0 がデフォルト。不明の場合は -1。
INITIAL_INSTS	DOUBLE		最初のルーチン呼び出し時に実行される命令の数の見積もり。0 がデフォルト。不明の場合は -1。
CARDINALITY	BIGINT		表関数の予測されるカーディナリティー。不明の場合、またはルーチンが表関数でない場合は -1。
SELECTIVITY[2]	DOUBLE		ユーザー定義述部用。ユーザー定義述部がない場合は -1。
RESULT_COLS	SMALLINT		表関数 (ROUTINETYPE = 'F' および FUNCTIONTYPE = 'T') の場合は結果表の列数。プロシージャー (ROUTINETYPE = 'P') の場合は 0、その他の場合は 1 が入ります。
IMPLEMENTATION	VARCHAR (762)	Y	ORIGIN= 'E' の場合、この関数を実現するためのパス/モジュール/関数。 ORIGIN= 'U'、かつソース関数が組み込み関数の場合、この列に含まれるソース関数の名前とシグニチャー。それ以外の場合、NULL 値です。
LIB_ID	INTEGER	Y	将来の使用のために予約済み。
TEXT_BODY_OFFSET	INTEGER		LANGUAGE = 'SQL' の場合、 CREATE ステートメントのテキスト全体の中の、 SQL プロシージャー本体の開始位置に対するオフセット。 LANGUAGE が 'SQL' でない場合は -1。
TEXT	CLOB(2M)	Y	LANGUAGE = 'SQL' の場合、CREATE FUNCTION、CREATE METHOD、または CREATE PROCEDURE ステートメントのフルテキスト。その他の場合は NULL 値。
NEWSAVEPOINTLEVEL	CHAR(1)		呼び出し時にルーチンが新しいセーブポイント・レベルを開始するかどうかを指示します。 • N = ルーチンの呼び出し時に新しいセーブポイント・レベルが開始されない。ルーチンは既存のセーブポイント・レベルを使用する。 • Y = ルーチンの呼び出し時に新しいセーブポイント・レベルが開始される • ブランク = 該当しない場合

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
DEBUG_MODE[3]	VARCHAR(8)		ルーチンを DB2 デバッガーを使ってデバッグできるかどうかを示します。 • DISALLOW = ルーチンはデバッグ不可 • ALLOW = ルーチンはデバッグ可能で、DB2 デバッガーでクライアント・デバッグ・セッションに関与できる • DISABLE = ルーチンはデバッグ不可で、この設定は、ルーチンをドロップして再作成しなければ変更できない • ブランク = ルーチン・タイプは現在 DB2 デバッガーによってサポートされていない
TRACE_LEVEL	VARCHAR(1)	Y	将来の使用のために予約済み。
DIAGNOSTIC_LEVEL	VARCHAR(1)	Y	将来の使用のために予約済み。
CHECKOUT_USERID	VARCHAR(128)	Y	オブジェクトのチェックアウトを実行したユーザーの ID。オブジェクトがチェックアウトされていない場合は NULL 値。
PRECOMPILE_OPTIONS	VARCHAR(1024)	Y	ルーチンに指定されたプリコンパイル・オプション。
COMPILE_OPTIONS	VARCHAR(1024)	Y	ルーチンに指定されたコンパイル・オプション。
EXECUTION_CONTROL	CHAR(1)		共通言語ランタイム (CLR) ルーチンの実行制御モード。使用できる値は以下のとおりです。 • N = Network • R = Fileread • S = Safe • U = Unsafe • W = Filewrite • ブランク = LANGUAGE は 'CLR' でない
CODEPAGE	SMALLINT		ルーチン・コード・ページ。これは、すべての文字パラメーター・タイプ、結果タイプ、およびルーチン本体の中のローカル変数のデフォルト・コード・ページを指定します。
COLLATIONSCHEMA	VARCHAR(128)		ルーチンの照合のスキーマ名。
COLLATIONNAME	VARCHAR(128)		ルーチンの照合の非修飾名。
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		ルーチンの ORDER BY 節の照合のスキーマ名。
COLLATIONNAME_ORDERBY	VARCHAR(128)		ルーチンの ORDER BY 節の照合の非修飾名。

表 123. SYSCAT.ROUTINES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ENCODING_SCHEME	CHAR(1)		PARAMETER CCSID 節に指定された、ルーチンのコード化スキーム。使用できる値は以下のとおりです。 • A = ASCII • U = UNICODE • ブランク = PARAMETER CCSID 節は指定されなかった
LAST_REGEN_TIME	TIMESTAMP		SQL ルーチン・パック記述子が最後に再生成された時刻。
INHERITLOCKREQUEST	CHAR(1)		• N = この関数またはメソッドは、指定した isolation-clause (分離節) の一部として lock-request-clause (ロック要求節) が含まれている SQL ステートメントのコンテキストで呼び出すことができない • Y = この関数またはメソッドは呼び出し元のステートメントの分離レベルを継承する。指定した lock-request-clause (ロック要求節) も継承する • ブランク = LANGUAGE が 'SQL' でないか、ROUTINETYPE が 'P' である
DEFINER[4]	VARCHAR(128)		ルーチン作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. バージョン 8.2 よりも前に作成して、バージョン 9 にマイグレーションした SQL プロシージャーの場合は、'E' です ('Q' ではない)。
2. マイグレーション中は、どのようなユーザー定義ルーチンに関しても、パック記述子およびシステム・カタログにおいて SELECTIVITY 列は -1 に設定されます。ユーザー定義述部の場合、システム・カタログでの選択度は -1 になります。この場合、オプティマイザーが使用する選択度の値は 0.01 です。
3. Java ルーチンの場合、DEBUG_MODE 設定は、Java ルーチンが実際にデバッグ・モードでコンパイルされたかどうか、またはデバッグ Jar がサーバーにインストールされたかどうかを示しません。
4. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.ROUTINESFEDERATED

各行はフェデレーテッド・プロシージャーを表します。

表 124. SYSCAT.ROUTINESFEDERATED カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ROUTINESCHEMA	VARCHAR(128)		ルーチンのスキーマ名。
ROUTINENAME	VARCHAR(128)		ルーチンの非修飾名。
ROUTINETYPE	CHAR(1)		ルーチンのタイプ • P = プロシージャー
OWNER	VARCHAR(128)		ルーチン作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
ROUTINEID	INTEGER		ルーチンの ID。
PARM_COUNT	SMALLINT		ルーチン・パラメーター数。
DETERMINISTIC	CHAR(1)		• N = 結果は非決定論的 (同じパラメーターが指定されても、異なるルーチンによる呼び出しでは異なる結果を出す可能性がある) • Y = 結果は決定論的
EXTERNAL_ACTION	CHAR(1)		• E = ルーチンに外部副次作用がある (したがって、呼び出しの数が重要) • N = 副次作用がない
SQL_DATA_ACCESS	CHAR(1)		ルーチンに含まれている SQL ステートメント (もしあれば) のタイプに関して、データベース・マネージャーが想定する内容を示します。 • C = SQL が含まれている (副照会のない単純式のみ) • M = データを変更する SQL ステートメントが含まれている • N = SQL ステートメントが含まれていない • R = 読み取り専用 SQL ステートメントが含まれている
COMMIT_ON_RETURN	CHAR(1)		正常にこのプロシージャーが戻ったときにトランザクションがコミットされるかどうかを示します。 • N = 作業単位はコミットされない • Y = 作業単位はコミットされる • ブランク = ROUTINETYPE は 'P' でない
RESULT_SETS	SMALLINT		結果セットの最大数の見積もり。
CREATE_TIME	TIMESTAMP		ルーチンが作成された時刻。

表 124. SYSCAT.ROUTINESFEDERATED カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ALTER_TIME	TIMESTAMP		ルーチンが最後に変更された時刻。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。非修飾参照を完了するために使用します。
RESULT_COLS	SMALLINT		プロシージャー (ROUTINETYPE = 'P') の場合は 0、その他の場合は 1 が入ります。
CODEPAGE	SMALLINT		ルーチン・コード・ページ。これは、すべての文字パラメーター・タイプ、結果タイプ、およびルーチン本体の中のローカル変数のデフォルト・コード・ページを指定します。
LAST_REGEN_TIME	TIMESTAMP		SQL ルーチン・パック記述子が最後に再生成された時刻。
REMOTE_PROCEDURE	VARCHAR(128)	Y	フェデレーテッド・ルーチンの作成の対象となった、ソース・プロシージャーの非修飾名。
REMOTE_SCHEMA	VARCHAR(128)	Y	フェデレーテッド・ルーチンの作成の対象となった、ソース・プロシージャーのスキーマ名。
SERVERNAME	VARCHAR(128)	Y	フェデレーテッド・ルーチンの作成の対象となった、ソース・プロシージャーを含むデータ・ソースの名前。
REMOTE_PACKAGE	VARCHAR(128)	Y	ソース・プロシージャーが属するパッケージの名前 (Oracle データ・ソースのラッパーにのみ適用)。
REMOTE_PROCEDURE_ ALTER_TIME	VARCHAR(128)	Y	将来の使用のために予約済み。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SCHEMAAUTH

各行は、スキーマに対して 1 つ以上の特権を付与されたユーザーまたはグループを表します。

表 125. SYSCAT.SCHEMAAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
SCHEMANAME	VARCHAR(128)		この特権が適用されるスキーマの名前。
ALTERINAUTH	CHAR(1)		指定されたスキーマ内のオブジェクトの変更、またはコメント付けを行う特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
CREATEINAUTH	CHAR(1)		指定されたスキーマにオブジェクトを作成する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
DROPINAUTH	CHAR(1)		指定されたスキーマからオブジェクトをドロップする特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

SYSCAT.SCHEMATA

各行はスキーマを表します。

表 126. SYSCAT.SCHEMATA カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SCHEMANAME	VARCHAR(128)		スキーマの名前。
OWNER	VARCHAR(128)		スキーマおよびその中のすべてのオブジェクトをドロップする権限のある、スキーマの許可 ID。暗黙的に作成されたスキーマの値は 'SYSIBM'。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
DEFINER	VARCHAR(128)		スキーマ作成時の許可 ID。
DEFINERTYPE	CHAR(1)		• S = 定義者はシステム • U = 定義者は個々のユーザー
CREATE_TIME	TIMESTAMP		スキーマが作成された時刻。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SECURITYLABELACCESS

各行は、データベース許可 ID に付与されたセキュリティー・ラベルを表します。

表 127. SYSCAT.SECURITYLABELACCESS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR (128)		セキュリティー・ラベルの認可者。
GRANTEE	VARCHAR (128)		セキュリティー・ラベルの保有者。
GRANTEETYPE	CHAR (1)		• U = GRANTEE は個々のユーザー。 • R = GRANTEE はユーザーのロール。 • G = GRANTEE はグループ・ユーザー。
SECLABELID	INTEGER		セキュリティー・ラベルの ID。セキュリティー・ラベルの名前については、SYSCAT.SECURITYLABELS カタログ・ビューにある、対応する SECLABELID 値の SECLABELNAME 列を選択します。
SECPOLICYID	INTEGER		セキュリティー・ラベルに関連したセキュリティー・ポリシーの ID。セキュリティー・ポリシーの名前については、SYSCAT.SECURITYPOLICIES カタログ・ビューにある、対応する SECPOLICYID 値の SECPOLICYNAME 列を選択します。
ACCESSTYPE	CHAR (1)		• B = 読み取りと書き込み両方のアクセス • R = 読み取りアクセス • W = 書き込みアクセス
GRANT_TIME	TIMESTAMP		セキュリティー・ラベルが付与された時刻。

SYSCAT.SECURITYLABELCOMPONENTELEMENTS

各行は、セキュリティー・ラベル・コンポーネントのエレメント値を表します。

表 128. SYSCAT.SECURITYLABELCOMPONENTELEMENTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
COMPID	INTEGER		セキュリティー・ラベル・コンポーネントの ID。
ELEMENTVALUE	VARCHAR(32)		セキュリティー・ラベル・コンポーネントのエレメント値。
ELEMENTVALUEENCODING	CHAR(8) FOR BIT DATA		エンコードされた形式のエレメント値。
PARENTELEMENTVALUE	VARCHAR(32)	Y	ツリー・コンポーネントのエレメントの親の名前。セットおよびアレイ・コンポーネント、およびツリー・コンポーネントの ROOT ノードの場合は NULL 値。

SYSCAT.SECURITYLABELCOMPONENTS

各行は、セキュリティー・ラベル・コンポーネントを表します。

表 129. SYSCAT.SECURITYLABELCOMPONENTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
COMPNAME	VARCHAR(128)		セキュリティー・ラベル・コンポーネントの名前。
COMPID	INTEGER		セキュリティー・ラベル・コンポーネントの ID。
COMPTYPE	CHAR(1)		セキュリティー・ラベル・コンポーネントのタイプ。 • A = アレイ • S = セット • T = ツリー
NUMELEMENTS	INTEGER		セキュリティー・ラベル・コンポーネント内のエレメントの数。
CREATE_TIME	TIMESTAMP		セキュリティー・ラベル・コンポーネントが作成された時刻。
REMARKS	VARCHAR(254)		ユーザー提供のコメントまたは NULL 値。

SYSCAT.SECURITYLABELS

各行は、セキュリティー・ラベルを表します。

表 130. SYSCAT.SECURITYLABELS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SECLABELNAME	VARCHAR(128)		セキュリティー・ラベルの名前。
SECLABELID	INTEGER		セキュリティー・ラベルの ID。
SECPOLICYID	INTEGER		セキュリティー・ラベルが属するセキュリティー・ポリシーの ID。
SECLABEL	SYSPROC. DB2SECURITYLABEL		セキュリティー・ラベルの内部表記。
CREATE_TIME	TIMESTAMP		セキュリティー・ラベルが作成された時刻。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SECURITYPOLICIES

各行は、セキュリティー・ポリシーを表します。

表 131. SYSCAT.SECURITYPOLICIES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SECPOLICYNAME	VARCHAR(128)		セキュリティー・ポリシーの名前。
SECPOLICYID	INTEGER		セキュリティー・ポリシーの ID。
NUMSECLABELCOMP	INTEGER		セキュリティー・ポリシー内のセキュリティー・ラベル・コンポーネントの数。
RWSECLABELREL	CHAR(1)		同じ許可 ID に認可された読み取りおよび書き込みアクセスのセキュリティー・ラベル同士の関係。 • S = ユーザーに認可された書き込みアクセスのセキュリティー・ラベルは、同じユーザーに認可された読み取りアクセスのセキュリティー・ラベルのサブセット
NOTAUTHWRITESECLABEL	CHAR(1)		INSERT または UPDATE ステートメントで指定されているセキュリティー・ラベルを書き込む許可をユーザーが持たない場合に取る処置。 • O = オーバーライドする • R = 制限する
CREATE_TIME	TIMESTAMP		セキュリティー・ポリシーが作成された時刻。
GROUPAUTHS	CHAR(1)		グループを表す許可 ID に付与されたセキュリティー・ラベルおよび免除の権限が使用されるか、それとも無視されるかを示す。 • I = 無視される • U = 使用される
ROLEAUTHS	CHAR(1)		ロールを表す許可 ID に付与されたセキュリティー・ラベルおよび免除の権限が使用されるか、それとも無視されるかを示す。 • I = 無視される • U = 使用される
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SECURITYPOLICYCOMPONENTRULES

各行は、セキュリティー・ポリシーのセキュリティー・ラベル・コンポーネントの読み取りおよび書き込みアクセス規則を表します。

表 132. SYSCAT.SECURITYPOLICYCOMPONENTRULES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SECPOLICYID	INTEGER		セキュリティー・ポリシーの ID。
COMPID	INTEGER		セキュリティー・ポリシーのセキュリティー・ラベル・コンポーネントの ID。
ORDINAL	INTEGER		セキュリティー・ポリシー内でセキュリティー・ラベル・コンポーネントが現れる位置 (1 から始まる)。
READACCESSRULENAME	VARCHAR(128)		セキュリティー・ラベル・コンポーネントに関連した読み取りアクセス規則の名前。
READACCESSRULETEXT	VARCHAR(512)		セキュリティー・ラベル・コンポーネントに関連した読み取りアクセス規則のテキスト。
WRITEACCESSRULENAME	VARCHAR(128)		セキュリティー・ラベル・コンポーネントに関連した書き込みアクセス規則の名前。
WRITEACCESSRULETEXT	VARCHAR(512)		セキュリティー・ラベル・コンポーネントに関連した書き込みアクセス規則のテキスト。

SYSCAT.SECURITYPOLICYEXEMPTIONS

各行は、データベース許可 ID に付与された、セキュリティー・ポリシーの免除を表します。

表 133. SYSCAT.SECURITYPOLICYEXEMPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR (128)		免除の認可者。
GRANTEE	VARCHAR (128)		免除の保有者。
GRANTEETYPE	CHAR (1)		• U = GRANTEE は個々のユーザー。 • R = GRANTEE はユーザーのロール。 • G = GRANTEE はグループ・ユーザー。
SECPOLICYID	INTEGER		免除が付与されたセキュリティー・ポリシーの ID。セキュリティー・ポリシーの名前については、SYSCAT.SECURITYPOLICIES カタログ・ビューにある、対応する SECPOLICYID 値の SECPOLICYNAME 列を選択します。
ACCESSRULENAME	VARCHAR (128)		免除が付与されたアクセス規則の ID。
ACCESSTYPE	CHAR (1)		規則が適用されるアクセスのタイプ。 • R = 読み取りアクセス • W = 書き込みアクセス
ORDINAL	INTEGER		規則が適用されるセキュリティー・ポリシー内でのセキュリティー・ラベル・コンポーネントの位置。
ACTIONALLOWED	CHAR (1)		規則が DB2LBACWRITEARRAY である場合は、次のようになります。 • D = 下方に書き込む • U = 上方に書き込む それ以外の場合、ブランクです。
GRANT_TIME	TIMESTAMP		免除が付与された時刻。

SYSCAT.SEQUENCEAUTH

各行は、シーケンスに対して 1 つ以上の特権を付与されたユーザーまたはグループを表します。

表 134. SYSCAT.SEQUENCEAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
SEQSCHEMA	VARCHAR(128)		シーケンスのスキーマ名。
SEQNAME	VARCHAR(128)		シーケンスの非修飾名。
ALTERAUTH	CHAR(1)		シーケンスを変更する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
USAGEAUTH	CHAR(1)		シーケンスを参照する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

SYSCAT.SEQUENCES

各行はシーケンスを表します。

表 135. SYSCAT.SEQUENCES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SEQSCHEMA	VARCHAR(128)		シーケンスのスキーマ名。
SEQNAME	VARCHAR(128)		シーケンスの非修飾名。
DEFINER[1]	VARCHAR(128)		シーケンス作成時の許可 ID。
DEFINERTYPE	CHAR(1)		• S = 定義者はシステム • U = 定義者は個々のユーザー
OWNER	VARCHAR(128)		シーケンス作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
SEQID	INTEGER		シーケンスの ID。
SEQTYPE	CHAR(1)		シーケンスのタイプ。 • I = ID シーケンス • S = 通常のシーケンス
INCREMENT	DECIMAL(31,0)		増分値。
START	DECIMAL(31,0)		シーケンスの開始値。
MAXVALUE	DECIMAL(31,0)		シーケンスの最大値。
MINVALUE	DECIMAL(31,0)		シーケンスの最小値。
NEXTCACHEFIRSTVALUE	DECIMAL (31,0)	Y	次のキャッシュ・ブロックで割り当てることのできる最初の値。キャッシュしない場合は、割り当てることのできる次の値。
CYCLE	CHAR(1)		その最大値または最小値に達した後、シーケンスが値の生成を続行できるかどうかを示します。 • N = シーケンスは循環できない • Y = シーケンスは循環できる
CACHE	INTEGER		アクセスを高速化するために、メモリーに事前割り振りするシーケンス値の数。 0 は、シーケンスの値が事前割り振りされないことを示します。パーティション・データベースでは、この値はそれぞれのデータベース・パーティションに適用されます。
ORDER	CHAR(1)		要求の順序でシーケンス番号が生成されるかどうかを示します。 • N = 要求の順序でシーケンス番号を生成しません。 • Y = 要求の順序でシーケンス番号を生成します。

表 135. SYSCAT.SEQUENCES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
DATATYPEID	INTEGER		組み込みタイプの場合は、組み込みタイプの内部 ID。特殊タイプの場合は、特殊タイプの内部 ID。
SOURCETYPEID	INTEGER		組み込みタイプの場合、この値は 0 になります。特殊タイプの場合は、特殊タイプのソース・タイプである組み込みタイプの内部 ID です。
CREATE_TIME	TIMESTAMP		シーケンスが作成された時刻。
ALTER_TIME	TIMESTAMP		シーケンスが最後に変更された時刻。
PRECISION	SMALLINT		シーケンスのデータ・タイプの精度。使用できる値は以下のとおりです。 • 5 = SMALLINT • 10 = INTEGER • 19 = BIGINT DECIMAL では、指定した DECIMAL データ・タイプの精度です。
ORIGIN	CHAR(1)		シーケンスの発生元。 • S = システム生成シーケンス • U = ユーザー生成シーケンス
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.SERVEROPTIONS

各行は、サーバー固有のオプション値を表します。

表 136. SYSCAT.SERVEROPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WRAPNAME	VARCHAR(128)	Y	ラッパーの名前。
SERVERNAME	VARCHAR(128)	Y	サーバー名 (大文字)。
SERVERTYPE	VARCHAR(30)	Y	サーバーのタイプ。
SERVERVERSION	VARCHAR(18)	Y	サーバーのバージョン。
CREATE_TIME	TIMESTAMP		項目が作成された時刻。
OPTION	VARCHAR(128)		サーバー・オプションの名前。
SETTING	VARCHAR (2048)		サーバー・オプションの値。
SERVEROPTIONKEY	VARCHAR(18)		行を固有識別する。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SERVERS

各行はデータ・ソースを表します。

表 137. SYSCAT.SERVERS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WRAPNAME	VARCHAR(128)		ラッパーの名前。
SERVERNAME	VARCHAR(128)		サーバー名 (大文字)。
SERVERTYPE	VARCHAR(30)	Y	サーバーのタイプ。
SERVERVERSION	VARCHAR(18)	Y	サーバーのバージョン。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.SERVICECLASSES

各行はサービス・クラスを表します。

表 138. SYSCAT.SERVICECLASSES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SERVICECLASSNAME	VARCHAR(128)		サービス・クラスの名前。
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	親サービス・スーパークラスのサービス・クラス名。
SERVICECLASSID	SMALLINT		サービス・クラスの ID。
PARENTID	SMALLINT		このサービス・クラスの親サービス・クラスの ID。このサービス・クラスがスーパー・サービス・クラスの場合は 0 です。
CREATE_TIME	TIMESTAMP		サービス・クラスが作成された時刻。
ALTER_TIME	TIMESTAMP		サービス・クラスが最後に変更された時刻。
ENABLED	CHAR(1)		サービス・クラスの状態。 • N = 使用不可 • Y = 使用可能
AGENTPRIORITY	SMALLINT		DB2 スレッドの通常優先順位に対して相対的なサービス・クラス内のエージェントのスレッド優先順位。 • -20 から 20 (Linux および UNIX) • -6 から 6 (Windows) • -32768 = 設定しない
PREFETCHPRIORITY	CHAR(1)		サービス・クラス内のエージェントのプリフェッチ優先順位。 • H = 高 • L = 低 • M = 中 • ブランク = 設定しない
INBOUNDCORRELATOR	VARCHAR(128)	Y	将来の利用。
OUTBOUNDCORRELATOR	VARCHAR(128)	Y	サービス・クラスとオペレーティング・システムのワークロード・マネージャー・サービス・クラスを関連付けるために使用されるストリング。
COLLECTAGGACTDATA	CHAR(1)		該当するイベント・モニターにサービス・クラスでキャプチャーさせる集約アクティビティー・データを指定します。 • B = 基礎集約アクティビティー・データを収集する • E = 拡張集約アクティビティー・データを収集する • N = なし

表 138. SYSCAT.SERVICECLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTAGGREQDATA	CHAR(1)		該当するイベント・モニターにサービス・クラスでキャプチャーさせる集約アクティビティー・データを指定します。 • B = 基礎集約要求データを収集する • N = なし
COLLECTACTDATA	CHAR(1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • V = 詳細および値ありのアクティビティー・データ • W = 詳細なしのアクティビティー・データ
COLLECTACTPARTITION	CHAR(1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーターのデータベース・パーティション • D = すべてのデータベース・パーティション
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.STATEMENTS

各行はパッケージ内の SQL ステートメントを表します。

表 139. SYSCAT.STATEMENTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
PKGSCHEMA	VARCHAR(128)		パッケージのスキーマ名。
PKGNAME	VARCHAR(128)		パッケージの非修飾名。
STMTNO	INTEGER		アプリケーション・プログラムのソース・モジュールにおける SQL ステートメントの行番号。
SECTNO	SMALLINT		この SQL ステートメントを収めたパッケージ・セクションの番号。
SEQNO	INTEGER		常に 1。
TEXT	CLOB(2M)		SQL ステートメントのテキスト。
UNIQUE_ID	CHAR(8) FOR BIT DATA		同じ名前を持つ複数のパッケージが存在している場合の、特定のパッケージの ID。
VERSION	VARCHAR(64)	Y	パッケージのバージョン ID。

SYSCAT.SURROGATEAUTHIDS

各行は、ユーザーまたは PUBLIC に対する SETSESSIONUSER 特権を付与されているユーザーまたはグループを表します。

表 140. SYSCAT.SURROGATEAUTHIDS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		TRUSTEDID に代理を務める能力を付与した許可 ID。 TRUSTEDID がトラステッド・コンテキスト・オブジェクトを表す場合、このフィールドは、トラステッド・コンテキスト・オブジェクトを作成または変更した許可 ID を表します。
TRUSTEDID	VARCHAR(128)		代理を務めるよう信頼されたエンティティーの ID。
TRUSTEDIDTYPE	CHAR(1)		• C = トラステッド・コンテキスト • G = グループ • U = ユーザー
SURROGATEAUTHID	VARCHAR(128)		TRUSTEDID が担うことのできる代理許可 ID。'PUBLIC' は、TRUSTEDID がどんな許可 ID でも担うことができることを示します。
SURROGATEAUTHIDTYPE	CHAR(1)		• G = グループ • U = ユーザー
AUTHENTICATE	CHAR(1)		• N = 認証は必要なし • Y = 許可 ID を取得するには、ユーザーを認証するための許可 ID を持つ認証トークンが必要 • ブランク = TRUSTEDIDTYPE は 'C' でない
CONTEXTROLE	VARCHAR(128)	Y	取得した許可 ID に割り当てられる特定のロール。トラステッド・コンテキストに定義されたデフォルトのロールがある場合には、それを置き換えます。 TRUSTEDIDTYPE が 'C' でない場合、NULL 値。
GRANT_TIME	TIMESTAMP		付与が行われた時刻。

SYSCAT.TABAUTH

各行は、表またはビューに対して 1 つ以上の特権を付与されたユーザーまたはグループを表します。

表 141. SYSCAT.TABAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
TABSCHEMA	VARCHAR(128)		表またはビューのスキーマ名。
TABNAME	VARCHAR(128)		表またはビューの非修飾名。
CONTROLAUTH	CHAR(1)		CONTROL 特権。 • N = 保有しない • Y = 保有するが、付与はできない
ALTERAUTH	CHAR(1)		以下のことを行う特権。表を変更する。この表の親表に対し、その主キーまたはユニーク制約をドロップすることを許可する。ある表に対し、この表またはマテリアライズ照会内のビューを参照するマテリアライズ照会表になることを許可する。この表またはマテリアライズ照会内のビューを参照する表に対し、マテリアライズ照会表ではなくなることを許可する。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
DELETEAUTH	CHAR(1)		表または更新可能なビューから行を削除する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
INDEXAUTH	CHAR(1)		表の索引を作成する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

表 141. SYSCAT.TABAUTH カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
INSERTAUTH	CHAR(1)		表または更新可能なビューに行を挿入するか、または、表またはビューに対してインポート・ユーティリティーを実行する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
REFAUTH	CHAR(1)		親として表を参照する外部キーの作成やドロップを行う特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
SELECTAUTH	CHAR(1)		表またはビューの行の検索、表に対するビューの作成、または表またはビューに対してエクスポート・ユーティリティーを実行する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
UPDATEAUTH	CHAR(1)		表または更新可能なビューに対して UPDATE ステートメントを実行する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

SYSCAT.TABCONST

それぞれの行は、タイプ CHECK、UNIQUE、PRIMARY KEY、または FOREIGN KEY の表制約を示しています。表階層の場合、制約はそれぞれ作成された階層レベルでのみ記録されます。

表 142. SYSCAT.TABCONST カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
CONSTNAME	VARCHAR(128)		制約の名前。
TABSCHEMA	VARCHAR(128)		この制約が適用される表のスキーマ名。
TABNAME	VARCHAR(128)		この制約が適用される表の非修飾名。
OWNER	VARCHAR(128)		その制約を作成した許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
TYPE	CHAR(1)		制約の種類。 • F = 外部キー • I = 関数の従属関係 • K = 検査 • P = 主キー • U = ユニーク
ENFORCED	CHAR(1)		• N = 制約を実施しない • Y = 制約を実施
CHECKEXISTINGDATA	CHAR(1)		• D = 既存のデータの検査を据え置く • I = 即時に既存のデータを検査する • N = 既存のデータを検査しない
ENABLEQUERYOPT	CHAR(1)		• N = 照会最適化は使用不能 • Y = 照会最適化は使用可能
DEFINER[1]	VARCHAR(128)		その制約を作成した許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.TABDEP

各行は、他のオブジェクトに対するビューまたはマテリアライズ照会表の従属関係を表します。ビューまたはマテリアライズ照会表は、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更はビューまたはマテリアライズ照会表に影響します。また、ビューに対する特権が、基礎表またはビューに対する特権にどのように従属しているかもエンコードします。

表 143. SYSCAT.TABDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		ビューまたはマテリアライズ照会表のスキーマ名。
TABNAME	VARCHAR(128)		ビューまたはマテリアライズ照会表の非修飾名。
DTYPE	CHAR(1)		従属オブジェクトのタイプ • S = マテリアライズ照会表 • T = 表 (ステージングのみ) • V = ビュー (非型付き) • W = 型付きビュー
OWNER	VARCHAR(128)		ビューまたはマテリアライズ照会表の作成者の許可 ID。
OWNERTYPE	CHAR(1)		• U = 所有者は個々のユーザー
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • F = ルーチン・インスタンス • I = 索引 (基本表への従属関係を記録する場合) • N = ニックネーム • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • R = 構造化タイプ • S = マテリアライズ照会表 • T = 表 (非型付き) • U = 型付き表 • V = ビュー (非型付き) • W = 型付きビュー • Z = XSR オブジェクト • v = グローバル変数
BSCHEMA	VARCHAR(128)		ビューまたはマテリアライズ照会表が従属しているオブジェクトのスキーマ名。
BNAME	VARCHAR (128)		ビューまたはマテリアライズ照会表が従属しているオブジェクトの非修飾名。

表 143. SYSCAT.TABDEP カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V' または 'W' の場合、このビューまたはマテリアライズ照会表が従属する基礎表またはビューの特権をエンコードします。それ以外の場合は NULL 値。
DEFINER[1]	VARCHAR (128)		ビューまたはマテリアライズ照会表の作成者の許可 ID。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.TABDETACHEDDEP

各行は、デタッチされた従属表とデタッチされた表との間の、デタッチ従属関係を表します。

表 144. SYSCAT.TABDETACHEDDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		デタッチされた表のスキーマ名。
TABNAME	VARCHAR(128)		デタッチされた表の非修飾名。
DEPTABSCHEMA	VARCHAR(128)		デタッチされた従属表のスキーマ名。
DEPTABNAME	VARCHAR(128)		デタッチされた従属表の非修飾名。

SYSCAT.TABLES

各行は、表、ビュー、別名、またはニックネームを表します。表またはビューの各階層にはそれぞれ追加の行が 1 行あります。この行は階層をインプリメントする階層表または階層ビューを表しています。カタログ表およびカタログ・ビューが含まれています。

表 145. SYSCAT.TABLES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		オブジェクトのスキーマ名。
TABNAME	VARCHAR(128)		オブジェクトの非修飾名。
OWNER	VARCHAR(128)		表、ビュー、別名、またはニックネーム作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
TYPE	CHAR(1)		オブジェクトのタイプ。 • A = 別名 • G = グローバル一時表 • H = 階層表 • L = デタッチされた表 • N = ニックネーム • S = マテリアライズ照会表 • T = 表 (非型付き) • U = 型付き表 • V = ビュー (非型付き) • W = 型付きビュー
STATUS	CHAR(1)		オブジェクトの状況。 • C = SET INTEGRITY ペンディング • N = 正常 • X = 作動不能
BASE_TABSCHEMA	VARCHAR(128)	Y	TYPE= 'A' の場合、この別名によって参照される表、ビュー、別名、またはニックネームのスキーマ名。その他の場合は NULL 値。
BASE_TABNAME	VARCHAR(128)	Y	TYPE= 'A' の場合、この別名によって参照される表、ビュー、別名、またはニックネームの非修飾名。その他の場合は NULL 値。
ROWTYPESCHEMA	VARCHAR(128)	Y	該当する場合、この表の行タイプのスキーマ名。その他の場合は NULL 値。
ROWTYPENAME	VARCHAR(128)	Y	該当する場合、この表の行タイプの非修飾名。その他の場合は NULL 値。
CREATE_TIME	TIMESTAMP		オブジェクトが作成された時刻。
ALTER_TIME	TIMESTAMP		オブジェクトが最後に変更された時刻。
INVALIDATE_TIME	TIMESTAMP		オブジェクトが最後に無効にされた時刻。

表 145. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
STATS_TIME	TIMESTAMP	Y	このオブジェクトについて記録されている統計値が最後に変更された時刻。統計が収集されていない場合は、NULL 値。
COLCOUNT	SMALLINT		列の数。継承された列がある場合は、それも含む。
TABLEID	SMALLINT		内部論理オブジェクト ID。
TBSPACEID	SMALLINT		このオブジェクトの PRIMARY 表スペースの内部論理 ID。
CARD	BIGINT		行の総数。統計が収集されていない場合は -1。
NPAGES	BIGINT		表の行が存在しているページの総数。ビューまたは別名の場合、または統計が収集されていない場合は -1。副表および階層表の場合は -2。
FPAGES	BIGINT		ページの総数。ビューまたは別名の場合、または統計が収集されていない場合は -1。副表および階層表の場合は -2。
OVERFLOW	BIGINT		表のオーバーフロー・レコードの総数。ビューまたは別名の場合、または統計が収集されていない場合は -1。副表および階層表の場合は -2。
TBSPACE	VARCHAR(128)	Y	表の PRIMARY 表スペースの名前。他の表スペースが指定されていない場合、表のすべての部分がこの表スペースに保管されます。別名、ビュー、およびパーティション表の場合は、NULL 値です。
INDEX_TBSPACE	VARCHAR(128)	Y	この表に作成されたすべての索引が保管されている表スペースの名前。別名、ビュー、およびパーティション表の場合、または INDEX IN 節の指定がないか、または INDEX IN 節に CREATE TABLE ステートメントの IN 節と同じ値が指定されている場合は、NULL 値。
LONG_TBSPACE	VARCHAR(128)	Y	この表のすべての長形式データ (LONG または LOB の列タイプ) が入っている表スペースの名前。別名、ビュー、およびパーティション表の場合、または LONG IN 節の指定がないか、または LONG IN 節に CREATE TABLE ステートメントの IN 節と同じ値が指定されている場合は、NULL 値。
PARENTS	SMALLINT	Y	このオブジェクトの親表の数。つまり、このオブジェクトが従属オブジェクトになっている参照制約の数。

表 145. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
CHILDREN	SMALLINT	Y	このオブジェクトに従属している表の数。つまり、このオブジェクトが親になっている参照制約の数。
SELFREFS	SMALLINT	Y	このオブジェクトに対する自己参照になっている参照制約の数。つまり、このオブジェクトが親と従属の両方を兼ねる参照制約の数。
KEYCOLUMNS	SMALLINT	Y	主キーを構成する列の数。
KEYINDEXID	SMALLINT	Y	主キー索引の索引 ID。主キーがない場合は、0 または NULL 値。
KEYUNIQUE	SMALLINT		このオブジェクトに定義されたユニーク・キー制約 (主キー制約を除く) の数。
CHECKCOUNT	SMALLINT		このオブジェクトに定義されたチェック制約の数。
DATACAPTURE	CHAR(1)		• L = 表はデータ複製 (LONG VARCHAR および LONG VARGRAPHIC 列の複製を含む) に関与している • N = 表はデータ複製に関与していない • Y = 表はデータ複製 (LONG VARCHAR および LONG VARGRAPHIC 列の複製を除く) に関与している

表 145. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
CONST_CHECKED	CHAR(32)		• バイト 1 は、外部キー制約を表します。 • バイト 2 は、チェック制約を表します。 • バイト 5 は、マテリアライズ照会表を表します。 • バイト 6 は生成される列を表します。 • バイト 7 は、ステージング表を表します。 • バイト 8 は、データ・パーティション制約を表します。 • 他のバイトは、将来の使用のために予約されています。 使用できる値は以下のとおりです。 • F = バイト 5 では、マテリアライズ照会表をインクリメンタル更新できない。バイト 7 では、ステージング表の内容は不完全で、関連したマテリアライズ照会表のインクリメンタル更新に使用することができない。 • N = チェックなし • U = ユーザーによるチェック • W = 表が SET INTEGRITY ペンディング状態になったときに 'U' の状態だった • Y = システムによるチェック
PMAP_ID	SMALLINT	Y	この表によって現在使用されている分散マップの ID (別名またはビューの場合は NULL)。
PARTITION_MODE	CHAR(1)		パーティション・データベース・システム内のデータベース・パーティション間でどのようにデータが配分されるかを示します。 • H = ハッシュ • R = データベース・パーティション間で複製される • ブランク = データベース・パーティションなし
LOG_ATTRIBUTE	CHAR(1)		• 常に 0。この列は使用されなくなりました。
PCTFREE	SMALLINT		将来の挿入用に予約された各ページのパーセント。
APPEND_MODE	CHAR(1)		ページ上での行の挿入方法の制御。 • N = 新規行は使用可能な既存スペースに挿入される • Y = 新規行はデータの終わりに追加される

表 145. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REFRESH	CHAR(1)		リフレッシュ・モード。 • D = 据え置き • I = 即時 • O = 1 回 • ブランク = マテリアライズ照会表でない
REFRESH_TIME	TIMESTAMP	Y	REFRESH = 'D' または 'O' の場合、データが最後にリフレッシュされた (REFRESH TABLE ステートメント) 時刻。その他の場合、NULL 値。
LOCKSIZE	CHAR(1)		データ操作言語 (DML) ステートメントがアクセスする表の優先ロック細分性を示します。表のみに適用されます。使用できる値は以下のとおりです。 • I = ブロック挿入 • R = 行 • T = 表 • ブランク = 該当しない場合
VOLATILE	CHAR(1)		• C = 表のカーディナリティーは揮発性 • ブランク = 該当しない場合
ROW_FORMAT	CHAR(1)		使用されません。
PROPERTY	VARCHAR(32)		表のプロパティー。単一ブランクは、表にプロパティーがないことを示します。ストリング内の位置、値、および意味を以下に示します。 • 1、Y = ユーザー保守のマテリアライズ照会表 • 2、Y = ステージング表 • 3、Y = PROPAGATE IMMEDIATE (伝搬即時) • 11、Y = キャッシュされないニックネーム
STATISTICS_PROFILE	CLOB(10M)	Y	オブジェクトの統計プロファイルの登録に使用された RUNSTATS コマンド。
COMPRESSION	CHAR(1)		• B = 値と行の両方の圧縮がアクティブ。 • N = 圧縮はアクティブにされていない。圧縮をサポートしない行フォーマットが使用されている。 • R = 行の圧縮がアクティブで、圧縮をサポートする行フォーマットを使用できる。 • V = 値の圧縮がアクティブで、圧縮をサポートする行フォーマットが使用されている。 • ブランク = 該当しない場合

表 145. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACCESS_MODE	CHAR(1)		オブジェクトのアクセス制限の状態。これらの状態は、SET INTEGRITY によりペンディング状態にあるオブジェクト、または SET INTEGRITY ステートメントによって処理されたオブジェクトにのみ適用されます。使用できる値は以下のとおりです。 • D = データの移動不可 • F = フル・アクセス権限 • N = アクセス不可 • R = 読み取り専用アクセス
CLUSTERED	CHAR(1)	Y	• Y = テーブルはマルチディメンションに分けてクラスター化される (1 つしかディメンションがない場合でも) • NULL 値 = テーブルはマルチディメンションに分けてクラスター化されない。
ACTIVE_BLOCKS	BIGINT		表の中のアクティブ・ブロックの総数、または -1。マルチディメンション・クラスタリング表 (MDC) のみに適用されます。
DROPRULE	CHAR(1)		• N = 規則はなし • R = ドロップに制約規則が適用される
MAXFREESPACESEARCH	SMALLINT		将来の使用のために予約済み。
AVGCOMPRESSEDROWSIZE	SMALLINT		この表の中の圧縮された行の長さの平均 (バイト単位)。統計が収集されていない場合は -1。
AVGROWCOMPRESSIONRATIO	REAL		表の中の圧縮された行の場合、これは行の平均の圧縮率です。つまり、圧縮されていない行の平均の長さを、圧縮された行の平均の長さで除算したものです。統計が収集されていない場合は、-1。
AVGROWSIZE	SMALLINT		この表の中の圧縮された行と圧縮されていない行の両方の長さの平均 (バイト単位)。統計が収集されていない場合は -1。
PCTROWSCOMPRESSED	REAL		表の中の行の総数に対する圧縮された行のパーセンテージ。統計が収集されていない場合は -1。

表 145. SYSCAT.TABLES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
LOGINDEXBUILD	VARCHAR(3)	Y	表で索引の作成、再作成、または再編成の操作を行う際に実行されるロギングのレベル。 • OFF = 表での索引作成操作が最小限ログに記録される • ON = 表での索引作成操作が完全にログに記録される • NULL 値 = *logindexbuild* データベース構成パラメーターの値を使用して、索引作成操作を完全にログに記録するかどうかを指定する。
CODEPAGE	SMALLINT		オブジェクトのコード・ページ。これが、すべての文字の列、トリガー、チェック制約、および式で生成された列のデフォルトのコード・ページです。
COLLATIONSCHEMA	VARCHAR(128)		表の照合のスキーマ名。
COLLATIONNAME	VARCHAR(128)		表の照合の非修飾名。
COLLATIONSCHEMA_ORDERBY	VARCHAR(128)		表の ORDER BY 節の照合のスキーマ名。
COLLATIONNAME_ORDERBY	VARCHAR(128)		表の ORDER BY 節の照合の非修飾名。
ENCODING_SCHEME	CHAR(1)		• A = CCSID ASCII が指定された • U = CCSID UNICODE が指定された • ブランク = CCSID 節は指定されなかった
PCTPAGESSAVED	SMALLINT		行の圧縮の結果、表に保管されるページの概算パーセンテージ。この値は表内の各ユーザー・データ行のオーバーヘッド・バイトを含んでいますが、ディクショナリー・オーバーヘッドによって消費されるスペースは含みません。統計が収集されない場合は -1 です。
LAST_REGEN_TIME	TIMESTAMP	Y	表のビューまたはチェック制約が最後に再生成された時刻。
SECPOLICYID	INTEGER		表を保護しているセキュリティー・ポリシーの ID。無保護の表の場合は 0。
PROTECTIONGRANULARITY	CHAR(1)		• B = 列レベルと行レベルの両方の細分性 • C = 列レベルの細分性 • R = 行レベルの細分性 • ブランク = 無保護の表
AUDITPOLICYID	INTEGER	Y	監査ポリシーの ID。
AUDITPOLICYNAME	VARCHAR(128)	Y	監査ポリシーの名前。
DEFINER[1]	VARCHAR(128)		表、ビュー、別名、またはニックネーム作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.TABLESPACES

各行は、表スペースを表します。

表 146. SYSCAT.TABLESPACES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TBSPACE	VARCHAR(128)		表スペースの名前。
OWNER	VARCHAR(128)		表スペース作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
CREATE_TIME	TIMESTAMP		表スペースが作成された時刻。
TBSPACEID	INTEGER		表スペースの ID。
TBSPACETYPE	CHAR(1)		表スペースのタイプ。 • D = データベース管理スペース • S = システム管理スペース
DATATYPE	CHAR(1)		この表スペースに保管できるデータのタイプ。 • A = REGULAR 表スペースにおける永続データのすべての型。 • L = LARGE 表スペースにおける永続データのすべての型。 • T = システム一時表のみ • U = 宣言された一時表のみ
EXTENTSIZE	INTEGER		サイズ PAGESIZE をページ単位とする各エクステント・サイズ。表スペースの中のコンテナーにこの数のページが書き込まれたら、次のコンテナーに切り替えるようになります。
PREFETCHSIZE	INTEGER		プリフェッチの実行時に読み取るサイズ PAGESIZE のページ数。AUTOMATIC の場合は -1。
OVERHEAD	DOUBLE		コントローラー・オーバーヘッドおよびディスク・シークおよび待ち時間 (ミリ秒単位) (この表スペースのコンテナーにおける平均)。
TRANSFERRATE	DOUBLE		サイズ PAGESIZE から成る 1 ページをバッファーに読み込む時間 (この表スペースのコンテナーにおける平均)。
PAGESIZE	INTEGER		この表スペースのページのサイズ (バイト単位)。
DBPGNAME	VARCHAR(128)		この表スペースに関連したデータベース・パーティション・グループの名前。
BUFFERPOOLID	INTEGER		この表スペースで使用されるバッファー・プールの ID (1 はデフォルトのバッファー・プールを示します)。

表 146. SYSCAT.TABLESPACES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
DROP_RECOVERY	CHAR(1)		表のドロップ操作後にこの表スペース内の表を回復できるかどうかを示します。 • N = 表は回復不可能 • Y = 表は回復可能
NGNAME[1]	VARCHAR(128)		この表スペースに関連したデータベース・パーティション・グループの名前。
DEFINER[2]	VARCHAR(128)		表スペース作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. NGNAME 列は、後方互換性のために含まれています。DBPGNAME を参照してください。
2. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.TABOPTIONS

各行は、リモート表に関連したオプションを表します。

表 147. SYSCAT.TABOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		表、ビュー、別名、またはニックネームのスキーマ名。
TABNAME	VARCHAR(128)		表、ビュー、別名、またはニックネームの非修飾名。
OPTION	VARCHAR(128)		表オプションの名前。
SETTING	CLOB (32K)		表オプションの値。

SYSCAT.TBSPACEAUTH

各行は、データベース内の特定の表スペースに対する USE 特権を付与されているユーザーまたはグループを表します。

表 148. SYSCAT.TBSPACEAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
TBSPACE	VARCHAR(128)		表スペースの名前。
USEAUTH	CHAR(1)		表スペース内で表を作成する特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

SYSCAT.THRESHOLDS

各行はしきい値を表します。

表 149. SYSCAT.THRESHOLDS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
THRESHOLDNAME	VARCHAR(128)		しきい値の名前。
THRESHOLDID	INTEGER		しきい値の ID。
ORIGIN	CHAR(1)		しきい値の作成元。 • U = しきい値はユーザーによって作成された • W = しきい値は作業アクション・セットから作成された
THRESHOLDCLASS	CHAR(1)		しきい値の分類。 • A = 集約のしきい値 • C = アクティビティーのしきい値
THRESHOLDPREDICATE	VARCHAR (15)		しきい値のタイプ。使用できる値は以下のとおりです。 • CONCDBC • CONCWCN • CONCWOC • CONNIDLETIME • DBCONN • ESTSQLCOST • ROWSRET • SCCONN • TEMPSPACE • TOTALTIME
THRESHOLDPREDICATEID	SMALLINT		しきい値の述部の ID。
DOMAIN	CHAR (2)		しきい値のドメイン。 • DB = データベース • SB = サービス・サブクラス • SP = サービス・スーパークラス • WA = 作業アクション・セット • WD = ワークロード定義
DOMAINID	INTEGER		しきい値が関連付けられているオブジェクトの ID。これにはサービス・クラス、作業アクション、またはワークロードの固有 ID を指定できます。これがデータベースのしきい値である場合は、値は 0 です。

表 149. SYSCAT.THRESHOLDS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ENFORCEMENT	CHAR (1)		しきい値の強制の有効範囲。 • D = データベース • P = データベース・パーティション • W = ワークロード・オカレンス
QUEUING	CHAR (1)		• N = しきい値はキューイングを行っていない • Y = しきい値はキューイングを行っている
MAXVALUE	BIGINT		しきい値によって指定された上限。
QUEUESIZE	INTEGER		QUEUING が 'Y' の場合は、キューのサイズ。それ以外の場合は -1。
COLLECTACTDATA	CHAR (1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • V = 詳細および値ありのアクティビティー・データ • W = 詳細なしのアクティビティー・データ
COLLECTACTPARTITION	CHAR (1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーターのデータベース・パーティション • D = すべてのデータベース・パーティション
EXECUTION	CHAR (1)		しきい値が超過した後、実行を続行するかどうかを指定します。 • C = 実行を続行する • S = 実行を停止する
ENABLED	CHAR (1)		• N = このしきい値は使用不可。 • Y = このしきい値は使用可能。
CREATE_TIME	TIMESTAMP		しきい値が作成された時刻。
ALTER_TIME	TIMESTAMP		しきい値が最後に変更された時刻。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.TRANSFORMS

各行は、ユーザー定義タイプから基本 SQL タイプへ、またはその逆のトランスフォーメーションを扱う関数を表します。

表 150. SYSCAT.TRANSFORMS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TYPEID	SMALLINT		データ・タイプの ID。
TYPESCHEMA	VARCHAR(128)		データ・タイプのスキーマ名。組み込みタイプのスキーマ名は 'SYSIBM' です。
TYPENAME	VARCHAR(128)		データ・タイプの非修飾名。
GROUPNAME	VARCHAR(128)		トランスフォーム・グループの名前。
FUNCID	INTEGER		ルーチンの ID。
FUNCSCHEMA	VARCHAR(128)		ルーチンのスキーマ名。
FUNCNAME	VARCHAR(128)		ルーチンの非修飾名。
SPECIFICNAME	VARCHAR(128)		ルーチン・インスタンスの名前 (システム生成の場合もある)。
TRANSFORMTYPE	VARCHAR(8)		• 'FROM SQL' = トランスフォーム関数は SQL から構造化タイプをトランスフォームする • 'TO SQL' = トランスフォーム関数は SQL に構造化タイプをトランスフォームする
FORMAT	CHAR(1)		FROM SQL トランスフォームによって作成される形式。 • S = 構造化データ・タイプ • U = ユーザー定義
MAXLENGTH	INTEGER	Y	FROM SQL トランスフォームからの出力の最大長 (バイト単位)。TO SQL トランスフォームの場合は NULL 値。
ORIGIN	CHAR(1)		トランスフォームのこのグループのソース。 • O = オリジナル・トランスフォーム・グループ (組み込みまたはシステム定義) • R = 再定義トランスフォーム・グループ (組み込みグループのみ再定義可能)
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.TRIGDEP

各行は、何か他のオブジェクトへのトリガーの従属関係を表します。トリガーは、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更はトリガーに影響します。

表 151. SYSCAT.TRIGDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TRIGSCHEMA	VARCHAR(128)		トリガーのスキーマ名。
TRIGNAME	VARCHAR(128)		トリガーの非修飾名。
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • B = トリガー • F = ルーチン・インスタンス • H = 階層表 • K = パッケージ • L = デタッチされた表 • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • Q = シーケンス • R = 構造化タイプ • S = マテリアライズ照会表 • T = 表 (型付きではない) • U = 型付き表 • V = ビュー (型付きではない) • W = 型付きビュー • X = 索引拡張 • Z = XSR オブジェクト • v = グローバル変数
BSCHEMA	VARCHAR(128)		従属関係があるオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		従属関係があるオブジェクトの非修飾名。ルーチン (BTYPE = 'F') の場合、これは特定名です。
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、従属トリガーで必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.TRIGGERS

各行はトリガーを表します。表階層の場合、トリガーはそれぞれ作成された階層レベルでのみ記録されます。

表 152. SYSCAT.TRIGGERS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TRIGSCHEMA	VARCHAR(128)		トリガーのスキーマ名。
TRIGNAME	VARCHAR(128)		トリガーの非修飾名。
OWNER	VARCHAR(128)		トリガー作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
TABSCHEMA	VARCHAR(128)		このトリガーが適用される表またはビューのスキーマ名。
TABNAME	VARCHAR(128)		このトリガーを適用される表またはビューの非修飾名。
TRIGTIME	CHAR(1)		トリガーを起動したイベントと比較して、いつトリガー・アクションが基本表へ適用されるか。 • A = トリガーはイベントの後に適用される • B = トリガーはイベントの前に適用される • I = トリガーはイベントの代わりに適用される
TRIGEVENT	CHAR(1)		トリガーを起動するイベント。 • D = 削除操作 • I = 挿入操作 • U = 更新操作
GRANULARITY	CHAR(1)		トリガーが実行される単位。 • R = 行 • S = ステートメント
VALID	CHAR(1)		• N = トリガーは無効 • Y = トリガーは有効 • X = トリガーは作動不能で、再作成が必要
CREATE_TIME	TIMESTAMP		トリガーが定義された時刻。関数とタイプの解決に使用されます。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。非修飾参照を完了するために使用します。
FUNC_PATH	CLOB(2K)		トリガーの定義された時点の SQL パス。関数とタイプの解決に使用されます。
TEXT	CLOB(2M)		入力されたとおりの CREATE TRIGGER ステートメントのテキスト全体。
LAST_REGEN_TIME	TIMESTAMP		トリガーのパック記述子が最後に再生成された時刻。

表 152. SYSCAT.TRIGGERS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLATIONSCHEMA	VARCHAR(128)		トリガーの照合のスキーマ名。
COLLATIONNAME	VARCHAR(128)		トリガーの照合の非修飾名。
COLLATIONSCHEMA_ ORDERBY	VARCHAR(128)		トリガーの ORDER BY 節の照合のスキーマ名。
COLLATIONNAME_ORDERBY	VARCHAR(128)		トリガーの ORDER BY 節の照合の非修飾名。
DEFINER[1]	VARCHAR(128)		トリガー作成時の許可 ID。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.TYPEMAPPINGS

各行は、ローカルで定義されたデータ・タイプとデータ・ソースのデータ・タイプとの間のマッピングを表します。マッピング・タイプ (マッピング方向) には、次の 2 種類があります。

- フォワード・タイプ・マッピングでは、データ・ソースのデータ・タイプをローカルで定義されたデータ・タイプへマップします。
- リバース・タイプ・マッピングでは、ローカルで定義されたデータ・タイプをデータ・ソースのデータ・タイプへマップします。

表 153. SYSCAT.TYPEMAPPINGS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TYPE_MAPPING	VARCHAR(18)		タイプ・マッピングの名前 (システム生成の場合もある)。
MAPPINGDIRECTION	CHAR(1)		このタイプ・マッピングが、フォワードまたはリバースのどちらのタイプのマッピングであるかを示します。 • F = フォワード・タイプ・マッピング • R = リバース・タイプ・マッピング
TYPESCHEMA	VARCHAR(128)	Y	データ・タイプ・マッピングでのローカル・タイプのスキーマ名。組み込みタイプの場合は NULL 値。
TYPENAME	VARCHAR(128)		データ・タイプ・マッピングでのローカル・タイプの非修飾名。
TYPEID	SMALLINT		データ・タイプの ID。
SOURCETYPEID	SMALLINT		ソース・タイプの ID。
OWNER	VARCHAR(128)		このタイプ・マッピングの作成に使用された許可 ID。 'SYSIBM' は、組み込みタイプ・マッピングを表します。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
LENGTH	INTEGER	Y	このマッピングでのローカル・データ・タイプの最大長または精度。NULL 値の場合、システムが最大長または精度を決定します。文字タイプの場合は、最大バイト数を表します。
SCALE	SMALLINT	Y	このマッピングでのローカル 10 進数値の小数部分の最大桁数。NULL 値の場合、システムが最大数を決定します。
LOWER_LEN	INTEGER	Y	このマッピングでのローカル・データ・タイプの最小長または精度。NULL 値の場合、システムが最小長または精度を決定します。文字タイプの場合は、バイトの最小数を表します。

表 153. SYSCAT.TYPEMAPPINGS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
UPPER_LEN	INTEGER	Y	このマッピングでのローカル・データ・タイプの最大長または精度。NULL 値の場合、システムが最大長または精度を決定します。文字タイプの場合は、最大バイト数を表します。
LOWER_SCALE	SMALLINT	Y	このマッピングでのローカル 10 進数値の小数部分の最小桁数。NULL 値の場合、システムが最小数を決定します。
UPPER_SCALE	SMALLINT	Y	このマッピングでのローカル 10 進数値の小数部分の最大桁数。NULL 値の場合、システムが最大数を決定します。
S_OPR_P	CHAR(2)	Y	このマッピングでのローカル 10 進数値の位取りと精度との間の関係。基本比較演算子 (=、<、>、<=、>=、<>) を使用できます。NULL 値であれば、特定の関係は不要です。
BIT_DATA	CHAR(1)	Y	この文字タイプがビット・データ用であるかどうかを示します。使用できる値は以下のとおりです。 • N = この型はビット・データ用ではない。 • Y = この型はビット・データ用です。 • NULL 値 = 文字データ・タイプではないか、またはシステムがビット・データ属性を決定する。
WRAPNAME	VARCHAR(128)	Y	このマッピングが適用されるデータ・アクセス・プロトコル (ラッパー)。
SERVERNAME	VARCHAR(128)	Y	サーバー名 (大文字)。
SERVERTYPE	VARCHAR(30)	Y	サーバーのタイプ。
SERVERVERSION	VARCHAR(18)	Y	サーバーのバージョン。
REMOTE_TYPESCHEMA	VARCHAR(128)	Y	データ・ソースのデータ・タイプのスキーマ名。
REMOTE_TYPENAME	VARCHAR(128)		データ・ソースのデータ・タイプの非修飾名。
REMOTE_META_TYPE	CHAR(1)	Y	このリモート・タイプがシステム組み込みタイプであるか特殊タイプであるかを示します。 • S = システム組み込みタイプ • T = 特殊タイプ

表 153. SYSCAT.TYPEMAPPINGS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REMOTE_LOWER_LEN	INTEGER	Y	このマッピングでのリモート・データ・タイプの最小長または精度、または NULL 値。文字タイプの場合は、文字 (バイトではない) の最小数を表します。バイナリー形式の場合は、バイトの最小数を表します。-1 という値は、デフォルトの長さまたは精度を使用することを示すか、またはリモート・タイプは長さも精度も持たないことを示します。
REMOTE_UPPER_LEN	INTEGER	Y	このマッピングでのリモート・データ・タイプの最大長または精度、または NULL 値。文字タイプの場合は、文字 (バイトではない) の最大数を表します。バイナリー形式の場合は、最大バイト数を表します。-1 という値は、デフォルトの長さまたは精度を使用することを示すか、またはリモート・タイプは長さも精度も持たないことを示します。
REMOTE_LOWER_SCALE	SMALLINT	Y	このマッピングでのリモート 10 進数値の小数部分の最小桁数、または NULL 値。
REMOTE_UPPER_SCALE	SMALLINT	Y	このマッピングでのリモート 10 進数値の小数部分の最大桁数、または NULL 値。
REMOTE_S_OPR_P	CHAR(2)	Y	このマッピングでのリモート 10 進数値の位取りと精度との間の関係。基本比較演算子 (=、<、>、<=、>=、<>) を使用できます。NULL 値であれば、特定の関係は不要です。
REMOTE_BIT_DATA	CHAR(1)	Y	このリモート文字タイプがビット・データ用であるかどうかを示します。使用できる値は以下のとおりです。 • N = この型はビット・データ用ではない。 • Y = この型はビット・データ用です。 • NULL 値 = 文字データ・タイプではないか、またはシステムがビット・データ属性を決定する。
USER_DEFINED	CHAR(1)		マッピングがユーザー定義であるかどうかを示します。値は常に 'Y' です。つまり、マッピングは常にユーザー定義です。
CREATE_TIME	TIMESTAMP		マッピングが作成された時刻。
DEFINER[1]	VARCHAR(128)		このタイプ・マッピングの作成に使用された許可 ID。 'SYSIBM' は、組み込みタイプ・マッピングを表します。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.USEROPTIONS

各行は、サーバー固有のユーザー・オプション値を表します。

表 154. SYSCAT.USEROPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
AUTHID	VARCHAR(128)		ローカル許可 ID (大文字)。
AUTHIDTYPE	CHAR(1)		• U = GRANTEE は個々のユーザー。
SERVERNAME	VARCHAR(128)		ユーザーが定義されたサーバーの名前。
OPTION	VARCHAR(128)		ユーザー・オプションの名前。
SETTING	VARCHAR (2048)		ユーザー・オプションの値。

SYSCAT.VARIABLEAUTH

各行は、データベースのグローバル変数に対して特定の認可者により 1 つ以上の特権を付与されたユーザー、グループ、またはロールを表します。

表 155. SYSCAT.VARIABLEAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
VARSCHEMA	VARCHAR(128)		グローバル変数のスキーマ名。
VARNAME	VARCHAR(128)		グローバル変数の非修飾名。
VARID	INTEGER		グローバル変数の ID。
READAUTH	CHAR(1)		グローバル変数を読み取る特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する
WRITEAUTH	CHAR(1)		グローバル変数を書き込む特権。 • G = 保有しており、付与可能 • N = 保有しない • Y = 保有する

SYSCAT.VARIABLEDEP

各行は、何らかの他のオブジェクトに対するグローバル変数の従属関係を表します。グローバル変数は、名前 BNAME のタイプ BTYPE のオブジェクトに依存するため、このオブジェクトの変更はグローバル変数に影響します。

表 156. SYSCAT.VARIABLEDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
VARSCHEMA	VARCHAR(128)		別のオブジェクトに従属するグローバル変数のスキーマ名。
VARNAME	VARCHAR(128)		別のオブジェクトに従属するグローバル変数の非修飾名。
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • F = ルーチン・インスタンス • H = 階層表 • N = ニックネーム • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • S = マテリアライズ照会表 • T = 表 (型付きではない) • U = 型付き表 • V = ビュー (型付きではない) • W = 型付きビュー • v = グローバル変数
BSCHEMA	VARCHAR(128)		従属関係があるオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		従属関係があるオブジェクトの非修飾名。ルーチン (BTYPE = 'F') の場合、これは特定名です。
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、グローバル変数で必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.VARIABLES

各行はグローバル変数を表します。

表 157. SYSCAT.VARIABLES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
VARSCHEMA	VARCHAR(128)		グローバル変数のスキーマ名。
VARNAME	VARCHAR(128)		グローバル変数の非修飾名。
VARID	INTEGER		グローバル変数の ID。
OWNER	VARCHAR(128)		グローバル変数の所有者の許可 ID。
OWNERTYPE	CHAR(1)		• U = 所有者は個々のユーザー
CREATE_TIME	TIMESTAMP		グローバル変数が作成された時刻。
LAST_REGEN_TIME	TIMESTAMP		デフォルトの式が最後に再生成された時刻。
TYPESCHEMA	VARCHAR(128)		データ・タイプのスキーマ名。組み込みタイプのスキーマ名は 'SYSIBM' です。
TYPENAME	VARCHAR(128)		データ・タイプの非修飾名。
LENGTH	INTEGER		グローバル変数の最大長。
SCALE	SMALLINT		グローバル変数の位取り。
CODEPAGE	SMALLINT		グローバル変数のコード・ページ。
COLLATIONSCHEMA	VARCHAR(128)		変数の照合のスキーマ名。
COLLATIONNAME	VARCHAR (128)		変数の照合の非修飾名。
COLLATIONSCHEMA_ORDERBY	VARCHAR (128)		変数の ORDER BY 節の照合のスキーマ名。
COLLATIONNAME_ORDERBY	VARCHAR(128)		変数の ORDER BY 節の照合の非修飾名。
SCOPE	CHAR(1)		グローバル変数の有効範囲。 • S = セッション
DEFAULT	CLOB(64K)	Y	最初に参照されるときのグローバル変数の初期値を計算するのに使用される式。
QUALIFIER	VARCHAR(128)	Y	変数が定義された時点のデフォルト・スキーマの値。
FUNC_PATH	CLOB(2K)	Y	変数が定義された時点の SQL パス。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.VIEWS

各行はビューを表します。

表 158. SYSCAT.VIEWS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
VIEWSCHEMA	VARCHAR(128)		ビューのスキーマ名。
VIEWNAME	VARCHAR(128)		ビューの非修飾名。
OWNER	VARCHAR(128)		ビュー作成時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
SEQNO	SMALLINT		常に 1。
VIEWCHECK	CHAR(1)		ビュー検査のタイプ • C = カスケード検査オプション • L = ローカル検査オプション • N = 検査オプションはない
READONLY	CHAR(1)		• N = ビューは適切な許可のあるユーザーによって更新可能 • Y = ビューはその定義により読み取り専用
VALID	CHAR(1)		• X = ビューまたはマテリアライズ照会表の定義が作動不能である。再作成が必要。 • Y = ビューまたはマテリアライズ照会表の定義が有効である。
QUALIFIER	VARCHAR(128)		オブジェクト定義時のデフォルト・スキーマの値。非修飾参照を完了するために使用します。
FUNC_PATH	CLOB(2K)		ビューが定義された時点で有効だった SQL パス。ビューをデータ操作言語 (DML) ステートメントで参照する場合、ビューにおける関数呼び出しを解決するのにこのパスを使用する必要があります。バージョン 2 より前のビューの場合、'SYSIBM'。
TEXT	CLOB(2M)		入力されたとおりの CREATE VIEW ステートメントのテキスト全体。
DEFINER[1]	VARCHAR(128)		ビュー作成時の許可 ID。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.WORKACTIONS

各行は、作業アクション・セットで定義されている作業アクションを表します。

表 159. SYSCAT.WORKACTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ACTIONNAME	VARCHAR(128)		作業アクションの名前。
ACTIONID	INTEGER		作業アクションの ID。
ACTIONSETNAME	VARCHAR(128)	Y	作業アクション・セットの名前。
ACTIONSETID	INTEGER		この作業アクションが属する作業アクション・セットの ID。この列は、SYSCAT.WORKACTIONSETS ビューの ACTIONSETID 列を参照します。
WORKCLASSNAME	VARCHAR(128)	Y	作業クラスの名前。
WORKCLASSID	INTEGER		作業クラスの ID。この列は、SYSCAT.WORKCLASSES ビューの WORKCLASSID 列を参照します。
CREATE_TIME	TIMESTAMP		作業アクションが作成された時刻。
ALTER_TIME	TIMESTAMP		作業アクションが最後に変更された時刻。
ENABLED	CHAR(1)		• N = この作業アクションは使用不可です。 • Y = この作業アクションは使用可能です。

表 159. SYSCAT.WORKACTIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACTIONTYPE	CHAR(1)		作業クラス属性 (突き合わせ有効範囲内の作業クラスで指定された) と一致する各 DB2 アクティビティーで実行されるアクション・タイプ。この列の記述の OBJECTTYPE は、SYSCAT.WORKACTIONSETS の列 OBJECTTYPE を参照します。 • B - 基礎集約アクティビティー・データを収集します。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。 • C - この作業アクションが関連付けられている作業クラスに該当するすべての DB2 アクティビティーの実行を許可し、作業クラスのカウンターを増分します。 • D - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細を含むアクティビティー・データを収集します。 • E - 拡張された集約アクティビティー・データを収集します。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。 • M - サービス・サブクラスにマップします。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。 • P - この作業アクションが関連付けられている作業クラスに該当する DB2 アクティビティーが実行されないようにします。 • T - アクションはしきい値の形式になります。このアクション・タイプは、OBJECTTYPE がブランク (データベース) の場合のみ指定できます。 • U - ネスト・レベルがゼロのアクティビティーおよびこのアクティビティーの下にネストされているアクティビティーすべてをサービス・サブクラスにマップします。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。

表 159. SYSCAT.WORKACTIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACTIONTYPE (cont'd)			• V - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細と値を含むアクティビティー・データを収集します。 • W - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細を含まないアクティビティー・データを収集します。 • X - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細を含むアクティビティー・データと、すべてのデータベース・パーティションにおけるアクティビティー・データを収集します。 • Y - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細と値を含むアクティビティー・データと、すべてのデータベース・パーティションにおけるアクティビティー・データを収集します。 • Z - すべてのデータベース・パーティションにおける、詳細を含まないアクティビティー・データを収集します。
REFOBJECTID	INTEGER	Y	ACTIONTYPE が 'M' (マップ) または 'N' (ネストされたマップ) の場合、この値は DB2 アクティビティーのマップ先のサービス・サブクラスの ID に設定されます。ACTIONTYPE が 'T' (しきい値) の場合、この値は、使用されるしきい値の ID に設定されます。それ以外のすべてのアクションの場合、この値は NULL になります。
REFOBJECTTYPE	VARCHAR (30)		ACTIONTYPE が 'M' または 'N' の場合、この値は 'SERVICE CLASS' に設定されます。ACTIONTYPE が 'T' の場合、この値は 'THRESHOLD' になり、それ以外の場合は NULL 値になります。

SYSCAT.WORKACTIONSETS

各行は、作業アクション・セットを表します。

表 160. SYSCAT.WORKACTIONSETS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ACTIONSETNAME	VARCHAR (128)		作業アクション・セットの名前。
ACTIONSETID	INTEGER		作業アクション・セットの ID。
WORKCLASSSETNAME	VARCHAR (128)	Y	作業クラス・セットの名前。
WORKCLASSSETID	INTEGER		OBJECTID で指定されたオブジェクトにマップされる作業クラス・セットの ID。この列は SYSCAT.WORKCLASSSETS ビューの WORKCLASSSETID を参照します。
CREATE_TIME	TIMESTAMP		作業アクション・セットが作成された時刻。
ALTER_TIME	TIMESTAMP		作業アクション・セットが最後に変更された時刻。
ENABLED	CHAR (1)		• N = この作業アクション・セットは使用不可です。 • Y = この作業アクション・セットは使用可能です。
OBJECTTYPE	CHAR (1)		• b = サービス・スーパークラス • ブランク = データベース
OBJECTNAME	VARCHAR (128)	Y	サービス・クラスの名前。
OBJECTID	INTEGER		作業クラス・セット (WORKCLASSSETID で指定される) のマップ先のオブジェクトの ID。OBJECTTYPE がブランクの場合、OBJECTID は -1 です。 OBJECTTYPE が 'b' の場合、OBJECTID はサービス・スーパークラスの ID です。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKCLASSES

各行は、作業クラス・セットで定義されている作業クラスを表します。

表 161. SYSCAT.WORKCLASSES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKCLASSNAME	VARCHAR(128)		作業クラスの名前。
WORKCLASSSETNAME	VARCHAR(128)	Y	作業クラス・セットの名前。
WORKCLASSID	INTEGER		作業クラスの ID。
WORKCLASSSETID	INTEGER		この作業クラスが属する作業クラス・セットの ID。この列は、SYSCAT.WORKCLASSSETS ビューの WORKCLASSSETID 列を参照します。
CREATE_TIME	TIMESTAMP		作業クラスが作成された時刻。
ALTER_TIME	TIMESTAMP		作業クラスが最後に変更された時刻。
WORKTYPE	SMALLINT		DB2 アクティビティーのタイプ。 • 1 = ALL • 2 = READ • 3 = WRITE • 4 = CALL • 5 = DML • 6 = DDL • 7 = LOAD
RANGEUNITS	CHAR(1)		上下範囲に使用する単位。 • C = カーディナリティー • T = Timeron • ブランク = 該当しない場合
FROMVALUE	DOUBLE	Y	RANGEUNITS で指定された単位による、範囲の低い値。 RANGEUNITS がブランクの場合は NULL 値になります。
TOVALUE	DOUBLE	Y	RANGEUNITS で指定された単位による、範囲の高い値。 RANGEUNITS がブランクの場合は NULL 値になります。値 -1 は上限なしを示すために使われます。
ROUTINESCHEMA	VARCHAR(128)	Y	CALL ステートメントから呼び出されるプロシージャーのスキーマ名。 WORKTYPE が 4 (CALL) または 1 (ALL) でない場合は NULL 値。
EVALUATIONORDER	SMALLINT		作業クラス・セット内の作業クラスを選択するときに使用される評価順序を一意的に識別します。

SYSCAT.WORKCLASSSETS

各行は、作業クラス・セットを表します。

表 162. SYSCAT.WORKCLASSSETS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKCLASSSETNAME	VARCHAR(128)		作業クラス・セットの名前。
WORKCLASSSETID	INTEGER		作業クラス・セットの ID。
CREATE_TIME	TIMESTAMP		作業クラス・セットが作成された時刻。
ALTER_TIME	TIMESTAMP		作業クラス・セットが最後に変更された時刻。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKLOADAUTH

各行は、ワークロードに対する USAGE 特権が付与されているユーザー、グループ、またはロールを表します。

表 163. SYSCAT.WORKLOADAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR(128)		ワークロードの名前。
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• U = GRANTEE は個々のユーザー。
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
USAGEAUTH	CHAR(1)		GRANTEE がワークロードに対する USAGE 特権を保有するかどうかを示します。 • N = 保有しない • Y = 保有する

SYSCAT.WORKLOADCONNATTR

各行は、ワークロードの定義における接続属性を表します。

表 164. SYSCAT.WORKLOADCONNATTR カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR(128)		ワークロードの名前。
CONNATTRTYPE	VARCHAR(30)		接続属性のタイプ。 • 1 = APPLNAME • 2 = SYSTEM_USER • 3 = SESSION_USER • 4 = SESSION_USER GROUP • 5 = SESSION_USER ROLE • 6 = CURRENT CLIENT_USERID • 7 = CURRENT CLIENT_APPLNAME • 8 = CURRENT CLIENT_WRKSTNNAME • 9 = CURRENT CLIENT_ACCTNG
CONNATTRVALUE	VARCHAR(1000)		接続属性の値。

SYSCAT.WORKLOADS

各行はワークロードを表します。

表 165. SYSCAT.WORKLOADS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR(128)		ワークロードの名前。
EVALUATIONORDER	SMALLINT		ワークロードの選択に使用される評価順序。
CREATE_TIME	TIMESTAMP		ワークロードが作成された時刻。
ALTER_TIME	TIMESTAMP		ワークロードが最後に変更された時刻。
ENABLED	CHAR(1)		• N = このワークロードは使用不可。 • Y = このワークロードは使用可能。
ALLOWACCESS	CHAR(1)		• N = このワークロードに関連付けられている UOW は拒否される。 • Y = このワークロードに関連付けられている作業単位 (UOW) はデータベースにアクセスできる。
SERVICECLASSNAME	VARCHAR(128)		(このワークロードに関連付けられている) 作業単位が割り当てられるサービス・サブクラスの名前。
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	(このワークロードに関連付けられている) 作業単位が割り当てられるサービス・スーパークラスの名前。
COLLECTAGGACTDATA	CHAR(1)		このワークロードについて、該当するイベント・モニターでキャプチャーする集約アクティビティー・データを指定します。 • N = なし
COLLECTACTDATA	CHAR(1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • V = 詳細および値ありのアクティビティー・データ (COLLECT 列が 'C' に設定されている場合に適用される) • W = 詳細なしのアクティビティー・データ
COLLECTACTPARTITION	CHAR(1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーターのデータベース・パーティション • D = すべてのデータベース・パーティション
EXTERNALNAME	VARCHAR(128)	Y	将来の使用のために予約済み。

表 165. SYSCAT.WORKLOADS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WRAPOPTIONS

各行は、ラッパー固有のオプションを表します。

表 166. SYSCAT.WRAPOPTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WRAPNAME	VARCHAR(128)		ラッパーの名前。
OPTION	VARCHAR(128)		ラッパー・オプションの名前。
SETTING	VARCHAR (2048)		ラッパー・オプションの値。

SYSCAT.WRAPPERS

各行は登録されたラッパーを表します。

表 167. SYSCAT.WRAPPERS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WRAPNAME	VARCHAR(128)		ラッパーの名前。
WRAPTYPE	CHAR(1)		ラッパーのタイプ • N = 非リレーショナル • R = リレーショナル
WRAPVERSION	INTEGER		ラッパーのバージョン。
LIBRARY	VARCHAR(255)		このラッパーに関連したデータ・ソースとの通信に使用するコードが入っているファイルの名前。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.XDBMAPGRAPHS

各行は XDB マップ (XSR オブジェクト) のスキーマ・グラフを表します。

表 168. SYSCAT.XDBMAPGRAPHS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
OBJECTID	BIGINT		XSR オブジェクトのユニーク生成 ID。
OBJECTSCHEMA	VARCHAR(128)		XSR オブジェクトのスキーマ名。
OBJECTNAME	VARCHAR(128)		XSR オブジェクトの非修飾名。
SCHEMAGRAPHID	INTEGER		スキーマ・グラフ ID。これは XDB マップ ID 内で固有です。
NAMESPACE	VARCHAR(1001)	Y	ルート・エレメントのネーム・スペース URI のストリング ID。
ROOTELEMENT	VARCHAR(1001)	Y	ルート・エレメントのエレメント名のストリング ID。

SYSCAT.XDBMAPSHREDTREES

各行は、指定のスキーマ・グラフ ID の 1 つの分解ツリーを表します。

表 169. SYSCAT.XDBMAPSHREDTREES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
OBJECTID	BIGINT		XSR オブジェクトのユニーク生成 ID。
OBJECTSCHEMA	VARCHAR(128)		XSR オブジェクトのスキーマ名。
OBJECTNAME	VARCHAR(128)		XSR オブジェクトの非修飾名。
SCHEMAGRAPHID	INTEGER		スキーマ・グラフ ID。これは XDB マップ ID 内で固有です。
SHREDTREEID	INTEGER		分解ツリー ID。これは XDB マップ ID 内で固有です。
MAPPINGDESCRIPTION	CLOB(1M)	Y	診断マッピング情報。

SYSCAT.XSROBJECTAUTH

各行は、特定の XSR オブジェクトに対する USAGE 特権を付与されているユーザーまたはグループを表します。

表 170. SYSCAT.XSROBJECTAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• S = 認可者はシステム • U = 認可者は個々のユーザー
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
OBJECTID	BIGINT		XSR オブジェクトの ID。
USAGEAUTH	CHAR(1)		XSR オブジェクトとそのコンポーネントを使用する特権。 • N = 保有しない • Y = 保有する

SYSCAT.XSROBJECTCOMPONENTS

各行は、XSR オブジェクトのコンポーネントを表します。

表 171. SYSCAT.XSROBJECTCOMPONENTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
OBJECTID	BIGINT		XSR オブジェクトのユニーク生成 ID。
OBJECTSCHEMA	VARCHAR(128)		XSR オブジェクトのスキーマ名。
OBJECTNAME	VARCHAR(128)		XSR オブジェクトの非修飾名。
COMPONENTID	BIGINT		XSR オブジェクト・コンポーネントのユニーク生成 ID。
TARGETNAMESPACE	VARCHAR(1001)	Y	ターゲット・ネーム・スペースのストリング ID。
SCHEMALOCATION	VARCHAR(1001)	Y	スキーマ・ロケーションのストリング ID。
COMPONENT	BLOB(30M)		コンポーネントの外部表記。
CREATE_TIME	TIMESTAMP		XSR オブジェクト・コンポーネントが登録された時刻。
STATUS	CHAR(1)		登録状況。 • C = 完了 • I = 未完了

SYSCAT.XSROBJECTDEP

各行は、何らかの他のオブジェクトに対する XSR オブジェクトの従属関係を表します。 XSR オブジェクトは、名前 BNAME のタイプ BTYPE のオブジェクトに従属するため、このオブジェクトの変更は XSR オブジェクトに影響します。

表 172. SYSCAT.XSROBJECTDEP カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
OBJECTID	BIGINT		XSR オブジェクトのユニーク生成 ID。
OBJECTSCHEMA	VARCHAR(128)		XSR オブジェクトのスキーマ名。
OBJECTNAME	VARCHAR(128)		XSR オブジェクトの非修飾名。
BTYPE	CHAR(1)		従属関係があるオブジェクトのタイプ。使用できる値は以下のとおりです。 • A = 別名 • B = トリガー • F = ルーチン・インスタンス • H = 階層表 • K = パッケージ • L = デタッチされた表 • O = 表またはビュー階層内のすべての副表またはサブビューに対する特権の従属関係 • Q = シーケンス • R = 構造化タイプ • S = マテリアライズ照会表 • T = 表 (型付きではない) • U = 型付き表 • V = ビュー (型付きではない) • W = 型付きビュー • X = 索引拡張 • Z = XSR オブジェクト • v = グローバル変数
BSCHEMA	VARCHAR(128)		従属関係があるオブジェクトのスキーマ名。
BNAME	VARCHAR(128)		従属関係があるオブジェクトの非修飾名。ルーチン (BTYPE = 'F') の場合、これは特定名です。
TABAUTH	SMALLINT	Y	BTYPE= 'O'、'S'、'T'、'U'、'V'、'W'、または 'v' の場合、従属トリガーで必要な表またはビューの特権をエンコードします。それ以外の場合は NULL 値。

SYSCAT.XSROBJECTHIERARCHIES

各行は、XSR オブジェクトとそのコンポーネントとの間の階層リレーションシップを表します。

表 173. SYSCAT.XSROBJECTHIERARCHIES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
OBJECTID	BIGINT		XSR オブジェクトの ID。
COMPONENTID	BIGINT		XSR コンポーネントの ID。
HTYPE	CHAR(1)		階層タイプ。 • D = 文書 • N = 最上位ネーム・スペース • P = 1 次文書
TARGETNAMESPACE	VARCHAR(1001)	Y	コンポーネントのターゲット・ネーム・スペースのストリング ID。
SCHEMALOCATION	VARCHAR(1001)	Y	コンポーネントのスキーマ・ロケーションのストリング ID。

SYSCAT.XSROBJECTS

各行は、XML スキーマ・リポジトリー・オブジェクトを表します。

表 174. SYSCAT.XSROBJECTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
OBJECTID	BIGINT		XSR オブジェクトのユニーク生成 ID。
OBJECTSCHEMA	VARCHAR(128)		XSR オブジェクトのスキーマ名。
OBJECTNAME	VARCHAR(128)		XSR オブジェクトの非修飾名。
TARGETNAMESPACE	VARCHAR(1001)	Y	ターゲット・ネーム・スペースのストリング ID または公開 ID。
SCHEMALOCATION	VARCHAR(1001)	Y	スキーマ・ロケーションのストリング ID、またはシステム ID。
OBJECTINFO	XML	Y	メタデータ文書。
OBJECTTYPE	CHAR(1)		XSR オブジェクト・タイプ。 • D = DTD • E = 外部エンティティー • S = XML スキーマ
OWNER	VARCHAR(128)		XSR オブジェクト登録時の許可 ID。
OWNERTYPE	CHAR(1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
CREATE_TIME	TIMESTAMP		オブジェクトが登録された時刻。
ALTER_TIME	TIMESTAMP		オブジェクトが最後に更新された (置き換えられた) 時刻。
STATUS	CHAR(1)		登録状況。 • C = 完了 • I = 未完了 • R = 置換 • T = 一時的
DECOMPOSITION	CHAR(1)		分解 (断片化) がこの XSR オブジェクトで使用可能かどうかを示します。 • N = 使用可能でない • X = 作動不能 • Y = 使用可能
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSIBM.SYSDUMMY1

1 つの行が入っています。このビューは、DB2 Universal Database for z/OS との互換性を必要とするアプリケーションで使用できます。

表 175. SYSIBM.SYSDUMMY1 カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
IBMREQD	CHAR(1)		'Y'

SYSSTAT.COLDIST

各行は、列の中で *n* 番目に高い頻度の値、または列の *n* 番目の変位 (累積分布) 値を表します。実表 (ビューではない) の列にのみ適用されます。型付き表の継承列の場合、統計は記録されません。

表 176. SYSSTAT.COLDIST カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
TABSCHEMA	VARCHAR(128)			統計が適用される表のスキーマ名。
TABNAME	VARCHAR(128)			統計が適用される表の非修飾名。
COLNAME	VARCHAR(128)			統計が適用される列の名前。
TYPE	CHAR(1)			• F = 頻度 • Q = 変位値
SEQNO	SMALLINT			TYPE= 'F' の場合、この列の値 *n* は最大頻度が *n* 番目であることを示す。TYPE= 'Q' の場合、この列の値 *n* は変位値が *n* 番目であることを示す。
COLVALUE[1]	VARCHAR(254)	Y	Y	データ値 (文字リテラルまたは NULL 値)。
VALCOUNT	BIGINT		Y	TYPE= 'F' の場合、VALCOUNT は、その列の中の COLVALUE の出現回数。TYPE= 'Q' の場合、VALCOUNT は、値が COLVALUE 以下の行の数。
DISTCOUNT[2]	BIGINT	Y	Y	TYPE= 'Q' の場合、この列は COLVALUE 以下の特殊値の数 (入手不能の場合は NULL) を記録します。

注:

1. カタログ・ビュー内の COLVALUE の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。
2. DISTCOUNT は、索引の最初のキー列である列でのみ収集されます。

SYSSTAT.COLGROUPDIST

各行は、列グループの中で *n* 番目に高い頻度の値または *n* 番目の変位値を構成する、列グループ内の列の値を表します。

表 177. SYSSTAT.COLGROUPDIST カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
COLGROUPID	INTEGER			列グループの ID。
TYPE	CHAR(1)			• F = 頻度 • Q = 変位値
ORDINAL	SMALLINT			列グループ内の列の順序数。
SEQNO	SMALLINT			TYPE= 'F' の場合、この列の値 *n* は最大頻度が *n* 番目であることを示す。TYPE= 'Q' の場合、この列の値 *n* は変位値が *n* 番目であることを示す。
COLVALUE	VARCHAR(254)		Y	データ値 (文字リテラルまたは NULL 値)。

SYSSTAT.COLGROUPDISTCOUNTS

各行は、列グループの中で *n* 番目に高い頻度の値、または列グループの中で *n* 番目の変位値に適用される分散統計を表します。

表 178. SYSSTAT.COLGROUPDISTCOUNTS カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
COLGROUPID	INTEGER			列グループの ID。
TYPE	CHAR(1)			• F = 頻度 • Q = 変位値
SEQNO	SMALLINT			*n* 番目の TYPE 値を表すシーケンス番号 *n*。
VALCOUNT	BIGINT		Y	TYPE= 'F' の場合、VALCOUNT は、この SEQNO を持つ列グループの中の COLVALUE の出現回数です。 TYPE= 'Q' の場合、VALCOUNT は、値がこの SEQNO を持つ列グループの COLVALUE 以下の行の数です。
DISTCOUNT	BIGINT		Y	TYPE= 'Q' の場合、この列はこの SQENO を持つ列グループの COLVALUE 以下の値の種類数 (入手不能の場合は NULL) を記録します。

SYSSTAT.COLGROUPS

各行は、列グループ、およびその列グループ全体に適用される統計を表します。

表 179. SYSSTAT.COLGROUPS カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
COLGROUPSCHEMA	VARCHAR(128)			列グループのスキーマ名。
COLGROUPNAME	VARCHAR(128)			列グループの非修飾名。
COLGROUPID	INTEGER			列グループの ID。
COLGROUPCARD	BIGINT		Y	列グループのカーディナリティー。
NUMFREQ_VALUES	SMALLINT			列グループに関して収集された頻度の数。
NUMQUANTILES	SMALLINT			列グループに関して収集された変位値の数。

SYSSTAT.COLUMNS

各行は、表、ビュー、またはニックネームに定義された列を表します。

表 180. SYSSTAT.COLUMNS カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
TABSCHEMA	VARCHAR(128)			この列のある表、ビュー、またはニックネームのスキーマ名。
TABNAME	VARCHAR(128)			この列のある表、ビュー、またはニックネームの非修飾名。
COLNAME	VARCHAR(128)			列の名前。
COLCARD	BIGINT		Y	列の特殊値の数。統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
HIGH2KEY[1]	VARCHAR(254)	Y	Y	2 番目に高いデータ値。数値データを文字リテラルに変更して表現。統計が収集されていない場合は空。継承列および階層表の列の場合は空。
LOW2KEY[1]	VARCHAR(254)	Y	Y	2 番目に低いデータ値。数値データを文字リテラルに変更して表現。統計が収集されていない場合は空。継承列および階層表の列の場合は空。
AVGCOLLEN	INTEGER		Y	列に必要な平均のスペース (バイト単位)。長形式フィールドまたは LOB の場合、または統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。
NUMNULLS	BIGINT		Y	列内の NULL 値の数。統計が収集されていない場合は -1。
SUB_COUNT	SMALLINT		Y	列のサブエレメントの平均数。文字ストリング列のみに適用されます。
SUB_DELIM_LENGTH	SMALLINT		Y	列内の各サブエレメントを区切る、区切り文字の平均の長さ。文字ストリング列のみに適用されます。
AVGCOLLENCHAR	INTEGER		Y	列に必要な平均文字数 (列で有効になっている照合に基づく)。長形式フィールドまたは LOB の場合、または統計が収集されていない場合は -1。継承列および階層表の列の場合は -2。

注:

1. カタログ・ビューでは、HIGH2KEY と LOW2KEY の値は常にデータベース・コード・ページ中に示されますが、これには置換文字を収めることができます。ただし、列の表のコード・ページ内で内部的に統計が収集されるので、照会の最適化時に適用するときは実際の列値が使用されます。

SYSSTAT.INDEXES

各行は、索引を表します。型付き表の索引は、2 つの行で表されます。1 つは型付き表の「論理索引」用、もう 1 つは階層表の「階層索引」用です。

表 181. SYSSTAT.INDEXES カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
INDSCHEMA	VARCHAR(128)			索引のスキーマ名。
INDNAME	VARCHAR(128)			索引の非修飾名。
TABSCHEMA	VARCHAR(128)			索引が定義されている表またはニックネームのスキーマ名。
TABNAME	VARCHAR(128)			索引が定義されている表またはニックネームの非修飾名。
COLNAMES	VARCHAR(640)			この列は使用されなくなりました。次のリリースで除去されます。
NLEAF	BIGINT		Y	リーフ・ページの数。統計が収集されていない場合は -1。
NLEVELS	SMALLINT		Y	索引レベルの数。統計が収集されていない場合は -1。
FIRSTKEYCARD	BIGINT		Y	最初のキーの値の種類数。統計が収集されていない場合は -1。
FIRST2KEYCARD	BIGINT		Y	索引の最初の 2 つの列を使用するキーの種類数。統計が収集されていない場合、または適用されない場合は -1。
FIRST3KEYCARD	BIGINT		Y	索引の最初の 3 つの列を使用するキーの種類数。統計が収集されていない場合、または適用されない場合は -1。
FIRST4KEYCARD	BIGINT		Y	索引の最初の 4 つの列を使用するキーの種類数。統計が収集されていない場合、または適用されない場合は -1。
FULLKEYCARD	BIGINT		Y	全キー値の種類数。統計が収集されていない場合は -1。
CLUSTERRATIO[4]	SMALLINT		Y	索引によるデータ・クラスタリングの程度。統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1 (それらの場合は CLUSTERFACTOR の方が使用されます)。
CLUSTERFACTOR[4]	DOUBLE		Y	より高い計算精度のクラスタリング。統計を収集していない場合、あるいはニックネームに索引が定義されている場合は -1。
SEQUENTIAL_PAGES	BIGINT		Y	索引キー順にディスクに存在し、それらの間にほとんど（またはまったく）大きなギャップがないようなリーフ・ページの数。統計が収集されていない場合は -1。

表 181. SYSSTAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	更新可能	説明
DENSITY	INTEGER		Y	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0 から 100 の整数)。統計が収集されていない場合は -1。
PAGE_FETCH_PAIRS[4]	VARCHAR(520)		Y	文字形式で表された整数ペアのリスト。それぞれのペアは、仮のバッファー内のページ数と、その仮のバッファーを使用した表のスキャンに必要なページ・フェッチ回数を表しています。データが利用できない場合は、長さゼロのストリング。
NUMRIDS[4]	BIGINT		Y	索引内の行 ID (RID) またはブロック ID (BID) の合計数。不明の場合、-1。
NUMRIDS_DELETED[4]	BIGINT		Y	削除対象としてマークされている、索引内の行 ID (またはブロック ID) の合計数 (すべての ID が削除対象としてマークされている、リーフ・ページ上の ID は除く)。
NUM_EMPTY_LEAFS	BIGINT		Y	すべての行 ID (またはブロック ID) が削除対象としてマークされている、索引リーフ・ページの合計数。
AVERAGE_RANDOM_ FETCH_PAGES[1,2,4]	DOUBLE		Y	索引を使用してフェッチする際の、順次ページ・アクセス間のランダム表ページの平均数。不明の場合は -1。
AVERAGE_RANDOM_ PAGES[2]	DOUBLE		Y	順次ページ・アクセス間のランダム表ページの平均数。不明の場合は -1。
AVERAGE_SEQUENCE_ GAP[2]	DOUBLE		Y	索引ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、索引ページ・シーケンスの間でランダムにフェッチしなければならない索引ページの平均数を表します。不明の場合は -1。
AVERAGE_SEQUENCE_ FETCH_GAP[1,2,4]	DOUBLE		Y	索引を使用してフェッチする際の、表ページ・シーケンス間のギャップ。各ギャップは索引リーフ・ページのスキャンにより検出され、一連の表ページの間でランダムにフェッチしなければならない表ページの平均数を表します。不明の場合は -1。
AVERAGE_SEQUENCE_ PAGES[2]	DOUBLE		Y	順次にアクセス可能な索引ページの平均数 (つまり、順番になっているものとしてプリフェッチャーが検出する索引ページの数)。不明の場合は -1。

表 181. SYSSTAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	更新可能	説明
AVERAGE_SEQUENCE_FETCH_PAGES[1,2,4]	DOUBLE		Y	索引を使用してフェッチする際の、順次にアクセス可能な表ページの平均数 (つまり、プリフェッチャーが順次として検出する表ページの数)。不明の場合は -1。
AVGPARTITION_CLUSTERRATIO[3,4]	SMALLINT		Y	単一のデータ・パーティション内でのデータ・クラスタリングの程度。表がパーティション化されていない場合、統計が収集されていない場合、または詳細な索引統計が収集されている場合 (その場合は AVGPARTITION_CLUSTERFACTOR の方が使用されます) は、-1。
AVGPARTITION_CLUSTERFACTOR[3,4]	DOUBLE		Y	単一のデータ・パーティション内でのクラスタリングの程度の詳細測定値。表がパーティション化されていない場合、統計が収集されていない場合、あるいはニックネームに索引が定義されている場合は -1。
AVGPARTITION_PAGE_FETCH_PAIRS[3,4]	VARCHAR(520)		Y	文字形式の整数ペアのリスト。各ペアは、潜在的なバッファー・プール・サイズと、表の単一データ・パーティションにアクセスするのに必要なページのフェッチ回数との対応を示します。データが利用できない場合、または表がパーティション化されていない場合は、長さゼロのストリング。
DATAPARTITION_CLUSTERFACTOR	DOUBLE		Y	データ・パーティションに関しての索引キーの「クラスタリング」を測定する統計。これは 0 から 1 の間の数値で、1 は完全なクラスタリングを表し、0 はクラスタリングがないことを表します。
INDCARD	BIGINT		Y	索引のカーディナリティー。表の行と索引項目との間に 1 対 1 の関係がない索引の場合、これは表のカーディナリティーと異なる場合があります。

表 181. SYSSTAT.INDEXES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	更新可能	説明

注:

1. DMS 表スペースの使用時には、この統計は計算されません。
2. LOAD...STATISTICS YES または CREATE INDEX...COLLECT STATISTICS の操作時や、データベース構成パラメーター *seqdetect* がオフになっているときは、プリフェッチ統計は集められません。
3. AVGPARTITION_CLUSTERRATIO、AVGPARTITION_CLUSTERFACTOR、および AVGPARTITION_PAGE_FETCH_PAIRS は、単一のデータ・パーティション内でのクラスタリング (ローカル・クラスタリング) の程度を測定します。CLUSTERRATIO、CLUSTERFACTOR、および PAGE_FETCH_PAIRS は、表全体におけるクラスタリング (グローバル・クラスタリング) の程度を測定します。表パーティション・キーが索引キーの接頭部でない場合、あるいは表パーティション・キーと索引キーが論理的に互いに独立している場合、グローバル・クラスタリングの値とローカル・クラスタリングの値の差が大きくなる場合があります。
4. 索引タイプが 'XPTH' (XML パス索引) である場合、この統計は更新できません。
5. XML 列上の論理索引には統計がないため、索引タイプが 'XVIL' の行は SYSSTAT.INDEXES カタログ・ビューから除外されています。

SYSSTAT.ROUTINES

各行はユーザー定義ルーチン (スカラー関数、表関数、ソース派生関数、メソッド、またはプロシージャー) を表します。組み込み関数は含まれません。

表 182. SYSSTAT.ROUTINES カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
ROUTINESCHEMA	VARCHAR(128)			ルーチンのスキーマ名。
ROUTINENAME	VARCHAR(128)			ルーチンの非修飾名。
ROUTINETYPE	CHAR(1)			ルーチンのタイプ • F = 関数 • M = メソッド • P = プロシージャー
SPECIFICNAME	VARCHAR(128)			ルーチン・インスタンスの名前 (システム生成の場合もある)。
IOS_PER_INVOC	DOUBLE		Y	呼び出しごとの入力/出力 (I/O) の推定数。0 がデフォルト。不明の場合は -1。
INSTS_PER_INVOC	DOUBLE		Y	呼び出しごとの命令の数の見積もり。デフォルト値は 450。不明の場合は -1。
IOS_PER_ARGBYTE	DOUBLE		Y	入力引数 1 バイトごとの入出力回数の見積もり。デフォルト値は 0。不明の場合は -1。
INSTS_PER_ARGBYTE	DOUBLE		Y	入力引数 1 バイトごとの命令数の見積もり。0 がデフォルト。不明の場合は -1。
PERCENT_ARGBYTES	SMALLINT		Y	ルーチンが実際に読み取る入力引数バイトの平均パーセント値の見積もり。デフォルト値は 100。不明の場合は -1。
INITIAL_IOS	DOUBLE		Y	最初のルーチン呼び出し時に実行される入出力回数の見積もり。0 がデフォルト。不明の場合は -1。
INITIAL_INSTS	DOUBLE		Y	最初のルーチン呼び出し時に実行される命令の数の見積もり。0 がデフォルト。不明の場合は -1。
CARDINALITY	BIGINT		Y	表関数の予測されるカーディナリティー。不明の場合、またはルーチンが表関数でない場合は -1。
SELECTIVITY	DOUBLE		Y	ユーザー定義述部用。ユーザー定義述部がない場合は -1。

SYSSTAT.TABLES

各行は、表、ビュー、別名、またはニックネームを表します。表またはビューの各階層にはそれぞれ追加の行が 1 行あります。この行は階層をインプリメントする階層表または階層ビューを表しています。カタログ表およびカタログ・ビューが含まれています。

表 183. SYSSTAT.TABLES カタログ・ビュー

列名	データ・タイプ	NULL 可能	更新可能	説明
TABSCHEMA	VARCHAR(128)			オブジェクトのスキーマ名。
TABNAME	VARCHAR(128)			オブジェクトの非修飾名。
CARD	BIGINT		Y	行の総数。統計が収集されていない場合は -1。
NPAGES	BIGINT		Y	表の行が存在しているページの総数。ビューまたは別名の場合、または統計が収集されていない場合は -1。副表および階層表の場合は -2。
FPAGES	BIGINT		Y	ページの総数。ビューまたは別名の場合、または統計が収集されていない場合は -1。副表および階層表の場合は -2。
OVERFLOW	BIGINT		Y	表のオーバーフロー・レコードの総数。ビューまたは別名の場合、または統計が収集されていない場合は -1。副表および階層表の場合は -2。
CLUSTERED	CHAR(1)	Y		• Y = テーブルはマルチディメンションに分けてクラスター化される (1 つしかディメンションがない場合でも) • NULL 値 = テーブルはマルチディメンションに分けてクラスター化されない。
ACTIVE_BLOCKS	BIGINT		Y	表の中のアクティブ・ブロックの総数、または -1。マルチディメンション・クラスタリング表 (MDC) のみに適用されます。
AVGCOMPRESSEDROWSIZE	SMALLINT		Y	この表の中の圧縮された行の長さの平均 (バイト単位)。統計が収集されていない場合は -1。
AVGROWCOMPRESSIONRATIO	REAL		Y	表の中の圧縮された行の場合、これは行の平均の圧縮率です。つまり、圧縮されていない行の平均の長さを、圧縮された行の平均の長さで除算したものです。統計が収集されていない場合は、-1。
AVGROWSIZE	SMALLINT			この表の中の圧縮された行と圧縮されていない行の両方の長さの平均 (バイト単位)。統計が収集されていない場合は -1。
PCTROWSCOMPRESSED	REAL		Y	表の中の行の総数に対する圧縮された行のパーセンテージ。統計が収集されていない場合は -1。
PCTPAGESSAVED	SMALLINT		Y	行の圧縮の結果、表に保管されるページの概算パーセンテージ。この値は表内の各ユーザー・データ行のオーバーヘッド・バイトを含んでいますが、ディクショナリー・オーバーヘッドによって消費されるスペースは含みません。統計が収集されない場合は -1 です。

付録 E. フェデレーテッド・システム

© Copyright IBM Corp. 1993, 2010

SQL ステートメントで有効なサーバーのタイプ

サーバー・タイプは、サーバー定義が表すデータ・ソースの種類を示します。

サーバー・タイプは、ベンダー、目的、およびオペレーティング・システムに応じて変わります。サポートされる値はデータ・ソースによって異なります。

ほとんどのデータ・ソースで、CREATE SERVER ステートメントに有効なサーバー・タイプを指定しなければなりません。

表 184. データ・ソースおよびサーバー・タイプ

データ・ソース	サーバー・タイプ
BioRS	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
Blast	BLASTN、BLASTP、BLASTX、TBLASTN、TBLASTX
Entrez	NUCLEOTIDE、OMIM、PUBMED
Excel	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
HMMER	PFAM、SEARCH
IBM DB2 Universal Database for Linux, UNIX, and Windows	DB2/UDB
IBM DB2 Universal Database for System i and AS/400®	DB2/ISERIES
IBM DB2 Universal Database for z/OS	DB2/ZOS
IBM DB2 for VM	DB2/VM
Informix	INFORMIX
Microsoft SQL Server	MSSQLSERVER (DataDirect Connect ODBC 4.2 以降のドライバー、または Microsoft SQL Server ODBC 3.0 以降のドライバーがサポートするデータ・ソースでは必須。)
ODBC	ODBC (ODBC 3.x ドライバーがサポートする ODBC データ・ソースでは必須。)
OLE DB	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
Oracle	ORACLE (Oracle NET8 クライアント・ソフトウェアがサポートする Oracle データ・ソースでは必須。)
Sybase (CTLIB)	SYBASE
表構造ファイル	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
Teradata	TERADATA
Web サービス	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。
WebSphere Business Integration	WBI
XML	サーバー・タイプは CREATE SERVER ステートメントでは必要ありません。

フェデレーテッド・システムの関数マッピング・オプション

フェデレーテッド・サーバーは、DB2 関数とデータ・ソース関数の間のデフォルトのマッピングを提供します。ほとんどのデータ・ソースの場合、ラッパー内にデフォルトの関数マッピングがあります。フェデレーテッド・サーバーが認識しないデータ・ソース関数を使用したり、デフォルトのマッピングを変更したりする場合は、関数マッピングを作成する必要があります。

関数マッピングを作成する時、データ・ソース関数の名前を指定し、マップされる関数を使用可能にする必要があります。その後マップされた関数を使用する時、照会オプティマイザーは、その関数をデータ・ソースで実行する場合のコストとその関数をフェデレーテッド・サーバーで実行する場合のコストを比較します。

表 185. 関数マッピングのオプション

名前	説明
DISABLE	デフォルト関数マッピングを使用可能または使用不可にします。有効な値は Y と N です。デフォルトは N です。
REMOTE_NAME	データ・ソース関数の名前。デフォルトはローカル名です。

デフォルトの順方向データ・タイプ・マッピング

データ・ソースのデータ・タイプとフェデレーテッド・データベースのデータ・タイプの間のマッピングには、順方向タイプのマッピングと逆方向タイプのマッピングの 2 種類があります。順方向タイプ・マッピング では、リモートのタイプから対応するローカル・タイプへのマッピングとなります。

デフォルトのタイプ・マッピングをオーバーライドすることも、 CREATE TYPE MAPPING ステートメントを使用して新しいタイプ・マッピングを作成することもできます。

これらのマッピングは、特に記述のないかぎり、サポート対象のすべてのバージョンで有効です。

データ・ソースからフェデレーテッド・データベースへのデフォルトの順方向データ・タイプ・マッピングのすべてについて、フェデレーテッド・スキーマは SYSIBM です。

次の表は、フェデレーテッド・データベースのデータ・タイプとデータ・ソースのデータ・タイプの間のデフォルトの順方向マッピングを示しています。

DB2 Database for Linux, UNIX, and Windows データ・ソース

以下の表は、DB2 Database for Linux, UNIX, and Windows データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 186. DB2 Database for Linux, UNIX, and Windows のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

REMOTE_TYPE NAME	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_BIT_DATA	REMOTE_DATA_OPERATORS	FEDERATED_TYPE NAME	FEDERATED_LENGTH	FEDERATED_SCALE	FEDERATED_BIT_DATA
BIGINT	-	-	-	-	-	-	BIGINT	-	0	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	-	-	-	-	-	-	CHAR	-	0	N
CHAR	-	-	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
LONGVAR	-	-	-	-	N	-	CLOB	-	-	-
LONGVAR	-	-	-	-	Y	-	BLOB	-	-	-
LONGVARG	-	-	-	-	-	-	DBCLOB	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	0	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for System i データ・ソース

以下の表は、DB2 for System i データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 187. DB2 for System i のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
NUMERIC	-	-	-	-	-	-	DECIMAL	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for VM and VSE データ・ソース

以下の表は、DB2 for VM and VSE データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 188. DB2 Server for VM and VSE のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBAHW	-	-	-	-	-	-	SMALLINT	-	0	-
DBAINT	-	-	-	-	-	-	INTEGER	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPH	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

DB2 for z/OS データ・ソース

以下の表は、DB2 for z/OS データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 189. DB2 for z/OS のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHAR	1	254	-	-	-	-	CHAR	-	0	N
CHAR	255	32672	-	-	-	-	VARCHAR	-	0	N
CHAR	1	254	-	-	Y	-	CHAR	-	0	Y
CHAR	255	32672	-	-	Y	-	VARCHAR	-	0	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	0	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
FLOAT	4	-	-	-	-	-	REAL	-	-	-
FLOAT	8	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	0	N
INTEGER	-	-	-	-	-	-	INTEGER	-	0	-
ROWID	-	-	-	-	Y	-	VARCHAR	40	-	Y
SMALLINT	-	-	-	-	-	-	SMALLINT	-	0	-
TIME	-	-	-	-	-	-	TIME	-	0	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
TIMESTMP	-	-	-	-	-	-	TIMESTAMP	-	0	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	0	N
VARCHAR	1	32672	-	-	Y	-	VARCHAR	-	0	Y
VARG	1	16336	-	-	-	-	VARGRAPHIC	-	0	N
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	0	N

Informix データ・ソース

以下の表は、Informix データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 190. Informix のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	-	-	-	-	-	-	BLOB	2147483647	-	-
BOOLEAN	-	-	-	-	-	-	CHARACTER	1	-	-
BYTE	-	-	-	-	-	-	BLOB	2147483647	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CLOB	-	-	-	-	-	-	CLOB	2147483647	-	-
DATE	-	-	-	-	-	-	DATE	4	-	-
DATETIME	0	4	0	4	-	-	DATE	4	-	-
DATETIME	6	10	6	10	-	-	TIME	3	-	-
DATETIME	0	4	6	15	-	-	TIMESTAMP	10	-	-
DATETIME	6	10	11	15	-	-	TIMESTAMP	10	-	-
DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
DECIMAL	32	130	-	-	-	-	DOUBLE	8	-	-
DECIMAL	1	32	255	255	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
INTERVAL	-	-	-	-	-	-	VARCHAR	25	-	-
INT8	-	-	-	-	-	-	BIGINT	19	0	-
LVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
MONEY	1	31	0	31	-	-	DECIMAL	-	-	-
MONEY	32	32	-	-	-	-	DOUBLE	8	-	-
NCHAR	1	254	-	-	-	-	CHARACTER	-	-	-
NCHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
NVARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
REAL	-	-	-	-	-	-	REAL	4	-	-
SERIAL	-	-	-	-	-	-	INTEGER	4	-	-
SERIAL8	-	-	-	-	-	-	BIGINT	-	-	-
SMALLFLOAT	-	-	-	-	-	-	REAL	4	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
TEXT	-	-	-	-	-	-	CLOB	2147483647	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-

注:

- Informix DATETIME データ・タイプでは、DB2 UNIX and Windows フェデレーテッド・サーバーは、Informix 高水準修飾子を REMOTE_LENGTH として、Informix 低水準修飾子を REMOTE_SCALE として使用します。

  Informix 修飾子は、Informix Client SDK `datatime.h` ファイルに定義されている "TU_" 定数です。定数は次のとおりです。

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)

表 190. Informix のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります) (続き)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
6 = HOUR		12 = FRACTION(2)								

Microsoft SQL Server データ・ソース

以下の表は、Microsoft SQL Server データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 191. Microsoft SQL のデフォルトの順方向データ・タイプ・マッピング

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
bigint [2]	-	-	-	-	-	-	BIGINT	-	-	-
バイナリー	1	254	-	-	-	-	CHARACTER	-	-	Y
バイナリー	255	8000	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	2	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	8000	-	-	-	-	VARCHAR	-	-	N
datetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
10 進	1	31	0	31	-	-	DECIMAL	-	-	-
10 進	32	38	0	38	-	-	DOUBLE	-	-	-
浮動	-	8	-	-	-	-	DOUBLE	8	-	-
浮動	-	4	-	-	-	-	REAL	4	-	-
image	-	-	-	-	-	-	BLOB	2147483647	-	Y
int	-	-	-	-	-	-	INTEGER	4	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	127	-	-	-	-	CHAR	-	-	N
nchar	128	4000	-	-	-	-	VARCHAR	-	-	N
数値	1	31	0	31	-	-	DECIMAL	-	-	-
数値	32	38	0	38	-	-	DOUBLE	8	-	-
ntext	-	-	-	-	-	-	CLOB	2147483647	-	Y
nvarchar	1	4000	-	-	-	-	VARCHAR	-	-	N
実数	-	-	-	-	-	-	REAL	4	-	-
smallint	-	-	-	-	-	-	SMALLINT	2	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	10	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
SQL_BIGINT	-	-	-	-	-	-	BIGINT	-	-	-
SQL_BINARY	1	254	-	-	-	-	CHARACTER	-	-	Y
SQL_BINARY	255	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	8000	-	-	-	-	VARCHAR	-	-	N
SQL_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_GUID	-	-	-	-	-	-	VARCHAR	-	-	Y
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N

表 191. Microsoft SQL のデフォルトの順方向データ・タイプ・マッピング (続き)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	-	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	8	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	8000	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	8000	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	254	-	-	-	-	CHARACTER	-	-	N
SQL_WCHAR	255	8800	-	-	-	-	VARCHAR	-	-	N
SQL_ WLONGVARCHAR	-	1073741823		-	-	-	CLOB	2147483647	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
テキスト	-	-	-	-	-	-	CLOB	-	-	N
タイム・スタンプ	-	-	-	-	-	-	VARCHAR	8		Y
tinyint	-	-	-	-	-	-	SMALLINT	2	-	-
uniqueidentifier	1	4000	-	-	Y	-	VARCHAR	16	-	Y
varbinary	1	8000	-	-	-	-	VARCHAR	-	-	Y
varchar	1	8000	-	-	-	-	VARCHAR	-	-	N

注:

1. このタイプ・マッピングは Microsoft SQL Server バージョン 2000 でのみ有効です。

ODBC データ・ソース

以下の表は、ODBC データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 192. ODBC のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
SQL_BIGINT	-	-	-	-	-	-	BIGINT	8	-	-
SQL_BINARY	1	254	-	-	-	-	CHARAC-TER	-	-	Y
SQL_BINARY	255	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_BIT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_CHAR	1	254	-	-	-	-	CHAR	-	-	N
SQL_CHAR	255	32672	-	-	-	-	VARCHAR	-	-	N
SQL_DECIMAL	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_DECIMAL	32	38	0	38	-	-	DOUBLE	8	-	-
SQL_DOUBLE	-	-	-	-	-	-	DOUBLE	8	-	-
SQL_FLOAT	-	8	-	-	-	-	FLOAT	8	-	-
SQL_FLOAT	-	4	-	-	-	-	FLOAT	4	-	-
SQL_INTEGER	-	-	-	-	-	-	INTEGER	4	-	-
SQL_ LONGVARCHAR	-	-	-	-	-	-	CLOB	2147483647	-	N
SQL_ LONGVARBINARY	-	-	-	-	-	-	BLOB	2147483647	-	Y
SQL_NUMERIC	1	31	0	31	-	-	DECIMAL	-	-	-
SQL_NUMERIC	32	32	0	31	-	-	DOUBLE	8	-	-
SQL_REAL	-	-	-	-	-	-	REAL	4	-	-
SQL_SMALLINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_TYPE_DATE	-	-	-	-	-	-	DATE	4	-	-
SQL_TYPE_TIME	-	-	-	-	-	-	TIME	3	-	-
SQL_TYPE_ TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	10	-	-
SQL_TINYINT	-	-	-	-	-	-	SMALLINT	2	-	-
SQL_VARBINARY	1	32672	-	-	-	-	VARCHAR	-	-	Y
SQL_VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	N
SQL_WCHAR	1	127	-	-	-	-	CHAR	-	-	N
SQL_WCHAR	128	16336	-	-	-	-	VARCHAR	-	-	N
SQL_WVARCHAR	1	16336	-	-	-	-	VARCHAR	-	-	N
SQL_ WLONGVARCHAR	-	1073741823	-	-	-	-	CLOB	2147483647	-	N

Oracle NET8 データ・ソース

以下の表は、Oracle NET8 データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 193. Oracle NET8 のデフォルトの順方向データ・タイプ・マッピング

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BLOB	0	0	0	0	-	¥0	BLOB	2147483647	0	Y
CHAR	1	254	0	0	-	¥0	CHAR	0	0	N
CHAR	255	2000	0	0	-	¥0	VARCHAR	0	0	N
CLOB	0	0	0	0	-	¥0	CLOB	2147483647	0	N
DATE	0	0	0	0	-	¥0	TIMESTAMP	0	0	N
FLOAT	1	126	0	0	-	¥0	DOUBLE	0	0	N
LONG	0	0	0	0	-	¥0	CLOB	2147483647	0	N
LONG RAW	0	0	0	0	-	¥0	BLOB	2147483647	0	Y
NUMBER	10	18	0	0	-	¥0	BIGINT	0	0	N
NUMBER	1	38	-84	127	-	¥0	DOUBLE	0	0	N
NUMBER	1	31	0	31	-	>=	DECIMAL	0	0	N
NUMBER	1	4	0	0	-	¥0	SMALLINT	0	0	N
NUMBER	5	9	0	0	-	¥0	INTEGER	0	0	N
NUMBER	-	10	0	0	-	¥0	DECIMAL	0	0	N
RAW	1	2000	0	0	-	¥0	VARCHAR	0	0	Y
ROWID	0	0	0	NULL	-	\0	CHAR	18	0	N
TIMESTAMP [1]	-	-	-	-	-	-	TIMESTAMP	10	-	-
VARCHAR2	1	4000	0	0	-	¥0	VARCHAR	0	0	N

注:

1. このタイプ・マッピングは、 Oracle 9i (またはそれ以降) のクライアントおよびサーバー構成でのみ有効です。

Sybase データ・ソース

以下の表は、Sybase データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 194. Sybase CTLIB のデフォルトの順方向データ・タイプ・マッピング

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
binary	1	254	-	-	-	-	CHAR	-	-	Y
binary	255	32672	-	-	-	-	VARCHAR	-	-	Y
bit	-	-	-	-	-	-	SMALLINT	-	-	-
char	1	254	-	-	-	-	CHAR	-	-	N
char	255	32672	-	-	-	-	VARCHAR	-	-	N
char null (varchar を参照)										
datetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
datetimn	-	-	-	-	-	-	TIMESTAMP	-	-	-
10 進	1	31	0	31	-	-	DECIMAL	-	-	-
10 進	32	38	0	38	-	-	DOUBLE	-	-	-
decimaln	1	31	0	31	-	-	DECIMAL	-	-	-
decimaln	32	38	0	38	-	-	DOUBLE	-	-	-
浮動	-	4	-	-	-	-	REAL	-	-	-
浮動	-	8	-	-	-	-	DOUBLE	-	-	-
floatn	-	4	-	-	-	-	REAL	-	-	-
floatn	-	8	-	-	-	-	DOUBLE	-	-	-
image	-	-	-	-	-	-	BLOB	-	-	-
int	-	-	-	-	-	-	INTEGER	-	-	-
intn	-	-	-	-	-	-	INTEGER	-	-	-
money	-	-	-	-	-	-	DECIMAL	19	4	-
moneyn	-	-	-	-	-	-	DECIMAL	19	4	-
nchar	1	254	-	-	-	-	CHAR	-	-	N
nchar	255	32672	-	-	-	-	VARCHAR	-	-	N
nchar null (nvarchar を参照)										
数値	1	31	0	31	-	-	DECIMAL	-	-	-
数値	32	38	0	38	-	-	DOUBLE	-	-	-
numericn	1	31	0	31	-	-	DECIMAL	-	-	-
numericn	32	38	0	38	-	-	DOUBLE	-	-	-
nvarchar	1	32672	-	-	-	-	VARCHAR	-	-	N
実数	-	-	-	-	-	-	REAL	-	-	-
smalldatetime	-	-	-	-	-	-	TIMESTAMP	-	-	-
smallint	-	-	-	-	-	-	SMALLINT	-	-	-
smallmoney	-	-	-	-	-	-	DECIMAL	10	4	-
sysname	-	-	-	-	-	-	VARCHAR	30	-	N
テキスト	-	-	-	-	-	-	CLOB	-	-	-

表 194. Sybase CTLIB のデフォルトの順方向データ・タイプ・マッピング (続き)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
タイム・スタンプ	-	-	-	-	-	-	VARCHAR	8	-	Y
tinyint	-	-	-	-	-	-	SMALLINT	-	-	-
unichar[1]	1	254	-	-	-	-	CHAR	-	-	N
unichar[1]	255	32672	-	-	-	-	VARCHAR	-	-	N
unichar null (univarchar を参照)										
univarchar[1]	1	32672	-	-	-	-	VARCHAR	-	-	N
varbinary	1	32672	-	-	-	-	VARCHAR	-	-	Y
varchar	1	32672	-	-	-	-	VARCHAR	-	-	N

注:

1. 非 Unicode のフェデレーテッド・データベースで有効。

Teradata データ・ソース

以下の表は、Teradata データ・ソースのデフォルトの順方向データ・タイプ・マッピングをリストしています。

表 195. Teradata のデフォルトの順方向データ・タイプ・マッピング (表示されていない列があります)

Remote Typename	Remote Lower Len	Remote Upper Len	Remote Lower Scale	Remote Upper Scale	Remote Bit Data	Remote Data Operators	Federated Typename	Federated Length	Federated Scale	Federated Bit Data
BYTE	1	254	-	-	-	-	CHAR	-	-	Y
BYTE	255	32672	-	-	-	-	VARCHAR	-	-	Y
BYTE	32673	64000	-	-	-	-	BLOB	-	-	-
BYTEINT	-	-	-	-	-	-	SMALLINT	-	-	-
CHAR	1	254	-	-	-	-	CHARACTER	-	-	-
CHAR	255	32672	-	-	-	-	VARCHAR	-	-	-
CHAR	32673	64000	-	-	-	-	CLOB	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DOUBLE PRECISION	-	-	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	-	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	1	127	-	-	-	-	GRAPHIC	-	-	-
GRAPHIC	128	16336	-	-	-	-	VARGRAPHIC	-	-	-
GRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
INTERVAL	-	-	-	-	-	-	CHAR	-	-	-
NUMERIC	1	18	0	18	-	-	DECIMAL	-	-	-
REAL	-	-	-	-	-	-	DOUBLE	-	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	0	21	0	21	-	-	TIME	-	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	-	-	-
VARBYTE	1	32762	-	-	-	-	VARCHAR	-	-	Y
VARBYTE	32763	64000	-	-	-	-	BLOB	-	-	-
VARCHAR	1	32672	-	-	-	-	VARCHAR	-	-	-
VARCHAR	32673	64000	-	-	-	-	CLOB	-	-	-
VARGRAPHIC	1	16336	-	-	-	-	VARGRAPHIC	-	-	-
VARGRAPHIC	16337	32000	-	-	-	-	DBCLOB	-	-	-

デフォルトの逆方向データ・タイプ・マッピング

ほとんどのデータ・ソースの場合、ラッパー内にデフォルトのタイプ・マッピングがあります。

データ・ソースのデータ・タイプとフェデレーテッド・データベースのデータ・タイプの間のマッピングには、順方向タイプのマッピングと逆方向タイプのマッピングの 2 種類があります。順方向タイプ・マッピング では、リモートのタイプから対応するローカル・タイプへのマッピングとなります。マッピングのもう一方のタイプは逆方向タイプ・マッピング であり、これはリモートの表を作成または変更するために、透過 DDL で使用されます。

DB2 ファミリーのデータ・ソース用のデフォルトのタイプ・マッピングは、 DRDA ラッパーにあります。 Informix 用のデフォルトのタイプ・マッピングは INFORMIX ラッパーにあります。その他のタイプ・マッピングについても同様です。

フェデレーテッド・データベースにリモートの表またはビューを定義する際、その定義には、逆方向タイプ・マッピングが含まれます。このマッピングには、それぞれの列のローカル のフェデレーテッド・データベースのデータ・タイプからのもの、および対応するリモート・データ・タイプのものがあります。例えば、ローカル・タイプ REAL が Informix タイプ SMALLFLOAT を指す、デフォルトの逆方向タイプ・マッピングがあります。

フェデレーテッド・データベースは、LONG VARCHAR、LONG VARGRAPHIC、およびユーザー定義タイプのマッピングをサポートしません。

CREATE TABLE ステートメントを使用してリモート表を作成する時に、リモート表に含めたいローカル・データ・タイプを指定します。これらのデフォルトの逆方向タイプ・マッピングは、これらの列に対応するリモート・タイプを割り当てます。例えば、CREATE TABLE ステートメントを使用して、列 C2 を持つ Informix の表を定義するとします。ステートメント内で C2 のデータ・タイプとして BIGINT を指定します。 BIGINT のデフォルトの逆方向タイプ・マッピングは、どのバージョンの Informix で表を作成しているかにより異なります。 Informix バージョン 8 では Informix 表の C2 のマッピングは DECIMAL となり、 Informix バージョン 9 では INT8 になります。

デフォルトの逆方向タイプ・マッピングをオーバーライドすることも、 CREATE TYPE MAPPING ステートメントを使用して新しい逆方向タイプ・マッピングを作成することもできます。

次の表は、フェデレーテッド・データベースのローカル・データ・タイプとリモート・データ・ソースのデータ・タイプの間のデフォルトの逆方向マッピングを示しています。

これらのマッピングは、特に記述のないかぎり、サポート対象のすべてのバージョンで有効です。

DB2 Database for Linux, UNIX, and Windows データ・ソース

以下の表は、DB2 Database for Linux, UNIX, and Windows データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 196. DB2 Database for Linux, UNIX, and Windows のデフォルトの逆方向データ・タイプ・マッピング (表示されていない列があります)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Federated Bit Data
BIGINT	-	8	-	-	-	-	BIGINT	-	-	-
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	-
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPHIC	-	-	N
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

DB2 for System i データ・ソース

以下の表は、DB2 for System i データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 197. DB2 for System i のデフォルトの逆方向データ・タイプ・マッピング (表示されていない列があります)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operations	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	N
CHARACTER	-	-	-	-	Y	-	CHARACTER	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	NUMERIC	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	FLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARG	-	-	N

DB2 for VM and VSE データ・ソース

以下の表は、DB2 for VM and VSE データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 198. DB2 for VM and VSE のデフォルトの逆方向データ・タイプ・マッピング (表示されていない列があります)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPH	-	-	-	-	-	-	VARGRAPH	-	-	N

DB2 for z/OS データ・ソース

以下の表は、DB2 for z/OS データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 199. DB2 for z/OS のデフォルトの逆方向データ・タイプ・マッピング (表示されていない列があります)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB	-	-	-	-	-	-	BLOB	-	-	-
CHARACTER	-	-	-	-	-	-	CHAR	-	-	N
CHARACTER	-	-	-	-	Y	-	CHAR	-	-	Y
CLOB	-	-	-	-	-	-	CLOB	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DBCLOB	-	-	-	-	-	-	DBCLOB	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	DOUBLE	-	-	–
FLOAT	-	8	-	-	-	-	DOUBLE	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	N
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	REAL	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	TIME	-	-	-
TIMESTAMP	-	10	-	-	-	-	TIMESTAMP	-	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	N
VARCHAR	-	-	-	-	Y	-	VARCHAR	-	-	Y
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	N

Informix データ・ソース

以下の表は、Informix データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 200. Informix のデフォルトの逆方向データ・タイプ・マッピング

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT [1]	-	-	-	-	-	-	DECIMAL	19	-	-
BIGINT [2]	-	-	-	-	-	-	INT8	-	-	-
BLOB	1	2147483647	-	-	-	-	BYTE	-	-	-
CHARACTER	-	-	-	-	N	-	CHAR	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB	1	2147483647	-	-	-	-	TEXT	-	-	-
DATE	-	4	-	-	-	-	DATE	-	-	-
DECIMAL	-	-	-	-	-	-	DECIMAL	-	-	-
DOUBLE	-	8	-	-	-	-	FLOAT	-	-	-
INTEGER	-	4	-	-	-	-	INTEGER	-	-	-
REAL	-	4	-	-	-	-	SMALLFLOAT	-	-	-
SMALLINT	-	2	-	-	-	-	SMALLINT	-	-	-
TIME	-	3	-	-	-	-	DATETIME	6	10	-
TIMESTAMP	-	10	-	-	-	-	DATETIME	0	15	-
VARCHAR	1	254	-	-	N	-	VARCHAR	-	-	-
VARCHAR[1]	255	32672	-	-	N	-	TEXT	-	-	-
VARCHAR	-	-	-	-	Y	-	BYTE	-	-	-
VARCHAR [2]	255	2048	-	-	N	-	LVARCHAR	-	-	-
VARCHAR [2]	2049	32672	-	-	N	-	TEXT	-	-	-

注:

1. このタイプ・マッピングは Informix サーバーのバージョン 8 以下でのみ有効。
2. このタイプ・マッピングは Informix サーバーのバージョン 9 以上でのみ有効。

Informix DATETIME データ・タイプでは、フェデレーテッド・サーバーは、Informix 高水準修飾子を REMOTE_LENGTH として、Informix 低水準修飾子を REMOTE_SCALE として使用します。

Informix 修飾子は、Informix Client SDK `datatime.h` ファイルに定義されている "TU_" 定数です。定数は次のとおりです。

0 = YEAR	8 = MINUTE	13 = FRACTION(3)
2 = MONTH	10 = SECOND	14 = FRACTION(4)
4 = DAY	11 = FRACTION(1)	15 = FRACTION(5)
6 = HOUR	12 = FRACTION(2)	

Microsoft SQL Server データ・ソース

以下の表は、Microsoft SQL Server データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 201. Microsoft SQL Server のデフォルトの逆方向データ・タイプ・マッピング (表示されていない列があります)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT [1]	-	-	-	-	-	-	bigint	-	-	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	4	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	8	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	int	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
REAL	-	4	-	-	-	-	real	-	-	-
TIME	-	3	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	10	-	-	-	-	datetime	-	-	-
VARCHAR	1	8000	-	-	N	-	varchar	-	-	-
VARCHAR	8001	32672	-	-	N	-	text	-	-	-
VARCHAR	1	8000	-	-	Y	-	varbinary	-	-	-
VARCHAR	8001	32672	-	-	Y	-	image	-	-	-

注:

1. このタイプ・マッピングは Microsoft SQL Server バージョン 2000 でのみ有効です。

Oracle NET8 データ・ソース

以下の表は、 Oracle NET8 データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 202. Oracle NET8 のデフォルトの逆方向データ・タイプ・マッピング

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT	0	8	0	0	N	¥0	NUMBER	19	0	N
BLOB	0	2147483647	0	0	Y	¥0	BLOB	0	0	Y
CHARACTER	1	254	0	0	N	¥0	CHAR	0	0	N
CHARACTER	1	254	0	0	Y	¥0	RAW	0	0	Y
CLOB	0	2147483647	0	0	N	¥0	CLOB	0	0	N
DATE	0	4	0	0	N	¥0	DATE	0	0	N
DECIMAL	0	0	0	0	N	¥0	NUMBER	0	0	N
DOUBLE	0	8	0	0	N	¥0	FLOAT	126	0	N
FLOAT	0	8	0	0	N	¥0	FLOAT	126	0	N
INTEGER	0	4	0	0	N	¥0	NUMBER	10	0	N
REAL	0	4	0	0	N	¥0	FLOAT	63	0	N
SMALLINT	0	2	0	0	N	¥0	NUMBER	5	0	N
TIME	0	3	0	0	N	¥0	DATE	0	0	N
TIMESTAMP [1]	0	10	0	0	N	¥0	DATE	0	0	N
TIMESTAMP [2]	0	10	0	0	N	¥0	TIMESTAMP	6	0	N
VARCHAR	1	4000	0	0	N	¥0	VARCHAR2	0	0	N
VARCHAR	1	2000	0	0	Y	¥0	RAW	0	0	Y

注:

1. このタイプ・マッピングは Oracle バージョン 8 でのみ有効です。
2. このタイプ・マッピングは Oracle バージョン 9 およびバージョン 10 でのみ有効です。

Sybase データ・ソース

以下の表は、Sybase データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 203. Sybase CTLIB のデフォルトの逆方向データ・タイプ・マッピング

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BIGINT	-	-	-	-	-	-	decimal	19	0	-
BLOB	-	-	-	-	-	-	image	-	-	-
CHARACTER	-	-	-	-	N	-	char	-	-	-
CHARACTER	-	-	-	-	Y	-	binary	-	-	-
CLOB	-	-	-	-	-	-	text	-	-	-
DATE	-	-	-	-	-	-	datetime	-	-	-
DECIMAL	-	-	-	-	-	-	decimal	-	-	-
DOUBLE	-	-	-	-	-	-	float	-	-	-
INTEGER	-	-	-	-	-	-	integer	-	-	-
REAL	-	-	-	-	-	-	real	-	-	-
SMALLINT	-	-	-	-	-	-	smallint	-	-	-
TIME	-	-	-	-	-	-	datetime	-	-	-
TIMESTAMP	-	-	-	-	-	-	datetime	-	-	-
VARCHAR[1]	1	255	-	-	N	-	varchar	-	-	-
VARCHAR[1]	256	32672	-	-	N	-	text	-	-	-
VARCHAR [2]	1	16384	-	-	N	-	varchar	-	-	-
VARCHAR [2]	16385	32672	-	-	N	-	text	-	-	-
VARCHAR[1]	1	255	-	-	Y	-	varbinary	-	-	-
VARCHAR[1]	256	32672	-	-	Y	-	image	-	-	-
VARCHAR [2]	1	16384	-	-	Y	-	varbinary	-	-	-
VARCHAR [2]	16385	32672	-	-	Y	-	image	-	-	-

注:

1. このタイプ・マッピングは、Sybase サーバーのバージョン 12.0 またはそれ以前で CTLIB を使用する場合にのみ有効です。
2. このタイプ・マッピングは、Sybase サーバーのバージョン 12.5 またはそれ以降で CTLIB を使用する場合にのみ有効です。

Teradata データ・ソース

以下の表は、Teradata データ・ソースのデフォルトの逆方向データ・タイプ・マッピングをリストしています。

表 204. Teradata のデフォルトの逆方向データ・タイプ・マッピング (表示されていない列があります)

Federated Typename	Federated Lower Len	Federated Upper Len	Federated Lower Scale	Federated Upper Scale	Federated Bit Data	Federated Data Operators	Remote Typename	Remote Length	Remote Scale	Remote Bit Data
BLOB [1]	1	64000	-	-	-	-	VARBYTE	-	-	-
CHARACTER	-	-	-	-	-	-	CHARACTER	-	-	-
CHARACTER	-	-	-	-	Y	-	BYTE	-	-	-
CLOB [2]	1	64000	-	-		-	VARCHAR	-	-	-
DATE	-	-	-	-	-	-	DATE	-	-	-
DBCLOB [3]	1	64000	-	-	-	-	VARGRAPHIC	-	-	-
DECIMAL	1	18	0	18	-	-	DECIMAL	-	-	-
DECIMAL	19	31	0	31	-	-	FLOAT	8	-	-
DOUBLE	-	-	-	-	-	-	FLOAT	-	-	-
GRAPHIC	-	-	-	-	-	-	GRAPHIC	-	-	-
INTEGER	-	-	-	-	-	-	INTEGER	-	-	-
REAL	-	-	-	-	-	-	FLOAT	8	-	-
SMALLINT	-	-	-	-	-	-	SMALLINT	-	-	-
TIME	-	-	-	-	-	-	TIME	15	-	-
TIMESTAMP	-	-	-	-	-	-	TIMESTAMP	26	-	-
VARCHAR	-	-	-	-	-	-	VARCHAR	-	-	-
VARCHAR	-	-	-	-	Y	-	VARBYTE	-	-	-
VARGRAPHIC	-	-	-	-	-	-	VARGRAPHIC	-	-	-

注:

1. Teradata VARBYTE データ・タイプには、指定した長さ (1 から 64000) の BLOB データ・タイプのみ含めることができます。
2. Teradata VARCHAR データ・タイプには、指定した長さ (1 から 64000) の CLOB データ・タイプのみ含めることができます。
3. Teradata VARGRAPHIC データ・タイプには、指定した長さ (1 から 32000) の DBCLOB データ・タイプのみ含めることができます。

付録 F. SAMPLE データベース

サンプル・データベースはさまざまな目的で使用できます。例えば、アプリケーションのテスト、DB2 の種々のフィーチャーの試用などです。DB2PATH/sqllib/samples の下にあるサンプル・アプリケーション・プログラムのほとんどは、様々な DB2 フィーチャーのそのテクノロジーを容易に理解できるように例示する目的でサンプル・データベースを使用します。

サンプル・データベースの作成後、次のスキーマが作成されます。

- 非 XML データの組織に関するスキーマ
- XML データの注文書スキーマ。

こうしたスキーマのデータおよびデータベース・オブジェクトは、スケールの小さなリアルタイム環境を使用して作成されます。

サンプル・データベース内の各表の説明を以下に示します。各表の初期データ値が提供されています。ダッシュ (-) は NULL 値を示します。

ACT 表

名前:	ACTNO	ACTKWD	ACTDESC
タイプ:	SMALLINT	CHAR(6)	VARCHAR(20)
値:	10	MANAGE	MANAGE/ADVISE
	20	ECOST	ESTIMATE COST
	30	DEFINE	DEFINE SPECS
	40	LEADPR	LEAD PROGRAM/DESIGN
	50	SPECS	WRITE SPECS
	60	LOGIC	DESCRIBE LOGIC
	70	CODE	CODE PROGRAMS
	80	TEST	TEST PROGRAMS
	90	ADMQS	ADM QUERY SYSTEM
	100	TEACH	TEACH CLASSES
	110	COURSE	DEVELOP COURSES
	120	STAFF	PERS AND STAFFING
	130	OPERAT	OPER COMPUTER SYS
	140	MAINT	MAINT SOFTWARE SYS
	150	ADMSYS	ADM OPERATING SYS
	160	ADMDB	ADM DATA BASES
	170	ADMDC	ADM DATA COMM
	180	DOC	DOCUMENT

© Copyright IBM Corp. 1993, 2010

ADEFUSR 表

名前:	WORKDEPT	NO_OF_EMPLOYEES
タイプ:	CHAR(3)	INTEGER
値:	A00	5
	B01	1
	C01	4
	D11	11
	D21	7
	E01	1
	E11	7
	E21	6

CL_SCHED 表

名前:	CLASS_CODE	DAY	STARTING	ENDING
タイプ:	CHAR(7)	SMALLINT	TIME	TIME
説明:	クラス・コード (部屋: 教師)	4 日間の日程のうちの日数	クラス開始時刻	クラス最終時刻
値:	042:BF	4	12:10:00	14:00:00
	553:MJA	1	10:30:00	11:00:00
	543:CWM	3	09:10:00	10:30:00
	778:RES	2	12:10:00	14:00:00
	044:HD	3	17:12:30	18:00:00

DEPT 表

名前:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
タイプ:	CHAR(3)	VARCHAR(36)	CHAR(6)	CHAR(3)	CHAR(16)
値:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	
	B01	PLANNING	000020	A00	
	C01	INFORMATION CENTER	000030	A00	
	D01	DEVELOPMENT CENTER		A00	
	D11	MANUFACTURING SYSTEMS	000060	D01	
	D21	ADMINISTRATION SYSTEMS	000070	D01	
	E01	SUPPORT SERVICES	000050	A00	
	E11	OPERATIONS	000090	E01	
	E21	SOFTWARE SUPPORT	000100	E01	
	F22	BRANCH OFFICE F2		E01	
	G22	BRANCH OFFICE G2		E01	
	H22	BRANCH OFFICE H2		E01	
	I22	BRANCH OFFICE I2		E01	
	J22	BRANCH OFFICE J2		E01	

部門 (DEPARTMENT) 表

名前:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
タイプ:	CHAR(3) NOT NULL	VARCHAR(29) NOT NULL	CHAR(6)	CHAR(3) NOT NULL	CHAR(16)
説明:	部署番号	部門の一般的なアクティビティーを説明する名前	部門管理者の従業員番号 (EMPNO)	この部門の報告先の部門 (DEPTNO)	リモート・ロケーションの名前
値:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	
	B01	PLANNING	000020	A00	
	C01	INFORMATION CENTER	000030	A00	
	D01	DEVELOPMENT CENTER		A00	
	D11	MANUFACTURING SYSTEMS	000060	D01	
	D21	ADMINISTRATION SYSTEMS	000070	D01	
	E01	SUPPORT SERVICES	000050	A00	
	E11	OPERATIONS	000090	E01	
	E21	SOFTWARE SUPPORT	000100	E01	
	F22	BRANCH OFFICE F2		E01	
	G22	BRANCH OFFICE G2		E01	
	H22	BRANCH OFFICE H2		E01	
	I22	BRANCH OFFICE I2		E01	
	J22	BRANCH OFFICE J2		E01	

EMPLOYEE および EMP 表

これら 2 つの表の内容は同一です。

名前:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
タイプ:	CHAR(6) NOT NULL	VARCHAR(12) NOT NULL	CHAR(1) NOT NULL	VARCHAR(15) NOT NULL	CHAR(3)	CHAR(4)	DATE
説明:	従業員番号	ファーストネーム	ミドルネームのイニシャル	ラストネーム	従業員が勤務する部門 (DEPTNO)	電話番号	雇用日

+

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
CHAR(8)	SMALLINT NOT NULL	CHAR(1)	DATE	DECIMAL(9,2)	DECIMAL(9,2)	DECIMAL(9,2)
職務	学校教育の年数	性別 (M 男性、F 女性)	生年月日	年収	年次賞与	年次歩合

以下の表には、EMPLOYEE 表の値が含まれています。

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
CHAR(6) NOT NULL	VARCHAR(12) NOT NULL	CHAR(1) NOT NULL	VARCHAR(15) NOT NULL	CHAR(3)	CHAR(4)	DATE	CHAR(8)	SMALLINT NOT NULL	CHAR(1)	DATE	DECIMAL (9,2)	DECIMAL (9,2)	DECIMAL (9,2)

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

EMP_ACT 表

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
タイプ:	CHAR(6) NOT NULL	CHAR(6) NOT NULL	SMALLINT NOT NULL	DEC(5,2)	DATE	DATE
説明:	従業員番号	プロジェクト番号	アクティビティー番号	従業員がプロジェクトに費やした時間の比率	アクティビティーの開始日	アクティビティーの終了日
値:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000270	AD3113	70	1.00	1982-10-15	1983-02-01
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

EMP_PHOTO 表

名前:	EMPNO	PHOTO_FORMAT	PICTURE
タイプ:	CHAR(6) NOT NULL	VARCHAR(10) NOT NULL	BLOB(100K)
説明:	従業員番号	写真フォーマット	従業員の写真
値:	000130	ビットマップ	db200130.bmp
	000130	gif	db200130.gif
	000140	ビットマップ	db200140.bmp
	000140	gif	db200140.gif
	000150	ビットマップ	db200150.bmp
	000150	gif	db200150.gif
	000190	ビットマップ	db200190.bmp
	000190	gif	db200190.gif

EMPPROJACT 表

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
タイプ:	CHAR(6)	CHAR(6)	SMALLINT	DEC(5,2)	DATE	DATE
値:	000070	AD3110	10	1.00	01/01/1982	02/01/1983
	000230	AD3111	60	1.00	01/01/1982	03/15/1982
	000230	AD3111	60	0.50	03/15/1982	04/15/1982
	000230	AD3111	70	0.50	03/15/1982	10/15/1982
	000230	AD3111	80	0.50	04/15/1982	10/15/1982
	000230	AD3111	180	0.50	10/15/1982	01/01/1983
	000240	AD3111	70	1.00	02/15/1982	09/15/1982
	000240	AD3111	80	1.00	09/15/1982	01/01/1983
	000250	AD3112	60	1.00	01/01/1982	02/01/1982
	000250	AD3112	60	0.50	02/01/1982	03/15/1982
	000250	AD3112	60	1.00	01/01/1983	02/01/1983
	000250	AD3112	70	0.50	02/01/1982	03/15/1982
	000250	AD3112	70	1.00	03/15/1982	08/15/1982
	000250	AD3112	70	0.25	08/15/1982	10/15/1982
	000250	AD3112	80	0.25	08/15/1982	10/15/1982
	000250	AD3112	80	0.50	10/15/1982	12/01/1982
	000250	AD3112	180	0.50	08/15/1982	01/01/1983
	000260	AD3113	70	0.50	06/15/1982	07/01/1982
	000260	AD3113	70	1.00	07/01/1982	02/01/1983
	000260	AD3113	80	1.00	01/01/1982	03/01/1982
	000260	AD3113	80	0.50	03/01/1982	04/15/1982
	000260	AD3113	180	0.50	03/01/1982	04/15/1982
	000260	AD3113	180	1.00	04/15/1982	06/01/1982
	000260	AD3113	180	1.00	06/01/1982	07/01/1982
	000270	AD3113	60	0.50	03/01/1982	04/01/1982
	000270	AD3113	60	1.00	04/01/1982	09/01/1982
	000270	AD3113	60	0.25	09/01/1982	10/15/1982
	000270	AD3113	70	0.75	09/01/1982	10/15/1982
	000270	AD3113	70	1.00	10/15/1982	02/01/1983
	000270	AD3113	80	1.00	01/01/1982	03/01/1982
	000270	AD3113	80	0.50	03/01/1982	04/01/1982
	000030	IF1000	10	0.50	06/01/1982	01/01/1983
	000130	IF1000	90	1.00	10/01/1982	01/01/1983
	000130	IF1000	100	0.50	10/01/1982	01/01/1983
	000140	IF1000	90	0.50	10/01/1982	01/01/1983
	000030	IF2000	10	0.50	01/01/1982	01/01/1983
	000140	IF2000	100	1.00	01/01/1982	03/01/1982
	000140	IF2000	100	0.50	03/01/1982	07/01/1982

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000140	IF2000	110	0.50	03/01/1982	07/01/1982
	000140	IF2000	110	0.50	10/01/1982	01/01/1983
	000010	MA2100	10	0.50	01/01/1982	11/01/1982
	000110	MA2100	20	1.00	01/01/1982	03/01/1983
	000010	MA2110	10	1.00	01/01/1982	02/01/1983
	000200	MA2111	50	1.00	01/01/1982	06/15/1982
	000200	MA2111	60	1.00	06/15/1982	02/01/1983
	000220	MA2111	40	1.00	01/01/1982	02/01/1983
	000150	MA2112	60	1.00	01/01/1982	07/15/1982
	000150	MA2112	180	1.00	07/15/1982	02/01/1983
	000170	MA2112	60	1.00	01/01/1982	06/01/1983
	000170	MA2112	70	1.00	06/01/1982	02/01/1983
	000190	MA2112	70	1.00	01/01/1982	10/01/1982
	000190	MA2112	80	1.00	10/01/1982	10/01/1982
	000160	MA2113	60	1.00	07/15/1982	02/01/1983
	000170	MA2113	80	1.00	01/01/1982	02/01/1983
	000180	MA2113	70	1.00	04/01/1982	06/15/1982
	000210	MA2113	80	0.50	10/01/1982	02/01/1983
	000210	MA2113	180	0.50	10/01/1982	02/01/1983
	000050	OP1000	10	0.25	01/01/1982	02/01/1983
	000090	OP1010	10	1.00	01/01/1982	02/01/1983
	000280	OP1010	130	1.00	01/01/1982	02/01/1983
	000290	OP1010	130	1.00	01/01/1982	02/01/1983
	000300	OP1010	130	1.00	01/01/1982	02/01/1983
	000310	OP1010	130	1.00	01/01/1982	02/01/1983
	000050	OP1010	10	0.75	01/01/1982	02/01/1983
	000100	OP1010	10	1.00	01/01/1982	02/01/1983
	000320	OP2011	140	0.75	01/01/1982	02/01/1983
	000320	OP2011	150	0.25	01/01/1982	02/01/1983
	000330	OP2012	140	0.25	01/01/1982	02/01/1983
	000330	OP2012	160	0.75	01/01/1982	02/01/1983
	000340	OP2013	140	0.50	01/01/1982	02/01/1983
	000340	OP2013	170	0.50	01/01/1982	02/01/1983
	000020	PL2100	30	1.00	01/01/1982	09/15/1982

EMP_RESUME 表

名前:	EMPNO	RESUME_FORMAT	RESUME
タイプ:	CHAR(6) NOT NULL	VARCHAR(10) NOT NULL	CLOB(5K)
説明:	従業員番号	履歴書形式	従業員の履歴書
値:	000130	ascii	db200130.asc

名前:	EMPNO	RESUME_FORMAT	RESUME
	000130	html	db200130.htm
	000140	ascii	db200140.asc
	000140	html	db200140.htm
	000150	ascii	db200150.asc
	000150	html	db200150.htm
	000190	ascii	db200190.asc
	000190	html	db200190.htm

IN_TRAY 表

名前:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
タイプ:	TIMESTAMP	CHAR(8)	CHAR(64)	VARCHAR(3000)
説明:	受信した日付と時間	メモの送信担当者のユーザー ID	要旨	メモ
	1988-12-25-17.12.30.000000	BADAMSON	FWD: 素晴らしい年! 第 4 四半期ボーナス。	To: JWALKER Cc: QUINTANA、NICHOLLS Jim、激務が報われそうです。お祝いに来てくれるなら、冷蔵庫にいいビールが入ってますよ。 Delores と Heather も、来ませんか? Bruce <ISTERN から転送> 件名: FWD: 素晴らしい年!第 4 四半期ボーナス。 To: Dept_D11 おめでとう。仕事、よく頑張りました。今年のボーナスを楽しみにしてください。Irv <CHAAS から転送> 件名: 素晴らしい年!第 4 四半期ボーナス。 To: All_Managers 第 4 四半期の業績がまとまりました。1 つのチームとして力を合わせた結果、計画を上回りました! 嬉しいことに、今年のボーナスは 18 % です。休日を楽しんでください。 Christine Haas

名前:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
	1988-12-23-08.53.58.000000	ISTERN	FWD: 素晴らしい年! 第 4 四半期ボーナス。	To: Dept_D11 おめでとう。仕事、よく頑張りました。今年のボーナスを楽しみにしてください。Irv <CHAAS から転送> 件名: 素晴らしい年!第 4 四半期ボーナス。 To: All_Managers 第 4 四半期の業績がまとまりました。1 つのチームとして力を合わせた結果、計画を上回りました! 嬉しいことに、今年のボーナスは 18 % です。休日を楽しんでください。 Christine Haas
	1988-12-22-14.07.21.136421	CHAAS	素晴らしい年! 第 4 四半期ボーナス。	To: All_Managers 第 4 四半期の業績がまとまりました。1 つのチームとして力を合わせた結果、計画を上回りました! 嬉しいことに、ボーナスは今年の 18 % です。休日を楽しんでください。 Christine Haas

ORG 表

名前:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
タイプ:	SMALLINT NOT NULL	VARCHAR(14)	SMALLINT	VARCHAR(10)	VARCHAR(13)
説明:	部署番号	部署名	管理者番号	会社の部門	City
値:	10	Head Office	160	Corporate	New York
	15	New England	50	Eastern	Boston
	20	Mid Atlantic	10	Eastern	Washington
	38	South Atlantic	30	Eastern	Atlanta
	42	Great Lakes	100	Midwest	Chicago
	51	Plains	140	Midwest	Dallas
	66	Pacific	270	Western	San Francisco
	84	Mountain	290	Western	Denver

PROJ 表

名前:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
タイプ:	CHAR(6)	VARCHAR(36)	CHAR(3)	CHAR(6)	DEC(5,2)	DATE	DATE	CHAR(6)
値:	AD3100	ADMIN SERVICES	D01	000010	6.50	01/01/1982	02/01/1983	
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6.00	01/01/1982	02/01/1983	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2.00	01/01/1982	02/01/1983	AD3100
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1.00	01/01/1982	02/01/1983	AD3100
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2.00	01/01/1982	02/01/1983	AD3100
	IF1000	QUERY SERVICES	C01	000030	2.00	01/01/1982	02/01/1983	
	IF2000	USER EDUCATION	C01	000030	1.00	01/01/1982	02/01/1983	
	MA2100	WELD LINE AUTOMATION	D01	000010	12.00	01/01/1982	02/01/1983	
	MA2110	W L PROGRAMMING	D11	000060	9.00	01/01/1982	02/01/1983	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2.00	01/01/1982	12/01/1982	MA2100
	MA2112	W L ROBOT DESIGN	D11	000150	3.00	01/01/1982	12/01/1982	MA2100
	MA2113	W L PROD CONT PROGS	D11	000160	3.00	02/15/1982	12/01/1982	MA2100
	OP1000	OPERATION SUPPORT	E01	000050	6.00	01/01/1982	02/01/1983	
	OP1010	OPERATION	E11	000090	5.00	01/01/1982	02/01/1983	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5.00	01/01/1982	02/01/1983	
	OP2010	SYSTEMS SUPPORT	E21	000100	4.00	01/01/1982	02/01/1983	OP2010
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1.00	01/01/1982	02/01/1983	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1.00	01/01/1982	02/01/1983	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1.00	01/01/1982	02/01/1983	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1.00	01/01/1982	09/15/1982	MA2100

PROJACT 表

名前:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
タイプ:	CHAR(6)	SMALLINT	DEC(5,2)	DATE	DATE
値:	AD3100	10		01/01/1982	
	AD3110	10		01/01/1982	
	AD3111	60		01/01/1982	
	AD3111	60		03/15/1982	
	AD3111	70		03/15/1982	
	AD3111	80		04/15/1982	

名前:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
	AD3111	180		10/15/1982	
	AD3111	70		02/15/1982	
	AD3111	80		09/15/1982	
	AD3112	60		01/01/1982	
	AD3112	60		02/01/1982	
	AD3112	60		01/01/1983	
	AD3112	70		02/01/1982	
	AD3112	70		03/15/1982	
	AD3112	70		08/15/1982	
	AD3112	80		08/15/1982	
	AD3112	80		10/15/1982	
	AD3112	180		08/15/1982	
	AD3113	70		06/15/1982	
	AD3113	70		07/01/1982	
	AD3113	80		01/01/1982	
	AD3113	80		03/01/1982	
	AD3113	180		03/01/1982	
	AD3113	180		04/15/1982	
	AD3113	180		06/01/1982	
	AD3113	60		03/01/1982	
	AD3113	60		04/01/1982	
	AD3113	60		09/01/1982	
	AD3113	70		09/01/1982	
	AD3113	70		10/15/1982	
	IF1000	10		06/01/1982	
	IF1000	90		10/01/1982	
	IF1000	100		10/01/1982	
	IF2000	10		01/01/1982	
	IF2000	100		01/01/1982	
	IF2000	100		03/01/1982	
	IF2000	110		03/01/1982	
	IF2000	110		10/01/1982	
	MA2100	10		01/01/1982	
	MA2100	20		01/01/1982	
	MA2110	10		01/01/1982	
	MA2111	50		01/01/1982	
	MA2111	60		06/15/1982	
	MA2111	40		01/01/1982	
	MA2112	60		01/01/1982	
	MA2112	180		07/15/1982	
	MA2112	70		06/01/1982	

名前:	PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
	MA2112	70		01/01/1982	
	MA2112	80		10/01/1982	
	MA2113	60		07/15/1982	
	MA2113	80		01/01/1982	
	MA2113	70		04/01/1982	
	MA2113	80		10/01/1982	
	MA2113	180		10/01/1982	
	OP1000	10		01/01/1982	
	OP1010	10		01/01/1982	
	OP1010	130		01/01/1982	
	OP2010	10		01/01/1982	
	OP2011	140		01/01/1982	
	OP2011	150		01/01/1982	
	OP2012	140		01/01/1982	
	OP2012	160		01/01/1982	
	OP2013	140		01/01/1982	
	OP2013	170		01/01/1982	
	PL2100	30		01/01/1982	

PROJECT 表

名前:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
タイプ:	CHAR(6) NOT NULL	VARCHAR(24) NOT NULL	CHAR(3) NOT NULL	CHAR(6) NOT NULL	DEC(5,2)	DATE	DATE	CHAR(6)
説明:	プロジェクト番号	プロジェクト名	担当部門	担当従業員	推定平均スタッフ	推定開始日付	推定終了日付	主要なプロジェクト (サブプロジェクト用)
値:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010

名前:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

SALES 表

名前:	SALES_DATE	SALES_PERSON	REGION	SALES
タイプ:	DATE	VARCHAR(15)	VARCHAR(15)	INTEGER
説明:	販売日	従業員のラストネーム	販売地域	販売数
値:	12/31/2005	LUCCHESSI	Ontario-South	1
	12/31/2005	LEE	Ontario-South	3
	12/31/2005	LEE	Quebec	1
	12/31/2005	LEE	Manitoba	2
	12/31/2005	GOUNOT	Quebec	1
	03/29/2006	LUCCHESSI	Ontario-South	3
	03/29/2006	LUCCHESSI	Quebec	1
	03/29/2006	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/2006	LEE	Quebec	3
	03/29/2006	LEE	Manitoba	5
	03/29/2006	GOUNOT	Ontario-South	3
	03/29/2006	GOUNOT	Quebec	1
	03/29/2006	GOUNOT	Manitoba	7
	03/30/2006	LUCCHESSI	Ontario-South	1
	03/30/2006	LUCCHESSI	Quebec	2
	03/30/2006	LUCCHESSI	Manitoba	1
	03/30/2006	LEE	Ontario-South	7
	03/30/2006	LEE	Ontario-North	3
	03/30/2006	LEE	Quebec	7
	03/30/2006	LEE	Manitoba	4
	03/30/2006	GOUNOT	Ontario-South	2
	03/30/2006	GOUNOT	Quebec	18
	03/30/2006	GOUNOT	Manitoba	1
	03/31/2006	LUCCHESSI	Manitoba	1
	03/31/2006	LEE	Ontario-South	14
	03/31/2006	LEE	Ontario-North	3
	03/31/2006	LEE	Quebec	7
	03/31/2006	LEE	Manitoba	3
	03/31/2006	GOUNOT	Ontario-South	2
	03/31/2006	GOUNOT	Quebec	1
	04/01/2006	LUCCHESSI	Ontario-South	3
	04/01/2006	LUCCHESSI	Manitoba	1
	04/01/2006	LEE	Ontario-South	8

名前:	SALES_DATE	SALES_PERSON	REGION	SALES
	04/01/2006	LEE	Ontario-North	-
	04/01/2006	LEE	Quebec	8
	04/01/2006	LEE	Manitoba	9
	04/01/2006	GOUNOT	Ontario-South	3
	04/01/2006	GOUNOT	Ontario-North	1
	04/01/2006	GOUNOT	Quebec	3
	04/01/2006	GOUNOT	Manitoba	7

STAFF 表

名前:	ID	名前	DEPT	JOB	YEARS	SALARY	COMM
タイプ:	SMALLINT NOT NULL	VARCHAR(9)	SMALLINT	CHAR(5)	SMALLINT	DECIMAL(7,2)	DECIMAL(7,2)
説明:	従業員番号	従業員名	部署番号	ジョブ・タイプ	勤務年数	現在の給与	歩合
値:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

PRODUCT 表

名前:	PID	名前	PRICE	PROMOPRICE	PROMOSTART	PROMOEND	DESCRIPTION
タイプ:	VARCHAR(10) NOT NULL	VARCHAR(128)	DECIMAL(30,2)	DECIMAL(30,2)	DATE	DATE	XML
値:	100-100-01	Snow Shovel, Basic 22 inch	9.99	7.25	11/19/2004	12/19/2004	`p1.xml`
	100-101-01	Snow Shovel, Deluxe 24 inch	19.99	15.99	12/18/2005	02/28/2006	`p2.xml`
	100-103-01	Snow Shovel, Super Deluxe 26 inch	49.99	39.99	12/22/2005	02/22/2006	`p3.xml`
	100-201-01	Ice Scraper, Windshield 4 inch	3.99	--	--	--	`p4.xml`

上の表product.xsd の XML 列の XML スキーマ定義ファイルを示します。

PURCHASEORDER 表

名前:	POID	STATUS	CUSTID	ORDERDATE	PORDER	COMMENTS
タイプ:	BIGINT NOT NULL	VARCHAR(10) NOT NULL	BIGINT	DATE	XML	VARCHAR(1000)
値:	5000	Unshipped	1002	02/18/2006	po1.xml	THIS IS A NEW PURCHASE ORDER
	5001	Shipped	1003	02/03/2005	po2.xml	THIS IS A NEW PURCHASE ORDER
	5002	Shipped	1001	02/29/2004	po3.xml	THIS IS A NEW PURCHASE ORDER
	5003	Shipped	1002	02/28/2005	po4.xml	THIS IS A NEW PURCHASE ORDER
	5004	Shipped	1005	11/18/2005	po5.xml	THIS IS A NEW PURCHASE ORDER
	5006	Shipped	1002	03/01/2006	po6.xml	THIS IS A NEW PURCHASE ORDER

上の表porder.xsd の XML 列の XML スキーマ定義ファイルを示します。

CUSTOMER 表

名前:	CID	INFO
タイプ:	BIGINT NOT NULL	XML
値:	1000	c1.xml
	1001	c2.xml
	1002	c3.xml
	1003	c4.xml
	1004	c5.xml
	1005	c6.xml

上の表customer.xsd の XML 列の XML スキーマ定義ファイルを示します。

CATALOG 表

名前:	名前	CATALOG
タイプ:	VARCHAR(128) NOT NULL	XML

上の表catalog.xsd の XML 列の XML スキーマ定義ファイルを示します。

INVENTORY 表

名前:	PID	QUANTITY	LOCATION
タイプ:	VARCHAR(10) NOT NULL	INTEGER	VARCHAR(128)
値:	100-100-01	5	--
	100-101-01	25	Store
	100-103-01	55	Store
	100-201-01	99	Warehouse

PRODUCTSUPPLIER 表

名前:	PID	SID
タイプ:	VARCHAR(10) 非ヌル	VARCHAR(10) 非ヌル
値:	100-101-01	100
	100-201-01	101

SUPPLIERS 表

名前:	SID	ADDR
タイプ:	VARCHAR(10) 非ヌル	XML
値:	100	s1.xml
	101	s2.xml

上の表supplier.xsd の XML 列の XML スキーマ定義ファイルを示します。

BLOB および CLOB データ・タイプのサンプル・ファイル

このセクションでは、EMP_PHOTO ファイル (従業員の写真) および EMP_RESUME ファイル (従業員の履歴書) にあるデータを示します。

Quintana の写真

図 15. Dolores M. Quintana

Quintana の履歴書

以下のテキストは、ファイル db200130.asc にあります。

Resume: Dolores M. Quintana

個人情報

Address:
1150 Eglinton Ave Mellonville, Idaho 83725

Phone: (208) 555-9933

Birthdate:
September 15, 1925

Sex: Female

Marital Status:
Married

Height:
5'2"

Weight:
120 lbs.

Department Information

Employee Number:
000130

Dept Number:
C01

Manager:
Sally Kwan

Position:
Analyst

Phone: (208) 555-4578

Hire Date:
1971-07-28

Education

1965 Math and English, B.A. Adelphi University

1960 Dental Technician Florida Institute of Technology

Work History

10/91 - present
Advisory Systems Analyst Producing documentation tools for engineering department.

12/85 - 9/91
Technical Writer, Writer, text programmer, and planner.

1/79 - 11/85
COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

Interests

- Cooking
- Reading
- Sewing
- Remodeling

以下は、ファイル db200130.htm の内容です。

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  Delores M. Quintana
<!-- Begin Header Records  ========================================== -->
```

```
<!-- DB200130 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:23                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:33:57">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:33:57">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:33:57"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  Delores M. Quintana<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  Delores M. Quintana<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  Delores M. Quintana<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>1150 Eglinton Ave
<BR>
Mellonville, Idaho 83725
<DT>Phone:
<DD>(208) 875-9933
<DT>Birthdate:
<DD>September 15, 1925
<DT>Sex:
<DD>Female
<DT>Marital Status:
<DD>Married
<DT>Height:
<DD>5'2"
<DT>Weight:
<DD>120 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000130
<DT>Dept Number:
<DD>C01
<DT>Manager:
<DD>Sally Kwan
<DT>Position:
<DD>Analyst
<DT>Phone:
<DD>(208) 385-4578
<DT>Hire Date:
<DD>1971-07-28<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1965
<DD>Math and English, B.A.
<BR>
Adelphi University
<P><DT>1960
<DD>Dental Technician
<BR>
Florida Institute of Technology<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>10/91 - present
<DD>Advisory Systems Analyst
<BR>
Producing documentation tools for engineering department.
<P><DT>12/85 - 9/91
<DD>Technical Writer
<BR>
Writer, text programmer, and planner.
<P><DT>1/79 - 11/85
<DD>COBOL Payroll Programmer
```

```
<BR>
Writing payroll programs for a diesel fuel company.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Cooking
<LI>Reading
<LI>Sewing
<LI>Remodeling<A NAME="Bot_Of_Page">
```

Nicholls の写真

図 16. Heather A. Nicholls

Nicholls の履歴書

以下のテキストは、ファイル db200140.asc にあります。

Resume: Heather A. Nicholls

個人情報

Address:
844 Don Mills Ave Mellonville, Idaho 83734

Phone: (208) 555-2310

Birthdate:
January 19, 1946

Sex: Female

Marital Status:
Single

Height:
5'8"

Weight:
130 lbs.

Department Information

Employee Number:
000140

Dept Number:
C01

Manager:
Sally Kwan

Position:
Analyst

Phone: (208) 555-1793

Hire Date:
1976-12-15

Education

1972 Computer Engineering, Ph.D. University of Washington

1969 Music and Physics, M.A. Vassar College

Work History

2/83 - present
Architect, OCR Development Designing the architecture of OCR products.

12/76 - 1/83
Text Programmer Optical CHARacter recognition (OCR) programming in PL/I.

9/72 - 11/76
Punch Card Quality Analyst Checking punch cards met quality specifications.

Interests

- Model railroading
- Interior decorating
- Embroidery
- Knitting

以下は、ファイル db200140.htm の内容です。

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  Heather A. Nicholls
<!-- Begin Header Records  ========================================== -->
<!-- DB200140 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:21                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:17">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:17">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:17"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  Heather A. Nicholls<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  Heather A. Nicholls<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  Heather A. Nicholls<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>844 Don Mills Ave
<BR>
```

```
Mellonville, Idaho 83734
<DT>Phone:
<DD>(208) 610-2310
<DT>Birthdate:
<DD>January 19, 1946
<DT>Sex:
<DD>Female
<DT>Marital Status:
<DD>Single
<DT>Height:
<DD>5'8"
<DT>Weight:
<DD>130 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000140
<DT>Dept Number:
<DD>C01
<DT>Manager:
<DD>Sally Kwan
<DT>Position:
<DD>Analyst
<DT>Phone:
<DD>(208) 385-1793
<DT>Hire Date:
<DD>1976-12-15<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1972
<DD>Computer Engineering, Ph.D.
<BR>
University of Washington
<P><DT>1969
<DD>Music and Physics, B.A.
<BR>
Vassar College<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>2/83 - present
<DD>Architect, OCR Development
<BR>
Designing the architecture of OCR products.
<P><DT>12/76 - 1/83
<DD>Text Programmer
<BR>
Optical CHARacter recognition (OCR) programming in PL/I.
<P><DT>9/72 - 11/76
<DD>Punch Card Quality Analyst
<BR>
Checking punch cards met quality specifications.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Model railroading
<LI>Interior decorating
<LI>Embroidery
<LI>Knitting<A NAME="Bot_Of_Page">
```

Adamson の写真

図 17. Bruce Adamson

Adamson の履歴書

以下のテキストは、ファイル db200150.asc にあります。

Resume: Bruce Adamson

個人情報

Address:
3600 Steeles Ave Mellonville, Idaho 83757

Phone: (208) 555-4489

Birthdate:
May 17, 1947

Sex: Male

Marital Status:
Married

Height:
6'0"

Weight:
175 lbs.

Department Information

Employee Number:
000150

Dept Number:
D11

Manager:
Irving Stern

Position:
Designer

Phone: (208) 555-4510

Hire Date:

1972-02-12

Education

1971 Environmental Engineering, M.Sc. Johns Hopkins University

1968 American History, B.A. Northwestern University

Work History

8/79 - present

Neural Network Design Developing neural networks for machine intelligence products.

2/72 - 7/79

Robot Vision Development Developing rule-based systems to emulate sight.

9/71 - 1/72

Numerical Integration Specialist Helping bank systems communicate with each other.

Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

以下は、ファイル db200150.htm の内容です。

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  Bruce Adamson
<!-- Begin Header Records  ========================================== -->
<!-- DB200150 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:21                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:39">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:39">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:39"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  Bruce Adamson<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  Bruce Adamson<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  Bruce Adamson<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>3600 Steeles Ave
<BR>
Mellonville, Idaho 83757
<DT>Phone:
<DD>(208) 725-4489
<DT>Birthdate:
<DD>May 17, 1947
<DT>Sex:
```

```
<DD>Male
<DT>Marital Status:
<DD>Married
<DT>Height:
<DD>6'0"
<DT>Weight:
<DD>175 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000150
<DT>Dept Number:
<DD>D11
<DT>Manager:
<DD>Irving Stern
<DT>Position:
<DD>Designer
<DT>Phone:
<DD>(208) 385-4510
<DT>Hire Date:
<DD>1972-02-12<P>
<H3><A NAME="Header_4">Education<DL>
<P><DT>1971
<DD>Environmental Engineering, M.Sc.
<BR>
Johns Hopkins University
<P><DT>1968
<DD>American History, B.A.
<BR>
Northwestern University<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>8/79 - present
<DD>Neural Network Design
<BR>
Developing neural networks for machine intelligence products.
<P><DT>2/72 - 7/79
<DD>Robot Vision Development
<BR>
Developing rule-based systems to emulate sight.
<P><DT>9/71 - 1/72
<DD>Numerical Integration Specialist
<BR>
Helping bank systems communicate with each other.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Racing motorcycles
<LI>Building loudspeakers
<LI>Assembling personal computers
<LI>Sketching<A NAME="Bot_Of_Page">
```

Walker の写真

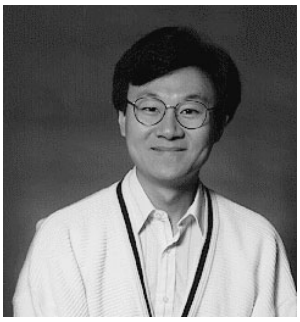


図 18. James H. Walker

Walker の履歴書

以下のテキストは、ファイル db200190.asc にあります。

Resume: James H. Walker

個人情報

Address:
3500 Steeles Ave Mellonville, Idaho 83757

Phone: (208) 555-7325

Birthdate:
June 25, 1952

Sex: Male

Marital Status:
Single

Height:
5'11"

Weight:
166 lbs.

Department Information

Employee Number:
000190

Dept Number:
D11

Manager:
Irving Stern

Position:
Designer

Phone: (208) 555-2986

Hire Date:
1974-07-26

Education

1974 Computer Studies, B.Sc. University of Massachusetts

1972 Linguistic Anthropology, B.A. University of Toronto

Work History

6/87 - present
Microcode Design Optimizing algorithms for mathematical functions.

4/77 - 5/87
Printer Technical Support Installing and supporting laser printers.

9/74 - 3/77

Maintenance Programming Patching assembly language compiler for mainframes.

Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

以下は、ファイル db200190.htm の内容です。

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3//EN">
<HTML><HEAD>
<TITLE>Resume:  James H. Walker
<!-- Begin Header Records  ========================================== -->
<!-- DB200190 SCRIPT A converted by B2H R4.1 (346) (CMS) by MJA at    -->
<!-- RCHVMW2 on 16 Aug 2000 at 11:35:20                               -->
<META HTTP-EQUIV="updated" CONTENT="Wed, 16 Aug 2000 11:34:59">
<META HTTP-EQUIV="review" CONTENT="Thu, 16 Aug 2001 11:34:59">
<META HTTP-EQUIV="expires" CONTENT="Fri, 16 Aug 2002 11:34:59"><BODY>
<!-- End Header Records  ========================================== -->
<A NAME="Top_Of_Page"><H1>Resume:  James H. Walker<HR>
<H2><A NAME="ToC">Table of Contents<A NAME="ToC_1" HREF="#Header_1">
Resume:  James H. Walker<BR>
<A NAME="ToC_2" HREF="#Header_2">Personal Information<BR>
<A NAME="ToC_3" HREF="#Header_3">Department Information<BR>
<A NAME="ToC_4" HREF="#Header_4">Education<BR>
<A NAME="ToC_5" HREF="#Header_5">Work History<BR>
<A NAME="ToC_6" HREF="#Header_6">Interests<BR>
<HR><P>
<P>
<H3><A NAME="Header_1">Resume:  James H. Walker<P>
<H3><A NAME="Header_2">Personal Information<DL COMPACT>
<DT>Address:
<DD>3500 Steeles Ave
<BR>
Mellonville, Idaho 83757
<DT>Phone:
<DD>(208) 725-7325
<DT>Birthdate:
<DD>June 25, 1952
<DT>Sex:
<DD>Male
<DT>Marital Status:
<DD>Single
<DT>Height:
<DD>5'11"
<DT>Weight:
<DD>166 lbs.<P>
<H3><A NAME="Header_3">Department Information<DL COMPACT>
<DT>Employee Number:
<DD>000190
<DT>Dept Number:
<DD>D11
<DT>Manager:
<DD>Irving Stern
<DT>Position:
<DD>Designer
<DT>Phone:
<DD>(208) 385-2986
<DT>Hire Date:
<DD>1974-07-26<P>
```

```
<H3><A NAME="Header_4">Education<DL>
<P><DT>1974
<DD>Computer Studies, B.Sc.
<BR>
University of Massachusetts
<P><DT>1972
<DD>Linguistic Anthropology, B.A.
<BR>
University of Toronto<P>
<H3><A NAME="Header_5">Work History<DL>
<P><DT>6/87 - present
<DD>Microcode Design
<BR>
Optimizing algorithms for mathematical functions.
<P><DT>4/77 - 5/87
<DD>Printer Technical Support
<BR>
Installing and supporting laser printers.
<P><DT>9/74 - 3/77
<DD>Maintenance Programming
<BR>
Patching assembly language compiler for mainframes.<P>
<H3><A NAME="Header_6">Interests<UL COMPACT>
<LI>Wine tasting
<LI>Skiing
<LI>Swimming
<LI>Dancing<A NAME="Bot_Of_Page">
```

付録 G. 予約済みスキーマ名と予約語

データベース・マネージャーに必要な特定の名前の使用には制限があります。名前によっては、予約済みで、アプリケーション・プログラムで使用できない名前があります。また、データベース・マネージャーによって、その使用は禁止されてはいませんが、アプリケーション・プログラムによる使用をお勧めできない名前もあります。

予約済みのスキーマ名は以下のとおりです。

- SYSCAT
- SYSFUN
- SYSIBM
- SYSIBMADM
- SYSPROC
- SYSSTAT

'SYS' は規則によりシステムで予約されている領域を示すのに使用されるので、'SYS' の接頭部で始まるスキーマ名は使用しないようにしてください。名前が 'SYS' で始まるスキーマには、別名、グローバル変数、トリガー、ユーザー定義関数、またはユーザー定義タイプを入れることができません (SQLSTATE 42939)。

DB2QP スキーマおよび SYSTOOLS スキーマは、DB2 ツールが使用するために確保されています。データベース・マネージャーによってこれらのスキーマの使用は禁止されてはいませんが、ユーザーがそれらを明示的に定義することはお勧めできません。

さらに、SESSION はスキーマ名としては使用しないようお勧めします。宣言済み一時表は SESSION で修飾しなければならないため、アプリケーションが持続表の名前と同じ名前を指定して一時表を宣言してしまい、アプリケーションのロジックが複雑になってしまう場合があります。このような事態を避けるため、宣言済み一時表を扱う場合を除いて、スキーマ SESSION を使用しないでください。

DB2 バージョン 9 では、特別な予約語はありません。キーワードも、SQL キーワードとして解釈される可能性があるコンテキスト以外であれば、通常 ID として使用できます。キーワードとして解釈されるコンテキストの場合は、その語を区切り ID として指定する必要があります。例えば、SELECT ステートメントに、区切り ID でない COUNT を列名として使用することはできません。

ISO/ANSI SQL2003 および他の DB2 データベース製品には、DB2 Database for Linux, UNIX, and Windows では使用されていない予約語が含まれていますが、移植性が低下するので、通常 ID として使用しないようにお勧めします。

さまざまな DB2 データベース製品間の移植性のためには、以下を予約語と見なす必要があります。

```
ACTIVATE        DOCUMENT        LOCK            ROUND_CEILING
ADD             DOUBLE          LOCKMAX         ROUND_DOWN
AFTER           DROP            LOCKSIZE        ROUND_FLOOR
```

© Copyright IBM Corp. 1993, 2010

ALIAS
ALL
ALLOCATE
ALLOW
ALTER
AND
ANY
AS
ASENSITIVE
ASSOCIATE
ASUTIME
AT
ATTRIBUTES
AUDIT
AUTHORIZATION
AUX
AUXILIARY
BEFORE
BEGIN
BETWEEN
BINARY
BUFFERPOOL
BY
CACHE
CALL
CALLED
CAPTURE
CARDINALITY
CASCADED
CASE
CAST
CCSID
CHAR
CHARACTER
CHECK
CLONE
CLOSE
CLUSTER
COLLECTION
COLLID
COLUMN
COMMENT
COMMIT
CONCAT
CONDITION
CONNECT
CONNECTION
CONSTRAINT
CONTAINS
CONTINUE
COUNT
COUNT_BIG
CREATE
CROSS
CURRENT
CURRENT_DATE
CURRENT_LC_CTYPE
CURRENT_PATH
CURRENT_SCHEMA
CURRENT_SERVER
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_TIMEZONE
CURRENT_USER
CURSOR
CYCLE
DATA
DSSIZE
DYNAMIC
EACH
EDITPROC
ELSE
ELSEIF
ENABLE
ENCODING
ENCRYPTION
END
END-EXEC
ENDING
ERASE
ESCAPE
EVERY
EXCEPT
EXCEPTION
EXCLUDING
EXCLUSIVE
EXECUTE
EXISTS
EXIT
EXPLAIN
EXTERNAL
EXTRACT
FENCED
FETCH
FIELDPROC
FILE
FINAL
FOR
FOREIGN
FREE
FROM
FULL
FUNCTION
GENERAL
GENERATED
GET
GLOBAL
GO
GOTO
GRANT
GRAPHIC
GROUP
HANDLER
HASH
HASHED_VALUE
HAVING
HINT
HOLD
HOUR
HOURS
IDENTITY
IF
IMMEDIATE
IN
INCLUDING
INCLUSIVE
INCREMENT
INDEX
INDICATOR
INF
INFINITY
INHERIT
INNER
INOUT
LONG
LOOP
MAINTAINED
MATERIALIZED
MAXVALUE
MICROSECOND
MICROSECONDS
MINUTE
MINUTES
MINVALUE
MODE
MODIFIES
MONTH
MONTHS
NAN
NEW
NEW_TABLE
NEXTVAL
NO
NOCACHE
NOCYCLE
NODENAME
NODENUMBER
NOMAXVALUE
NOMINVALUE
NONE
NOORDER
NORMALIZED
NOT
NULL
NULLS
NUMPARTS
OBID
OF
OLD
OLD_TABLE
ON
OPEN
OPTIMIZATION
OPTIMIZE
OPTION
OR
ORDER
OUT
OUTER
OVER
OVERRIDING
PACKAGE
PADDED
PAGESIZE
PARAMETER
PART
PARTITION
PARTITIONED
PARTITIONING
PARTITIONS
PASSWORD
PATH
PIECESIZE
PLAN
POSITION
PRECISION
PREPARE
PREVVAL
PRIMARY
PRIQTY
PRIVILEGES
ROUND_HALF_DOWN
ROUND_HALF_EVEN
ROUND_HALF_UP
ROUND_UP
ROUTINE
ROW
ROW_NUMBER
ROWNUMBER
ROWS
ROWSET
RRN
RUN
SAVEPOINT
SCHEMA
SCRATCHPAD
SCROLL
SEARCH
SECOND
SECONDS
SECQTY
SECURITY
SELECT
SENSITIVE
SEQUENCE
SESSION
SESSION_USER
SET
SIGNAL
SIMPLE
SNAN
SOME
SOURCE
SPECIFIC
SQL
SQLID
STACKED
STANDARD
START
STARTING
STATEMENT
STATIC
STATMENT
STAY
STOGROUP
STORES
STYLE
SUBSTRING
SUMMARY
SYNONYM
SYSFUN
SYSIBM
SYSPROC
SYSTEM
SYSTEM_USER
TABLE
TABLESPACE
THEN
TIME
TIMESTAMP
TO
TRANSACTION
TRIGGER
TRIM
TRUNCATE
TYPE
UNDO
UNION

DATABASE	INSENSITIVE	PROCEDURE	UNIQUE
DATAPARTITIONNAME	INSERT	PROGRAM	UNTIL
DATAPARTITIONNUM	INTEGRITY	PSID	UPDATE
DATE	INTERSECT	PUBLIC	USAGE
DAY	INTO	QUERY	USER
DAYS	IS	QUERYNO	USING
DB2GENERAL	ISOBID	RANGE	VALIDPROC
DB2GENRL	ISOLATION	RANK	VALUE
DB2SQL	ITERATE	READ	VALUES
DBINFO	JAR	READS	VARIABLE
DBPARTITIONNAME	JAVA	RECOVERY	VARIANT
DBPARTITIONNUM	JOIN	REFERENCES	VCAT
DEALLOCATE	KEEP	REFERENCING	VERSION
DECLARE	KEY	REFRESH	VIEW
DEFAULT	LABEL	RELEASE	VOLATILE
DEFAULTS	LANGUAGE	RENAME	VOLUMES
DEFINITION	LATERAL	REPEAT	WHEN
DELETE	LC_CTYPE	RESET	WHENEVER
DENSE_RANK	LEAVE	RESIGNAL	WHERE
DENSERANK	LEFT	RESTART	WHILE
DESCRIBE	LIKE	RESTRICT	WITH
DESCRIPTOR	LINKTYPE	RESULT	WITHOUT
DETERMINISTIC	LOCAL	RESULT_SET_LOCATOR	WLM
DIAGNOSTICS	LOCALDATE	RETURN	WRITE
DISABLE	LOCALE	RETURNS	XMLELEMENT
DISALLOW	LOCALTIME	REVOKE	XMLEXISTS
DISCONNECT	LOCALTIMESTAMP	RIGHT	XMLNAMESPACES
DISTINCT	LOCATOR	ROLE	YEAR
DO	LOCATORS	ROLLBACK	YEARS

以下は、上記のリストに載っていない、ISO/ANSI SQL2003 予約語のリストです。

ABS	GROUPING	REGR_INTERCEPT
ARE	INT	REGR_R2
ARRAY	INTEGER	REGR_SLOPE
ASYMMETRIC	INTERSECTION	REGR_SXX
ATOMIC	INTERVAL	REGR_SXY
AVG	LARGE	REGR_SYY
BIGINT	LEADING	ROLLUP
BLOB	LN	SCOPE
BOOLEAN	LOWER	SIMILAR
BOTH	MATCH	SMALLINT
CEIL	MAX	SPECIFICTYPE
CEILING	MEMBER	SQLEXCEPTION
CHAR_LENGTH	MERGE	SQLSTATE
CHARACTER_LENGTH	METHOD	SQLWARNING
CLOB	MIN	SQRT
COALESCE	MOD	STDDEV_POP
COLLATE	MODULE	STDDEV_SAMP
COLLECT	MULTISET	SUBMULTISET
CONVERT	NATIONAL	SUM
CORR	NATURAL	SYMMETRIC
CORRESPONDING	NCHAR	TABLESAMPLE
COVAR_POP	NCLOB	TIMEZONE_HOUR
COVAR_SAMP	NORMALIZE	TIMEZONE_MINUTE
CUBE	NULLIF	TRAILING
CUME_DIST	NUMERIC	TRANSLATE
CURRENT_DEFAULT_TRANSFORM_GROUP	OCTET_LENGTH	TRANSLATION
CURRENT_ROLE	ONLY	TREAT
CURRENT_TRANSFORM_GROUP_FOR_TYPE	OVERLAPS	TRUE
DEC	OVERLAY	UESCAPE
DECIMAL	PERCENT_RANK	UNKNOWN
DEREF	PERCENTILE_CONT	UNNEST
ELEMENT	PERCENTILE_DISC	UPPER
EXEC	POWER	VAR_POP
EXP	REAL	VAR_SAMP
FALSE	RECURSIVE	VARCHAR

FILTER	REF	VARYING
FLOAT	REGR_AVGX	WIDTH_BUCKET
FLOOR	REGR_AVGY	WINDOW
FUSION	REGR_COUNT	WITHIN

付録 H. トリガーと参照制約の間の相互作用の例

更新操作を行うと、トリガーと参照制約およびチェック制約の相互作用が発生することがあります。

図 19 とその後の説明は、データベースのデータを更新するステートメントに対して行われる典型的な処理を示しています。

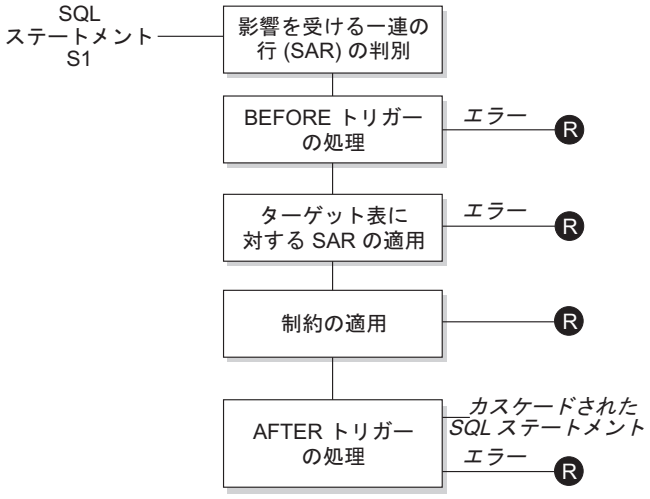


図 19. 関連するトリガーと制約を伴うステートメントの処理

図 19 は、表を更新するステートメントの一般的な処理の順序を示しています。ここでは、BEFORE トリガー、参照制約、チェック制約、および AFTER トリガーがカスケードしている表を想定しています。図 19 に示されているボックスやその他の項目について、以下に説明します。

- ステートメント S_1

  これは、プロセスを開始する DELETE、INSERT、または UPDATE ステートメントです。ステートメント S_1 は、この説明においてサブジェクト表 と呼ばれる表 (または表に対する更新可能なビュー) を指定しています。

- 影響を受ける一連の行の判別

  このステップは、CASCADE および SET NULL の参照制約の削除規則と、AFTER トリガーからのカスケード・ステートメントに対して繰り返されるプロセスの開始点です。

  このステップの目的は、そのステートメントで*影響を受ける一連の行* を判別することです。含まれる行の集合は、ステートメントに基づいて、以下のようになります。

  – DELETE の場合、ステートメントの検索条件を満たしているすべての行 (位置指定 DELETE の場合は現在行)

© Copyright IBM Corp. 1993, 2010

- INSERT の場合、VALUES 節または全選択によって指定される行
- UPDATE の場合、検索条件を満たしているすべての行 (位置指定 UPDATE の場合は現在行)

影響を受ける一連の行が空の場合、BEFORE トリガー、サブジェクト表に適用される変更、またはステートメントの処理に対する制約はありません。

- BEFORE トリガーの処理

  BEFORE トリガーの処理はすべて作成の昇順で行われます。各 BEFORE トリガーは、影響を受ける一連の行内の各行ごとに 1 回ずつトリガー・アクションを処理します。

  トリガー・アクションの処理の過程でエラーが生じることがあり、そのような場合には元のステートメント S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

  BEFORE トリガーがない場合、または影響を受ける一連の行が空の場合、このステップはスキップされます。

- サブジェクト表への影響を受ける一連の行の適用

  データベース内のサブジェクト表への実際の削除、挿入、または更新は影響を受ける一連の行を使用して適用されます。

  影響を受ける一連の行の適用時にエラーが生じることがあり (ユニーク索引のあるロケーションに重複するキーを持つ行を挿入しようとした場合など)、そのような場合は元のステートメント S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

- 制約の適用

  影響を受ける一連の行が空でない場合には、サブジェクト表に関連した制約が適用されます。この制約には、ユニーク制約、ユニーク索引、参照制約、チェック制約、ビューに対する WITH CHECK OPTION に関連した検査などがあります。カスケード削除規則または NULL 設定のある参照制約では、追加のトリガーが活動化されることがあります。

  何らかの制約または WITH CHECK OPTION に違反するとエラーが発生し、S_1 の結果として行われた変更 (その時点までの) はロールバックされます。

- AFTER トリガーの処理

  S_1 によって活動化された AFTER トリガーは、すべて作成の昇順に処理されます。

  FOR EACH STATEMENT トリガーでは、影響を受ける一連の行が空の場合にも、1 回だけトリガー・アクションが処理されます。 FOR EACH ROW トリガーでは、影響を受ける一連の行内の各行ごとに 1 回ずつトリガー・アクションが処理されます。

  トリガー・アクションの処理の過程でエラーが生じることがあり、そのような場合は元の S_1 の結果としての変更内容 (これまでの) がすべてロールバックされます。

トリガーのトリガー・アクションには、トリガーによって実行される DELETE、INSERT、または UPDATE などのステートメントが入っている場合があります。この説明では、そのような各ステートメントは、*カスケードしたステートメント*と見なされます。

カスケードしたステートメントは、AFTER トリガーのトリガー・アクションの一部として処理される DELETE、INSERT、または UPDATE ステートメントです。そのステートメントによって、カスケード・レベルのトリガー処理が開始されます。これは、新しい S_1 としてトリガー・ステートメントを割り当てて、ここで説明した手順をすべて再帰的に実行することと見なすことができます。

各 S_1 ごとに起動されるすべての AFTER トリガーによって実行されるすべてのステートメントの処理が完了すると、元の S_1 の処理が完了します。

- R = 変更を S_1 の前までロールバックする操作

  制約違反も含めて、処理中にエラーが発生すると、元のステートメント S_1 の結果として直接または間接になされたすべての変更がロールバックされます。その場合、データベースは、元のステートメント S_1 の実行直前と同じ状態に戻ります。

付録 I. Explain 表

Explain 表は、Explain 機能がアクティブ化された時点のアクセス・プランをキャプチャーします。 Explain 表は、Explain 機能を呼び出す前に作成しておく必要があります。以下のいずれかの方法を使用して作成できます。

- 以下のように SYSPROC.SYSINSTALLOBJECTS プロシージャーを呼び出します。

  ```
  db2 CONNECT TO database-name
  db2 CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'C',
         CAST (NULL AS VARCHAR(128)), CAST (NULL AS VARCHAR(128)))
  ```

  この呼び出しは、SYSTOOLS スキーマの下で Explain 表を作成します。別のスキーマの下で作成するには、呼び出し内の最後のパラメーターとしてスキーマ名を指定します。

- 以下のように EXPLAIN.DDL DB2 コマンド・ファイルを実行します。

  ```
  db2 CONNECT TO database-name
  db2 -tf EXPLAIN.DDL
  ```

  このコマンド・ファイルは、現行スキーマの下で Explain 表を作成します。これは、Windows オペレーティング・システムでは DB2PATH¥misc ディレクトリーにあり、Linux および UNIX オペレーティング・システムでは INSTHOME/sqllib/misc ディレクトリーにあります。DB2PATH は DB2 コピーのインストール場所で、INSTHOME はインスタンスのホーム・ディレクトリーです。

Explain 機能は、データの取り込み先である Explain 表を修飾するときに、スキーマとして以下の ID を使用します。

- 動的 SQL のセッション許可 ID
- 静的 SQL のステートメント許可 ID

そのスキーマは、一連の Explain 表に関連付けられている場合もあれば、別のスキーマの下で一連の Explain 表を参照する別名に関連付けられている場合もあります。そのスキーマの下で Explain 表が検出されなかった場合、Explain 機能は、SYSTOOLS スキーマの下に Explain 表があるかどうかをチェックし、その表を使用しようとします。

Explain 機能によって Explain 表にデータを追加しても、トリガーが起動したり、参照制約またはチェック制約が起動したりすることはありません。例えば、EXPLAIN_INSTANCE 表に対する挿入トリガーを定義して、該当するステートメントが Explain されても、そのトリガーは起動しません。

パーティション・データベース・システムにおいて Explain 機能のパフォーマンスを改善するには、単一のデータベース・パーティション・グループ内に Explain 表を、できれば照会のコンパイル時に接続する予定のものと同じデータベース・パーティション上で作成することをお勧めします。

© Copyright IBM Corp. 1993, 2010

ADVISE_INDEX 表

ADVISE_INDEX 表は、推奨索引を示しています。

表 205. ADVISE_INDEX 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	いいえ	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	いいえ	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	いいえ	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	いいえ	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	いいえ	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	いいえ	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	いいえ	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	いいえ	この Explain 情報に関連したパッケージ内のセクションの番号。
QUERYNO	INTEGER	いいえ	いいえ	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。
QUERYTAG	CHAR(20)	いいえ	いいえ	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLP」です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
NAME	VARCHAR(128)	いいえ	いいえ	索引の名前。
CREATOR	VARCHAR(128)	いいえ	いいえ	索引名の修飾子。
TBNAME	VARCHAR(128)	いいえ	いいえ	索引が定義されている表またはニックネームの名前。
TBCREATOR	VARCHAR(128)	いいえ	いいえ	表名の修飾子。
COLNAMES	CLOB(2M)	いいえ	いいえ	列名のリスト。
UNIQUERULE	CHAR(1)	いいえ	いいえ	ユニーク規則。 • D = 重複可 • P = 1 次索引 • U = ユニークな項目のみ可

表 205. ADVISE_INDEX 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
COLCOUNT	SMALLINT	いいえ	いいえ	キー内の列数と組み込み列 (もしあれば) の数の合計。
IID	SMALLINT	いいえ	いいえ	索引の内部 ID。
NLEAF	BIGINT	いいえ	いいえ	リーフ・ページの数。統計が収集されていない場合は -1。
NLEVELS	SMALLINT	いいえ	いいえ	索引レベルの数。統計が収集されていない場合は -1。
FIRSTKEYCARD	BIGINT	いいえ	いいえ	第 1 キーの値の数。統計が収集されていない場合は -1。
FULLKEYCARD	BIGINT	いいえ	いいえ	個別の全キー値の数。統計が収集されていない場合は -1。
CLUSTERRATIO	SMALLINT	いいえ	いいえ	索引によるデータ・クラスタリングの程度。統計が収集されていない場合、または詳細な索引統計が収集されている場合は -1 (それらの場合は CLUSTERFACTOR の方が使用されます)。
AVGPARTITION_ CLUSTERRATIO	SMALLINT	いいえ	いいえ	単一のデータ・パーティション内でのデータ・クラスタリングの程度。表が表パーティション化されていない場合、統計が収集されていない場合、または詳細な索引統計が収集されている場合 (その場合は CLUSTERFACTOR の方が使用されます) は、-1。
AVGPARTITION_ CLUSTERFACTOR	DOUBLE	いいえ	いいえ	単一のデータ・パーティション内でのクラスタリングの程度の詳細測定値。表が表パーティション化されていない場合、統計が収集されていない場合、あるいはニックネームに索引が定義されている場合は -1。
AVGPARTITION_PAGE_ FETCH_PAIRS	VARCHAR(520)	いいえ	いいえ	文字形式の整数ペアのリスト。各ペアは、潜在的なバッファー・プール・サイズと、表の単一データ・パーティションにアクセスするのに必要なページのフェッチ回数との対応を示します。データが利用できない場合、または表が表パーティション化されていない場合は、長さゼロのストリング。
DATAPARTITION_ CLUSTERFACTOR	DOUBLE	いいえ	いいえ	データ・パーティションに関しての索引キーの「クラスタリング」を測定する統計。このフィールドには、0 から 1 までの数値が入ります。1 は完全なクラスタリングを表し、0 はクラスタリングがないことを表します。
CLUSTERFACTOR	DOUBLE	いいえ	いいえ	より高い計算精度のクラスタリング。詳細索引統計を収集していない場合、あるいはニックネームに索引が定義されていない場合は -1。
USERDEFINED	SMALLINT	いいえ	いいえ	ユーザーによる定義。

表 205. ADVISE_INDEX 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
SYSTEM_REQUIRED	SMALLINT	いいえ	いいえ	• 次の条件のいずれかを満たす場合は 1。 – この索引が主キー制約またはユニーク・キー制約で必要です。またはこの索引がマルチディメンション・クラスタリング (MDC) 表のディメンション・ブロック索引または複合ブロック索引です。 – これは型付き表のオブジェクト ID (OID) 列に対する索引です。 • 次の条件の両方を満たす場合は 2。 – この索引は主キー制約またはユニーク・キー制約に必要です。または、この索引は MDC 表に対するディメンション・ブロック索引またはコンポジット・ブロック索引です。 – これは型付き表のオブジェクト ID (OID) 列に対する索引です。 • それ以外の場合は 0。
CREATE_TIME	TIMESTAMP	いいえ	いいえ	索引の作成された時刻。
STATS_TIME	TIMESTAMP	はい	いいえ	この索引について記録されている統計値が最後に変更された時刻。統計が利用できない場合は、NULL。
PAGE_FETCH_PAIRS	VARCHAR(520)	いいえ	いいえ	文字形式で表された整数ペアのリスト。それぞれのペアは、仮のバッファー内のページ数と、その仮のバッファーを使用した表のスキャンに必要なページ・フェッチ回数を表しています。 (データが利用できない場合は、長さ 0 のストリング。)
REMARKS	VARCHAR(254)	はい	いいえ	ユーザー提供のコメントまたは NULL 値。
DEFINER	VARCHAR(128)	いいえ	いいえ	索引を作成したユーザー。
CONVERTED	CHAR(1)	いいえ	いいえ	将来の利用のために予約済み。
SEQUENTIAL_PAGES	BIGINT	いいえ	いいえ	索引キーの順序でディスクに存在し、それらの間に大きなギャップがないか、わずかなギャップしかないリーフ・ページの数。 (統計が入手できない場合は -1。)
DENSITY	INTEGER	いいえ	いいえ	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表されます (0 から 100 の整数。統計が入手できない場合は -1)。
FIRST2KEYCARD	BIGINT	いいえ	いいえ	索引の最初の 2 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。
FIRST3KEYCARD	BIGINT	いいえ	いいえ	索引の最初の 3 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。

表 205. ADVISE_INDEX 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
FIRST4KEYCARD	BIGINT	いいえ	いいえ	索引の最初の 4 つの列を使用するキーの種類数 (統計がない場合、または適用されない場合は -1)。
PCTFREE	SMALLINT	いいえ	いいえ	索引を最初に作成する際に予約する索引リーフ・ページのパーセント。このスペースは、索引の作成後に行う挿入用に使用可能です。
UNIQUE_COLCOUNT	SMALLINT	いいえ	いいえ	ユニーク・キーに必要な列の数。常に <=COLCOUNT。組み込み列がある場合にのみ < COLCOUNT。索引にユニーク・キーがない場合は -1 (重複可能)。
MINPCTUSED	SMALLINT	いいえ	いいえ	ゼロでない場合は、オンライン索引デフラグが使用可能になり、その値は、ページをマージをする前に使用される最小スペースのしきい値です。
REVERSE_SCANS	CHAR(1)	いいえ	いいえ	• Y = 索引は逆スキャンをサポートする • N = 索引は逆スキャンをサポートしない
USE_INDEX	CHAR(1)	はい	いいえ	• Y = 推奨または評価された索引 • N = 推奨されない索引 • R = 既存のクラスタリング RID 索引の非クラスター化が推奨された (設計アドバイザーによって)。これは、表に対して新規のクラスタリング RID 索引が推奨されたケースです。
CREATION_TEXT	CLOB(2M)	いいえ	いいえ	索引の作成に使用された SQL ステートメント。
PACKED_DESC	BLOB(1M)	はい	いいえ	表の内部記述。
RUN_ID	TIMESTAMP	はい	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。
INDEXTYPE	VARCHAR(4)	いいえ	いいえ	索引のタイプ。 • CLUS = クラスタリング • REG = 通常 • DIM = ディメンション・ブロック索引 • BLOK = ブロック索引
EXISTS	CHAR(1)	いいえ	いいえ	データベース・カタログ内に索引が存在する場合は、'Y' に設定します。
RIDTOBLOCK	CHAR(1)	いいえ	いいえ	設計アドバイザー内にブロック索引を作成するのに RID 索引が使われていた場合は、'Y' に設定します。

ADVISE_INSTANCE 表

ADVISE_INSTANCE 表には、開始時刻などの db2advis の実行に関する情報が入っています。 db2advis の実行ごとに行が 1 つずつ入ります。他の ADVISE 表には、同じ設計アドバイザーの実行中に作成された行ごとに、 ADVISE_INSTANCE 表の START_TIME 列にリンクする外部キー (RUN_ID) が添付されます。

表 206. ADVISE_INSTANCE 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
START_TIME	TIMESTAMP	いいえ	PK	db2advis の実行が開始された時刻。
END_TIME	TIMESTAMP	いいえ	いいえ	db2advis の実行が終了した時刻。
MODE	VARCHAR(4)	いいえ	いいえ	設計アドバイザー上で -m オプションを使って指定された値。たとえば、'MC' は MQT と MDC を指定します。
WKLD_COMPRESSION	CHAR(4)	いいえ	いいえ	設計アドバイザーの実行で使われたワークロード圧縮。
STATUS	CHAR(9)	いいえ	いいえ	設計アドバイザーの実行の状況。状況は'STARTED' または 'COMPLETED' (正常完了時) になりますが、内部エラーの場合は 'EI' を、外部エラーの場合は 'EX' を前に付けられたエラー番号になります。後者の場合のエラー番号は SQLCODE を表します。

ADVISE_MQT 表

ADVISE_MQT 表には、設計アドバイザーから推奨されたマテリアライズ照会表 (MQT) に関する情報が入っています。

表 207. ADVISE_MQT 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	いいえ	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	いいえ	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	いいえ	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	いいえ	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	いいえ	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	いいえ	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	いいえ	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	いいえ	この Explain 情報に関連したパッケージ内のステートメントの番号。
NAME	VARCHAR(128)	いいえ	いいえ	MQT 名。
CREATOR	VARCHAR(128)	いいえ	いいえ	MQT 作成者名
IID	SMALLINT	いいえ	いいえ	内部 ID
CREATE_TIME	TIMESTAMP	いいえ	いいえ	MQT が作成された時刻。
STATS_TIME	TIMESTAMP	はい	いいえ	統計がとられた時刻。
NUMROWS	DOUBLE	いいえ	いいえ	MQT 内の推定行数。
NUMCOLS	SMALLINT	いいえ	いいえ	MQT に定義されている列の数。
ROWSIZE	DOUBLE	いいえ	いいえ	MQT 内の行の平均の長さ (バイト単位)。
BENEFIT	FLOAT	いいえ	いいえ	将来の使用のために予約済み。
USE_MQT	CHAR(1)	はい	いいえ	MQT が推奨された場合は 'Y' に設定します。
MQT_SOURCE	CHAR(1)	はい	いいえ	MQT 候補が生成された場所を示します。 MQT 候補が即時リフレッシュ MQT の場合は 'I' に、リフレッシュの据え置き MQT としてのみ作成できる場合は 'D' に設定します。
QUERY_TEXT	CLOB(2M)	いいえ	いいえ	MQT を定義する照会が入っています。
CREATION_TEXT	CLOB(2M)	いいえ	いいえ	MQT 用の CREATE TABLE DDL が入っています。
SAMPLE_TEXT	CLOB(2M)	いいえ	いいえ	MQT に関する詳細な統計を得るのに使われるサンプリング照会が入っています。詳細な統計が設計アドバイザーで必要な場合のみ使用します。その結果のサンプル化統計がこの表に示されます。これが NULL の場合、この MQT のサンプリング照会は作成されませんでした。

表 207. ADVISE_MQT 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
COLSTATS	CLOB(2M)	いいえ	いいえ	MQT の列統計が入っています (NULL でない場合)。その統計は XML フォーマットになっていて、中には列名、列カーディナリティー、および選択しだいで HIGH2KEY 値と LOW2KEY 値が組み入れられます。
EXTRA_INFO	BLOB(2M)	いいえ	いいえ	その他の出力用に予約されています。
TBSPACE	VARCHAR(128)	いいえ	いいえ	MQT に対して推奨される表スペース。
RUN_ID	TIMESTAMP	はい	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。
REFRESH_TYPE	CHAR(1)	いいえ	いいえ	即時の場合は 'I' に、据え置きの場合は 'D' に設定します。
EXISTS	CHAR(1)	いいえ	いいえ	データベース・カタログ内に MQT が存在する場合は、'Y' に設定します。

ADVISE_PARTITION 表

ADVISE_PARTITION 表には、設計アドバイザーから推奨されたデータベース・パーティションに関する情報が入りますが、この表にデータを追加できるのはパーティション・データベース環境においてのみです。

表 208. ADVISE_PARTITION 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	いいえ	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	いいえ	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	いいえ	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	いいえ	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	いいえ	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	いいえ	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	いいえ	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	いいえ	この Explain 情報に関連したパッケージ内のステートメントの番号。
QUERYNO	INTEGER	いいえ	いいえ	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。
QUERYTAG	CHAR(20)	いいえ	いいえ	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLP」です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
TBNAME	VARCHAR(128)	はい	いいえ	表名を指定します。
TBCREATOR	VARCHAR(128)	はい	いいえ	表作成者名を指定します。
PMID	SMALLINT	はい	いいえ	分散マップ ID を指定します。
TBSPACE	VARCHAR(128)	はい	いいえ	表を置く表スペースのページ・サイズを指定します。
COLNAMES	CLOB(2M)	はい	いいえ	データベース・パーティション列名をコンマで区切って指定します。
COLCOUNT	SMALLINT	はい	いいえ	データベース・パーティションの列の数を指定します。

表 208. ADVISE_PARTITION 表 (続き)： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
REPLICATE	CHAR(1)	はい	いいえ	データベース・パーティションを複製するかどうかを指定します。
COST	DOUBLE	はい	いいえ	データベース・パーティションの使用コストを指定します。
USEIT	CHAR(1)	はい	いいえ	EVALUATE PARTITION モードでデータベース・パーティションを使用するかどうかを指定します。 USEIT を 'Y' または 'y' に設定した場合は、データベース・パーティションを使用します。
RUN_ID	TIMESTAMP	はい	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。

ADVISE_TABLE 表

ADVISE_TABLE 表には、マテリアライズ照会表 (MQT)、マルチディメンション・クラスター表 (MDC)、およびデータベース・パーティション化に関する設計アドバイザーの最終勧告を使用して表を作成するためのデータ定義言語 (DDL) が保管されます。

表 209. ADVISE_TABLE 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
RUN_ID	TIMESTAMP	はい	FK	ADVISE_INSTANCE 表内の行の START_TIME に対応し、同じ設計アドバイザーの実行にリンクする値。
TABLE_NAME	VARCHAR(128)	いいえ	いいえ	表の名前。
TABLE_SCHEMA	VARCHAR(128)	いいえ	いいえ	表作成者の名前。
TABLESPACE	VARCHAR(128)	いいえ	いいえ	表の作成場所である表スペース。
SELECTION_FLAG	VARCHAR(4)	いいえ	いいえ	勧告タイプを示します。有効値は、MQT の場合は 'M'、データベース・パーティションの場合は 'P'、MDC の場合は 'C' です。このフィールドには、これらの値の任意のサブセットを入れることができます。たとえば、'MC' は、表を MQT 表と MDC 表とすることが推奨されたことを示します。
TABLE_EXISTS	CHAR(1)	いいえ	いいえ	データベース・カタログ内に表が存在する場合は、'Y' に設定します。
USE_TABLE	CHAR(1)	いいえ	いいえ	表に関する設計アドバイザーからの勧告がある場合は、'Y' に設定します。
GEN_COLUMNS	CLOB(2M)	いいえ	いいえ	表 DDL の作成内に生成された列を必要とする MDC 勧告がこの行に入っている場合、生成された列ストリングが入ります。
ORGANIZE_BY	CLOB(2M)	いいえ	いいえ	MDC 勧告の場合は、表 DDL の作成の ORGANIZE BY 節が入ります。
CREATION_TEXT	CLOB(2M)	いいえ	いいえ	表 DDL の作成が入ります。
ALTER_COMMAND	CLOB(2M)	いいえ	いいえ	表の ALTER TABLE ステートメントが入ります。

ADVISE_WORKLOAD 表

ADVISE_WORKLOAD 表は、ワークロードを構成するステートメントを示しています。

表 210. ADVISE_WORKLOAD 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
WORKLOAD_NAME	CHAR(128)	いいえ	いいえ	このステートメントが属している SQL ステートメント (ワークロード) の集合の名前。
STATEMENT_NO	INTEGER	いいえ	いいえ	ワークロード内のステートメントのうち、この Explain 情報に関連するもののステートメント番号。
STATEMENT_TEXT	CLOB(1M)	いいえ	いいえ	SQL ステートメントの内容。
STATEMENT_TAG	VARCHAR(256)	いいえ	いいえ	Explain 対象の各 SQL ステートメントの ID タグ。
FREQUENCY	INTEGER	いいえ	いいえ	ワークロード内でこのステートメントが使用される回数。
IMPORTANCE	DOUBLE	いいえ	いいえ	ステートメントの重要性。
WEIGHT	DOUBLE	いいえ	いいえ	ステートメントの優先度。
COST_BEFORE	DOUBLE	はい	いいえ	推奨が作成されていない場合、照会のコスト (timerons 単位)
COST_AFTER	DOUBLE	はい	いいえ	推奨が作成されている場合、照会のコスト (timerons 単位) COST_AFTER は、クラスター索引およびマルチディメンション・クラスタリング (MDC) に関係していないすべての推奨を反映します。
COMPILABLE	CHAR(17)	はい	いいえ	ステートメントを準備しようとしている間に生じた照会コンパイル・エラーを示します。この列が NULL であるか、または SQLCA で始まっていない場合、 SQL 照会は db2advis でコンパイルできます。コンパイル・エラーが db2advis または設計アドバイザーによって検出される場合、 COMPILABLE 列値は、8 バイトの長さの SQLCA.sqlcaid フィールド、コロン (:)、および 8 バイトの長さの SQLCA.sqlstate フィールドで構成されますが、この値は SQL ステートメントの戻りコードです。

EXPLAIN_ARGUMENT 表

EXPLAIN_ARGUMENT 表は、個々の演算子にユニークな特性がある場合、それを示します。

表 211. EXPLAIN_ARGUMENT 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のセクションの番号。
OPERATOR_ID	INTEGER	いいえ	いいえ	この照会内の演算子の固有な ID。
ARGUMENT_TYPE	CHAR(8)	いいえ	いいえ	この演算子の引数のタイプ。
ARGUMENT_VALUE	VARCHAR(1024)	はい	いいえ	この演算子の引数の値。値が LONG_ARGUMENT_VALUE にある場合は NULL。
LONG_ARGUMENT_VALUE	CLOB(2M)	はい	いいえ	この演算子の引数の値。(テキストが ARGUMENT_VALUE に収まらない場合。) 値が ARGUMENT_VALUE にある場合は NULL。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	部分集約標識。
BITFLTR	INTEGER FALSE	ハッシュ結合で使われるビット・フィルターのサイズ。
BLD_LEVEL	DB2 ビルド ID	ソース・コード・バージョンの内部識別ストリング。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE NONE SHARE UPDATE	ブロック・レベル・ロック意図。
CSERQY	TRUE FALSE	リモート照会が共通副次式。
CSETEMP	TRUE FALSE	共通副次式フラグに対する一時表。
DIRECT	TRUE	取り出し標識を指示する。
DPESTFLG	TRUE FALSE	DPNUMPRT 値が見積もりに基づいているかどうかを示す。可能な値は 'TRUE' (DPNUMPRT は、アクセスしたデータ・パーティションの見積もり数を表す) または 'FALSE' (DPNUMPRT はアクセスしたデータ・パーティションの実数を表す) です。
DPLSTPRT	NONE CHARACTER	アクセスしたデータ・パーティションを表す。これは、アクセスしたデータ・パーティションのコンマで区切られたリスト (例: 1,3,5) またはハイフンで範囲指定されたリスト (例: 1-5) です。値 'NONE' は、指定した述部が適用された後、データ・パーティションが残らないことを示しています。
DPNUMPRT	INTEGER	アクセスしたデータ・パーティションの実数または見積もり数を表す。
DSTSEVER	サーバー名	宛先 (配送元) サーバー。
DUPLWARN	TRUE FALSE	警告標識を複写する。
EARLYOUT	LEFT RIGHT GROUPBY NONE	EARLYOUT 標識。 LEFT は、外部表からの各行が内部表からの最大 1 行にだけ結合される必要があることを示します。RIGHT は、内部表からの各行が外部表からの最大 1 行にだけ結合される必要があることを示します。NONE は、EARLYOUT 処理が行われないことを示します。 GROUPBY は、GROUP BY 操作のために EARLYOUT 処理が許可されていることを示します。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
ENVVAR	このタイプの各行には以下のものが入ります。 • 環境変数名 • 環境変数値	オプティマイザーに影響する環境変数
ERRTOL	このタイプの各行には SQLSTATE と SQLCODE の対が入ります。	許容されるエラーのリスト。
FETCHMAX	IGNORE INTEGER	FETCH 演算子の MAXPAGES 引数の値をオーバーライドする。
GREEDY	TRUE	オプティマイザーが貪欲型アルゴリズムを使用してアクセスをプランしたことを示す。
GLOBLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE NO LOCK OBTAINED SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	パーティション表オブジェクトについてのグローバル・ロックの意図情報を表す。
GROUPBYC	TRUE FALSE	Group By 列が与えられているかどうか。
GROUPBYN	整数	比較列の数。
GROUPBYR	このタイプの各行には以下のものが入ります。 • group by 節内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。	Group By 要件。
HASHCODE	24 32	ハッシュ結合に使われるハッシュ・コードのサイズ (ビット単位)。
INNERCOL	このタイプの各行には以下のものが入ります。 • 配列内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	内部配列の列。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
INPUTXID	コンテキスト・ノード ID	INPUTXID は、XSCAN 演算子によって使用される入力コンテキスト・ノードを示します。
ISCANMAX	IGNORE INTEGER	ISCAN 演算子の MAXPAGES 引数の値をオーバーライドする。
JN INPUT	INNER OUTER	演算子が内部または外部のどちらの結合を送る演算子であるかを示す。
LISTENER	TRUE FALSE	Listener 表キューの標識。
MAXPAGES	ALL NONE INTEGER	プリフェッチのための最大ページ数。
MAXRIDS	NONE INTEGER	個々のリスト・プリフェッチ要求に組み込まれる最大行 ID。
NUMROWS	INTEGER	ソートされるべき行の数。
ONEFETCH	TRUE FALSE	1 つの取り出し標識。
OUTERCOL	このタイプの各行には以下のものが入ります。 • 配列内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	外部配列の列。
OUTERJN	LEFT RIGHT FULL LEFT (ANTI) RIGHT (ANTI)	外部結合の標識。
PARTCOLS	列の名前	演算子のパーティション列。
PREFETCH	LIST NONE SEQUENTIAL	プリフェッチ有資格属性のタイプ。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
REOPT	ALWAYS ONCE	パラメーター・マーカー、ホスト変数、および特殊レジスターに bind-in 値を使ってステートメントを最適化します。
RMTQTEXT	照会テキスト	リモート照会テキスト。
RNG_PROD	関数名	拡張索引アクセスのための範囲生成関数。
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	行ロック意図。
ROWWIDTH	INTEGER	ソートされる行の幅。
RSUFFIX	照会テキスト	リモート SQL の接尾部。
SCANDIR	FORWARD REVERSE	スキャンの方向。
SCANGRAN	INTEGER	パーティション内並列処理、パーティション内並列処理のスキャンの細分性。 SCANUNIT の単位で表される。
SCANTYPE	LOCAL PARALLEL	パーティション内並列処理、索引または表のスキャン。
SCANUNIT	ROW PAGE	パーティション内並列処理、スキャンの細分性の単位。
SHARED	TRUE	パーティション内並列処理、共用 TEMP 標識。
SLOWMAT	TRUE FALSE	低速マテリアライズ・フラグ。
SNGLPROD	TRUE FALSE	パーティション内並列処理、単一エージェントによるソートまたは一時作成。
SORTKEY	このタイプの各行には以下のものが入ります。 • キー内の列の順序値 (後に、コロンとスペースが続く) • 列の名前。 • 昇順逆順指定値 **(A)** 昇順 **(D)** 降順	ソート・キーの列。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	パーティション内並列処理、ソート・タイプ。
SRCSEVER	サーバー名	ソース (配送先) サーバー。
SPILLED	INTEGER	SORT スピル内の推定ページ数。
SQLCA	警告情報	Explain 操作中に発行された警告と理由コード。
STMTHEAP	INTEGER	ステートメントのコンパイルの開始時のステートメント・ヒープのサイズ。
STREAM	TRUE FALSE	リモート・ソースがストリーミング。
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	表ロック意図。
TEMPSIZE	INTEGER	一時表のページ・サイズ。
TQDEGREE	INTEGER	パーティション内並列処理、表キューにアクセスするサブエージェントの数。
TQMERGE	TRUE FALSE	マージする (ソート済み) 表キューの標識。
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	表キューの読み取り特性。
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	表キューの送信特性。
TQ TYPE	LOCAL	パーティション内並列処理、表キュー。
TQ_ORIGIN	ASYNCHRONY	表キューがアクセス・プランに導入された理由。

表 212. ARGUMENT_TYPE および ARGUMENT_VALUE 列の値 (続き)

ARGUMENT_TYPE の値	可能な ARGUMENT_VALUE 値	説明
TRUNCTQ	INPUT OUTPUT INPUT AND OUTPUT	切り捨てられる表キューの標識。INPUT は、表キューへの入力時に切り捨てられることを示します。 OUTPUT は、表キューからの出力時に切り捨てられることを示します。 INPUT AND OUTPUT は、表キューへの入力時と表キューからの出力時の両方に切り捨てられることを示します。
TRUNCSRT	TRUE	切り捨てソート (作成される行の数を制限する)。
UNIQUE	TRUE FALSE	固有性の標識。
UNIQKEY	このタイプの各行には以下のものが入ります。 • キー内の列の順序値 (後に、コロンとスペースが続く) • 列の名前	ユニーク・キーの列。
VOLATILE	TRUE	VOLATILE 表。
XDFOUT	DECIMAL	XDFOUT は、コンテキスト・ノードごとに XISCAN 演算子によって戻される文書数の予期値を示します。
XLOGID	SQL スキーマ名および XML データに対する索引の名前で構成される ID	XLOGID は、XISCAN 演算子のためにオプティマイザーによって選択された XML データに対する索引を示します。
XPATH	内部形式による XPATH 式および結果セット	この引数は、XSCAN 演算子による XPATH 式の評価を示します。
XPHYID	SQL スキーマ名および物理 XML データに対する索引の名前で構成される ID	XPHYID は、XISCAN 演算子によって使用される XML データに対する索引に関連した物理索引を示します。

EXPLAIN_DIAGNOSTIC 表

EXPLAIN_DIAGNOSTIC 表には、EXPLAIN_STATEMENT 表内の EXPLAIN されたステートメントの特定のインスタンスについて生成された、各診断メッセージの項目が入ります。

EXPLAIN_GET_MSGS 表関数は、EXPLAIN_DIAGNOSTIC および EXPLAIN_DIAGNOSTIC_DATA Explain 表を照会し、定様式メッセージを戻します。

表 213. EXPLAIN_DIAGNOSTIC 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	PK、FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	PK、FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	PK、FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	PK、FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	PK、FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	PK、FK	この行に関連する Explain 情報のレベル。有効な値は次のとおりです。 **O** 元のテキスト (ユーザーが入力したもの) **P** PLAN SELECTION
STMTNO	INTEGER	いいえ	PK、FK	この Explain 情報に関連したパッケージ内のステートメントの番号。動的 Explain SQL ステートメントの場合は 1。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS システム・カタログ・ビューで使用されているものと同じです。
SECTNO	INTEGER	いいえ	PK、FK	パッケージ内のセクションのうち、この SQL ステートメントを収めたもののセクション番号です。動的 Explain SQL ステートメントの場合、これは、実行時にこのステートメントのセクションを保持するのに使用されるセクション番号です。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS システム・カタログ・ビューで使用されているものと同じです。
DIAGNOSTIC_ID	INTEGER	いいえ	PK	EXPLAIN_STATEMENT 表内の特定のステートメントのインスタンスの診断 ID。

表 213. EXPLAIN_DIAGNOSTIC 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
CODE	INTEGER	いいえ	いいえ	各診断メッセージに割り当てられた固有の番号。この番号は、メッセージ API が診断メッセージのフルテキストを取り出すために使用します。

EXPLAIN_DIAGNOSTIC_DATA 表

EXPLAIN_DIAGNOSTIC_DATA 表には、EXPLAIN_DIAGNOSTIC 表に記録されている特定の診断メッセージのためのメッセージ・トークンが入ります。メッセージ・トークンは、メッセージを生成した SQL ステートメントの実行に固有の追加情報を提供します。

EXPLAIN_GET_MSGS 表関数は、EXPLAIN_DIAGNOSTIC および EXPLAIN_DIAGNOSTIC_DATA Explain 表を照会し、定様式メッセージを戻します。

表 214. EXPLAIN_DIAGNOSTIC_DATA 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	FK	この行に関連する Explain 情報のレベル。有効な値は次のとおりです。 **O** 元のテキスト (ユーザーが入力したもの) **P** PLAN SELECTION
STMTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。動的 Explain SQL ステートメントの場合は 1。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS システム・カタログ・ビューで使用されているものと同じです。
SECTNO	INTEGER	いいえ	FK	パッケージ内のセクションのうち、この SQL ステートメントを収めたもののセクション番号です。動的 Explain SQL ステートメントの場合、これは、実行時にこのステートメントのセクションを保持するのに使用されるセクション番号です。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS システム・カタログ・ビューで使用されているものと同じです。
DIAGNOSTIC_ID	INTEGER	いいえ	PK	EXPLAIN_STATEMENT 表内の特定のステートメントのインスタンスの診断 ID。
ORDINAL	INTEGER	いいえ	いいえ	フル・メッセージ・テキストの中のトークンの位置。
TOKEN	VARCHAR(1000)	はい	いいえ	フル・メッセージ・テキストに挿入されるメッセージ・トークン。切り捨てられる可能性があります。

表 214. EXPLAIN_DIAGNOSTIC_DATA 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
TOKEN_LONG	BLOB(3M)	はい	いいえ	使用可能な場合は、詳細情報。

EXPLAIN_INSTANCE 表

EXPLAIN_INSTANCE 表は、すべての Explain 情報用の主コントロール表です。 Explain 表中のデータの各行は、この表内のある固有の 1 行に明示的にリンクされます。 EXPLAIN_INSTANCE 表は、Explain 対象の SQL ステートメントのソースに関する基本情報、および Explain 機能の環境に関する情報を提供します。

表 215. EXPLAIN_INSTANCE 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	PK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	PK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	PK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	PK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	PK	Explain 要求のソースのバージョン。
EXPLAIN_OPTION	CHAR(1)	いいえ	いいえ	この要求に関して要求された Explain 情報の内容を示します。 可能な値は以下のとおりです。 **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	いいえ	いいえ	この要求に関して Explain スナップショットが取られたかどうかを示します。 使用できる値は以下のとおりです。 **Y** Yes。1 つまたは複数の Explain スナップショットが取られ、EXPLAIN_STATEMENT 表に保管されました。正規の Explain 情報もキャプチャーされました。 **N** Explain スナップショットは取られませんでした。正規の Explain 情報がキャプチャーされました。 **O** Explain スナップショットだけが取られました。正規の Explain 情報はキャプチャーされませんでした。
DB2_VERSION	CHAR(7)	いいえ	いいえ	この Explain 要求を処理した DB2 製品のリリース番号。フォーマットは *vv.rr.m* です。ただし、 **vv** バージョン番号 **rr** リリース番号 **m** 保守リリース番号

表 215. EXPLAIN_INSTANCE 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
SQL_TYPE	CHAR(1)	いいえ	いいえ	Explain インスタンスが静的と動的のどちらの SQL に関するものであったかどうかを示します。 使用できる値は以下のとおりです。 **S** 静的 SQL **D** 動的 SQL
QUERYOPT	INTEGER	いいえ	いいえ	Explain 呼び出しの時点で SQL コンパイラーが使用する照会最適化クラスを示します。値は、Explain 中の SQL ステートメントについて SQL コンパイラーが実行したのが、どのレベルの照会最適化であるかを示します。
BLOCK	CHAR(1)	いいえ	いいえ	SQL ステートメントのコンパイル時に使用されたカーソルのブロッキング・タイプを示します。詳細については、SYSCAT.PACKAGES の BLOCK 列を参照してください。 使用できる値は以下のとおりです。 **N** ブロッキングなし **U** 確定カーソルのブロック **B** すべてのカーソルのブロック
ISOLATION	CHAR(2)	いいえ	いいえ	SQL ステートメントのコンパイル時に使用された分離レベルの種類を示します。詳細については、SYSCAT.PACKAGES の ISOLATION 列を参照してください。 使用できる値は以下のとおりです。 **RR** 反復可能読み取り **RS** 読み取り固定 **CS** カーソル固定 **UR** 非コミット読み取り
BUFFPAGE	INTEGER	いいえ	いいえ	Explain の呼び出しの時点で設定された BUFFPAGE データベース構成の値が入っています。
AVG_APPLS	INTEGER	いいえ	いいえ	Explain 呼び出し時に、 **avg_appls** データベース構成パラメーターの値が入れられます。
SORTHEAP	INTEGER	いいえ	いいえ	Explain 呼び出し時に、 **sortheap** データベース構成パラメーターの値が入れられます。
LOCKLIST	INTEGER	いいえ	いいえ	Explain 呼び出し時に、 **locklist** データベース構成パラメーターの値が入れられます。
MAXLOCKS	SMALLINT	いいえ	いいえ	Explain 呼び出し時に、 **maxlocks** データベース構成パラメーターの値が入れられます。

表 215. EXPLAIN_INSTANCE 表 (続き)： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
LOCKS_AVAIL	INTEGER	いいえ	いいえ	オプティマイザーによって各ユーザーごとに使用可能と見なされるロックの数が入っています。 (**locklist** および **maxlocks** から派生したもの。)
CPU_SPEED	DOUBLE	いいえ	いいえ	Explain 呼び出し時に、 **cpuspeed** データベース・マネージャー構成パラメーターの値が入れられます。
REMARKS	VARCHAR(254)	はい	いいえ	ユーザーが入力したコメント。
DBHEAP	INTEGER	いいえ	いいえ	Explain 呼び出し時に、 **dbheap** データベース構成パラメーターの値が入れられます。
COMM_SPEED	DOUBLE	いいえ	いいえ	Explain 呼び出し時に、 **comm_bandwidth** データベース構成パラメーターの値が入れられます。
PARALLELISM	CHAR(2)	いいえ	いいえ	使用できる値は以下のとおりです。 • N = 並列処理なし • P = パーティション内並列処理 • IP = パーティション間並列処理 • BP = パーティション内並列処理とパーティション間並列処理
DATAJOINER	CHAR(1)	いいえ	いいえ	使用できる値は以下のとおりです。 • N = 非フェデレーテッド・システム・プラン • Y = フェデレーテッド・システム・プラン

EXPLAIN_OBJECT 表

EXPLAIN_OBJECT 表は、 SQL ステートメントを満たすために生成されるアクセス・プランが必要とするデータ・オブジェクトを指定します。

表 216. EXPLAIN_OBJECT 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のセクションの番号。
OBJECT_SCHEMA	VARCHAR(128)	いいえ	いいえ	このオブジェクトが属しているスキーマ。
OBJECT_NAME	VARCHAR(128)	いいえ	いいえ	オブジェクトの名前。
OBJECT_TYPE	CHAR(2)	いいえ	いいえ	オブジェクトのタイプの記述ラベル。
CREATE_TIME	TIMESTAMP	はい	いいえ	オブジェクトの作成時刻。表関数の場合は NULL。
STATISTICS_TIME	TIMESTAMP	はい	いいえ	このオブジェクトを最後に更新した時刻。このオブジェクトに統計がない場合は NULL 値。
COLUMN_COUNT	SMALLINT	いいえ	いいえ	このオブジェクトの列数。
ROW_COUNT	INTEGER	いいえ	いいえ	このオブジェクトの行数の見積もり。
WIDTH	INTEGER	いいえ	いいえ	オブジェクトの平均幅 (バイト数)。索引の場合は -1 に設定します。
PAGES	BIGINT	いいえ	いいえ	オブジェクトがバッファー・プールで占有するページ数の見積もり。表関数には、-1 に設定します。
DISTINCT	CHAR(1)	いいえ	いいえ	オブジェクト内の行が固有かどうか (つまり、重複しているかどうか) を示します。 可能な値は以下のとおりです。 **Y** はい **N** いいえ
TABLESPACE_NAME	VARCHAR(128)	はい	いいえ	このオブジェクトが保管されている表スペースの名前。表スペースが関係していない場合は、NULL 値に設定する。

表 216. EXPLAIN_OBJECT 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
OVERHEAD	DOUBLE	いいえ	いいえ	指定された表スペースにランダム入出力を 1 回行うためのオーバーヘッドの見積合計 (ミリ秒)。コントローラー・オーバーヘッド、ディスク・シーク、および待ち時間が組み入れられます。表スペースが関係していない時は -1 に設定します。
TRANSFER_RATE	DOUBLE	いいえ	いいえ	指定の表スペースからデータ・ページを読み取るための時間の見積もり (ミリ秒単位)。表スペースが関係していない時は -1 に設定します。
PREFETCHSIZE	INTEGER	いいえ	いいえ	プリフェッチの実行時に読み取るデータ・ページの数。表関数には、-1 に設定します。
EXTENTSIZE	INTEGER	いいえ	いいえ	データ・ページを単位とするエクステント・サイズ。表スペースの中のコンテナーにこの数のページが書き込まれたら、次のコンテナーに切り替えるようになります。表関数には、-1 に設定します。
CLUSTER	DOUBLE	いいえ	いいえ	索引とのデータ・クラスタリングの程度。 >= 1 の場合は CLUSTERRATIO。 0 以上かつ 1 より小さい場合、これは CLUSTERFACTOR。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
NLEAF	BIGINT	いいえ	いいえ	この索引オブジェクトの値が占めるリーフ・ページの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
NLEVELS	INTEGER	いいえ	いいえ	この索引オブジェクトのツリー内の索引レベルの数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FULLKEYCARD	BIGINT	いいえ	いいえ	この索引オブジェクトに組み込まれる個別全キー値の数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
OVERFLOW	BIGINT	いいえ	いいえ	表内のオーバーフロー・レコードの合計数。索引、表関数について、またはこの統計が使用不可である場合は、-1 に設定します。
FIRSTKEYCARD	BIGINT	いいえ	いいえ	最初のキーの値の種類数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FIRST2KEYCARD	BIGINT	いいえ	いいえ	索引の最初の {2,3,4} 列を使用する最初のキー値の種類数。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
FIRST3KEYCARD	BIGINT	いいえ	いいえ	
FIRST4KEYCARD	BIGINT	いいえ	いいえ	
SEQUENTIAL_PAGES	BIGINT	いいえ	いいえ	索引キーの順序でディスクに存在し、それらの間に大きなギャップがないか、わずかなギャップしかないリーフ・ページの数。 表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。

表 216. EXPLAIN_OBJECT 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
DENSITY	INTEGER	いいえ	いいえ	索引によって占有されているページの範囲内の、ページ数に対する SEQUENTIAL_PAGES の比率。パーセントで表現される (0 から 100 の整数)。表、表関数について、またはこの統計が使用不可の場合は、-1 に設定します。
STATS_SRC	CHAR(1)	いいえ	いいえ	統計のソースを示します。単一のノードからの場合は 1 に設定します。
AVERAGE_SEQUENCE_GAP	DOUBLE	いいえ	いいえ	シーケンス間のギャップ。
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	いいえ	いいえ	索引を使用してフェッチするときの、シーケンス間のギャップ。
AVERAGE_SEQUENCE_PAGES	DOUBLE	いいえ	いいえ	順次にアクセスできる索引ページの平均数。
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	いいえ	いいえ	索引を使用してフェッチする際の、順次にアクセスできる表ページの平均数。
AVERAGE_RANDOM_PAGES	DOUBLE	いいえ	いいえ	順次ページ・アクセスの間のランダム索引ページの平均数。
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	いいえ	いいえ	索引を使用してフェッチする際の、順次ページ・アクセスの間のランダム表ページの平均数。
NUMRIDS	BIGINT	いいえ	いいえ	索引内の行 ID の合計数。
NUMRIDS_DELETED	BIGINT	いいえ	いいえ	索引内の疑似削除された行 ID の合計数。
NUM_EMPTY_LEAFS	BIGINT	いいえ	いいえ	索引内の空白リーフ・ページの合計数。
ACTIVE_BLOCKS	BIGINT	いいえ	いいえ	表内のアクティブなマルチディメンション・クラスタリング (MDC) ブロックの合計数。
NUM_DATA_PART	INTEGER	いいえ	いいえ	パーティション表のデータ・パーティションの数。表がパーティション化されていない場合、1 に設定します。

表 217. 可能な OBJECT_TYPE 値

値	説明
IX	索引
NK	ニックネーム
RX	RCT 索引
DP	データ・パーティション表
TA	表
TF	表関数
+A	コンパイラー参照の別名
+C	コンパイラー参照の制約
+F	コンパイラー参照の関数
+G	コンパイラー参照のトリガー
+N	コンパイラー参照のニックネーム

表 217. 可能な OBJECT_TYPE 値 (続き)

値	説明
+T	コンパイラー参照の表
+V	コンパイラー参照のビュー
XI	論理 XML 索引
PI	物理 XML 索引

EXPLAIN_OPERATOR 表

EXPLAIN_OPERATOR 表は、照会コンパイラーが照会ステートメントを満たすために必要とするすべての演算子を格納します。

表 218. EXPLAIN_OPERATOR 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のセクションの番号。
OPERATOR_ID	INTEGER	いいえ	いいえ	この照会内の演算子の固有な ID。
OPERATOR_TYPE	CHAR(6)	いいえ	いいえ	演算子のタイプの記述ラベル。
TOTAL_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの実行にかかる合計コスト (timeron 単位) の累積の見積もり。
IO_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの実行にかかる入出力コスト (データ・ページの入出力単位) の累積の見積もり。
CPU_COST	DOUBLE	いいえ	いいえ	選択したアクセス・プランをこの演算子まで実行した場合にかかる CPU コスト (命令数) の累積の見積もり。
FIRST_ROW_COST	DOUBLE	いいえ	いいえ	この演算子までのアクセス・プランで 1 行目を取り出した場合にかかる累積合計 (timeron 数) の見積もり。この値には、必要な初期オーバーヘッドが入ります。
RE_TOTAL_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの次の行の取り出しにかかるコスト (timeron 単位) の累積の見積もり。
RE_IO_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの次の行の取り出しにかかる入出力コスト (データ・ページの入出力単位) の累積の見積もり。

表 218. EXPLAIN_OPERATOR 表 (続き)： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
RE_CPU_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの次の行のフェッチにかかる CPU コスト (命令数) の累積の見積もり。
COMM_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの実行にかかる通信コスト (TCP/IP フレーム単位) の累積の見積もり。
FIRST_COMM_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したアクセス・プランの最初の行の取り出しにかかる通信コスト (TCP/IP フレーム単位) の累積の見積もり。この値には、必要な初期オーバーヘッドが入ります。
BUFFERS	DOUBLE	いいえ	いいえ	この演算子とその入力に必要なバッファーの見積もり。
REMOTE_TOTAL_COST	DOUBLE	いいえ	いいえ	リモート・データベース操作の実行にかかる合計コスト (timeron 単位) の累積の見積もり。
REMOTE_COMM_COST	DOUBLE	いいえ	いいえ	この演算子に至るまで (この演算子を含む) の、選択したリモート・アクセス・プランの実行にかかる通信コストの累積の見積もり。

表 219. OPERATOR_TYPE の値

値	説明
DELETE	削除
EISCAN	拡張索引スキャン
FETCH	フェッチ
FILTER	行フィルター
GENROW	生成行
GRPBY	グループ化
HSJOIN	ハッシュ結合
INSERT	挿入
IXAND	動的ビットマップ索引 ANDing
IXSCAN	リレーショナル索引スキャン
MSJOIN	マージ・スキャン結合
NLJOIN	ネスト・ループの結合
RETURN	結果
RIDSCN	行 ID (RID) スキャン
RPD	リモート・プッシュダウン
SHIP	リモート・システムへの照会の配送
SORT	ソート
TBSCAN	表スキャン
TEMP	一時表構造
TQ	テーブル・キュー

表 219. OPERATOR_TYPE の値 (続き)

値	説明
UNION	和集合
UNIQUE	複写除去
UPDATE	更新
XISCAN	XML データに対する索引スキャン
XSCAN	XML 文書のナビゲーション・スキャン
XANDOR	XML データに対する索引の ANDing および ORing

EXPLAIN_PREDICATE 表

EXPLAIN_PREDICATE は、特定の演算子によって適用される述部を指定します。

表 220. EXPLAIN_PREDICATE 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のセクションの番号。
OPERATOR_ID	INTEGER	いいえ	いいえ	この照会内の演算子の固有な ID。
PREDICATE_ID	INTEGER	いいえ	いいえ	特定の演算子のための述部の固有な ID。 Explain ツールによって構成された演算子述部で、オプティマイザー・オブジェクトではなく、オプティマイザー・プランに存在しないものについては、「-1」の値が示されます。
HOW_APPLIED	CHAR(10)	いいえ	いいえ	特定の演算子によって述部がどのように使用されているか。

表 220. EXPLAIN_PREDICATE 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
WHEN_EVALUATED	CHAR(3)	いいえ	いいえ	この述部で使用される副照会をいつ評価するかの指示。 可能な値は以下のとおりです。 **ブランク** この述部には副照会が入っていません。 **EAA** この述部で使用される副照会は、適用時に評価されます (EAA)。つまり、指定の演算子によって行が処理されるたびに、述部が適用されるので副照会が再評価されます。 **EAO** この述部で使用される副照会は、オープン時に評価されます (EAO)。つまり、指定の演算子に対して副照会は 1 回だけ再評価され、結果はそれぞれの行へ述部を適用する際に再利用されます。 **MUL** この述部の副照会のタイプは複数あります。
RELOP_TYPE	CHAR(2)	いいえ	いいえ	この述部で使用される関係演算子のタイプ。
SUBQUERY	CHAR(1)	いいえ	いいえ	この述部に対して、副照会からのデータ・ストリームが要求されているか。複数の副照会ストリームが要求されることがあります。 可能な値は以下のとおりです。 **N** 副照会ストリームは要求されていない **Y** 1 つ以上の副照会ストリームが要求されている
FILTER_FACTOR	DOUBLE	いいえ	いいえ	この述部が修飾する小数部の行数の見積もり。 FILTER_FACTOR が適用されない場合は「-1」の値が示されます。 Explain ツールによって構成された演算子述部で、オプティマイザー・オブジェクトではなく、オプティマイザー・プランに存在しないものについては、FILTER_FACTOR を適用できません。

表 220. EXPLAIN_PREDICATE 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
PREDICATE_TEXT	CLOB(2M)	はい	いいえ	SQL または XQuery ステートメントの内部表示から再作成された述部のテキスト。ステートメントのコンパイル時にホスト変数、特殊レジスター、またはパラメーター・マーカーの値が使われると、コメントに囲まれた述部テキストの末尾にこの値が示されます。 値が EXPLAIN_PREDICATE 表に保管されるのは、DBADM 権限をもったユーザーがステートメントを実行した場合か、または DB2 レジストリー変数 DB2_VIEW_REOPT_VALUES が YES に設定されている場合のみです。それ以外の場合、述部テキストの末尾に空のコメントが表示されます。 使用不可の場合は NULL 値。
RANGE_NUM	INTEGER	はい	いいえ	データ・パーティションの除去述部の範囲。範囲によってデータ・パーティションの除去に使用される述部のグループ化を可能にします。その他のすべての述部タイプは NULL 値。

表 221. 可能な HOW_APPLIED 値

値	説明
BSARG	各ブロックにつき 1 つの検索引数述部と評価された。
DPSTART	データ・パーティションの除去で使用される開始キー述部。
DPSTOP	データ・パーティションの除去で使用される停止キー述部。
JOIN	表の結合に使用される。
RESID	残余述部と評価された。
SARG	索引またはデータ・ページの検索引数述部と評価された。
START	開始条件として使用される。
STOP	停止条件として使用される。

表 222. 可能な RELOP_TYPE の値

値	説明
ブランク	該当なし
EQ	等しい
GE	より大きいかまたは等しい
GT	より大きい
IN	リストされている
LE	より小さいかまたは等しい
LK	類似
LT	より小さい

表 222. 可能な RELOP_TYPE の値 (続き)

値	説明
NE	等しくない
NL	NULL 値
NN	NULL 値以外

EXPLAIN_STATEMENT 表

EXPLAIN_STATEMENT 表には、さまざまなレベルの Explain 情報に関する SQL ステートメントのテキストが入ります。この表には、ユーザーが入力した元の SQL ステートメントと、その SQL ステートメントを満たすアクセス・プランを選択するのに (オプティマイザーで) 使用されるバージョンとが保管されます。後のバージョンは、書き直されているか、 SQL コンパイラーで判別された追加の述部によって拡張されているか、またはその両方であるので、元のバージョンとはあまり類似していないことがあります。 PLAN SELECTION ステートメントは、有効な SQL ステートメントとして使用されることを目的としたものではありません。

表 223. EXPLAIN_STATEMENT 表： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	PK、FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	PK、FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	PK、FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	PK、FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	PK	この行に関連する Explain 情報のレベル。 有効な値は次のとおりです。 **O** 元のテキスト (ユーザーが入力したもの) **P** PLAN SELECTION
STMTNO	INTEGER	いいえ	PK	この Explain 情報に関連したパッケージ内のステートメントの番号。動的 Explain SQL ステートメントの場合は 1。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。
SECTNO	INTEGER	いいえ	PK	パッケージ内のセクションのうち、この SQL ステートメントを収めたもののセクション番号です。動的 Explain SQL ステートメントの場合、これは、実行時にこのステートメントのセクションを保持するのに使用されるセクション番号です。静的 SQL ステートメントの場合、この値は SYSCAT.STATEMENTS カタログ・ビューで使用されているものと同じです。

表 223. EXPLAIN_STATEMENT 表 (続き)： PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
QUERYNO	INTEGER	いいえ	いいえ	Explain 対象の SQL ステートメントの数値 ID。 CLP または CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントを除く)、デフォルト値は 1 ずつ順番に大きくなる値です。そうでない場合、デフォルト値は静的 SQL ステートメントでは STMTNO の値で、動的 SQL ステートメントでは 1 です。
QUERYTAG	CHAR(20)	いいえ	いいえ	Explain 対象の各 SQL ステートメントの ID タグ。 CLP を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLP」です。 CLI を介して発行された動的 SQL ステートメントの場合 (EXPLAIN SQL ステートメントは除く)、デフォルト値は「CLI」です。それ以外の場合、使用されるデフォルト値はブランクです。
STATEMENT_TYPE	CHAR(2)	いいえ	いいえ	Explain 対象の照会のタイプの記述ラベル。 使用できる値は以下のとおりです。 **CL** 呼び出し **CP** コンパウンド SQL (動的) **D** 削除 **DC** カーソルの現在位置の削除 **I** 挿入 **M** マージ **S** 選択 **SI** 保全性の設定または表のリフレッシュ **U** 更新 **UC** カーソルの現在位置の更新
UPDATABLE	CHAR(1)	いいえ	いいえ	このステートメントが更新可能であると見なされるかどうかを示します。これは特に、潜在的に更新可能であると見なされる可能性のある SELECT ステートメントに関係しています。 使用できる値は以下のとおりです。 **' '** 該当しない (ブランク) **N** いいえ **Y** はい

表 223. EXPLAIN_STATEMENT 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
DELETABLE	CHAR(1)	いいえ	いいえ	このステートメントが削除可能であると見なされるかどうかを示します。これは特に、潜在的に削除可能であると見なされる可能性のある SELECT ステートメントに関係しています。 使用できる値は以下のとおりです。 ' ' 該当しない (ブランク) **N** いいえ **Y** はい
TOTAL_COST	DOUBLE	いいえ	いいえ	このステートメントについて選択されたアクセス・プランの実行のための合計コストの見積もり (timeron 単位)。 EXPLAIN_LEVEL が *O* (オリジナル・テキスト) の場合は、この時点で選択されているアクセス・プランがないため、ゼロに設定されます。
STATEMENT_TEXT	CLOB(2M)	いいえ	いいえ	Explain 対象の SQL ステートメントのテキスト、またはその一部。 Explain 機能のプラン選択レベルで表示されるテキストは、内部表記から再構成されたものであり、本質的に SQL ステートメントに類似したものです。再構成されたステートメントは、正しい SQL 構文に必ず準拠しているとは限りません。
SNAPSHOT	BLOB(10M)	はい	いいえ	示されている Explain_Level での、この SQL ステートメントの内部表記のスナップショット。 この列は、DB2 Visual Explain で使用することを意図したものです。 EXPLAIN_LEVEL が *0* (元のステートメント) の場合は、ステートメントのこの特定バージョンがキャプチャーされた時点でアクセス・プランが選択されていないため、この列は NULL 値に設定されます。
QUERY_DEGREE	INTEGER	いいえ	いいえ	Explain の呼び出し時のパーティション内並列処理の度合い。元のステートメントの場合、ここには指定された度合いのパーティション内並列処理が入ります。 PLAN SELECTION の場合、ここには使用のプランに応じて生成されたパーティション内並列処理の度合いが入ります。

EXPLAIN_STREAM 表

EXPLAIN_STREAM 表は、個々の演算子とデータ・オブジェクトの間の、入出力データ・ストリームを表します。データ・オブジェクト自体は、EXPLAIN_OBJECT 表に示されています。データ・ストリームに関連する演算子は、EXPLAIN_OPERATOR 表にあります。

表 224. EXPLAIN_STREAM 表: PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
EXPLAIN_REQUESTER	VARCHAR(128)	いいえ	FK	この Explain 要求を開始した許可 ID。
EXPLAIN_TIME	TIMESTAMP	いいえ	FK	Explain 要求の開始時刻。
SOURCE_NAME	VARCHAR(128)	いいえ	FK	動的ステートメントに Explain 要求を出したときに実行していたパッケージの名前、または静的 SQL に Explain 要求を出したときのソース・ファイルの名前。
SOURCE_SCHEMA	VARCHAR(128)	いいえ	FK	Explain 要求のソースのスキーマ、または修飾子。
SOURCE_VERSION	VARCHAR(64)	いいえ	FK	Explain 要求のソースのバージョン。
EXPLAIN_LEVEL	CHAR(1)	いいえ	FK	この行に関連する Explain 情報のレベル。
STMTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のステートメントの番号。
SECTNO	INTEGER	いいえ	FK	この Explain 情報に関連したパッケージ内のセクションの番号。
STREAM_ID	INTEGER	いいえ	いいえ	このデータ・ストリームに対する特定の演算子のユニーク ID。
SOURCE_TYPE	CHAR(1)	いいえ	いいえ	このデータ・ストリームのソースを示します。 **O** 演算子 **D** データ・オブジェクト
SOURCE_ID	SMALLINT	いいえ	いいえ	このデータ・ストリームのソースである照会内の、演算子に対するユニーク ID。SOURCE_TYPE が「D」の場合は -1 に設定されます。
TARGET_TYPE	CHAR(1)	いいえ	いいえ	このデータ・ストリームのターゲットを示します。 **O** 演算子 **D** データ・オブジェクト
TARGET_ID	SMALLINT	いいえ	いいえ	このデータ・ストリームのターゲットである照会内の、演算子に対するユニーク ID。TARGET_TYPE が「D」の場合は -1 に設定されます。
OBJECT_SCHEMA	VARCHAR(128)	はい	いいえ	影響を受けるデータ・オブジェクトが属するスキーマ。 SOURCE_TYPE および TARGET_TYPE が共に「O」である場合、NULL に設定します。

表 224. EXPLAIN_STREAM 表 (続き): PK は、その列が主キーの一部であることを意味します。 FK は、その列が外部キーの一部であることを意味します。

列名	データ・タイプ	NULL 可能	キー?	説明
OBJECT_NAME	VARCHAR(128)	はい	いいえ	データ・ストリームのサブジェクトであるオブジェクトの名前。 SOURCE_TYPE および TARGET_TYPE が共に「O」である場合、NULL に設定します。
STREAM_COUNT	DOUBLE	いいえ	いいえ	データ・ストリームのカーディナリティー推定値。
COLUMN_COUNT	SMALLINT	いいえ	いいえ	データ・ストリーム内の列数。
PREDICATE_ID	INTEGER	いいえ	いいえ	ストリームが述部に対する副照会の一部である場合、述部 ID が反映される。それ以外は列は -1 に設定される。
COLUMN_NAMES	CLOB(2M)	はい	いいえ	この列には、このストリームに関連した列の名前や配列情報が入っています。 名前は以下のフォーマットに従います。 `NAME1(A)+NAME2(D)+NAME3+NAME4` ここで、*(A)* は昇順の列、 *(D)* は降順の列を示し、配列情報がないものは、列が配列されていないか、配列が関係ないかのいずれかを示します。
PMID	SMALLINT	いいえ	いいえ	分散マップ ID。
SINGLE_NODE	CHAR(5)	はい	いいえ	このデータ・ストリームが、単一または複数のデータベース・パーティションのどちらにあるかを示します。 **MULT** 複数のデータベース・パーティションにある **COOR** コーディネーター・パーティションにある **HASH** ハッシュを使用して指定される **RID** 行 ID を使用して指定される **FUNC** 関数を使用して指定される (HASHEDVALUE() または DBPARTITIONNUM()) **CORR** 相関値を使用して指定される **Numberic** 事前に決められた単一データベース・パーティションに指定される
PARTITION_COLUMNS	CLOB(2M)	はい	いいえ	データ・ストリームが配分される列のリスト。
SEQUENCE_SIZES	CLOB(2M)	はい	いいえ	データ・ストリーム内の XML 列に対して予期されるシーケンス・サイズをリストします。非 XML 列については「NA」(適用外) を示します。 データ・ストリーム内に 1 つも XML 列がない場合には、NULL に設定されます。

付録 J. Explain レジスター値

以下は CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値の相互作用、また PREP および BIND コマンドとの相互作用の説明です。

動的 SQL で CURRENT EXPLAIN MODE および CURRENT EXPLAIN SNAPSHOT 特殊レジスターの値には以下の相互作用があります。

表 225. Explain 特殊レジスターの値の相互作用 (動的 SQL)

	EXPLAIN MODE の値					
EXPLAIN SNAPSHOT の値	**NO**	**YES**	**EXPLAIN**	**REOPT**	**RECOMMEND INDEXES**	**EVALUATE INDEXES**
NO	• 戻された照会の結果	• データを追加された Explain 表 • 戻された照会の結果	• データを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • 戻された照会の結果	• データを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 評価された索引
YES	• とられた Explain スナップショット • 戻された照会の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻された照会の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • とられた Explain スナップショット • 戻された照会の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 評価された索引

© Copyright IBM Corp. 1993, 2010

表 225. Explain 特殊レジスターの値の相互作用 (動的 SQL) (続き)

EXPLAIN SNAPSHOT の値	EXPLAIN MODE の値					
	NO	YES	EXPLAIN	REOPT	RECOMMEND INDEXES	EVALUATE INDEXES
EXPLAIN	• とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • とられた Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 評価された索引
REOPT	• 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻された照会の結果	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻された照会の結果	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻された照会の結果	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果 • 推奨された索引	• データを追加された Explain 表 • 実行時にステートメントが再最適化の対象となったときにとられる Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的または追加バインド・ステートメント) の結果 • 評価された索引

CURRENT EXPLAIN MODE 特殊レジスターは、動的 SQL に対して以下のような方法で EXPLAIN BIND オプションと相互作用します。

表 226. EXPLAIN BIND オプションと CURRENT EXPLAIN MODE の相互作用

EXPLAIN MODE の値	EXPLAIN BIND オプションの値			
	NO	YES	REOPT	ALL
NO	• 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果
YES	• 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻された照会の結果
EXPLAIN	• 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果

表 226. EXPLAIN BIND オプションと CURRENT EXPLAIN MODE の相互作用 (続き)

EXPLAIN MODE の値	EXPLAIN BIND オプションの値			
	NO	**YES**	**REOPT**	**ALL**
REOPT	• 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻された照会の結果
RECOMMEND INDEXES	• 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 推奨索引

表 226. EXPLAIN BIND オプションと CURRENT EXPLAIN MODE の相互作用 (続き)

EXPLAIN MODE の値	EXPLAIN BIND オプションの値			
	NO	YES	REOPT	ALL
EVALUATE INDEXES	• 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価	• 実行時にステートメントが再最適化の対象となったときに静的 SQL のデータを追加される Explain 表。 • 実行時にステートメントが再最適化の対象となったときに動的 SQL のデータを追加される Explain 表。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価	• 静的 SQL のデータを追加された Explain 表 • 動的 SQL のデータを追加された Explain 表 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果 • 索引の評価

CURRENT EXPLAIN SNAPSHOT 特殊レジスターは、動的 SQL に関して次ページのような方法で EXPLSNAP BIND オプションと相互作用します。

表 227. EXPLSNAP BIND オプションと CURRENT EXPLAIN SNAPSHOT の相互作用

EXPLAIN SNAPSHOT の値	EXPLSNAP BIND オプションの値			
	NO	YES	REOPT	ALL
NO	• 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻された照会の結果

表 227. EXPLSNAP BIND オプションと CURRENT EXPLAIN SNAPSHOT の相互作用 (続き)

EXPLAIN SNAPSHOT の値	EXPLSNAP BIND オプションの値			
	NO	**YES**	**REOPT**	**ALL**
YES	• とられた動的 SQL の Explain スナップショット • 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻された照会の結果
EXPLAIN	• とられた動的 SQL の Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻されなかった照会 (実行されなかった動的ステートメント) の結果	• とられた静的 SQL の Explain スナップショット • とられた動的 SQL の Explain スナップショット • 戻されなかった照会 (実行されなかった動的ステートメント) の結果
REOPT	• 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果	• 実行時にステートメントが再最適化の対象となったときにとられた静的 SQL の Explain スナップショット。 • 実行時にステートメントが再最適化の対象となったときにとられた動的 SQL の Explain スナップショット。 • 戻された照会の結果

付録 K. 例外表

例外表は、IMMEDIATE CHECKED オプションを指定した SET INTEGRITY ステートメントを使用して検査する対象として指定された表の定義を模倣して作成されたユーザー作成の表です。例外表は、検査対象の表の行のうち制約に違反しているもののコピーを保管するのに使用されます。

ロード・ユーティリティーによって使用される例外表はここで説明されているものと同じなので、それらは SET INTEGRITY ステートメントでの検査中に再使用できます。

例外表の作成規則

例外表を作成する際の規則は、次のとおりです。

- 表がセキュリティー・ポリシーによって保護される場合、例外表も同じセキュリティー・ポリシーによって保護されることが必要です。
- 例外表の最初の『n』列は、検査対象の表の列と同じです。名前、データ・タイプ、および長さなど、すべての列属性が同じでなければなりません。保護されている列については、列を保護するセキュリティー・ラベルが両方の表で同じであることが必要です。
- 例外表のすべての列は、制約やトリガーに束縛されないようにする必要があります。制約には、参照保全、チェック制約、さらには挿入時にエラーの原因となるユニーク索引の制約が付帯します。
- 例外表の第『(n+1)』列は、オプションの TIMESTAMP 列です。これは、データを検査するための SET INTEGRITY を発行する前に例外表内の行が削除されなかった場合に、同じ表の SET INTEGRITY ステートメントによる検査の連続呼び出しを検出するのに使用します。
- 第『(n+2)』列は、CLOB(32K) タイプまたはそれより大きいタイプでなければなりません。この列は、行内のデータが違反している制約の名前を示すために使用されるもので、オプションではありますが、なるべく使用するようにしてください。この列を用意しなかった場合 (たとえば、元の表の列数がすでに可能な最大値になっていた場合にはそれが可能)、制約違反が検出された行だけがコピーされます。
- (n+1) 列と (n+2) 列の両方を備えた例外表を作成する必要があります。
- 上記の追加列の特定の名前には制約がありません。しかし、タイプの指定は、正確でなければなりません。
- それ以外の列は使用できません。
- 生成される列 (IDENTITY プロパティーも含む) が元の表にある場合は、例外表の対応する列に、生成されるプロパティーを指定しないでください。
- データのチェックのために SET INTEGRITY ステートメントを呼び出すユーザーは、例外表に対して INSERT 特権を保持している必要があります。
- 例外表は、データ・パーティション表、範囲クラスター表、またはデタッチされた表であってはなりません。
- 例外表は、マテリアライズ照会表またはステージング表であってはなりません。

© Copyright IBM Corp. 1993, 2010

- 例外表には、従属する REFRESH IMMEDIATE マテリアライズ照会表、または従属する PROPAGATE IMMEDIATE ステージング表があってはなりません。

「メッセージ」列の情報の構造は次のとおりです。

表 228. 例外表のメッセージ列の構造

フィールド番号	内容	サイズ	コメント
1	制約違反の数	5 バイト	先頭に '0' を付加して右揃え
2	最初の制約違反の種類	1 バイト	'K' - チェック制約違反 'F' - 外部キー違反 'G' - 生成列違反 'I' - ユニーク索引違反 [a] 'D' - 削除規則 : カスケード違反 'P' - データ・パーティション違反 'S' - 無効な行セキュリティー・ラベル 'L' - DB2 LBAC 書き込み規則
3	制約/列[b] /索引 ID[c] の長さ	5 バイト	先頭に '0' を付加して右揃え
4	制約名/列名[b]/索引 ID[c]	直前のフィールドで指定される長さ	
5	区切り記号	3 バイト	<space><colon><space>
6	次の制約違反の種類	1 バイト	'K' - チェック制約違反 'F' - 外部キー違反 'G' - 生成列違反 'I' - ユニーク索引違反 'D' - 削除規則 : カスケード違反 'P' - データ・パーティション違反 'S' - 無効な行セキュリティー・ラベル 'L' - DB2 LBAC 書き込み規則
7	制約/列/索引 ID の長さ	5 バイト	先頭に '0' を付加して右揃え
8	制約名/列名/索引 ID	直前のフィールドで指定される長さ	
.....	.....	.....	違反ごとにフィールド 5 から 8 を繰り返す。

- [a] SET INTEGRITY ステートメントを使用した検査の際には、それがアタッチ操作の後でない限り、ユニーク索引違反は起こりません。しかし、FOR EXCEPTION オプションを選択した場合に LOAD を実行すると、それが報告されます。一方、LOAD はチェック制約、生成列、外部キー、カスケード削除、またはデータ・パーティションに関する違反を例外表に報告しません。
- [b] 生成列の式をカタログ・ビューから取り出すには、select ステートメントを使用します。たとえば、フィールド 4 が MYSCHEMA.MYTABLE.GEN_1 の場合、SELECT SUBSTR(TEXT, 1, 50) FROM SYSCAT.COLUMNS WHERE TABSCHEMA='MYSCHEMA' AND TABNAME='MYNAME' AND COLNAME='GEN_1'; は、式の最初の 50 バイトを "AS (<expression>)" の形式で戻します。
- [c] カタログ・ビューから索引 ID を取り出すには、 select ステートメントを使用します。たとえば、フィールド 4 が 1234 であれば、 SELECT INDSCHEMA, INDNAME FROM SYSCAT.INDEXES WHERE IID=1234 となります。

例外表での行の処理

例外表の情報は、さまざまな方法で処理できます。データを訂正して、行を元の表に再挿入できます。

元の表に INSERT トリガーがないなら、例外表に対する副照会の入った INSERT ステートメントを発行することによって、修正した行を転送します。

INSERT トリガーがあり、トリガーを起動することなく例外表からの修正済みの行によるロード操作を完了したい場合は、次のような方法があります。

- 目的に合わせて明示的に定義された列において、値に応じて起動されるように INSERT トリガーを設計します。
- 例外表からのデータをアンロードして、ロード・ユーティリティーを使用してそれを付加します。その場合、データを再検査するにあたっては、制約検査の対象は付加された行だけに限定されないことに注意してください。
- 関連するシステム・カタログ・ビューのトリガー定義テキストを保存します。次に、INSERT トリガーをドロップし、INSERT を使用して修正された行を例外表から転送します。最後に、保存したトリガー定義を使ってトリガーを再作成します。

例外表から行を挿入する際にトリガーが起動しないように防止する明示的な機能はありません。

ユニーク索引の違反に対しては、行ごとに 1 つの違反しか報告されません。

LONG ストリングまたは LOB データ・タイプの値が表の中に入っている場合、ユニーク索引違反があっても、その値は例外表に挿入されません。

例外表の照会

例外表のメッセージ列の構造は、前述の制約の名前、長さ、および区切り文字を連結したリストです。この情報は、照会可能です。

たとえば、すべての違反のリストを取得し、各行ごとに制約名だけを繰り返すとします。元の表 T1 に C1 と C2 という 2 つの列があるとします。また、対応する例外表 E1 には、T1 のものと対応する列 C1 および C2 があり、さらにメッセージ列 MSGCOL があると想定します。以下の照会では再帰を使用して、行ごとに 1 つの制約名を示します (複数の違反がある行は繰り返します)。

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
 (SELECT C1, C2, MSGCOL,
     CHAR(SUBSTR(MSGCOL, 12,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
     1,
     15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
    FROM E1
  UNION ALL
  SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, J+6,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV;
```

特定の制約に違反したすべての行のリストを作成するには、前述の照会を次のように拡張します。

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, I, J) AS
 (SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, 12,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
      1,
      15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
    FROM E1
  UNION ALL
  SELECT C1, C2, MSGCOL,
      CHAR(SUBSTR(MSGCOL, J+6,
                 INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
      I+1,
      J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
    FROM IV
    WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTNAME = 'constraintname';
```

次の照会を使用して、すべてのチェック制約違反を取得できます。

```
WITH IV  (C1, C2, MSGCOL, CONSTNAME, CONSTTYPE, I, J) AS
  (SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, 12,
                   INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, 6, 1)),
       1,
       15+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,7,5)),5,0))
     FROM E1
   UNION ALL
   SELECT C1, C2, MSGCOL,
       CHAR(SUBSTR(MSGCOL, J+6,
                  INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0)))),
       CHAR(SUBSTR(MSGCOL, J, 1)),
       I+1,
       J+9+INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,J+1,5)),5,0))
     FROM IV
     WHERE I < INTEGER(DECIMAL(VARCHAR(SUBSTR(MSGCOL,1,5)),5,0))
 ) SELECT C1, C2, CONSTNAME FROM IV WHERE CONSTTYPE = 'K';
```

付録 L. ルーチンで使用可能な SQL ステートメント

下記の表は、SQL ステートメント (第 1 列で指定されている) を、指定された SQL データ・アクセス指示を使ってルーチンで実行できるかどうかを示します。NO SQL と定義されたルーチン内に実行可能な SQL ステートメントが出現すると、 SQLSTATE 38001 が戻されます。その他の実行コンテキストの場合、どのコンテキストでもサポートされていない SQL ステートメントは SQLSTATE 38003 を戻します。 CONTAINS SQL コンテキスト内で使用できないその他の SQL ステートメントの場合は SQLSTATE 38004 が戻されます。 READS SQL DATA コンテキストの場合は SQLSTATE 38002 が戻されます。 SQL ルーチンの作成中、SQL データ・アクセス指示と一致しないステートメントによって、 SQLSTATE 42985 が戻されます。

ステートメントがルーチンを呼び出す際、そのステートメントの有効な SQL データ・アクセス指示は、以下の SQL データ・アクセス指示よりも優先されます。

- 下記の表のステートメントの SQL データ・アクセス指示。
- ルーチンの作成時に指定されたルーチンの SQL データ・アクセス指示。

例えば、CALL ステートメントには CONTAINS SQL の SQL データ・アクセス指示があります。ただし、READS SQL DATA と定義されたストアード・プロシージャーが呼び出されると、 CALL ステートメントの有効 SQL データ・アクセス指示は READS SQL DATA になります。

ルーチンが SQL ステートメントを呼び出す際、そのステートメントの有効な SQL データ・アクセス指示は、ルーチンに対して宣言された SQL データ・アクセス指示を超えてはなりません。例えば、READS SQL DATA と定義された関数は、MODIFIES SQL DATA と定義されたストアード・プロシージャーを呼び出せません。

表 229. SQL ステートメントと SQL データ・アクセス指示

SQL ステートメント	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
ALTER...	N	N	N	Y
AUDIT	N	N	N	Y
BEGIN DECLARE SECTION	Y(1)	Y	Y	Y
CALL	N	Y	Y	Y
CLOSE	N	N	Y	Y
COMMENT ON	N	N	N	Y
COMMIT	N	N(4)	N(4)	N(4)
COMPOUND SQL	N	Y	Y	Y
CONNECT(2)	N	N	N	N
CREATE	N	N	N	Y
DECLARE CURSOR	Y(1)	Y	Y	Y
DECLARE GLOBAL TEMPORARY TABLE	N	N	N	Y

© Copyright IBM Corp. 1993, 2010

表 229. SQL ステートメントと SQL データ・アクセス指示 (続き)

SQL ステートメント	NO SQL	CONTAINS SQL	READS SQL DATA	MODIFIES SQL DATA
DELETE	N	N	N	Y
DESCRIBE	N	Y	Y	Y
DISCONNECT(2)	N	N	N	N
DROP ...	N	N	N	Y
END DECLARE SECTION	Y(1)	Y	Y	Y
EXECUTE	N	Y(3)	Y(3)	Y
EXECUTE IMMEDIATE	N	Y(3)	Y(3)	Y
EXPLAIN	N	N	N	Y
FETCH	N	N	Y	Y
FREE LOCATOR	N	Y	Y	Y
FLUSH EVENT MONITOR	N	N	N	Y
GRANT ...	N	N	N	Y
INCLUDE	Y(1)	Y	Y	Y
INSERT	N	N	N	Y
LOCK TABLE	N	Y	Y	Y
MERGE	N	N	N	Y
OPEN	N	N	Y (5)	Y
PREPARE	N	Y	Y	Y
REFRESH TABLE	N	N	N	Y
RELEASE CONNECTION(2)	N	N	N	N
RELEASE SAVEPOINT	N	N	N	Y
RENAME TABLE	N	N	N	Y
REVOKE ...	N	N	N	Y
ROLLBACK	N	N(4)	N(4)	N(4)
ROLLBACK TO SAVEPOINT	N	N	N	Y
SAVEPOINT	N	N	N	Y
SELECT INTO	N	N	Y (5)	Y
SET CONNECTION(2)	N	N	N	N
SET INTEGRITY	N	N	N	Y
SET 特殊レジスター	N	Y	Y	Y
SET 変数	N	Y(6)	Y(5)	Y
TRANSFER OWNERSHIP	N	N	N	Y
UPDATE	N	N	N	Y
VALUES INTO	N	N	Y	Y
WHENEVER	Y(1)	Y	Y	Y

注:

1. NO SQL オプションは SQL ステートメントを指定できないことを暗黙指定しますが、実行不能ステートメントに対する制限はありません。
2. どのルーチン実行コンテキストでも、接続管理ステートメントは使用できません。
3. これは、実行しようとするステートメントによって異なります。 EXECUTE ステートメントで指定するステートメントは、有効な個々の SQL アクセス・レベルのコンテキストで使用できるものでなければなりません。例えば、SQL アクセス・レベル READS SQL DATA が有効な場合は、 INSERT、UPDATE、または DELETE ステートメントは指定できません。
4. TO SAVEPOINT 節以外の COMMIT ステートメントおよび ROLLBACK ステートメントは、ストアード・プロシージャーで使用できます。ただし、ストアード・プロシージャーがアプリケーションから直接、またはアプリケーションからネスト・ストアード・プロシージャー呼び出しを介して間接的に呼び出される場合に限ります。 (トリガー、関数、メソッド、またはアトミック・コンパウンド・ステートメントのいずれかが、ストアード・プロシージャーに対する呼び出しチェーンにある場合、作業単位の COMMIT または ROLLBACK は使用できません。)
5. SQL アクセス・レベル READS SQL DATA が有効な場合は、 SELECT INTO ステートメント、OPEN ステートメントで参照されるカーソル、または SET 変数ステートメントの右辺の式に、一切 SQL データ変更ステートメントを組み込むことができません。
6. SQL アクセス・レベル CONTAINS SQL が有効な場合は、SET 変数ステートメントの右辺の式に、一切スカラー全選択を組み込むことができません。

付録 M. コンパイル済みステートメントから呼び出される CALL

データベースに保管されたプロシージャーを呼び出します。たとえば、プロシージャーは、データベース側で実行されて、クライアント・アプリケーションにデータを戻します。

SQL CALL ステートメントを使用するプログラムは、クライアントとサーバーの 2 つの部分で実行されるように設計されます。データベースのサーバー・プロシージャーは、クライアント・アプリケーションと同じトランザクション内で実行されます。クライアント・アプリケーションとプロシージャーが同じデータベース・パーティションにある場合、ストアード・プロシージャーはローカル側で実行されます。

注: この形式の CALL ステートメントは推奨されていません。この形式は、DB2 の以前のバージョンと互換性を保つことのみを目的として提供されています。

呼び出し

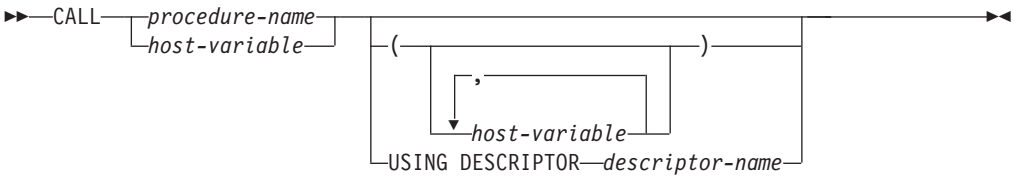
この形式の CALL ステートメントは、CALL_RESOLUTION DEFERRED を指定してプリコンパイルされたアプリケーション・プログラムに組み込むことしかできません。これは、フェデレーテッド・プロシージャーを呼び出すことはできません。これをトリガー、SQL プロシージャー、または他の非アプリケーション・コンテキストで使用することはできません。これは、動的に作成できない実行可能ステートメントです。ただし、プロシージャー名はホスト変数によって指定することができます。これは、USING DESCRIPTOR 節の使用とを組み合わせることによって、プロシージャー名とパラメーター・リストの両方を実行時に用意することができます。これで、動的準備が可能なステートメントに似た効果が得られます。

許可

*実行時*にステートメントの許可 ID がもっている特権には、以下の特権のうちの少なくとも 1 つが入っていなければなりません。

- プロシージャーに関連したパッケージに対する EXECUTE 特権。そのプロシージャーに対する EXECUTE 特権は検査されません。
- プロシージャーに関連したパッケージに対する CONTROL 特権。
- SYSADM または DBADM 権限

構文



© Copyright IBM Corp. 1993, 2010

説明

procedure-name **または** *host-variable*

呼び出すプロシージャーを指定します。プロシージャー名は、直接指定するかまたはホスト変数の中に指定できます。指定するプロシージャーは、現行サーバーに存在していなければなりません (SQLSTATE 42724)。

procedure-name を指定する場合、それは 254 バイト以下の通常 ID でなければなりません。通常 ID だけを指定できるので、ブランクや特殊文字を使用することができません。値は大文字に変換されます。小文字名、ブランク、または特殊文字を使用する必要がある場合は、名前を *host-variable* (ホスト変数) によって指定する必要があります。

host-variable を指定する場合、それは、長さ属性が 254 バイト以下の CHAR または VARCHAR 変数でなければならず、標識変数を組み入れることはできません。値は大文字に変換されません。文字ストリングは、左揃えでなければなりません。

プロシージャー名は、いくつかの形式のいずれかを使用して指定できます。

procedure-name

実行するプロシージャーの名前 (拡張子なし)。呼び出されるプロシージャーは、以下のように決定されます。

1. *procedure-name* を使用して、一致するプロシージャーがあるかどうか定義済みプロシージャー (SYSCAT.ROUTINES の) が探索されます。一致するプロシージャーは、以下の手順で決定されます。
   a. カタログ (SYSCAT.ROUTINES) から、 ROUTINENAME が指定の *procedure-name* と一致し、 ROUTINESCHEMA が SQL パス (CURRENT PATH 特殊レジスター) 内のスキーマ名であるプロシージャー (ROUTINETYPE は 'P') を見つけます。スキーマ名が明示的に指定されている場合、SQL パスは無視され、指定されたスキーマ名のプロシージャーのみが考慮されます。
   b. 次に、CALL ステートメントで指定された引数の数と同数のパラメーターを持たないプロシージャーをすべて除去します。
   c. 残りのプロシージャーから、SQL パスの最初のプロシージャーを選択します。

   プロシージャーが選択されると、DB2 は外部名によって定義されたそのプロシージャーを呼び出します。
2. マッチングするプロシージャーが見つからない場合、*procedure-name* は、プロシージャー・ライブラリーの名前と、そのライブラリー中の関数名の両方として使用されます。たとえば、*procedure-name* が `proclib` の場合、DB2 サーバーは、`proclib` という名前のプロシージャー・ライブラリーをロードし、そのライブラリー中の関数ルーチン `proclib()` を実行します。

   UNIX システムでは、プロシージャー・ライブラリーのデフォルト・ディレクトリーは `sqllib/function` です。 unfenced プロシージャーのデフォルト・ディレクトリーは、`sqllib/function/unfenced` です。

Windows ベースのシステムでは、プロシージャー・ライブラリーのデフォルト・ディレクトリーは sqllib¥function です。 unfenced プロシージャーのデフォルト・ディレクトリーは、sqllib¥function¥unfenced です。

ライブラリーまたは関数が見つからない場合、エラー (SQLSTATE 42884) になります。

procedure-library!function-name
感嘆符 (!) は、プロシージャーのライブラリー名と関数名との間の区切り文字です。たとえば、proclib!func と指定した場合は、 proclib がメモリーにロードされ、そのライブラリー中の関数 func が実行されます。これによって、1 つのプロシージャー・ライブラリーの中に複数の関数を入れることができるようになります。

procedure-name の部分で説明したように、プロシージャー・ライブラリーは、ディレクトリーに入れるか、あるいは LIBPATH 変数で指定されます。

absolute-path!function-name
absolute-path (絶対パス) には、ストアード・プロシージャー・ライブラリーまでの絶対パス名を指定します。

たとえば、UNIX システムの場合に、/u/terry/proclib!func を指定すると、プロシージャー・ライブラリーの proclib がディレクトリー /u/terry から取り出され、そのライブラリー内の関数 func が実行されます。

いずれの場合も、暗黙の絶対パスまたは明示指定の絶対パスの入ったプロシージャー名の全体の長さは、 254 バイトを超えてはなりません。

(host-variable,...)
それぞれの *host-variable* (ホスト変数) の指定は、CALL ステートメントのパラメーターです。 CALL の *n* 番目のパラメーターは、サーバーのプロシージャーの *n* 番目のパラメーターに対応します。

各 *host-variable* は、クライアントとサーバー間の双方向のデータ交換に使用されるものと見なされます。クライアントとサーバーが不要なデータをやり取りしないようにするため、クライアント・アプリケーションでは、各パラメーターに標識変数を指定して、そのパラメーターがプロシージャーへのデータの送信に使用されない場合にその標識を -1 に設定する必要があります。プロシージャーは、クライアント・アプリケーションにデータを返すために使用されないパラメーターすべてについて、標識変数を -128 に設定する必要があります。

サーバーが DB2 9.1 データベース・サーバーの場合、クライアントとサーバーの両方のプログラムで、パラメーターのデータ・タイプが一致している必要があります。

USING DESCRIPTOR *descriptor-name*
ホスト変数の有効な記述の入った SQLDA を指定します。 *n* 番目の SQLVAR エレメントは、サーバーのプロシージャーの *n* 番目のパラメーターに対応します。

CALL ステートメントが処理される前に、アプリケーションでは、 SQLDA 中の以下のフィールドを設定する必要があります。

- SQLDA に用意する SQLVAR のエレメント数を示す SQLN

- SQLDA に割り振るストレージのバイト数を示す SQLDABC
- ステートメントの処理時にその SQLDA の使用される変数の数を示す SQLD
- 変数の属性を示す SQLVAR のオカレンス渡される各基本 SQLVAR エレメントの次のフィールドは、初期化しておく必要があります。
  - SQLTYPE
  - SQLLEN
  - SQLDATA
  - SQLIND

  渡される各 2 次 SQLVAR エレメントの次のフィールドは、初期化しておく必要があります。
  - LEN.SQLLONGLEN
  - SQLDATALEN
  - SQLDATATYPE_NAME

SQLDA は、クライアントとサーバーの間の双方向のデータ交換に使用されるものと見なされます。クライアントとサーバーが不要なデータをやり取りしないようにするため、クライアント・アプリケーションでは、プロシージャーへのデータの送信にパラメーターが使用されない場合に、SQLIND フィールドを -1 に設定する必要があります。プロシージャーは、クライアント・アプリケーションにデータを返すために使用されないパラメーターすべてについて、SQLIND フィールドを -128 に設定する必要があります。

注

- ***ラージ・オブジェクト (LOB) データ・タイプの使用***

  クライアントとサーバー・アプリケーションで、 SQLDA から LOB データを指定して、SQLVAR 項目数の 2 倍を割り振る必要があります。

  LOB データ・タイプは、DB2 バージョン 2 以降のプロシージャーによってサポートされています。LOB データ・タイプは、それより下位レベルのクライアントまたはサーバーでは、サポートされていません。

- ***SQL プロシージャーからの DB2_RETURN_STATUS の検索:***

  SQL プロシージャーが RETURN ステートメントを状況値とともに正常に発行すると、この値が SQLCA の最初の SQLERRD フィールドに戻されます。 SQL プロシージャーで CALL ステートメントが発行される場合、 GET DIAGNOSTICS ステートメントを使用して DB2_RETURN_STATUS 値を検索します。 SQLSTATE がエラーを示す場合は、値は -1 になります。

- ***プロシージャーから戻される結果セット:***

  クライアント・アプリケーション・プログラムが CLI を使用して作成されている場合、結果セットをクライアント・アプリケーションに直接戻すことができます。プロシージャーは、結果セットにカーソルを宣言して、その結果セットでカーソルをオープンし、プロシージャー終了時にカーソルをオープンしたままにすることによって、結果セットを戻すよう指定します。

  プロシージャーの終了時に、

- オープンされたままのカーソルのすべてについて、結果セットがアプリケーションに戻されます。
- 複数のカーソルがオープンされたままの場合、結果セットは、それらのカーソルがオープンされた順序で戻されます。
- 未読の行だけが戻されます。たとえば、カーソルの結果セットに 500 行が入っていて、そのうち 150 行がプロシージャーの終了時にプロシージャーによって読み取られた場合、第 151 行から第 500 行までがプロシージャーに戻されます。

- ***特殊レジスターの取り扱い***

  呼び出し側の特殊レジスターの設定値は、起動時にプロシージャーに継承され、呼び出し側に戻されるとただちにリストアされます。プロシージャー内で特殊レジスターを変更してもかまいませんが、その変更で呼び出し側に影響を与えることはありません。ただし、既存のプロシージャー (パラメーター・スタイル DB2DARI で定義されているか、またはデフォルト・ライブラリーにあるもの) の場合はそうではなく、プロシージャー内で特殊レジスターに対して加えた変更は、呼び出し側の設定値になります。

- ***互換性***

  アプリケーションに組み込むこと (アプリケーションを CALL_RESOLUTION IMMEDIATE オプションを指定してプリコンパイルすることより)、または動的に準備することが可能な、新しく、推奨される形式の CALL ステートメントがあります。

例

例 1:

C において、TEAMWINS というプロシージャーを ACHIEVE ライブラリーから呼び出し、ホスト変数 HV_ARGUMENT に保管されているパラメーターをそれに渡します。

```
strcpy(HV_PROCNAME, "ACHIEVE!TEAMWINS");
CALL :HV_PROCNAME (:HV_ARGUMENT);
```

例 2:

C において、:SALARY_PROC というプロシージャーを、 INOUT_SQLDA という名前の SQLDA を使用して呼び出します。

```
struct sqlda *INOUT_SQLDA;
/* Setup code for SQLDA variables goes here */
CALL :SALARY_PROC
USING DESCRIPTOR :*INOUT_SQLDA;
```

例 3:

Java プロシージャーが、以下のステートメントを使用してデータベースに定義されています。

```
CREATE PROCEDURE PARTS_ON_HAND (IN PARTNUM INTEGER,
                                OUT COST DECIMAL(7,2),
                                OUT QUANTITY INTEGER)
   EXTERNAL NAME 'parts!onhand'
   LANGUAGE JAVA
   PARAMETER STYLE DB2GENERAL;
```

Java アプリケーションは、以下のコードを使用してこのプロシージャーを呼び出します。

```
...
CallableStatement stpCall;

String sql = "CALL PARTS_ON_HAND (?,?,?)";

stpCall = con.prepareCall( sql ) ; /* con is the connection */

stpCall.setInt( 1, variable1 ) ;
stpCall.setBigDecimal( 2, variable2 ) ;
stpCall.setInt( 3, variable3 ) ;

stpCall.registerOutParameter( 2, Types.DECIMAL, 2 ) ;
stpCall.registerOutParameter( 3, Types.INTEGER ) ;

stpCall.execute() ;

variable2 = stpCall.getBigDecimal(2) ;
variable3 = stpCall.getInt(3) ;
...
```

このアプリケーションのコード部分は、クラス *parts* の Java メソッド *onhand* を呼び出します。これは、CALL ステートメントで指定されたプロシージャー名がデータベースで検出され、外部名 'parts!onhand' を持っているためです。

付録 N. 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/

© Copyright IBM Corp. 1993, 2010

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、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 9.5 のマニュアル (PDF 形式)は、それぞれ www.ibm.com/support/docview.wss?uid=swg27009727 および www.ibm.com/support/docview.wss?uid=swg27009728 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、以下の表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 230. 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	SC88-4431-03	入手可能	2010 年 12 月
管理ルーチンおよびビュー	SC88-4435-03	入手不可	2010 年 12 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	SC88-4433-03	入手可能	2010 年 12 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	SC88-4434-03	入手可能	2010 年 12 月
コマンド・リファレンス	SC88-4432-03	入手可能	2010 年 12 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	SC88-4421-03	入手可能	2010 年 12 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	SC88-4423-03	入手可能	2010 年 12 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	SC88-4259-03	入手可能	2010 年 12 月
データベース・セキュリティー・ガイド	SC88-4418-03	入手可能	2010 年 12 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 230. 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	SC88-4427-03	入手可能	2010 年 12 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	SC88-4429-03	入手可能	2010 年 12 月
データベース・アプリケーション開発の基礎	GC88-4430-03	入手可能	2010 年 12 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GC88-4439-03	入手可能	2010 年 12 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	GI88-4109-00	入手不可	
メッセージ・リファレンス 第 2 巻	GI88-4110-00	入手不可	
マイグレーション・ガイド	GC88-4438-03	入手可能	2010 年 12 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	SC88-4419-03	入手可能	2010 年 12 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	GC88-4441-03	入手不可	2010 年 12 月
DB2 サーバー機能 概説およびインストール	GC88-4440-03	入手可能	2010 年 12 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	SC88-4436-03	入手可能	2010 年 12 月
SQL リファレンス 第 2 巻	SC88-4437-03	入手可能	2010 年 12 月

表 230. 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	SC88-4422-03	入手可能	2010 年 12 月
Text Search ガイド	SC88-4710-02	入手可能	2010 年 12 月
問題判別ガイド	GI88-4108-03	入手不可	2010 年 12 月
データベース・パフォーマンスのチューニング	SC88-4417-03	入手可能	2010 年 12 月
Visual Explain チュートリアル	SC88-4449-00	入手不可	
新機能	SC88-4445-03	入手可能	2010 年 12 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	SC88-4446-03	入手可能	2010 年 12 月
pureXML ガイド	SC88-4447-03	入手可能	2010 年 12 月
XQuery リファレンス	SC88-4448-02	入手不可	2009 年 4 月

表 231. 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	GC88-4443-03	入手可能	2010 年 12 月
DB2 Connect サーバー機能 概説およびインストール	GC88-4444-03	入手可能	2010 年 12 月
DB2 Connect ユーザーズ・ガイド	SC88-4442-03	入手可能	2010 年 12 月

表 232. 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	
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	

表 232. 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	
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	

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、 1026 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 9.8 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/v9r8/ です。

DB2 バージョン 9.7 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/v9r7/ です。

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 9.1 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、*DB2* 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 O. 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。

〒242-8502
神奈川県大和市下鶴間1623番14号
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、

© Copyright IBM Corp. 1993, 2010

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
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
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 ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml 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。

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]

あいまい参照エラー 57
アクセス・プラン
  説明 49
アスタリスク
  列名の選択 622
  列名の副選択 622
アスタリスク (*)
  選択の列名における 622
  副選択の列名における 622
  COUNT 内 314
  COUNT_BIG 内 315
値
  シーケンス 240
  定義 82
  NULL 82
値からタイム・スタンプを戻す
  TIMESTAMP 関数 519
値から月を戻す
  MONTH 関数 450
値から秒を戻す
  SECOND 関数 498
値から分を戻す
  MINUTE 関数 448
値からマイクロ秒を戻す
  MICROSECOND 関数 445
値の時間部分を戻す
  HOUR 関数 410
アプリケーション
  リクエスター 30
アプリケーション設計
  特殊文字のコード・ポイント 29
  2 バイト文字サポート (DBCS) 29
  SQL ステートメントの文字変換 29
アプリケーション・プロセス
  定義 18
暗号化
  ENCRYPT 関数 394
  GETHINT 関数 402
  XMLGROUP 関数 331
  XMLROW 関数 588
イベント・モニター
  概要 31
  名前 57
  EVENT_MON_STATE 関数 396
エラー・メッセージ
  SQLCA 定義 689
演算子
  算術 199
オブジェクト表 57
オプティマイザー
  説明 49
オペランド
  結果データ・タイプ 130
  ストリング 199
  整数 199
  浮動小数点数 199
  10 進 199
オンライン分析処理 (OLAP)
  関数 228

[カ行]

カーソル固定 (CS)
  分離レベル 20
カーソル名
  定義 57
回帰関数
  説明 323
  REGR_AVGX 323
  REGR_AVGY 323
  REGR_COUNT 323
  REGR_ICPT 323
  REGR_INTERCEPT 323
  REGR_R2 323
  REGR_SLOPE 323
  REGR_SXX 323
  REGR_SXY 323
  REGR_SYY 323
解決
  関数 181
  メソッド 190
外部関数
  概要 181
外部結合
  中間結合表 622
型付きビュー
  説明 9
  名前 57
型付き表
  名前 57
カタログ・ビュー
  概要 707, 709
  更新可能 707
  説明 18
  読み取り専用 707

© Copyright IBM Corp. 1993, 2010

カタログ・ビュー *(続き)*
ATTRIBUTES 714
AUDITPOLICIES 716
AUDITUSE 718
BUFFERPOOLDBPARTITIONS 719
BUFFERPOOLS 720
CASTFUNCTIONS 721
CHECKS 722
COLAUTH 723
COLCHECKS 724
COLDIST 725, 887
COLGROUPCOLS 726
COLGROUPDIST 727, 888
COLGROUPDISTCOUNTS 728, 889
COLGROUPS 729, 890
COLIDENTATTRIBUTES 730
COLOPTIONS 731
COLUMNS 732, 891
COLUSE 737
CONSTDEP 738
CONTEXTATTRIBUTES 739
CONTEXTS 740
DATAPARTITIONEXPRESSION 741
DATAPARTITIONS 742
DATATYPES 744
DBAUTH 747
DBPARTITIONGROUPDEF 749
DBPARTITIONGROUPS 750
EVENTMONITORS 751
EVENTS 753
EVENTTABLES 754
FULLHIERARCHIES 755
FUNCMAPOPTIONS 756
FUNCMAPPARMOPTIONS 757
FUNCMAPPINGS 758
HIERARCHIES 759
HISTOGRAMTEMPLATEBINS 760
HISTOGRAMTEMPLATES 761
HISTOGRAMTEMPLATEUSE 762
INDEXAUTH 763
INDEXCOLUSE 764
INDEXDEP 765
INDEXES 766, 892
INDEXEXPLOITRULES 772
INDEXEXTENSIONDEP 773
INDEXEXTENSIONMETHODS 774
INDEXEXTENSIONPARMS 775
INDEXEXTENSIONS 776
INDEXOPTIONS 777
INDEXXMLPATTERNS 778
KEYCOLUSE 779
NAMEMAPPINGS 780
NICKNAMES 781
PACKAGEAUTH 784
PACKAGEDEP 785
PACKAGES 787

カタログ・ビュー *(続き)*
PARTITIONMAPS 794
PASSTHRUAUTH 795
PREDICATESPECS 796
REFERENCES 797
ROLEAUTH 798
ROLES 799
ROUTINEAUTH 800
ROUTINEDEP 802
ROUTINEOPTIONS 803
ROUTINEPARMOPTIONS 804
ROUTINEPARMS 805
ROUTINES 807, 896
ROUTINESFEDERATED 816
SCHEMAAUTH 818
SCHEMATA 819
SECURITYLABELACCESS 820
SECURITYLABELCOMPONENTELEMENTS 821
SECURITYLABELCOMPONENTS 822
SECURITYLABELS 823
SECURITYPOLICIES 824
SECURITYPOLICYCOMPONENTRULES 825
SECURITYPOLICYEXEMPTIONS 826
SEQUENCEAUTH 827
SEQUENCES 828
SERVEROPTIONS 830
SERVERS 831
SERVICECLASSES 832
STATEMENTS 834
SURROGATEAUTHIDS 835
SYSDUMMY1 886
TABAUTH 836
TABCONST 838
TABDEP 839
TABDETACHEDDEP 841
TABLES 842, 897
TABLESPACES 849
TABOPTIONS 851
TBSPACEAUTH 852
THRESHOLDS 853
TRANSFORMS 855
TRIGDEP 856
TRIGGERS 857
TYPEMAPPINGS 859
USEROPTIONS 862
VARIABLEAUTH 863
VARIABLEDEP 864
VARIABLES 865
VIEWS 866
WORKACTIONS 867
WORKACTIONSETS 870
WORKCLASSES 871
WORKCLASSSETS 872
WORKLOADAUTH 873
WORKLOADCONNATTR 874
WORKLOADS 875

カタログ・ビュー *(続き)*
WRAPOPTIONS 877
WRAPPERS 878
XDBMAPGRAPHS 879
XDBMAPSHREDTREES 880
XSROBJECTAUTH 881
XSROBJECTCOMPONENTS 882
XSROBJECTDEP 883
XSROBJECTHIERARCHIES 884
XSROBJECTS 885
可変長 GRAPHIC ストリング 92
可変長文字ストリング 88
空ストリング
文字 88
GRAPHIC 92
関数
外部 181
キャスト
CAST 218
XMLCAST 223
行 181
組み込み 181
サポートされている 278
式 277
集約
説明 307
ARRAY_AGG 309
COUNT 314
MIN 322
XMLAGG 329
スカラー
説明 181, 334
ABS 335
ABSVAL 335
ACOS 336
ASCII 337
ASIN 338
ATAN 339
ATAN2 340
ATANH 341
AVG 311
BIGINT 342
BITAND 344
BITANDNOT 344
BITNOT 344
BITOR 344
BITXOR 344
BLOB 346
CARDINALITY 347
CEIL 348
CEILING 348
CHAR 349
CHARACTER_LENGTH 354
CHR 356
CLOB 357
COALESCE 358
関数 *(続き)*
スカラー *(続き)*
COLLATION_KEY_BIT 359
COMPARE_DECFLOAT 361
CONCAT 363
COS 364
COSH 365
COT 366
DATE 368
DAY 369
DAYNAME 370
DAYOFWEEK 371
DAYOFWEEK_ISO 372
DAYOFYEAR 373
DAYS 374
DBCLOB 375
DBPARTITIONNUM 376
DECFLOAT 378
DECIMAL 380
DECODE 384
DECRYPTBIN 386
DECRYPTCHAR 386
DEGREES 388
DEREF 389
DIFFERENCE 390
DIGITS 391
DOUBLE 392
DOUBLE_PRECISION 392
ENCRYPT 394
EVENT_MON_STATE 396
EXP 397
FLOAT 398
FLOOR 399
GENERATE_UNIQUE 400
GETHINT 402
GRAPHIC 403
GREATEST 405
GROUPING 318
HASHEDVALUE 406
HEX 408
HOUR 410
IDENTITY_VAL_LOCAL 411
INSERT 415
INTEGER 419
JULIAN_DAY 421
LCASE 422
LCASE (ロケール依存) 423
LEAST 424
LEFT 425
LENGTH 429
LN 431
LOCATE 432
LOG10 436
LONG_VARCHAR 437
LONG_VARGRAPHIC 438
LOWER 439

関数 *(続き)*
スカラー *(続き)*
LOWER (ロケール依存) 440
LTRIM 442
MAX 443
MAX_CARDINALITY 444
MICROSECOND 445
MIDNIGHT_SECONDS 446
MIN 447
MINUTE 448
MOD 449
MONTH 450
MONTHNAME 451
MULTIPLY_ALT 452
NODENUMBER (DBPARTITIONNUM を参照) 376
NORMALIZE_DECFLOAT 454
NULLIF 455
NVL 456
OCTET_LENGTH 457
OVERLAY 458
PARAMETER 462
PARTITION (HASHEDVALUE を参照) 406
POSITION 463
POSSTR 466
POWER 468, 472
QUANTIZE 469
QUARTER 471
RAISE_ERROR 473
RAND 475
REAL 476
REC2XML 477
REPEAT 482
REPLACE 483
RID 485
RID_BIT 485
RIGHT 487
ROUND 491
RTRIM 493
SECLABEL 494
SECLABEL_BY_NAME 495
SECLABEL_TO_CHAR 496
SECOND 498
SIGN 499
SIN 500
SINH 501
SMALLINT 502
SOUNDEX 503
SPACE 504
SQRT 505
STRIP 506
SUBSTR 507
SUBSTRING 510
TABLE_NAME 513
TABLE_SCHEMA 514
TAN 516
TANH 517
関数 *(続き)*
スカラー *(続き)*
TIME 518
TIMESTAMP 519
TIMESTAMPDIFF 526
TIMESTAMP_FORMAT 521
TIMESTAMP_ISO 525
TOTALORDER 530
TO_CHAR 528
TO_DATE 529
TRANSLATE 532
TRIM 535
TRUNC 537
TRUNCATE 537
TYPE_ID 539
TYPE_NAME 540
TYPE_SCHEMA 541
UCASE 542
UCASE (ロケール依存) 543
UPPER 544
UPPER (ロケール依存) 545
VALUE 547
VARCHAR 548
VARCHAR_FORMAT 551
VARGRAPHIC 555
WEEK 557
WEEK_ISO 558
XMLATTRIBUTES 559
XMLCOMMENT 561
XMLCONCAT 562
XMLDOCUMENT 564
XMLELEMENT 566
XMLFOREST 574
XMLGROUP 331
XMLNAMESPACES 578
XMLPARSE 580
XMLPI 583
XMLQUERY 585
XMLROW 588
XMLSERIALIZE 590
XMLTEXT 592
XMLVALIDATE 594
XMLXSROBJECTID 599
XSLTRANSFORM 600
YEAR 604
説明 277
ソース派生 181
多重定義 181
引数 277
ビット操作 344
表
XMLTABLE 605
表関数
説明 181, 604
プロシージャー 611
ユーザー定義 181, 609

関数 *(続き)*
列
回帰関数 323
説明 181, 307
ARRAY_AGG 309
AVG 311
CORR 313
CORRELATION 313
COUNT 314
COUNT_BIG 315
COVAR 317
COVARIANCE 317
MAX 320
MIN 322
REGR_AVGX 323
REGR_AVGY 323
REGR_COUNT 323
REGR_ICPT 323
REGR_INTERCEPT 323
REGR_R2 323
REGR_SLOPE 323
REGR_SXX 323
REGR_SXY 323
REGR_SYY 323
STDDEV 326
SUM 327
VARIANCE、オプション 328
VARIANCE、結果 328
VAR、オプション 328
VAR、結果 328
XMLAGG 329
OLAP (オンライン分析処理) 228
SQL 181
SQL 言語エレメント 181
Unicode データベース 334
関数シグニチャー 181
関数マッピング
オプション 901
マッピング名 57
関数名 57
間接参照操作 226
管理 SQL ルーチン
サポートされている 278
期間
概要 210
基本述部 253
逆方向タイプ・マッピング
デフォルト・マッピング 916
キャスト
構造化タイプの式をサブタイプへ 239
参照タイプ 106
データ・タイプ 218
データ・タイプ間の 106
ユーザー定義タイプ 106
XML 値 223
行
検索条件、構文 250
COUNT_BIG 関数 315
GROUP BY 節 622
HAVING 節 622
SELECT 節
構文図 622
行関数
説明 181
共通表式
再帰的 668
定義 668
SELECT ステートメント 668
共用ロック 20
許可 ID 57
許可名
説明 57
定義 57
適用される制限 57
切り捨て
数値 115
区切り文字
トークン 55
組み込み SQL アプリケーション
Java 3
組み込み関数
ストリング単位 88
説明 181
グループ
名前の定義 57
グループ化集合の結合 622
グローバル変数 179
グローバル・カタログ
説明 48
クロス集計行 622
結果データ・タイプ
オペランド 130
セット演算子 130
複数行の VALUES 節 130
CASE の結果式 130
COALESCE の引数 130
結果表
照会 621
結果列
副選択 622
結合
全選択の副選択コンポーネント 622
タイプ
全外部 622
内部 622
左外部 622
右外部 622
表
副選択節 622
権限レベル
特権を参照 11

検索条件
説明 250
評価順序 250
AND 論理演算子 250
HAVING 節
引数と規則 622
NOT 論理演算子 250
OR 論理演算子 250
WHERE 節 622
語
SQL 予約語 955
コード化スキーム
文字変換 26
コード・ページ
説明 51
属性 26
定義 26
コード・ポイント
文字変換 26
コール・レベル・インターフェース (CLI)
定義 2
更新
更新可能な特殊レジスター 145
DB2 インフォメーション・センター 1031
更新ロック 20
構造化照会言語 (SQL)
基本オペランド、割り当てと比較 115
代入 115
比較演算の概要 115
構造化タイプ
式
サブタイプへキャスト 239
説明 101
ホスト変数 57
構文
説明 xi
互換性
演算タイプに関する規則 115
規則 115
データ・タイプ 115
固定長 GRAPHIC ストリング 92
固定長文字ストリング 88
コミット
ロックの解放 18
コメント
ホスト言語、形式 55
SQL、形式 55
ご利用条件
資料の使用 1034
コロケーション
表 32
混合データ
定義 88
LIKE 述部 261

[サ行]

サーバー
名前 57
サーバー定義
説明 38
サーバー・オプション
一時 38
説明 38
サーバー・タイプ
有効なフェデレーテッド・タイプ 900
再帰的共通表式 668
再帰的照会 668
サイズの制限
ID の長さ 679
SQL 679
最適化
関数 181
メソッド 190
作業単位 (UOW)
定義 18
索引
概要 6
説明 6
名前
定義 57
サブストリング 507
サポートされている関数 278
算術
演算子 199
回帰関数 323
最大値の検出 320
式、値の加算 (SUM) 327
式から短精度整数値を戻す 502
数式からの 10 進値 380
数式からの浮動小数点値 392, 476
整数値、式から戻される 342, 419
列、値の加算 (SUM) 327
AVG 関数、演算 311
CORRELATION 関数の演算 313
COVARIANCE 関数の演算 317
STDDEV 関数 326
VARIANCE 関数の演算 328
参照タイプ
キャスト 106
説明 101
比較 115
DEREF 関数 389
シーケンス
値 240
時間
CHAR、形式変換に使用する 349
式
構造化タイプ 239
説明 199
副選択 622

式 *(続き)*
CASE 215
GROUP BY 節 622
ORDER BY 節 622
ROW CHANGE 244
SELECT 節 622
式からの整数値
INTEGER 関数 419
式からの短精度整数値
SMALLINT 関数 502
シグニチャー
関数 181
メソッド 190
時刻
時間値、式での使用 (HOUR) 410
式での時刻の使用 518
式内、TIME 関数 518
ストリング表記フォーマット 96
戻す
値からタイム・スタンプを 519
時刻に基づく値 518
日時値から分を 448
日時値からマイクロ秒を 445
秒、日時値から 498
システム・カタログ
システム表のビュー 707
システム・カタログ・ビュー
説明 18
シノニム
列名を修飾する 57
シフトイン文字
代入で切り捨てられない 115
修飾子
オブジェクト名 57
予約済み 955
修飾列名 57
集約関数
説明 307
ARRAY_AGG 309
COUNT 314
MIN 322
述部
概要 246
基本 253
照会処理 247
比較 254
BETWEEN 257
EXISTS 258
IN 259
LIKE 261
NULL 267
TYPE 268
VALIDATED 270
XMLEXISTS 273
順序
順序付け 400
順方向タイプ・マッピング
デフォルト・マッピング 902
仕様
ARRAY エレメント 225
CAST 218
OLAP 228
XMLCAST 223
照会
許可 ID 621
再帰的 668
説明 2
定義 621
フラグメント 49
例
SELECT ステートメント 668
照会の最適化
説明 49
小計行 622
条件名
SQL プロシージャー 57
照合順序
計画 51
ストリング比較の規則 115
説明 51
COLLATION_KEY_BIT スカラー関数 359
情報の暗号化解除 386
初期化
全選択 668
所有権
データベース・オブジェクト 11
資料
印刷 1026
注文 1029
概要 1025
使用に関するご利用条件 1034
PDF 1026
真理値表 250
真理値論理 250
スーパー集約行 622
スーパータイプ
ID 名 57
スーパー・グループ 622
水平相関 622
数値
スケール 695
精度 695
データ・タイプ 85
比較 115
SQL 操作での割り当て 115
スカラー関数
スカラー関数 334
通常の関数 181
DECIMAL 関数 380
VARCHAR_BIT_FORMAT 550
VARCHAR_FORMAT_BIT 554
スカラー全選択式 199

スキーマ
説明 3
名前 57
予約済み 955
スケール
数値
SQLLEN 変数によって決まる 695
データ
SQL における数値変換 115
SQL における比較 115
SQLLEN 変数によって決まる 695
ステートメント
名前 57
ストアード・プロシージャー
CALL ステートメント 1019
XSR_ADDSCHEMADOC 611
XSR_COMPLETE 612
XSR_DTD 613
XSR_ENTITY 615
XSR_REGISTER 616
XSR_UPDATE 618
ストリング
照合順序 51
代入変換規則 115
定義 26
Unicode での比較 136
ストリングからサブストリングを戻す
SUBSTR 関数 507
ストリング単位
組み込み関数 88
スペース
制御規則 55
セーブポイント
名前 57
制限
ID の長さ 679
SQL 679
整合性
点 18
整合点
データベース 18
整数
10 進数への変換のサマリー 115
ORDER BY 節 622
整数定数
説明 140
静的 SQL
説明 1
精度
数値
SQLLEN 変数 695
精度整数 DECIMAL 関数 380
制約
説明 5
名前、定義 57
Explain 表 963
セキュリティー管理者権限 (SECADM) 11
セキュリティー・ラベル (LBAC)
コンポーネント名の長さ 679
名前の長さ 679
ポリシー
名前の長さ 679
セクション
定義 11
セット演算子
結果データ・タイプ 130
EXCEPT、相違の比較 663
INTERSECT、比較における AND の役割 663
UNION、OR に対応 663
宣言
XMLNAMESPACES 578
全選択
詳細な構文 663
初期化 668
スカラー 199
反復 668
表参照 622
副照会の役割、検索条件 57
複数の演算、実行の順序 663
例 663
ORDER BY 節 622
全選択の EXCEPT 演算子 663
選択リスト
説明 622
適用規則と構文 622
表記法の規則と規約 622
ソース派生関数 181
ソート 51
結果の順序付け 115
ストリングの比較 115
相関参照
スカラー全選択での 57
ネストされた表の式での 57
副照会での 57
副選択での 622
相関名
規則 57
修飾子付き参照 57
定義 57
FROM 節
副選択の規則 622
SELECT 節の
構文図 622
総計行 622
操作
間接参照 226
代入 115
比較 115
日付/時刻 210
属性
属性名 57

[タ行]

対称スーパー集約行 622
代入
  基本 SQL 操作 115
タイプ
  構造化 101
  参照 101
  特殊
    ユーザー定義 101
タイプ保持メソッド 190
タイプ名 57
タイプ・マッピング
  名前 57
タイム・スタンプ
  ストリング表記フォーマット 96
  GENERATE_UNIQUE 400
対話式 SQL 1
大/小文字の区別
  トークン ID 55
多重定義関数
  複数の関数インスタンス 181
多重定義メソッド 190
単項演算子
  正符号 199
  負符号 199
短精度整数
  「SMALLINT データ・タイプ」を参照 85
単精度浮動小数点データ・タイプ 85
チュートリアル
  トラブルシューティング 1034
  問題判別 1034
  Visual Explain 1033
中間結果表 622
抽出
  副選択 tablesample-clause 622
長精度整数 85
直接的な相関名
  FROM 節 57
通常トークン 55
データ
  パーティション 32
データ構造
  パック 10 進数 695
データ定義言語 (DDL)
  ステートメント
    説明 1
  説明 1
データベース
  作成
    サンプル 927
  サンプルの消去 927
  パーティション 1
  分散 1
  リレーショナル 1
データベース・パーティションの互換性
  概要 138
データベース・マネージャー
  制限 679
  SQL の解釈 1
データ・ソース 36, 37
  識別 57
  説明 36
  有効なサーバー・タイプ 900
データ・ソース・オブジェクト
  説明 40
データ・タイプ
  キャスト 106
  結果列 622
  サポートされない 42
  数値
    概要 85
  パーティションの互換性 138
  バイナリー・ストリング 93
  浮動小数点
    概要 85
  プロモーション
    概要 104
  文字ストリング 88
  ユーザー定義
    概要 101
  BIGINT 85
  BLOB 93
  CHAR 88
  CLOB 88
  DATE 96
  datetime 96
  DBCLOB 92
  DECIMAL
    概要 85
  DOUBLE 85
  GRAPHIC ストリング 92
  INTEGER
    概要 85
  REAL 85
  SMALLINT 85
  SQL
    概要 82
  TIME 96
  TIMESTAMP 96
  TYPE_ID 関数 539
  TYPE_NAME 関数 540
  TYPE_SCHEMA 関数 541
  VARCHAR
    概要 88
  VARGRAPHIC 92
  XML
    値 100
データ・タイプ・マッピング
  逆方向 916
  順方向 902

データ・タイプ・マッピング *(続き)*
説明 42
定数
整数 140
浮動小数点 140
文字ストリング 140
ユーザー定義タイプの 140
10 進 140
16 進 140
GRAPHIC ストリング 140
SQL 言語エレメント 140
デクラスタリング
部分 32
トークン
区切り文字 55
大/小文字の区別 55
通常 55
SQL 言語エレメント 55
動的 SQL
SQLDA
説明 695
動的ディスパッチ
説明 190
特殊タイプ
概要 101
算術オペランド 199
定数 140
名前 57
比較
概要 115
連結 199
特殊レジスター
更新可能 145
相互作用、Explain 1005
CLIENT ACCTNG 148
CLIENT APPLNAME 149
CURRENT CLIENT_ACCTNG 148
CURRENT CLIENT_APPLNAME 149
CURRENT CLIENT_USERID 150
CURRENT CLIENT_WRKSTNNAME 151
CURRENT DATE 152
CURRENT DBPARTITIONNUM 153
CURRENT DECFLOAT ROUNDING MODE 154
CURRENT DEFAULT TRANSFORM GROUP 155
CURRENT DEGREE 156
CURRENT EXPLAIN MODE 157
CURRENT EXPLAIN SNAPSHOT 158
CURRENT FEDERATED ASYNCHRONY 159
CURRENT FUNCTION PATH 167
CURRENT IMPLICIT XMLPARSE OPTION 160
CURRENT ISOLATION 161
CURRENT LOCK TIMEOUT 162
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 163
CURRENT MDC ROLLOUT MODE 164
特殊レジスター *(続き)*
CURRENT NODE (CURRENT DBPARTITIONNUM を参照) 153
CURRENT OPTIMIZATION PROFILE 165
CURRENT PACKAGE PATH 166
CURRENT PATH 167
CURRENT QUERY OPTIMIZATION 168
CURRENT REFRESH AGE 169
CURRENT SCHEMA 170
CURRENT SERVER 171
CURRENT SQLID 170
CURRENT TIME 172
CURRENT TIMESTAMP 173
CURRENT TIMEZONE 174
CURRENT USER 175
SESSION USER 176
SQL 言語エレメント 145
SYSTEM USER 177
USER 178
特定名
定義 57
特記事項 1037
特権
階層 11
概要 11
個別の 11
所有権 (CONTROL) 11
パッケージの場合は暗黙 11
EXECUTE
関数 181
メソッド 190
トラブルシューティング
オンライン情報 1034
チュートリアル 1034
トリガー
カスケード 8
制約、相互作用 959
説明 8
相互作用 959
名前 57
名前の最大長 679
Explain 表 963

[ナ行]

長さ
LENGTH スカラー関数 429
名前
副選択の列 622
ニックネーム
説明 40
定義 57
列名を修飾する 57
FROM 節
間接的な名前 57
直接的な名前 57

ニックネーム *(続き)*
FROM 節 *(続き)*
副選択 622
SELECT 節 622
ニックネーム列オプション
説明 41
ネストされた表式 622

[ハ行]

パーティション表
ラージ・オブジェクト (LOB) 34
パーティション・データベース環境
概要 32
パーティションの互換性 138
倍精度浮動小数点データ・タイプ
概要 85
排他ロック 20
バイト長さ
データ・タイプ値 429
バイナリー・ストリング・データ・タイプ 93
バイナリー・ラージ・オブジェクト (BLOB)
スカラー関数 346
定義 93
バインディング
関数のセマンティクス 181
データの取得
最適化での役割 1
メソッドのセマンティクス 181
パス
SQL 181
パターン・マッチング
Unicode データベース 136
パッケージ
概要 11
許可 ID
動的ステートメント 57
バインディング 57
名前
概要 57
ハッシュ・パーティション 32
バッファー・プール
名前 57
パラメーター
命名規則 57
パラメーター・マーカー
動的 SQL
ホスト変数 57
反復可能読み取り (RR)
説明 20
反復全選択 668
比較 115
値とコレクション 257
2 つの述部、真の条件 253, 268
SQL 操作 115
比較述部 254
非コミット読み取り (UR)
分離レベル 20
日付
ストリング表記フォーマット 96
日付関数
DAY 369
DAYS 374
MONTH 450
YEAR 604
日付データ・タイプ
操作 210
ビット操作関数 344
ビット・データ 88
ビュー
概要 9
説明 9
別名 11
列名を修飾する 57
FROM 節
副選択での命名規則 622
FROM 節中の直接的な名前 57
FROM 節における間接的な名前 57
FROM 節における名前 622
SELECT 節における名前
構文図 622
ビュー名
定義 57
表
コロケーション 32
参照 622
式
共通 2
共通表式 668
説明 2
システム表のカタログ・ビュー 707
指定子、あいまいさを避けるための 57
修飾子付き列名 57
スカラー全選択 57
説明 5
相関名 57
名前
説明 57
FROM 節中の 622
SELECT 節の、構文図 622
ネストされた表の式 57
副照会 57
別名 11
ユニーク相関名 57
例外 1011
FROM 節
副選択での命名規則 622
FROM 節中の直接的な名前 57
FROM 節における間接的な名前 57
SAMPLE データベース 927
評価順序
式 199

表関数
説明 181, 604
表構造ファイル
サポートされるバージョン 44
表参照
ニックネーム 622
ネストされた表式 622
ビュー名 622
表名 622
別名 622
標識変数
説明 57
ホスト変数、宣言での使用 57
表スペース
説明 24
名前 57
非リレーショナル・データ・ソース
データ・タイプ・マッピングの指定 42
ファイル参照変数
BLOB 57
CLOB 57
DBCLOB 57
フェデレーテッド・サーバー 36
説明 42
フェデレーテッド・システム
概要 35
フェデレーテッド・データベース
システム・カタログ 48
説明 36
ラッパー 37
ラッパー・モジュール 37
複合列値 622
副照会
検索条件としての全選択の使用 57
HAVING 節 622
WHERE 節 622
複数行の VALUES 節
結果データ・タイプ 130
副選択
一連の操作の例 622
説明 622
例 622
FROM 節
副選択 622
プッシュダウン分析
説明 49
浮動小数点
定数 140
10 進数への変換 115
不明条件
NULL 値 250
フラット・ファイル
「表構造ファイル」も参照 44
プロシージャー
名前
概要 57
分散データベース管理システム 35
分散リレーショナル・データベース
接続 30
分離レベル
カーソル固定 20
説明 20
反復可能読み取り (RR) 20
非コミット読み取り (UR) 20
読み取り固定 (RS) 20
DELETE ステートメントにおける 668
並行トランザクション 30
別名
概要 11
説明 57
チェーニング、処理 11
別名 57
TABLE_NAME 関数 513
TABLE_SCHEMA 関数 514
ヘルプ
言語の構成 1030
SQL ステートメント 1030
変換
規則
ストリング結合演算 135
ストリングの比較 135
代入 115
比較 115
数式からの 10 進値 380
数式からの浮動小数点値 392, 476
数値の位取りと精度のまとめ 115
日時からストリング変数 115
文字ストリングからタイム・スタンプ 519
2 バイト文字ストリング 555
CHAR、変換された日時値を戻す 349
DBCS から SBCS と DBCS の混合 555
変数
グローバル 179
ホスト ID
概要 57
ホスト変数
概要 57
構文図 57
標識変数 57
BLOB 57
CLOB 57
DBCLOB 57

[マ行]

幻像読み取り行 20
マルチバイト文字サポート
特殊文字のコード・ポイント 29
未定義の参照エラー 57
命名規則
修飾子付き列の規則 57
ID 57

メソッド
外部 190
組み込み 190
タイプ保持 190
多重定義 190
動的ディスパッチ 190
ユーザー定義 190
呼び出し 237
SQL 190
SQL 言語エレメント 190
メソッド名 57
メソッド・シグニチャー 190
文字
変換 26
SQL 言語エレメント 54
文字サブタイプ 88
文字ストリング
ストリング定数 140
ストリング変換の構文 532
代入 115
データ・タイプ 88
等価の
照合シーケンスの例 115
定義 115
比較 115
ホスト変数名から戻す 532
2 バイト文字ストリング 555
BLOB ストリング表記 346
POSSTR スカラー関数 466
VARCHAR スカラー関数 548
VARGRAPHIC スカラー関数 555
文字セット
説明 51
定義 26
文字変換
ストリング結合演算での規則 135
ストリングを比較する規則 135
比較に関する規則 115
割り当てに関する規則 115
SQL ステートメントのコーディング 29
モニター
データベース・イベント 31
問題判別
チュートリアル 1034
利用できる情報 1034

[ヤ行]

ユーザー定義関数
説明 181, 277, 609
ユーザー定義タイプ
キャスト 106
構造化タイプ 101
参照タイプ 101
説明 101
ユーザー定義タイプ (続き)
特殊タイプ
説明 101
ユーザー定義タイプ (UDT)
サポートされないデータ・タイプ 42
ユーザー定義メソッド
説明 190
ユーザー・マッピング
説明 39
保管 39
有効範囲
定義 101
優先順位
概要 199
ユニーク相関名
表指定子 57
呼び出し
関数 181
メソッド 237
読み取り固定 (RS)
説明 20
予約済み
語 955
修飾子 955
スキーマ 955

[ラ行]

ラージ・オブジェクト (LOB)
説明 94
動作 34
パーティション化された表を伴う 34
ロケーター 94
ラッパー
説明 37
名前 57
ラベル
期間 210
SQL プロシージャー内のオブジェクト名 57
ラベル・ベースのアクセス制御 (LBAC)
制限 679
例外表 1011
ランタイム許可 ID 57
リテラル
説明 140
リモート
関数名 57
タイプ名 57
リモート許可名 57
リモート・カタログ情報 48
ルーチン
使用可能な SQL ステートメント 1015
プロシージャー 611
SQL 管理
サポートされている 278

例外表
構造 1011
列
あいまいな名前参照エラー 57
値セットの平均 (AVG) 311
値の加算 (SUM) 327
共分散 317
結果データ 622
差異 328
最大値の検出 320
修飾子付き列名の規則 57
スカラー全選択 57
ストリング代入の規則 115
相関 313
名前
修飾条件 57
非修飾条件 57
ORDER BY 節 622
ネストされた表の式 57
標準偏差 326
副照会 57
マッチング・ストリングでの BASIC 述部 253
マッチング・ストリングでの BETWEEN 述部 257
未定義の名前参照エラー 57
命名規則 57
列名
使用法 57
定義 57
COMMENT ON ステートメントにおける修飾 57
EXISTS 述部、マッチング・ストリング 258
GROUP BY
SELECT 節で列の制限に使用 622
GROUP BY でのグループ化列名 622
HAVING
SELECT 節で列の制限に使用 622
HAVING 節
検索名の規則 622
IN 述部、全選択、戻り値 259
LIKE 述部、マッチング・ストリング 261
NULL 値
結果列における 622
SELECT 節の構文図 622
WHERE 節を使用した検索 622
列オプション
説明 41
列データベース関数
説明 181
連結
演算子 199
特殊タイプ 199
ローカル・カタログ
「グローバル・カタログ」を参照 48
ロールバック
定義 18
ロケーター
変数の説明 57
ロケーター *(続き)*
ラージ・オブジェクト (LOB) 94
ロック
共用 (S) 20
更新 (U) 20
定義 18
排他 (X) 20
論理演算子
検索規則 250

[ワ行]

ワイルドカード
LIKE 述部 261

[数字]

10 進数変換 115
10 進定数 140
16 進定数
説明 140
2 バイト文字セット (DBCS)
ストリングを戻す 555
代入時に切り捨てられる文字 115

A

ABS スカラー関数 335
ABSVAL スカラー関数 335
ACOS スカラー関数
値および引数 336
説明 336
ADVISE_INDEX 表 964
ADVISE_INSTANCE 表 968
ADVISE_MQT 表 969
ADVISE_PARTITION 表 971
ADVISE_TABLE 表 973
ADVISE_WORKLOAD 表 974
ALL オプション 663
ALL 節
比較述部 254
SELECT ステートメント 622
AND 真理値表 250
ANY 節 254
ARRAY エレメント
仕様 225
ARRAY_AGG 関数 309
AS 節
ORDER BY 節 622
SELECT 節の 622
ASC 節
SELECT ステートメント 622
ASCII スカラー関数
値および引数 337
説明 337

ASIN スカラー関数
値および引数 338
説明 338
ATAN スカラー関数
値および引数 339
説明 339
ATAN2 スカラー関数
値および引数 340
説明 340
ATANH スカラー関数
値および引数 341
説明 341
AVG 集約関数 311

B

BETWEEN 述部 257
BIGINT 関数 342
BIGINT データ・タイプ 85
符号と精度 85
BITAND 関数 344
BITANDNOT 関数 344
BITNOT 関数 344
BITOR 関数 344
BITXOR 関数 344
BLAST
サポートされるバージョン 44
BLOB (バイナリー・ラージ・オブジェクト)
スカラー関数 346
説明 93

C

CALL ステートメント
コンパイル済みステートメントからの呼び出し 1019
CARDINALITY 関数 347
CASE
式 215
CASE の結果式
結果データ・タイプ 130
CAST
仕様 218
CEIL スカラー関数
値および引数 348
説明 348
CEILING スカラー関数
値および引数 348
説明 348
CHAR スカラー関数
説明 349
CHAR データ・タイプ
説明 88
CHARACTER_LENGTH スカラー関数
説明 354
CHR スカラー関数
値および引数 356
説明 356
CLI (コール・レベル・インターフェース)
定義 2
CLIENT USERID 特殊レジスター 150
CLIENT WRKSTNNAME 特殊レジスター 151
CLOB (文字ラージ・オブジェクト)
関数
値および引数 357
説明 357
データ・タイプ
説明 88
CLSCHED サンプル表 927
COALESCE 関数 358
COALESCE の引数 130
COLLATING_SEQUENCE サーバー・オプション
例 51
COLLATION_KEY_BIT スカラー関数
説明 359
COMPARE_DECFLOAT スカラー関数
説明 361
component-name
説明 57
CONCAT スカラー関数
値および引数 363
説明 363
CORRELATION 関数 313
COS スカラー関数
値および引数 364
説明 364
COSH スカラー関数
値および引数 365
説明 365
COT スカラー関数
値および引数 366
説明 366
COUNT 関数 314
COUNT_BIG 関数
値および引数 315
詳しい書式の説明 315
COVARIANCE 関数 317
CREATE SERVER ステートメント 42
CS (カーソル固定)
分離レベル 20
CUBE グループ
照会の説明 622
例 622
CURRENT CLIENT_ACCTNG 特殊レジスター 148
CURRENT CLIENT_APPLNAME 特殊レジスター 149
CURRENT CLIENT_USERID 特殊レジスター 150
CURRENT CLIENT_WRKSTNNAME 特殊レジスター 151
CURRENT DATE 特殊レジスター 152
CURRENT DBPARTITIONNUM 特殊レジスター 153
CURRENT DECFLOAT ROUNDING MODE 特殊レジスター 154

CURRENT DEFAULT TRANSFORM GROUP 特殊レジスター 155
CURRENT DEGREE 特殊レジスター
説明 156
CURRENT EXPLAIN MODE 特殊レジスター
説明 157
CURRENT EXPLAIN SNAPSHOT 特殊レジスター
説明 158
CURRENT FEDERATED ASYNCHRONY 特殊レジスター 159
CURRENT FUNCTION PATH 特殊レジスター
説明 167
CURRENT IMPLICIT XMLPARSE OPTION 特殊レジスター 160
CURRENT ISOLATION 特殊レジスター 161
CURRENT LOCK TIMEOUT 特殊レジスター 162
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスター 163
CURRENT MDC ROLLOUT MODE 特殊レジスター 164
CURRENT OPTIMIZATION PROFILE 特殊レジスター 165
CURRENT PACKAGE PATH 特殊レジスター 166
CURRENT PATH 特殊レジスター
説明 167
CURRENT QUERY OPTIMIZATION 特殊レジスター
説明 168
CURRENT REFRESH AGE 特殊レジスター
説明 169
CURRENT SCHEMA 特殊レジスター 170
CURRENT SERVER 特殊レジスター 171
CURRENT SQLID 特殊レジスター 170
CURRENT TIME 特殊レジスター 172
CURRENT TIMESTAMP 特殊レジスター 173
CURRENT TIMEZONE 特殊レジスター 174
CURRENT USER 特殊レジスター 175

D

DATALINK データ・タイプ
サポートされない 42
DATAPARTITIONNUM スカラー関数 367
DATE 関数 368
DATE データ・タイプ
概要 96
WEEK_ISO スカラー関数 558
DAY スカラー関数 369
DAYNAME スカラー関数
説明 370
DAYOFWEEK スカラー関数
説明 371
DAYOFWEEK_ISO スカラー関数
説明 372
DAYOFYEAR スカラー関数
説明 373
DAYS スカラー関数 374
DB2 for Linux, UNIX and Windows
サポートされるバージョン 44
デフォルトの逆方向タイプ・マッピング 916
DB2 for Linux, UNIX and Windows *(続き)*
デフォルトの順方向タイプ・マッピング 902
DB2 for System i
サポートされるバージョン 44
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
DB2 for VM and VSE
サポートされるバージョン 44
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
DB2 for z/OS
サポートされるバージョン 44
DB2 for z/OS and OS/390
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
DB2 インフォメーション・センター
言語 1030
更新 1031
バージョン 1030
別の言語で表示する 1030
DB2 資料の印刷方法 1029
db2nodes.cfg ファイル
DBPARTITIONNUM 関数 376
DBCLOB 関数
説明 375
DBCLOB データ・タイプ
説明 92
DBPARTITIONNUM 関数
説明 376
DDL (データ定義言語)
データ定義言語 (DDL) を参照。 1
DECFLOAT スカラー関数 378
DECIMAL スカラー関数 380
DECIMAL データ・タイプ
精度 85
符号 85
変換
浮動小数点 115
DECODE スカラー関数
説明 384
DECRYPT_BIN 関数 386
DECRYPT_CHAR 関数 386
DEGREES スカラー関数
説明 388
DEPARTMENT サンプル表 927
DEREF 関数
説明 389
descriptor-name
構文図 57
DIFFERENCE スカラー関数
説明 390
DIGITS 関数 391
DISABLE 関数マッピング・オプション
有効な設定値 901
DISTINCT キーワード
集約関数 307

DISTINCT キーワード *(続き)*
副選択ステートメント 622
DOUBLE スカラー関数 392
DOUBLE データ・タイプ
精度 85
符号 85
CHAR スカラー関数 349

E

EMPACT サンプル表 927
EMPLOYEE サンプル表 927
EMPPHOTO サンプル表 927
EMPRESUME サンプル表 927
ENCRYPT スカラー関数 394
Entrez
サポートされるバージョン 44
ESCAPE 節
LIKE 述部 261
Excel ファイル
サポートされるバージョン 44
EXECUTE IMMEDIATE ステートメント
動的 SQL 1
EXECUTE ステートメント
動的 SQL 1
EXECUTE 特権
関数 181
メソッド 190
EXISTS 述部 258
EXP スカラー関数
説明 397
Explain 表
概要 963
EXPLAIN_ARGUMENT 表 975
EXPLAIN_DIAGNOSTIC 表 982
EXPLAIN_DIAGNOSTIC_DATA 表 984
EXPLAIN_INSTANCE 表 986
EXPLAIN_OBJECT 表 989
EXPLAIN_OPERATOR 表 993
EXPLAIN_PREDICATE 表 996
EXPLAIN_STATEMENT 表 1000
EXPLAIN_STREAM 表 1003

F

FLOAT 関数
値および引数 398
説明 398
FLOAT データ・タイプ
符号と精度 85
FLOOR 関数
値および引数 399
説明 399
FOR FETCH ONLY 節
SELECT ステートメント 668
FOR READ ONLY 節
SELECT ステートメント 668
FROM 節
間接的な名前の説明 57
相関名の使用 57
相関名の例 57
直接的な名前の説明 57
副選択の構文 622
FROM 節中の間接的な相関名 57

G

GENERATE_UNIQUE 関数
構文 400
GETHINT 関数
値および引数 402
説明 402
GRAPHIC 関数
値および引数 403
説明 403
GRAPHIC ストリング
ストリング変換の構文 532
ホスト変数名から戻す 532
GRAPHIC ストリング定数
説明 140
GRAPHIC ストリング・データ・タイプ
説明 92
GRAPHIC スペース 29
GRAPHIC データ・タイプ
説明 92
GREATEST 関数 405
GROUP BY 節
副選択での規則と構文 622
副選択の結果 622
GROUP BY 節の ROLLUP グループ化 622
GROUPING 関数 318
grouping-expression 622

H

HASHEDVALUE 関数 406
HAVING 節 622
HEX 関数 408
HMMER データ・ソース
サポートされるバージョン 44
HOUR スカラー関数
説明 410

I

ID
区切り 57
通常 57
長さの制限 679
ホスト 57

ID *(続き)*
SQL 57
ID 列の値を返す
IDENTITY_VAL_LOCAL 関数 411
IDENTITY_VAL_LOCAL 関数
値および引数 411
説明 411
IMPLICIT_SCHEMA 権限 3
IN 述部 259
Informix
サポートされるバージョン 44
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
INSERT 関数
値および引数 415
説明 415
INTEGER 関数
値および引数 419
説明 419
INTEGER データ・タイプ
符号と精度 85
INTERSECT 演算子
全選択の、比較での役割 663
重複行、ALL の使用 663
INTO 節
値、アプリケーション・プログラムからの 57
FETCH ステートメント、ホスト変数の使用 57
SELECT INTO ステートメント、ホスト変数での使用 57
INTRAY サンプル表 927

J

Java Database Connectivity (JDBC)
概要 3
組み込み SQL for Java (SQLJ) 3
JULIAN_DAY スカラー関数
説明 421

L

LBAC (ラベル・ベースのアクセス制御)
制限 679
セキュリティー・ポリシー
名前の長さ 679
セキュリティー・ラベル
コンポーネント名の長さ 679
名前の長さ 679
例外表 1011
LCASE (ロケール依存) スカラー関数
概要 422, 423
LEAST 関数 424
LEFT スカラー関数
値および引数 425
説明 425
LENGTH スカラー関数
値および引数 429
説明 429
LIKE 述部 261
LN 関数
値および引数 431
説明 431
LOB (ラージ・オブジェクト)
説明 94
LOB ロケーター 94
LOCATE スカラー関数
値および引数 432
説明 432
LOG10 スカラー関数
値および引数 436
説明 436
LONG_VARCHAR 関数
説明 437
LONG_VARGRAPHIC 関数
説明 438
LOWER スカラー関数
値および引数 439
説明 439
LOWER (ロケール依存) スカラー関数
値および引数 440
説明 440
LTRIM スカラー関数
値および引数 442
説明 442

M

MAX 関数 443
値および引数 320
詳しい書式の説明 320
MAX_CARDINALITY 関数 444
MICROSECOND 関数
値および引数 445
説明 445
Microsoft Excel
「Excel ファイル」を参照 44
Microsoft SQL Server
サポートされるバージョン 44
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
MIDNIGHT_SECONDS 関数
値および引数 446
説明 446
MIN 関数 322, 447
MINUTE スカラー関数
値および引数 448
説明 448
MOD 関数
値および引数 449
説明 449

MONTH 関数
値および引数 450
説明 450
MONTHNAME 関数
値および引数 451
説明 451
MULTIPLY_ALT 関数
値と引数、規則 452
詳しい書式の説明 452

N

NODENUMBER 関数
DBPARTITIONNUM 376
NORMALIZE_DECFLOAT スカラー関数
説明 454
NOT NULL 節
NULL 述部内 267
NUL 文字で終了する文字ストリング 88
NULL 値
SQL
グループ化式における使用法 622
結果列 622
代入 115
重複行における出現 622
定義 82
標識変数によって指定される 57
不明条件 250
NULL 述部の規則 267
NULLIF 関数
値および引数 455
説明 455
NUMERIC または DECIMAL データ・タイプ
符号と精度 85
NVL スカラー関数
概要 456

O

OCTET_LENGTH スカラー関数
説明 457
ODBC
サポートされるバージョン 44
デフォルトの順方向タイプ・マッピング 902
ODBC (Open Database Connectivity)
説明 2
OLAP
仕様 228
OLAP (オンライン分析処理)
関数 228
OLE DB
サポートされるバージョン 44
Open Database Connectivity (ODBC) 2
OR 真理値表 250
Oracle
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
ORDER BY 節
言語文化的に正しい照合 359
SELECT ステートメント 622
ORG サンプル表 927
OVERLAY スカラー関数
説明 458

P

PARAMETER 関数
説明 462
PARTITION 関数
HASHEDVALUE 名の置換 406
POSITION スカラー関数 463
POSSTR 関数 466
POWER スカラー関数
説明 468
PREPARE ステートメント
動的 SQL 1
PROJECT サンプル表 927

Q

QUANTIZE スカラー関数
説明 469
QUARTER 関数
値および引数 471
説明 471

R

RADIANS スカラー関数
値および引数 472
説明 472
RAISE_ERROR スカラー関数
値および引数 473
説明 473
RAND スカラー関数
値および引数 475
説明 475
REAL SQL データ・タイプ
符号と精度 85
REAL 関数
値および引数 476
説明 476
単精度変換 476
REC2XML スカラー関数
値および引数 477
説明 477
remote-object-name 57
remote-schema-name 57
remote-table-name 57

REMOTE_NAME 関数マッピング・オプション
有効な設定値 901
REPEAT スカラー関数
値および引数 482
説明 482
REPLACE スカラー関数
値および引数 483
説明 483
RID 関数 485
RID_BIT 関数 485
RIGHT スカラー関数
値および引数 487
説明 487
ROUND スカラー関数
値および引数 491
説明 491
ROW CHANGE
式 244
RR (反復可能読み取り) 分離レベル
説明 20
RS (読み取り固定) 分離レベル
説明 20
RTRIM スカラー関数
説明 493

S

SALES サンプル表 927
SAMPLE データベース
作成 927
消去 927
説明 927
SBCS (1 バイト文字セット) データ
概要 88
Script
サポートされるバージョン 44
SECADM
データベース権限 11
SECLABEL
スカラー関数 494
SECLABEL_BY_NAME スカラー関数
説明 495
SECLABEL_TO_CHAR スカラー関数
説明 496
SECOND スカラー関数
値および引数 498
説明 498
security-label-name 57
security-policy-name 57
SELECT ステートメント
全選択の詳細な構文 663
定義 668
副選択 622
例 668
VALUES 節 663
SELECT 節
リストの表記法
列参照 622
DISTINCT キーワードを使用 622
SESSION_USER 特殊レジスター 176
SET SERVER OPTION ステートメント
一時的にオプションを設定する 38
SIGN スカラー関数
値および引数 499
説明 499
SIN スカラー関数
値および引数 500
説明 500
SINH スカラー関数
値および引数 501
説明 501
SMALLINT 関数
値および引数 502
説明 502
SMALLINT データ・タイプ
符号と精度 85
SOME 比較述部 254
SOUNDEX スカラー関数
値および引数 503
説明 503
SPACE スカラー関数
値および引数 504
説明 504
SQL 関数 181
SQL (構造化照会言語)
制限 679
パス 181
SQL 構文
回帰関数の結果 323
基本述部、詳細ダイアグラム 253
検索条件、形式と規則の詳細 250
複数の演算、実行の順序 663
2 つの述部の比較、真の条件 253, 268
AVG 集約関数、列セットでの結果 311
BETWEEN 述部、規則 257
CORRELATION 集約関数の結果 313
COUNT_BIG 関数、引数と結果 315
COVARIANCE 集約関数の結果 317
EXISTS 述部 258
GENERATE_UNIQUE 関数 400
GROUP BY 節
副選択 622
IN 述部の説明 259
LIKE 述部、規則 261
SELECT 節の説明 622
STDDEV 集約関数、結果 326
TYPE 述部 268
VARIANCE 集約関数の結果 328
WHERE 節の検索条件 622
SQL コンパイラー
フェデレーテッド・システム内の 37

SQL ステートメント
静的
定義 1
対話式 SQL
定義 1
動的 SQL
即時実行 1
定義 1
動的 SQL の準備と実行 1
ヘルプを表示する 1030
ルーチンで使用可能 1015
CALL 1019
SQL 操作
基本 115
SQL パス
組み込み 181
SQL 副照会
WHERE 節 622
SQL 変数名 57
SQLCA (SQL 連絡域)
エラー報告 689
説明 689
対話式の表示 689
パーティション・データベース・システム 689
SQLD フィールド、SQLDA の 695
SQLDA (SQL 記述子域)
内容 695
SQLDABC フィールド、SQLDA の 695
SQLDAID フィールド、SQLDA の 695
SQLDATA フィールド、SQLDA の 695
SQLDATALEN フィールド、SQLDA の 695
SQLDATATYPE_NAME フィールド、SQLDA の 695
SQLIND フィールド、SQLDA の 695
SQLJ
接続 3
SQLLEN フィールド、SQLDA の 695
SQLLONGLEN フィールド、SQLDA の 695
SQLN フィールド、SQLDA の 695
SQLNAME フィールド、SQLDA の 695
SQLSTATE
RAISE_ERROR 関数内 473
SQLTYPE フィールド、SQLDA の 695
SQLVAR フィールド、SQLDA の 695
SQRT スカラー関数
説明 505
STAFF サンプル表 927
STAFFG サンプル表 927
STDDEV 関数 326
STRIP スカラー関数
説明 506
SUBSTR 関数
フラグメント 507
SUBSTR スカラー関数
値および引数 507
説明 507
SUBSTRING スカラー関数
説明 510
SUM 関数
値および引数 327
詳しい書式の説明 327
Sybase
サポートされるバージョン 44
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
SYSTEM USER 特殊レジスター 177

T

TABLE 節
表参照 622
TABLE_NAME 関数
値および引数 513
説明 513
別名 513
TABLE_SCHEMA 関数
値および引数 514
説明 514
別名 514
TAN スカラー関数
値および引数 516
説明 516
TANH スカラー関数
値および引数 517
説明 517
Teradata
デフォルトの逆方向タイプ・マッピング 916
デフォルトの順方向タイプ・マッピング 902
TIME 関数
値および引数 518
説明 518
TIME データ・タイプ
説明 96
操作 210
TIMESTAMP 関数
値および引数 519
説明 519
TIMESTAMP データ・タイプ
説明 96
WEEK スカラー関数 557
WEEK_ISO スカラー関数 558
TIMESTAMPDIFF スカラー関数
値および引数 526
説明 526
TIMESTAMP_FORMAT 関数
値および引数 521
説明 521
TIMESTAMP_ISO 関数
値および引数 525
説明 525
TOTALORDER スカラー関数
説明 530

TO_CHAR 関数
値および引数 528
説明 528
TO_DATE 関数
値および引数 529
説明 529
TRANSLATE スカラー関数
値および引数 532
説明 532
文字ストリング 532
GRAPHIC ストリング 532
TRIM スカラー関数
説明 535
TRUNCATE スカラー関数 537
TYPE 述部
形式 268
TYPE_ID 関数
値および引数 539
説明 539
データ・タイプ 539
TYPE_NAME 関数
値および引数 540
説明 540
TYPE_SCHEMA 関数
値および引数 541
説明 541
データ・タイプ 541

U

UCASE スカラー関数 542
UCASE (ロケール依存) スカラー関数 543
UDF
ユーザー定義関数を参照 609
Unicode
大文字への変換 55
Unicode (UCS-2)
関数 334
ストリングの比較 136
パターン・マッチング 136
UNION 演算子
全選択の比較での役割 663
UPPER スカラー関数
値および引数 544
説明 544
UPPER (ロケール依存) スカラー関数
値および引数 545
説明 545
UR (非コミット読み取り) 分離レベル 20
USER 特殊レジスター 178

V

VALIDATED 述部
説明 270
VALUE 関数
値および引数 547
説明 547
VALUES 節
全選択 663
VARCHAR 関数
値および引数 548
説明 548
VARCHAR データ・タイプ
説明 88
DOUBLE スカラー関数 392
WEEK スカラー関数 557
WEEK_ISO スカラー関数 558
VARCHAR_BIT_FORMAT 関数
説明 550
VARCHAR_FORMAT 関数
値および引数 551
説明 551
VARCHAR_FORMAT_BIT 関数
説明 554
VARGRAPHIC 関数
値および引数 555
説明 555
VARGRAPHIC データ・タイプ
説明 92
VARIANCE 集約関数 328
Visual Explain
チュートリアル 1033

W

WEEK スカラー関数
説明 557
WEEK_ISO スカラー関数
説明 558
WHERE 節
全選択の副選択コンポーネント 622
WITH 共通表式
select-statement 668

X

XML
サポートされるバージョン 44
データ・タイプ 100
XMLAGG 集約関数
説明 329
XMLATTRIBUTES スカラー関数
説明 559
XMLCAST 指定
説明 223
XMLCOMMENT スカラー関数
説明 561
XMLCONCAT スカラー関数
説明 562

XMLDOCUMENT スカラー関数
説明 564
XMLELEMENT スカラー関数
説明 566
XMLEXISTS 述部 273
XMLFOREST スカラー関数
説明 574
XMLGROUP スカラー関数 331
XMLNAMESPACES 宣言
説明 578
XMLPARSE スカラー関数
説明 580
XMLPI スカラー関数
説明 583
XMLQUERY スカラー関数
説明 585
XMLROW スカラー関数 588
XMLSERIALIZE スカラー関数
説明 590
XMLTABLE 表関数
説明 605
XMLTEXT スカラー関数
説明 592
XMLVALIDATE スカラー関数
説明 594
XMLXSROBJECTID スカラー関数
説明 599
XSLTRANSFORM スカラー関数
説明 600
XSR_ADDSCHEMADOC ストアード・プロシージャー 611
XSR_COMPLETE ストアード・プロシージャー 612
XSR_DTD ストアード・プロシージャー 613
XSR_ENTITY ストアード・プロシージャー 615
XSR_REGISTER ストアード・プロシージャー 616
XSR_UPDATE ストアード・プロシージャー 618

Y

YEAR スカラー関数
値および引数 604
説明 604

Printed in Japan

SC88-4436-03

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21

Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows　バージョン 9 リリース 5

SQL リファレンス 第 1 巻