これが
知りたかった!

Oracle開発者のための

DB2 UDB SQLリファレンス

今すぐ使える実例集

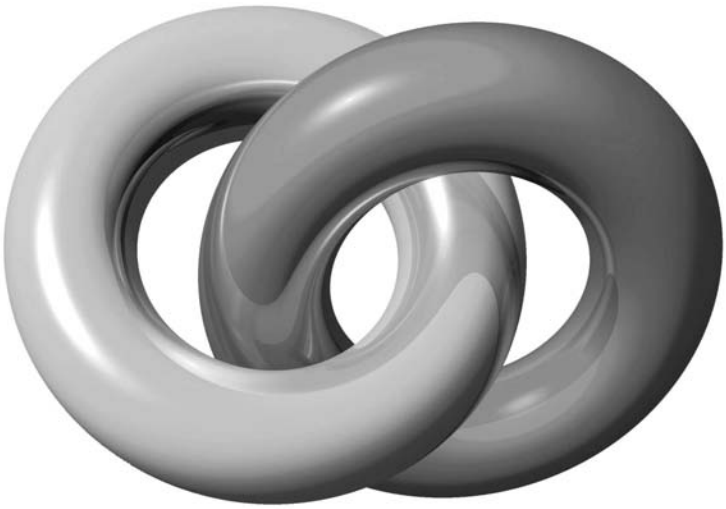
株式会社システム・テクノロジー・アイ

これが
知りたかった!

Oracle開発者のための

DB2 UDB SQLリファレンス

今すぐ使える実例集



株式会社システム・テクノロジー・アイ

はじめに

本書について

本書は、Oracleデータベースを用いたソフトウェア開発およびデータベース管理の経験のある方、またはそれと同等の知識を保有する方が、DB2 Universal Database（以下、DB2 UDB）を用いて開発を行う際に役立つリファレンスです。

本書は、Oracleの用語を使って解説しています（ただし、「集計ファンクション」については、ISO/ANSI SQL標準およびDB2 UDBのマニュアルに従い「集約関数」と表記しています）。基本的なSQLの使い方から知って得するテクニックまで掲載しています。また、OracleとDB2 UDBの各コマンドの実行結果を対比しているので、スムーズに知識を移行できるようになっています。

なお、本書の執筆にあたっては、Oracle Database 10gおよびDB2 UDB V8.2にて検証を行っています。

本書の使い方

最初の章から読み進めていただければ、DB2 UDBにおけるSQL学習書としてご利用いただけますし、すでにSQLを使った開発経験のある方は、巻末の機能索引から「○○を実現したいときのコマンド」を逆引きで探すこともできます。もちろん、キーワードを掲載した通常の索引も付属しています。

なお、実行例のSQLはOracleのサンプル・データベースをDB2 UDBへ移行した表に対して実行しているので、DB2 UDBがデフォルトで作成するサンプル・データベースの表名・列名とは異なります。

実行例のインターフェース

Oracleでは、コマンドの実行インターフェースとしてSQL*Plusを使用します。

一方、DB2 UDBは、コマンド行プロセッサー（Command Line Processor：CLP）とコマンド・エディターの2つのインターフェースを使って実行できます。コマンド・エディターはGUIで操作が可能で、CLPに比べると使いやすく機能も豊富です。

■ コマンド行プロセッサー（CLP）の例

■ コマンド・エディターの例 ①

■ コマンド・エディターの例 ②

謝辞

本書の制作にあたっては、日本アイ・ビー・エム株式会社 ソフトウェア事業 インフォメーション・マネジメント事業部の皆様に多大なるご協力をいただきました。心より感謝いたします。

2005 年 12 月

株式会社システム・テクノロジー・アイ

目次

1 データの検索 …… 1

1.1 表の全列検索 …… 2
1.2 指定した列の検索 …… 3
1.3 日付データを検索条件に指定した検索 …… 4
1.4 数値データを検索条件に指定した検索 …… 5
1.5 文字データを検索条件に指定した検索 …… 6
1.6 指定した条件に一致しないレコードの検索 …… 7
1.7 複数の検索条件のすべてを満たす行を表示する …… 8
1.8 複数の検索条件のいずれかを満たす行を表示する …… 9
1.9 特定の範囲に一致する行を表示する …… 10
1.10 上限が決まらない検索条件 …… 11
1.11 下限が決まらない検索条件 …… 12
1.12 値のリストに一致する行を表示する …… 13
1.13 テキストのあいまい検索 …… 14
1.14 1文字だけ不明な場合の検索 …… 15
1.15 複数のワイルドカードを使用した検索 …… 16
1.16 NULL値のレコードの検索 …… 17
1.17 NULL値以外のレコードの検索 …… 18
1.18 検索結果のレコードの列の加算 …… 19
1.19 検索結果のレコードの列の減算 …… 20
1.20 検索結果のレコードの列の乗算 …… 21
1.21 検索結果のレコードの列の除算 …… 22
1.22 複数の列を1つにまとめる（連結） …… 23
1.23 n件だけデータを表示する …… 24
1.24 列の一意な値を表示する（重複した値は排除） …… 25
1.25 表名に別名を付けて検索する …… 26
1.26 列名に別名を付けて検索する …… 27
1.27 予約語を使用した列別名を指定する …… 28
1.28 検索結果を昇順に並べ替える …… 29
1.29 検索結果を降順に並べ替える …… 30
1.30 複数の列を指定し、検索結果を昇順に並べ替える …… 31
1.31 検索結果の1列目を昇順、2列目を降順に並べ替える …… 32
1.32 列番号を指定して検索結果を並べ替える …… 33
1.33 検索結果を列別名で並べ替える …… 34
1.34 階層構造データの検索（上から順に） …… 35
1.35 特定の階層を取り除いた階層構造データの検索（上から順に） …… 38
1.36 特定の階層を選択した階層構造データの検索（上から順に） …… 40
1.37 階層構造データの検索（下から順に） …… 42

1.38 基準日からの経過日数を表示する …… 44
1.39 検索条件に単一引用符（'）を使用する …… 45
1.40 %を含む文字列を検索する …… 46

2 データの集計 ……47

2.1 平均値を求める …… 48
2.2 表内の行数のカウント …… 49
2.3 列内の値のカウント …… 50
2.4 列内の異なる値のカウント …… 51
2.5 最大値を求める …… 52
2.6 最小値を求める …… 53
2.7 合計を求める …… 54
2.8 検索結果をグループ化して表示する …… 55
2.9 集合値に条件を付ける …… 56
2.10 重複しているレコードの件数を検索する …… 57
2.11 重複したレコードを除いて表示する …… 58
2.12 指定した行だけをグループ化して表示する …… 59
2.13 小計と総計 …… 60
2.14 第1グループの小計、第2グループの小計と総計 …… 62
2.15 複数グループごとの小計 …… 64
2.16 年を単位にグルーピングした結果を求める …… 66
2.17 指定した列の値が最大値のレコードを検索する …… 67

3 関数の利用方法 ……69

3.1 ABS — 絶対値を求める …… 70
3.2 ASCII — 最初の文字の10進表記を求める …… 71
3.3 ASCIISTR — 文字列のASCII表記を求める …… 72
3.4 CHR — 指定された文字コードを文字に変換する …… 73
3.5 NCHR — 指定された文字コードを文字に変換する …… 74
3.6 COALESCE …… 75
3.7 CONCAT — 文字列を連結する …… 76
3.8 COUNT — 件数をカウントする …… 77
3.9 CURRENT_DATE …… 78
3.10 CURRENT_TIMESTAMP …… 79
3.11 DBTIMEZONE …… 80
3.12 DECODE …… 81
3.13 DENSE_RANK …… 82
3.14 EXTRACT …… 83

3.15 FIRST …… 84
3.16 FIRST_VALUE …… 85
3.17 FROM_TZ …… 86
3.18 GROUP_ID …… 87
3.19 GROUPING …… 89
3.20 GROUPING_ID …… 90
3.21 LAG …… 92
3.22 LAST …… 93
3.23 LAST_VALUE …… 94
3.24 LEAD …… 95
3.25 LENGTH ― 文字列の長さを返す …… 96
3.26 LENGTHB ― 文字列の長さを返す …… 97
3.27 LTRIM ― 文字列の先頭の指定された文字列を削除する …… 98
3.28 LOWER ― すべての文字列を小文字にする …… 100
3.29 MAX ― 最大値を求める …… 101
3.30 MIN ― 最小値を求める …… 102
3.31 MOD ― 余りを求める …… 103
3.32 POWER ― べき乗を求める …… 104
3.33 REPLACE …… 105
3.34 SIGN …… 106
3.35 RTRIM …… 107
3.36 SOUNDEX …… 109
3.37 SUBSTR …… 110
3.38 SUBSTRB …… 111
3.39 SUM …… 112
3.40 TRANSLATE …… 113
3.41 UPPER …… 114
3.42 USER …… 115
3.43 ACOS …… 116
3.44 ASIN …… 117
3.45 ATAN …… 118
3.46 ATAN2 …… 119
3.47 COS …… 120
3.48 COSH …… 121
3.49 SIN …… 122
3.50 SINH …… 123
3.51 TAN …… 124
3.52 TANH …… 125
3.53 CEIL …… 126
3.54 FLOOR …… 127
3.55 ROUND（数値型：小数点ｎ桁あるいは小数点の左ｎ桁に四捨五入） … 128
3.56 ROUND（日付型：時間、日にち、月を四捨五入） …… 129
3.57 ROUND（日付型：日にちを四捨五入） …… 130
3.58 ROUND（日付型：月を四捨五入） …… 132
3.59 TRUNC（数値型：小数点以下ｍ桁あるいは小数点の左ｍ桁を
切り捨て） …… 133

3.60 TRUNC（日付型：時間、日にち、月の切り捨て） …… 134
3.61 TRUNC（日付型：日にちの切り捨て） …… 135
3.62 TRUNC（日付型：月の切り捨て） …… 136
3.63 EXP …… 137
3.64 LN …… 138
3.65 LOCALTIMESTAMP …… 139
3.66 SQRT …… 140
3.67 AVG …… 141
3.68 INSTR …… 142
3.69 INSTRB ― 出現回数目に一致したものを戻す …… 143
3.70 INSTRB ― 1回目に一致したものを戻す …… 144
3.71 LOG …… 145
3.72 RAWTOHEX …… 146
3.73 NVL …… 147
3.74 SYSDATE …… 148
3.75 TO_MULTI_BYTE …… 149
3.76 VSIZE …… 150
3.77 VARIANCE …… 151
3.78 BIN_TO_NUM …… 152
3.79 BITAND …… 153
3.80 CAST …… 154
3.81 INITCAP …… 155
3.82 LPAD …… 156
3.83 RPAD …… 157
3.84 NLS_INITCAP …… 158
3.85 NLS_UPPER …… 159
3.86 NLS_LOWER …… 160
3.87 NTILE …… 161
3.88 HARTOROWID …… 162
3.89 CONVERT …… 163
3.90 ROWIDTOCHAR …… 164
3.91 TO_DATE …… 165
3.92 TO_NUMBER …… 166
3.93 TO_SINGLE_BYTE …… 167
3.94 ADD_MONTH …… 168
3.95 LAST_DAY …… 169
3.96 LEAST …… 170
3.97 MONTHS_BETWEEN …… 171
3.98 NEXT_DAY …… 172
3.99 NEW_TIME …… 173
3.100 GREATEST …… 174
3.101 NULLIF …… 175
3.102 NUMTODSINTERVAL …… 176
3.103 NUMTOYMINTERVAL …… 177
3.104 NVL2 …… 178
3.105 UID …… 179

3.106 USERENV …… 180
3.107 MEDIAN …… 181

4 複雑な問い合わせ ……183

4.1 WHERE句に使用する問い合わせ …… 184
4.2 1行のみ戻す副問い合わせ …… 186
4.3 複数行戻す副問い合わせ …… 187
4.4 副問い合わせの結果のいずれかに一致する行を表示する …… 189
4.5 副問い合わせの結果のすべてに一致する行を表示する …… 190
4.6 副問い合わせの結果の最小値より大きい行を表示する …… 191
4.7 副問い合わせの結果の最大値より大きい行を表示する …… 193
4.8 副問い合わせの結果の最小値より小さい行を表示する …… 194
4.9 副問い合わせの結果の最大値より小さい行を表示する …… 195
4.10 EXISTSを使用した副問い合わせ …… 197
4.11 NOT EXISTSを使用した副問い合わせ …… 198
4.12 複数列を使用した副問い合わせ …… 199
4.13 複数の副問い合わせを使用した問い合わせ …… 201
4.14 相関副問い合わせ …… 203
4.15 SET句（UPDATE文）に使用する問い合わせ …… 204
4.16 相関副問い合わせを使用したUPDATE …… 205
4.17 FROM句に使用する問い合わせ …… 206
4.18 ソート済み結果への連番表示 …… 207
4.19 上位5件までのデータを表示する …… 209

5 表の結合 ……211

5.1 共通な列を持つ表の結合（列名、データ型が一致）……212
5.2 共通な列を持つ表の結合（列名のみ一致） …… 214
5.3 一部の列を使用した表の結合 …… 216
5.4 等価条件以外での結合 …… 218
5.5 表の異なる行同士での結合 …… 220
5.6 3つ以上の表の結合 …… 222
5.7 一致する行に加えてJOIN句の左側の表にしかない行（一致しない行）も表示する外部結合 …… 224
5.8 一致する行に加えてJOIN句の右側の表にしない行（一致しない行）も表示する外部結合 …… 226
5.9 一致する行に加えて互いに一致しない行を表示する外部結合 …… 228
5.10 一致しない行のみ表示する結合 …… 230
5.11 検索条件を付加した結合 …… 232

5.12 ORDER BY句を付加した結合 …… 233
5.13 複数の問い合わせから戻される行を表示する（重複行を排除） …… 235
5.14 複数の問い合わせから戻される行を表示する（重複行を含む） …… 236
5.15 最初の問い合わせから戻される行のみを表示する …… 238
5.16 複数の問い合わせから戻される共通の行のみを表示する …… 239
5.17 2つの表を総当たりで結合する …… 240

6 **データベース管理 ……243**

6.1 表一覧の取得 …… 244
6.2 新規表の作成 …… 245
6.3 表作成時に文字列をデフォルト値として設定する …… 246
6.4 表作成時に今日の日付をデフォルト値として設定する …… 247
6.5 ほかの表を元に新規表を作成する …… 248
6.6 ほかの表の特定列を元に新規表を作成する …… 249
6.7 ほかの表を元に新規表を作成する際に列名を変更する …… 250
6.8 ほかの表の特定行を元に新規表を作成する …… 251
6.9 ほかの表の定義だけ複写して新規表を作成する …… 252
6.10 表を削除する …… 253
6.11 表削除時に参照整合性制約を無効にする …… 254
6.12 表名を変更する …… 255
6.13 ほかのスキーマの表名を変更する …… 256
6.14 列を追加する …… 257
6.15 列を削除する …… 258
6.16 列に未使用マークを付ける …… 259
6.17 列名を変更する …… 260
6.18 列の長さを大きくする …… 261
6.19 列の長さを小さくする …… 262
6.20 列のデータ型を変更する …… 263
6.21 表の列情報の取得 …… 264
6.22 表を移動する …… 265
6.23 表の再編成を行う …… 266
6.24 表の定義を確認する …… 267
6.25 ほかのユーザーに表に対する問い合わせの権限を与える …… 268
6.26 ほかのユーザーに表に対する挿入、削除、更新の権限を与える …… 269
6.27 ほかのユーザーに表の特定の列に対する挿入の権限を与える …… 270
6.28 ほかのユーザーに表の特定の列に対する更新の権限を与える …… 271
6.29 ほかのユーザーに表の特定の列に対する問い合わせの権限を与える …… 272
6.30 ほかのユーザーに権限付与操作の許可を与える …… 273
6.31 与えたオブジェクト権限を取り消す …… 274
6.32 与えられている権限を一覧する …… 275
6.33 スキーマにアクセスできるユーザーを一覧する …… 276
6.34 表にコメントを定義する …… 277

6.35 列にコメントを定義する …… 278
6.36 表作成時に列制約を宣言する …… 279
6.37 表作成時に表制約を宣言する …… 280
6.38 主キー制約を削除するときに参照整合性制約も削除する …… 281
6.39 主キー制約を追加する …… 282
6.40 主キー制約を削除する …… 283
6.41 一意制約を追加する …… 284
6.42 一意制約を削除する …… 285
6.43 CHECK制約を追加する …… 286
6.44 CHECK制約を削除する …… 287
6.45 外部キーを追加する …… 288
6.46 外部キーを削除する …… 289
6.47 NOT NULLを削除する …… 290
6.48 NOT NULLを追加する …… 291
6.49 遅延制約を定義する …… 292
6.50 即時制約を定義する …… 293
6.51 制約を無効にする …… 294
6.52 制約を有効にする …… 295
6.53 データ削除時に子表行を削除する参照整合性制約 …… 296
6.54 データ削除時に子表行をNULL値に更新する参照整合性制約 …… 297
6.55 ビューの作成 …… 298
6.56 特定列を指定したビューの作成 …… 299
6.57 特定行を指定したビューの作成 …… 300
6.58 複数の表を使用したビューの作成 …… 301
6.59 ビューに対する問い合わせ …… 302
6.60 ビューから行を挿入する …… 303
6.61 ビューから列を更新する …… 304
6.62 ビューから行を削除する …… 305
6.63 ビューの変更 …… 306
6.64 ビューの削除 …… 307
6.65 ビュー同士を結合する …… 308
6.66 一意索引を作成する …… 310
6.67 非一意索引を作成する …… 311
6.68 複数列索引を作成する …… 312
6.69 索引を削除する …… 313
6.70 索引を再構築する …… 314
6.71 索引を再編成する …… 315
6.72 索引の使用状況を監視する …… 316
6.73 索引の一覧を表示する …… 317
6.74 複数列索引の列の並び順を調べる …… 318
6.75 順序を作成する …… 319
6.76 順序を変更する …… 320
6.77 順序を採番する …… 321
6.78 最新の順序値を確認する …… 322
6.79 行挿入時に順序を使用する …… 323
6.80 列更新時に順序を使用する …… 324

6.81 順序を削除する …… 325
6.82 シノニムを作成する …… 326
6.83 パブリック・シノニムを作成する …… 327
6.84 シノニムを削除する …… 328
6.85 全行を高速に削除する（TRUNCATE） …… 329

7 レコードの操作 ……331

7.1 行の挿入 …… 332
7.2 挿入列を省略した行の挿入 …… 333
7.3 特定の列を指定した行の挿入 …… 334
7.4 ほかの表から既存表へ行を挿入 …… 335
7.5 ほか表からの既存表に列を挿入 …… 336
7.6 数値データの挿入 …… 337
7.7 文字データの挿入 …… 338
7.8 日付データの挿入 …… 339
7.9 NULLデータの挿入 …… 340
7.10 デフォルト値を使用した挿入 …… 341
7.11 今日の日付の挿入 …… 342
7.12 書式を指定した日付の挿入 …… 343
7.13 単一引用符（'）をデータとして挿入する …… 344
7.14 複数行の挿入 …… 345
7.15 行をほかの表にコピーし、列の値によって、コピーする先を変更する …… 346
7.16 1行の更新 …… 347
7.17 特定行の更新 …… 348
7.18 全行の更新 …… 349
7.19 更新対象の行が存在しなかった場合の処理 …… 350
7.20 文字データの更新 …… 351
7.21 数値データの更新 …… 352
7.22 計算式を使用した数値データの更新 …… 353
7.23 日付データの更新 …… 354
7.24 書式を指定した日付の更新 …… 355
7.25 複数列の更新 …… 356
7.26 ほかの表の値を元に更新する …… 357
7.27 行の削除 …… 358
7.28 特定行の削除 …… 359
7.29 すべてのレコードを一括して削除する …… 360
7.30 全行を高速に削除する …… 361
7.31 重複する行を削除する …… 362
7.32 削除対象の行が存在しなかった …… 363
7.33 トランザクションを確定する …… 364
7.34 トランザクションを破棄する …… 365
7.35 トランザクションのセーブポイントを設定する …… 366

7.36 トランザクションに複数のセーブポイントを設定する …… 367
7.37 トランザクションを指定されたセーブポイントまで戻す …… 368
7.38 2つの表データをマージする …… 369

機能索引 …… **370**

キーワード索引 …… **373**

1

データの検索

1.1 表の全列検索

表のすべての列を検索します。

Oracle の場合

書式 **SELECT * FROM 表名**

実行例

```
SQL> SELECT * FROM DEPT;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

4行が選択されました。
```

DB2 の場合

書式 **SELECT * FROM 表名**

実行例

```
db2=> SELECT * FROM DEPT

DEPTNO DNAME          LOC
------ -------------- -------------
   10. ACCOUNTING     NEW YORK
   20. RESEARCH       DALLAS
   30. SALES          CHICAGO
   40. OPERATIONS     BOSTON

  4 レコードが選択されました。
```

注意点 表のすべての列の値を表示させたい場合は、アスタリスク（*）を使用します。表を定義したときに指定した列の順に表示されます。

1.2 指定した列の検索

表の指定した列を検索します。

Oracleの場合

書式 **SELECT 列名1,列名2[,列名n,・・・] FROM 表名**

実行例

```
SQL> SELECT DNAME,LOC FROM DEPT;

DNAME          LOC
-------------- -------------
ACCOUNTING     NEW YORK
RESEARCH       DALLAS
SALES          CHICAGO
OPERATIONS     BOSTON

4行が選択されました。
```

DB2の場合

書式 **SELECT 列名1,列名2[,列名n,・・・] FROM 表名**

実行例

```
db2=> SELECT DNAME,LOC FROM DEPT

DNAME          LOC
-------------- -------------
ACCOUNTING     NEW YORK
RESEARCH       DALLAS
SALES          CHICAGO
OPERATIONS     BOSTON

  4 レコードが選択されました。
```

注意点 SELECT句に複数列指定する場合は、カンマ(,)で区切って指定します。表を定義したときの列の並び順ではなく、SELECT句に指定した順に出力されます。

1.3 日付データを検索条件に指定した検索

指定した日付データを検索します。

Oracle の場合

書式

SELECT 列名 FROM 表名 WHERE 日付型列 = 'yy-mm-dd'
（ここでは、デフォルトの日付書式である yyyy-mm-dd の例を示します）

実行例

```
SQL> SELECT ENAME FROM EMP WHERE HIREDATE = '80-12-17';

ENAME
--------------------
SMITH

1行が選択されました。
```

DB2 の場合

書式

SELECT 列名 FROM 表名 WHERE 日付型列 = 'yyyy-mm-dd'
（デフォルトの日付のフォーマットとして yyyy-mm-dd、mm/dd/yyyy、dd.mm.yyyy が使えます）

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT ENAME FROM EMP WHERE HIREDATE = '1980-12-17';
--------------------------------------------------------------------------

ENAME
-----
SMITH

  1 レコードが選択されました。
```

注意点

Oracle のデフォルトの日付書式は yy-mm-dd ですが、デフォルト書式は変更もできます。設定されている書式に合った日付データを単一引用符（ ' ）で囲んで指定します。
DB2 UDB のデフォルトの日付書式は ISO や JIS などの標準フォーマットに合わせてあり、yyyy-mm-dd、mm/dd/yyyy、dd.mm.yyyy です。単一引用符（ ' ）で囲んで指定します。これ以外の書式を使用する場合は変換関数を用います。

1.4 数値データを検索条件に指定した検索

数値データを指定して検索します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 WHERE 数値型の列名 = 検索する数値**

実行例

```
SQL> SELECT ENAME FROM EMP WHERE EMPNO = 7369;

ENAME
--------------------
SMITH

1行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 WHERE 数値型の列名 = 検索する数値**

実行例

```
db2=> SELECT ENAME FROM EMP WHERE EMPNO = 7369

ENAME
----------
SMITH

  1 レコードが選択されました。
```

注意点

数値データを指定する場合、単一引用符 (') で囲むべきではありません。
Oracle では暗黙型変換が行われるため、単一引用符 (') で囲んでもエラーにはなりませんが、大量のレコードを処理する検索などの場合、パフォーマンス劣化の原因になるため好ましくありません。
DB2 UDB では暗黙型変換は行われません。単一引用符 (') で囲むとデータ型の指定が不適切である旨のエラーが表示されます。

1.5 文字データを検索条件に指定した検索

文字列型のデータを指定して検索します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 WHERE 文字列型の列名 = '検索する文字列'**

実行例

```
SQL> SELECT EMPNO,HIREDATE FROM EMP WHERE ENAME = 'SMITH';

     EMPNO HIREDATE
---------- --------
      7369 80-12-17

1行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 WHERE 文字列型の列名 = '検索する文字列'**

実行例

```
db2=> SELECT EMPNO,HIREDATE FROM EMP WHERE ENAME = 'SMITH'

EMPNO  HIREDATE
------ ----------
 7369. 1980-12-17

  1 レコードが選択されました。
```

注意点 | 文字列を単一引用符（'）で囲んで指定します。アルファベットの大文字小文字は区別します。

1.6 指定した条件に一致しないレコードの検索

指定した条件に一致しないデータを検索します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE NOT (条件)**

実行例

```
SQL> SELECT * FROM DEPT WHERE NOT (DEPTNO = 10);

    DEPTNO DNAME          LOC
---------- -------------- -------------
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

3行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE NOT (条件)**

実行例

```
db2=> SELECT * FROM DEPT WHERE NOT (DEPTNO = 10)

DEPTNO DNAME          LOC
------ -------------- -------------
   20. RESEARCH       DALLAS
   30. SALES          CHICAGO
   40. OPERATIONS     BOSTON

  3 レコードが選択されました。
```

注意点 NOTは結果を反転させる論理演算子です。指定した条件に合致しない行を検索する場合に有益です。
NOTは、論理演算子（NOT、AND、OR）の中で最初に処理されます（優先順位が最も高い）。
なお、単一の不一致の比較条件は <> で指定します。!= や ^= も使用できます。

1.7 複数の検索条件のすべてを満たす行を表示する

複数の検索条件をAND演算子を用いて記述します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 検索条件1 AND 検索条件2**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE JOB = 'MANAGER' AND SAL >=2900;

     EMPNO ENAME
---------- --------------------
      7566 JONES

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 検索条件1 AND 検索条件2**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE JOB = 'MANAGER' AND SAL >=2900

EMPNO  ENAME
------ ----------
 7566. JONES

  1 レコードが選択されました。
```

注意点 複数の条件を使用する場合、論理演算子（AND、OR、NOT）を使用して条件を指定します。すべての条件を満たした行だけを検索したい場合にはAND演算子を使用します。OR演算子と一緒に使用した場合は、AND演算子が先に処理されます（優先順位が高い）。演算子の優先順位を変更するには、括弧（( )）を使います。

1.8 複数の検索条件のいずれかを満たす行を表示する

複数の検索条件をOR演算子を用いて記述します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 検索条件1 OR 検索条件2**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE JOB = 'MANAGER' OR SAL >=2900;

     EMPNO ENAME
---------- --------------------
      7566 JONES
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7902 FORD

6行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 検索条件1 OR 検索条件2**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE JOB = 'MANAGER' OR SAL >=2900

EMPNO  ENAME
------ ----------
 7566. JONES
 7698. BLAKE
 7782. CLARK
 7788. SCOTT
 7839. KING
 7902. FORD

  6 レコードが選択されました。
```

注意点 複数の条件を使用する場合、論理演算子（AND、OR、NOT）を使用して条件を指定します。指定した条件のいずれか1つを満たしている行を検索したい場合にはOR演算子を使用します。AND演算子と一緒に使用した場合は、AND演算子が先に処理されます（優先順位が高い）。演算子の優先順位を変更するには、括弧（( )）を使います。

1.9 特定の範囲に一致する行を表示する

検索する範囲の上限と下限の値を指定します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 BETWEEN 下限値 AND 上限値**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE SAL BETWEEN 2000 AND 3000;

     EMPNO ENAME
---------- --------------------
      7566 JONES
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7902 FORD

5行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 BETWEEN 下限値 AND 上限値**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE SAL BETWEEN 2000 AND 3000

EMPNO  ENAME
------ ----------
 7566. JONES
 7698. BLAKE
 7782. CLARK
 7788. SCOTT
 7902. FORD

  5 レコードが選択されました。
```

注意点

数値データだけでなく、日付や文字列の範囲も指定できます。日付や文字列を使用する場合は、単一引用符 (') で囲む必要があります。文字列の範囲 (下限、上限) はデータベースに設定されている文字コードと照合順序に依存します。
Oracle では、文字コードと照合順序は初期化パラメータ NLS_SORT で指定し、DB2 UDB では、データベース作成時に COLLATE を指定します。

1.10 上限が決まらない検索条件

下限の値を指定します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 >= 下限値**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE SAL >= 2000;

     EMPNO ENAME
---------- --------------------
      7566 JONES
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7902 FORD

6行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 >= 下限値**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE SAL >= 2000

EMPNO  ENAME
------ ----------
 7566. JONES
 7698. BLAKE
 7782. CLARK
 7788. SCOTT
 7839. KING
 7902. FORD

  6 レコードが選択されました。
```

注意点

数値データだけでなく、日付や文字列の範囲も指定できます。日付や文字列を使用する場合は、単一引用符 (') で囲む必要があります。文字列の範囲 (下限、上限) はデータベースに設定されている文字コードと照合順序に依存します。

Oracle では、文字コードと照合順序は初期化パラメータ NLS_SORT で指定し、DB2 UDB では、データベース作成時に COLLATE を指定します。

比較する値を含めない場合は、比較演算子 > (より大きい) を使用します。

1.11 下限が決まらない検索条件

上限の値を指定します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 <= 上限値**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE SAL <= 1000;

     EMPNO ENAME
---------- --------------------
      7369 SMITH
      7900 JAMES

2行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 <= 上限値**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE SAL <= 1000

EMPNO  ENAME
------ ----------
 7369. SMITH
 7900. JAMES

  2 レコードが選択されました。
```

注意点

数値データだけでなく、日付や文字列の範囲も指定できます。日付や文字列を使用する場合は、単一引用符 (') で囲む必要があります。文字列の範囲 (下限、上限) はデータベースに設定されている文字コードに依存します。
比較する値を含めない場合は、比較演算子 < (より小さい) を使用します。

1.12 値のリストに一致する行を表示する

IN条件を用いて値のリストを指定します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 IN (値1,値2・・・)**

実行例

```
SQL> SELECT ENAME,JOB FROM EMP WHERE EMPNO IN (7369,7499);

ENAME                JOB
-------------------- ---------
SMITH                CLERK
ALLEN                SALESMAN

2行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 IN (値1,値2・・・)**

実行例

```
db2=> SELECT ENAME,JOB FROM EMP WHERE EMPNO IN (7369,7499)

ENAME      JOB
---------- ---------
SMITH      CLERK
ALLEN      SALESMAN

  2 レコードが選択されました。
```

注意点

数値データだけでなく、日付や文字列のリストを指定できます。日付や文字列を使用する場合は、単一引用符（'）で囲む必要があります。IN条件の代わりに、「列名 = 値1 OR 列名 = 値2」と指定することも可能です。

Oracleでは、IN条件の代わりに、比較条件のANYを使用することができます。

DB2 UDBでは、比較条件のANYは副問い合わせの場合にのみ使用します。値が定数の場合は、VALUESを指定して副問い合わせ（全選択）にしてしまう方法もあります。

たとえば、「EMPNO = ANY (VALUES 7369, 7499)」のようになります。ただし、IN条件で書いたほうが簡単なことが多いでしょう。

1.13 テキストのあいまい検索

一部の文字列しか明確でない場合、ワイルドカードの%を使って値を指定します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 LIKE '文字列%'**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE ENAME LIKE 'S%';

     EMPNO ENAME
---------- --------------------
      7369 SMITH
      7788 SCOTT

2行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 LIKE '文字列%'**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE ENAME LIKE 'S%'

EMPNO  ENAME
------ ----------
 7369. SMITH
 7788. SCOTT

  2 レコードが選択されました。
```

注意点

% (パーセント) はワイルドカードで、ゼロ文字以上の任意の文字列を表します。
Oracleは暗黙型変換が行われるため、列名に数値型および日付型の列を指定しても動作します。
DB2 UDBは暗黙型変換を行わないため、列名を文字列に型変換してからLIKE条件を使用するか、変換関数を使用する必要があります。

1.14 1文字だけ不明な場合の検索

文字列のどこかに1文字だけ不明な文字がある場合、ワイルドカードの _ を使って値を指定します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 LIKE '文字列_文字列'**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE ENAME LIKE 'SMI_H';

     EMPNO ENAME
---------- --------------------
      7369 SMITH

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 LIKE '文字列_文字列'**

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE ENAME LIKE 'SMI_H'

EMPNO  ENAME
------ ----------
 7369. SMITH

  1 レコードが選択されました。
```

注意点

_（アンダースコア）は、任意の1文字を表します。任意の位置に指定することができます。Oracleの _（アンダースコア）は半角全角どちらも同じ動作ですが、DB2 UDBでは非Unicodeデータベースの場合、半角の _（アンダースコア）は半角文字のみ、全角の _（アンダースコア）は全角文字のみを検索対象とします。

1.15 複数のワイルドカードを使用した検索

わかっているキーワードが文字列中のどこにあるかわからない場合に使用します。

Oracle の場合

書式

```
SELECT 列名 FROM 表名 WHERE 列名 LIKE '%文字列%文字列'
```

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE ENAME LIKE '%A%N';

     EMPNO ENAME
---------- --------------------
      7499 ALLEN
      7654 MARTIN

2行が選択されました。
```

DB2 の場合

書式

```
SELECT 列名 FROM 表名 WHERE 列名 LIKE '%文字列%文字列'
```

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP WHERE ENAME LIKE '%A%N'

EMPNO  ENAME
------ ----------
 7499. ALLEN
 7654. MARTIN

  2 レコードが選択されました。
```

注意点

不明な文字を表すワイルドカードの％（パーセンテージ）や _（アンダースコア）は、任意の位置に必要な数だけ指定することができます。「%A%N」は、先頭の文字列は不明だがどこかに「A」が含まれており、最後は「N」で終わることを意味します。

1.16 NULL値のレコードの検索

NULL値を検索するには、「IS NULL」を指定します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 IS NULL**

実行例

```
SQL> SELECT EMPNO,ENAME,COMM FROM EMP WHERE COMM IS NULL;

     EMPNO ENAME                          COMM
---------- -------------------- ----------
      7369 SMITH
      7566 JONES
      7698 BLAKE
      7782 CLARK
      7788 SCOTT
      7839 KING
      7876 ADAMS
      7900 JAMES
      7902 FORD
      7934 MILLER
      7950 MARY

11行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 IS NULL**

実行例

```
db2=> SELECT EMPNO,ENAME,COMM FROM EMP WHERE COMM IS NULL

EMPNO  ENAME      COMM
------ ---------- ---------
 7369. SMITH              -
 7566. JONES              -
 7698. BLAKE              -
 7782. CLARK              -
 7788. SCOTT              -
 7839. KING               -
 7876. ADAMS              -
 7900. JAMES              -
 7902. FORD               -
 7934. MILLER             -
 7950. MARY               -

  11 レコードが選択されました。
```

注意点

NULL値のデータの検索は、「=NULL」ではなく「IS NULL」を使用します。
Oracleでは「列名 = NULL」と指定しても望む値は得られません。エラーにならないので注意が必要です。
DB2 UDBでは、「列名 = NULL」と指定するとエラーになります。
OracleのSQL*Plusでは、デフォルトではNULLを空白で表示します。DB2 UDBのCLPでは「-」で表示されます。

1.17 NULL値以外のレコードの検索

NULL以外の値を検索するには、「IS NOT NULL」を指定します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 IS NOT NULL**

実行例

```
SQL> SELECT EMPNO,ENAME,COMM FROM EMP WHERE COMM IS NOT NULL;

     EMPNO ENAME                    COMM
---------- -------------------- ----------
      7499 ALLEN                       300
      7521 WARD                        500
      7654 MARTIN                     1400
      7844 TURNER                        0

4行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 IS NOT NULL**

実行例

```
db2=> SELECT EMPNO,ENAME,COMM FROM EMP WHERE COMM IS NOT NULL

EMPNO  ENAME      COMM
------ ---------- ---------
 7499. ALLEN         300.00
 7521. WARD          500.00
 7654. MARTIN       1400.00
 7844. TURNER          0.00

  4 レコードが選択されました。
```

注意点

NULL以外の値を検索するときには、比較演算子の <> (等しくない) を使用するのではなく、「IS NOT NULL」を使用します。
Oracleでは「列名 <> NULL」と指定しても望む値は得られません。エラーにならないので注意が必要です。
DB2 UDBでは、「列名 <> NULL」と指定するとエラーになります。

1.18 検索結果のレコードの列の加算

SELECT句で指定した列の値に足し算をします。

Oracleの場合

書式 **SELECT 列名1 + {列名2 | リテラル | 式} FROM 表名**

実行例

```
SQL> SELECT SAL,SAL+500 FROM EMP WHERE EMPNO = 7369;

       SAL    SAL+500
---------- ----------
       800       1300

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名1 + {列名2 | リテラル | 式} FROM 表名**

実行例

```
db2=> SELECT SAL,SAL+500 FROM EMP WHERE EMPNO = 7369

SAL        2
---------- ----------------
   800.00          1300.00

  1 レコードが選択されました。
```

注意点

数値型の列に対して加算を行うことができます。
Oracleは、式がそのまま検索結果の列名になります。
DB2 UDBは、列の番号が列名として表示されます。

1.19 検索結果のレコードの列の減算

SELECT句で指定した列の値に引き算をします。

Oracleの場合

書式 **SELECT 列名1 - {列名2 | リテラル | 式} FROM 表名**

実行例

```
SQL> SELECT SAL,SAL-500 FROM EMP WHERE EMPNO = 7369;

       SAL    SAL-500
---------- ----------
       800         300
```

1行が選択されました。

DB2の場合

書式 **SELECT 列名1 - {列名2 | リテラル | 式} FROM 表名**

実行例

```
db2=> SELECT SAL,SAL-500 FROM EMP WHERE EMPNO = 7369

SAL       2
--------- ----------------
   800.00           300.00
```

1 レコードが選択されました。

注意点

数値型の列に対して減算を行うことができます。
Oracleは、式がそのまま検索結果の列名になります。
DB2 UDBは、列の番号が列名として表示されます。
DB2 UDBでは、DATE、TIME、TIMESTAMPの各型同士の減算ができます。結果はそれぞれ、

- DATE間隔: DEC(8,0) yyyymmdd形式
- TIME間隔: DEC(6,0) hhmmss形式
- TIMESTAMP間隔: DEC(20,6) yyyymmddhhmmss.zzzzzz形式

となります。
この減算結果のDECIMALの値は、ほかのDATE、TIME、TIMESTAMP型データと加減算ができます。
「3.107 MEDIAN」の注意点に利用例を掲載しています。

1.20 検索結果のレコードの列の乗算

SELECT句で指定した列の値に掛け算をします。

Oracleの場合

書式 **SELECT 列名1 * {列名2 | リテラル | 式} FROM 表名**

実行例

```
SQL> SELECT SAL,SAL*1.5 FROM EMP WHERE EMPNO = 7369;

       SAL    SAL*1.5
---------- ----------
       800        1200

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名1 * {列名2 | リテラル | 式} FROM 表名**

実行例

```
db2=> SELECT SAL,SAL*1.5 FROM EMP WHERE EMPNO = 7369

SAL         2
--------- -----------
   800.00    1200.000

  1 レコードが選択されました。
```

注意点

数値型の列に対して乗算を行うことができます。式内の * (アスタリスク) は乗算を意味します。
Oracleは、式がそのまま検索結果の列名になります。
DB2 UDBは、列の番号が列名として表示されます。

1.21 検索結果のレコードの列の除算

SELECT句で指定した列の値に割り算をします。

Oracleの場合

書式 **SELECT 列名1 / {列名2 | リテラル | 式} FROM 表名**

実行例

```
SQL> SELECT SAL,SAL/2 FROM EMP WHERE EMPNO = 7369;

       SAL       SAL/2
---------- ----------
       800         400

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名1 / {列名2 | リテラル | 式} FROM 表名**

実行例

```
db2=> SELECT SAL,SAL/2 FROM EMP WHERE EMPNO = 7369

SAL        2
---------- ---------------------------------
   800.00     400.00000000000000000000000000

  1 レコードが選択されました。
```

注意点

数値型の列に対して除算を行うことができます。式内の / (スラッシュ) は除算を意味します。
ゼロで割るとゼロ除算エラーになります。
Oracleは、式がそのまま検索結果の列名になります。
DB2 UDBは、列の番号が列名として表示されます。
10進数の結果の桁数 (常に最大の31になる) と小数部の長さに注意が必要です。小数部の長さが長すぎる場合、DECIMAL関数で桁数と小数部の長さを調節できます。

1.22 複数の列を1つにまとめる（連結）

文字列を連結して1つの文字列として返します。

Oracleの場合

書式 **SELECT 列名1 || {列名2 | リテラル} FROM 表名**

実行例

```
SQL> SELECT EMPNO,ENAME||JOB FROM EMP WHERE EMPNO = 7369;

     EMPNO ENAME||JOB
---------- --------------------
      7369 SMITHCLERK

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名1 || {列名2 | リテラル} FROM 表名**

実行例

```
db2=> SELECT EMPNO,ENAME||JOB FROM EMP WHERE EMPNO = 7369

EMPNO  2
------ --------------------
 7369. SMITHCLERK

  1 レコードが選択されました。
```

注意点 文字列を連結して1つの文字列として扱いたい場合は、文字列連結演算子の || を使用します。いくつでも文字列を連結することができます。
Oracle、DB2 UDBのいずれも、CONCAT関数を使用して2つの文字列を連結できます。複数の関数を組み合わせることも可能ですが、3つ以上の文字列の連結を行う場合は、文字列連結演算子の || が便利です。

1.23 n件だけデータを表示する

先頭の数件（n件）分のデータを表示して値を確認します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名 WHERE ROWNUM <= 件数
```

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE ROWNUM <= 3;

     EMPNO ENAME
---------- --------------------
      7369 SMITH
      7499 ALLEN
      7521 WARD

3行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名 FETCH FIRST 件数 ROWS ONLY
```

実行例

```
db2=> SELECT EMPNO,ENAME FROM EMP FETCH FIRST 3 ROWS ONLY

EMPNO  ENAME
------ ----------
 7369. SMITH
 7499. ALLEN
 7521. WARD

  3 レコードが選択されました。
```

注意点

データベースにどんな内容のデータが入っているのか、確認のため数件を表示したいときに便利です。
Oracleは、ROWNUM擬似列を使用します。ROWNUMは行を取り出すときに割り振られる値です。
DB2 UDBは、「FIRST n ROWS ONLY」キーワードを使用し、最初のn件を確認できます。

1.24 列の一意な値を表示する（重複した値は排除）

重複するものを取り除いて出力するので、値の種類を確認できます。

Oracleの場合

書式 **SELECT DISTINCT 列名 FROM 表名**

実行例

```
SQL> SELECT DISTINCT JOB FROM EMP;

JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

5行が選択されました。
```

DB2の場合

書式 **SELECT DISTINCT 列名 FROM 表名**

実行例

```
db2=> SELECT DISTINCT JOB FROM EMP

JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

  5 レコードが選択されました。
```

注意点 同じ値の行（重複行）を省いて1行だけ表示するため、どんな種類の値が格納されているかを確認する場合に便利です。

1.25 表名に別名を付けて検索する

長い表名、わかりづらい表名に別名を付けて検索します。

Oracle の場合

書式 **SELECT 別名.列名 FROM 表名 [AS] 別名**

実行例

```
SQL> SELECT E.EMPNO,E.ENAME FROM EMP E WHERE E.EMPNO = 7369;

     EMPNO ENAME
---------- --------------------
      7369 SMITH

1行が選択されました。

SQL>
```

DB2 の場合

書式 **SELECT 相関名.列名 FROM 表名 [AS] 相関名**

実行例

```
db2=> SELECT E.EMPNO,E.ENAME FROM EMP E WHERE E.EMPNO = 7369

EMPNO  ENAME
------ ----------
 7369. SMITH

  1 レコードが選択されました。
```

注意点 FROM 句において、表名の後ろに AS キーワードに続けて別名（相関名）を指定できます。AS は省略できます。長い表名やわかりづらい表名を簡略化することができるため、結合などを行う場合に便利です。なお、別名（相関名）は命名規則に従う必要があります。DB2 UDB では相関名と呼びます。

1.26 列名に別名を付けて検索する

列名に別名を付けて検索します。

Oracleの場合

書式 **SELECT 列名 [AS] 列別名 FROM 表名**

実行例

```
SQL> SELECT ENAME,SAL*12 AS NENSYU FROM EMP WHERE EMPNO = 7369;

ENAME                    NENSYU
-------------------- ----------
SMITH                      9600

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名 [AS] 列別名 FROM 表名**

実行例

```
db2=> SELECT ENAME,SAL*12 AS NENSYU FROM EMP WHERE EMPNO = 7369

ENAME      NENSYU
---------- --------------------
SMITH                   9600.00

  1 レコードが選択されました。
```

注意点 SELECT句では、列名の後ろにASキーワードを使用して列の別名を指定できます。AS句は省略できます。計算式や関数を使用した場合、その内容がわかるような表示名を指定できます。別名は命名規則に従う必要があります。

1.27 予約語を使用した列別名を指定する

検索結果の列名に命名規則に違反した別名を使用します。

Oracle の場合

書式 **SELECT 列名 [AS] "列別名" FROM 表名**

実行例

```
SQL> SELECT ENAME AS "SELECT" FROM EMP WHERE EMPNO = 7369;

SELECT
----------
SMITH

1行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 [AS] "列別名" FROM 表名**

実行例

```
db2=> SELECT ENAME AS "SELECT" FROM EMP WHERE EMPNO = 7369

SELECT
----------
SMITH

  1 レコードが選択されました。
```

注意点 原則として、別名は命名規則に従う必要があります。しかし、命名規則に違反した（予約語や大文字小文字の区別、記号を使用した）別名を利用する必要がある場合は、二重引用符（ " ）を使用して指定することができます。
Oracle、DB2 UDB のどちらも、列名の後ろに 1 つ以上の空白をあけて指定した名前は、列の別名として扱われるため、AS 句を省略することが可能です。

1.28 検索結果を昇順に並べ替える

行を昇順（小さい順）に並べ替えて表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名 [ASC]**

実行例

```
SQL> SELECT * FROM DEPT ORDER BY DNAME;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        40 OPERATIONS     BOSTON
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO

4行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名 [ASC]**

実行例

```
db2=> SELECT * FROM DEPT ORDER BY DNAME

DEPTNO DNAME          LOC
------ -------------- -------------
   10. ACCOUNTING     NEW YORK
   40. OPERATIONS     BOSTON
   20. RESEARCH       DALLAS
   30. SALES          CHICAGO

  4 レコードが選択されました。
```

注意点 行を並べ替えて表示する場合は、ORDER BY句を使用します。数値型、日付型および文字列型の列を使用して、昇順（小さい順）または降順（大きい順）に並べ替えて表示できます。ORDER BY句を省略した場合（デフォルト時）、昇順に並べ替えて表示します。

1.29 検索結果を降順に並べ替える

行を降順（大きい順）に並べ替えて表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名 DESC**

実行例

```
SQL> SELECT * FROM DEPT ORDER BY DNAME DESC;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        30 SALES          CHICAGO
        20 RESEARCH       DALLAS
        40 OPERATIONS     BOSTON
        10 ACCOUNTING     NEW YORK

4行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名 DESC**

実行例

```
db2=> SELECT * FROM DEPT ORDER BY DNAME DESC

DEPTNO DNAME          LOC
------ -------------- -------------
   30. SALES          CHICAGO
   20. RESEARCH       DALLAS
   40. OPERATIONS     BOSTON
   10. ACCOUNTING     NEW YORK

  4 レコードが選択されました。
```

注意点 行を並べ替えて表示する場合は、ORDER BY句を使用します。数値型、日付型および文字列型の列を使用して、昇順（小さい順）または降順（大きい順）に並べ替えて表示できます。降順に並べ替えて表示する場合は、ORDER BY句で指定した列名の後ろにDESC（DESCENDING）キーワードを指定します。

1.30 複数の列を指定し、検索結果を昇順に並べ替える

複数の列を指定して行を昇順（小さい順）に並べ替えて表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名1 [ASC],列名2 [ASC]**

実行例

```
SQL> SELECT DEPTNO,SAL FROM EMP WHERE JOB IN ('SALESMAN','MANAGER')
➥ORDER BY DEPTNO,SAL;

    DEPTNO        SAL
---------- ----------
        10       2450
        20       2975
        30       1250
        30       1250
        30       1500
        30       1600
        30       2850

7行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名1 [ASC],列名2 [ASC]**

実行例

```
db2=> SELECT DEPTNO,SAL FROM EMP WHERE JOB IN ('SALESMAN','MANAGER')
➥ORDER BY DEPTNO,SAL

DEPTNO SAL
------ ---------
   10.   2450.00
   20.   2975.00
   30.   1250.00
   30.   1250.00
   30.   1500.00
   30.   1600.00
   30.   2850.00

  7 レコードが選択されました。
```

注意点 同じ値を持つ行が複数ある場合、1つの列だけではなく別の列も使用して並べ替えたい場合があります。ORDER BY句において、複数の列をカンマ（,）で区切って指定すると左から順に並べ替えをする列が決まります。行を並べ替える列とSELECT句で指定する表示する列の順番に関連性はありません。
また、表示されないが検索した表にある列名も指定できます。

1.31 検索結果の１列目を昇順、２列目を降順に並べ替える

複数の列にそれぞれ昇順（小さい順）または降順（大きい順）を指定して並べ替えて表示します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名1 [ASC],列名2 DESC**

実行例

```
SQL> SELECT DEPTNO,SAL FROM EMP WHERE JOB IN ('SALESMAN','MANAGER')
➥ORDER BY DEPTNO,SAL DESC;

    DEPTNO        SAL
---------- ----------
        10       2450
        20       2975
        30       2850
        30       1600
        30       1500
        30       1250
        30       1250

7行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 列名1 [ASC],列名2 DESC**

実行例

```
db2=> SELECT DEPTNO,SAL FROM EMP WHERE JOB IN ('SALESMAN','MANAGER')
➥ORDER BY DEPTNO DESC,SAL DESC

DEPTNO SAL
------ ---------
   30.   2850.00
   30.   1600.00
   30.   1500.00
   30.   1250.00
   30.   1250.00
   20.   2975.00
   10.   2450.00

  7 レコードが選択されました。
```

注意点

同じ値を持つ行が複数ある場合、１つの列だけではなく別の列も使用して並べ替えたい場合があります。ORDER BY句において、複数の列をカンマ（,）で区切って指定すると左から順に並べ替える列が決まります。しかし、すべての行を昇順にまたはすべて降順に並べ替えるとは限りません。所属する部門番号の昇順に並べ替えてから、同じ部門内の行は給与の降順に並べ替えたい場合も考えられます。ORDER BY句で指定する列は、列ごとに昇順、降順を指定できます。昇順の場合は、列名の後ろにASC（ASCENDING）、降順の場合はDESC（DESCENDING）を指定します。決して、ORDER BY句全体で並び順を１つだけ指定するわけではありません。デフォルトでは昇順（ASC）で並べ替えます。

1.32 列番号を指定して検索結果を並べ替える

ORDER BY 句に列名ではなく、列番号を指定します。

Oracle の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 1,2・・・**

実行例

```
SQL> SELECT DEPTNO,SAL FROM EMP WHERE JOB IN ('SALESMAN','MANAGER')
➥ORDER BY 1,2;

    DEPTNO        SAL
---------- ----------
        10       2450
        20       2975
        30       1250
        30       1250
        30       1500
        30       1600
        30       2850

7行が選択されました。
```

DB2 の場合

書式 **SELECT 列名 FROM 表名 ORDER BY 1,2・・・**

実行例

```
db2=> SELECT DEPTNO,SAL FROM EMP WHERE JOB IN ('SALESMAN','MANAGER')
➥ORDER BY 1,2

DEPTNO SAL
------ ---------
   10.   2450.00
   20.   2975.00
   30.   1250.00
   30.   1250.00
   30.   1500.00
   30.   1600.00
   30.   2850.00

  7 レコードが選択されました。
```

注意点 SELECT 句で指定した列は、左から 1、2 ・・・と列番号が割り当てられます。ORDER BY 句ではこの列番号を使用して並べ替える順番を指定できます。長い列名やわかりづらい列名の場合は、列番号を使用するほうが SQL 文が読みやすくなる場合があります。また、SELECT 句に指定する列が実行時に変わることが考えられる場合、「どの列が指定されても左の列から昇順に並べて表示する」とすることができ柔軟なコーディングを実現できます。

1.33 検索結果を列別名で並べ替える

SELECT句で指定した列の別名を使用して行を並べ替えます。

Oracleの場合

書式 **SELECT 列名1 AS 別名1 FROM 表名 ORDER BY 別名1**

実行例

```
SQL> SELECT ENAME,SAL*12 AS NENSYU FROM EMP WHERE DEPTNO = 10 ORDER BY
➥NENSYU;

ENAME                     NENSYU
-------------------- ----------
MILLER                     15600
CLARK                      29400
KING                       60000

3行が選択されました。
```

DB2の場合

書式 **SELECT 列名1 AS 別名1 FROM 表名 ORDER BY 別名1**

実行例

```
db2=> SELECT ENAME,SAL*12 AS NENSYU FROM EMP WHERE DEPTNO = 10 ORDER BY
➥NENSYU

ENAME      NENSYU
---------- --------------------
MILLER                  15600.00
CLARK                   29400.00
KING                    60000.00

  3 レコードが選択されました。
```

注意点 長い列名やわかりづらい列名がある場合、SELECT句で列の別名を指定すると便利です。この別名はORDER BY句でも使用できます。計算式の結果で行を並べ替えたい場合は、列別名を使用するほうが、SQL文が読みやすくなります。また、式に用いる値を変更した場合もSELECT句のみ変更し、ORDER BY句は変更する必要がないので柔軟なコーディングを実現できます。

1.34 階層構造データの検索（上から順に）

階層構造になったデータを階層の上から順に出力します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名 CONNECT BY PRIOR 親行の列 = 子行の列
START WITH 列名 = 階層の始点を表す値
```

実行例

```
SQL> SELECT LEVEL,LPAD(' ',LEVEL*2,' ')||ENAME AS ENAME,DEPTNO FROM EMP
  2  CONNECT BY PRIOR EMPNO = MGR START WITH JOB = 'PRESIDENT';

     LEVEL ENAME                    DEPTNO
---------- -------------------- ----------
         1   KING                       10
         2     JONES                    20
         3       SCOTT                  20
         4         ADAMS                20
         3       FORD                   20
         4         SMITH                20
         2     BLAKE                    30
         3       ALLEN                  30
         3       WARD                   30
         3       MARTIN                 30
         3       TURNER                 30
         3       JAMES                  30
         2     CLARK                    10
         3       MILLER                 10

14行が選択されました。
```

注意点

Oracleは、CONNECT BY句およびSTART WITH句を使用して階層関係にある行を表示できます。CONNECT BY句で行と行の親子関係を指定（PRIORキーワードを親行の値に指定）し、START WITH句でSQL文内で表示したい階層の始点（TOP）はどの行であるか指定します。このとき、階層の深さを表すLEVEL擬似列を使用できます。また、階層の深さを明示的に表現するために、LPAD関数を使ってインデント表示させることもできます。

DB2の場合

書式

```
WITH 共通表式名 AS (全選択 UNION ALL SELECT ··· FROM 共通表式名[, 表名]
WHERE 親行の列 = 子行の列) SELECT ···
```

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH
Recurse (level, empno, ename, deptno, ancestors) AS (
SELECT 1, empno, ename, deptno, CAST(CHAR(ename, 10) AS VARCHAR(100))
  FROM EMP
 WHERE job = 'PRESIDENT'
UNION ALL
SELECT level + 1
     , new.empno, new.ename, new.deptno
     , pre.ancestors || CHAR(new.ename, 10)
  FROM Recurse pre
     , EMP     new
```

```
 WHERE level < 10
   AND pre.empno = new.mgr
)
SELECT level, SUBSTR('                    ', 1, level*2) || ename as
ename, deptno
  FROM Recurse
 ORDER BY
       ancestors;
--------------------------------------------------------------------------

LEVEL       ENAME                           DEPTNO
----------- ------------------------------- ------
          1   KING                              10
          2     BLAKE                           30
          3       ALLEN                         30
          3       JAMES                         30
          3       MARTIN                        30
          3       TURNER                        30
          3       WARD                          30
          2     CLARK                           10
          3       MILLER                        10
          2     JONES                           20
          3       FORD                          20
          4         SMITH                       20
          3       SCOTT                         20
          4         ADAMS                       20

  14 レコードが選択されました。
```

注意点 | Oracleは、同じレベルのデータは表内の物理順に並んでいますが、DB2 UDBでは物理的並び（順番）とはなりません。出力の順番はORDER BY句で指定しなかったら不定となります。ただし、指定したSQLによっては、表スペースの順番や索引の順番に並ぶこともあります。たとえば、次の単純なSELECT文では、物理的並び（INSERT順）に行番号（ROWNUM）が付けられています。

```
------------------------------ 入力コマンド ------------------------------
SELECT e.*
     , ROWNUMBER() OVER() rownum
  FROM EMP e;
------------------------------------------------------------------------------
SELECT e.* , ROWNUMBER() OVER() rownum FROM EMP e

EMPNO  ENAME      JOB       MGR    HIREDATE   SAL       COMM      DEPTNO ROWNUM
------ ---------- --------- ------ ---------- --------- --------- ------ --------------------
  7369 SMITH      CLERK       7902 1980-12-17    800.00         -     20                    1
  7499 ALLEN      SALESMAN    7698 1981-02-20   1600.00    300.00     30                    2
  7521 WARD       SALESMAN    7698 1981-02-22   1250.00    500.00     30                    3
  7566 JONES      MANAGER     7839 1981-04-02   2975.00         -     20                    4
  7654 MARTIN     SALESMAN    7698 1981-09-28   1250.00   1400.00     30                    5
  7698 BLAKE      MANAGER     7839 1981-05-01   2850.00         -     30                    6
  7782 CLARK      MANAGER     7839 1981-06-09   2450.00         -     10                    7
  7788 SCOTT      ANALYST     7566 1982-12-09   3000.00         -     20                    8
  7839 KING       PRESIDENT      - 1981-11-17   5000.00         -     10                    9
  7844 TURNER     SALESMAN    7698 1981-09-08   1500.00      0.00     30                   10
  7876 ADAMS      CLERK       7788 1983-01-12   1100.00         -     20                   11
  7900 JAMES      CLERK       7698 1981-12-03    950.00         -     30                   12
  7902 FORD       ANALYST     7566 1981-12-03   3000.00         -     20                   13
  7934 MILLER     CLERK       7782 1982-01-23   1300.00         -     10                   14

  14 record(s) selected.
```

Oracleの実行結果と同じ並びにするには、次のように実行します。

```
------------------------------ 入力コマンド ------------------------------
WITH
Sequed (empno, ename, job, mgr, hiredate, sal, comm, deptno, rownum) AS (
SELECT e.*
     , ROWNUMBER() OVER() rownum
  FROM EMP e
)
,
Recurse (level, empno, ename, deptno, ancestors) AS (
SELECT 1, empno, ename, deptno, CAST(DIGITS(SMALLINT(rownum)) AS
VARCHAR(60))
  FROM Sequed
 WHERE job = 'PRESIDENT'
UNION ALL
SELECT level + 1
     , new.empno, new.ename, new.deptno
     , pre.ancestors || DIGITS(SMALLINT(rownum))
  FROM Recurse pre
     , Sequed  new
 WHERE level < 10
   AND pre.empno = new.mgr
)
SELECT level, SUBSTR('                    ', 1, level*2) || ename as
ename, deptno
  FROM Recurse
 ORDER BY
       ancestors;
------------------------------------------------------------------------

LEVEL       ENAME                          DEPTNO
----------- ------------------------------ ------
          1   KING                             10
          2     JONES                          20
          3       SCOTT                        20
          4         ADAMS                      20
          3       FORD                         20
          4         SMITH                      20
          2     BLAKE                          30
          3       ALLEN                        30
          3       WARD                         30
          3       MARTIN                       30
          3       TURNER                       30
          3       JAMES                        30
          2     CLARK                          10
          3       MILLER                       10

  14レコードが選択されました。
```

1.35 特定の階層を取り除いた階層構造データの検索（上から順に）

階層構造を持つデータから、特定の階層を取り除いて表示します。

Oracle の場合

書式

```
SELECT 列名 FROM 表名
CONNECT BY PRIOR 親行の列 = 子行の列 AND 対象階層行の検索条件
START WITH 列名 = 階層の始点を表す値
```

実行例

```
SQL> SELECT LEVEL,LPAD(' ',LEVEL*2,' ')||ENAME AS ENAME,DEPTNO FROM   EMP
  2  CONNECT BY PRIOR EMPNO = MGR AND ENAME <> 'JONES' START WITH JOB =
➥'PRESIDENT';

     LEVEL ENAME                     DEPTNO
---------- -------------------- ----------
         1   KING                         10
         2     BLAKE                      30
         3       ALLEN                    30
         3       WARD                     30
         3       MARTIN                   30
         3       TURNER                   30
         3       JAMES                    30
         2     CLARK                      10
         3       MILLER                   10

9行が選択されました。
```

注意点

WHERE句で対象外と指定された場合はその条件に一致する行だけが該当しますが、CONNECT BY PRIOR句で指定した場合は、その条件に一致する行より下の階層に含まれる行すべてが該当します。したがって、特定の階層だけを取り除いて表示できます。

DB2 の場合

書式

```
WITH 共通表式名 AS (全選択 UNION ALL SELECT ··· FROM 共通表式名[, 表名]
WHERE 親行の列 = 子行の列 AND 対象階層行の検索条件) SELECT ···
```

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH Recurse (level, empno, ename, deptno, ancestors) AS (
SELECT 1, empno, ename, deptno, CAST(CHAR(ename, 10) AS VARCHAR(100))
  FROM EMP
 WHERE job = 'PRESIDENT'
UNION ALL
SELECT level + 1
     , new.empno, new.ename, new.deptno
     , pre.ancestors || CHAR(new.ename, 10)
  FROM Recurse pre
     , EMP     new
 WHERE level < 10
   AND pre.empno = new.mgr
   AND new.ename <> 'JONES'
)
SELECT level, SUBSTR('                    ', 1, level*2) || ename as
➥ename, deptno
  FROM Recurse
```

```
 ORDER BY
       ancestors;
--------------------------------------------------------------------------------

LEVEL        ENAME                           DEPTNO
---------- ------------------------------ ------
         1   KING                               10
         2     BLAKE                            30
         3       ALLEN                          30
         3       JAMES                          30
         3       MARTIN                         30
         3       TURNER                         30
         3       WARD                           30
         2     CLARK                            10
         3       MILLER                         10
```

9レコードが選択されました。

1.36 特定の階層を選択した階層構造データの検索（上から順に）

階層構造になったデータを、階層の最上位のデータから順に関連性を表すのではなく、途中の階層だけ表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名 CONNECT BY PRIOR 親行の列 = 子行の列
START WITH 列名 = 階層の始点を表す値
```

実行例

```
SQL> SELECT LEVEL,LPAD(' ',LEVEL*2,' ')||ENAME AS ENAME,DEPTNO FROM EMP
  2  CONNECT BY PRIOR EMPNO = MGR START WITH ENAME = 'JONES';

     LEVEL ENAME                     DEPTNO
---------- -------------------- ----------
         1   JONES                       20
         2     SCOTT                     20
         3       ADAMS                   20
         2     FORD                      20
         3       SMITH                   20

5行が選択されました。
```

注意点

START WITH句で指定した行を階層の始点として扱います。したがって、階層関係にあるデータを最上位からではなく、階層の途中から表示したい場合は、始点（TOP）にする行を指定します。

DB2の場合

書式

```
WITH 共通表式名 AS (全選択 UNION ALL SELECT ··· FROM 共通表式名[, 表名]
WHERE 親行の列 = 子行の列 AND 対象階層行の検索条件) SELECT ···
```

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH Recurse (level, empno, ename, deptno, ancestors) AS (
SELECT 1, empno, ename, deptno, CAST(CHAR(ename, 10) AS VARCHAR(100))
  FROM EMP
 WHERE ename = 'JONES'
UNION ALL
SELECT level + 1
     , new.empno, new.ename, new.deptno
     , pre.ancestors || CHAR(new.ename, 10)
  FROM Recurse pre
     , EMP     new
 WHERE level < 10
   AND pre.empno = new.mgr
)
SELECT level, SUBSTR('                    ', 1, level*2) || ename as
➥ename, deptno
  FROM Recurse
 ORDER BY
       ancestors;
------------------------------------------------------------------------------

LEVEL       ENAME                          DEPTNO
```

```
---------- ------------------------------ ------
         1 JONES                            20
         2   FORD                           20
         3     SMITH                        20
         2   SCOTT                          20
         3     ADAMS                        20
```

5 レコードが選択されました。

注意点 「WITH 共通表式名 AS (全選択 UNION ALL ･･･」の全選択で指定した行を階層の始点（TOP）として扱います（実行例では、WHERE ename = 'JONES' で視点となる行を指定しています）。したがって、階層関係にあるデータを最上位からではなく、階層の途中から表示したい場合は、始点（TOP）にする行を指定します。

1.37 階層構造データの検索（下から順に）

階層構造になったデータを階層の下から順に関連性を表します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名 CONNECT BY PRIOR 子行の列 = 親行の列
START WITH 列名 = 階層の始点を表す値
```

実行例

```
SQL> SELECT LEVEL,LPAD(' ',LEVEL*2,' ')||ENAME AS ENAME,DEPTNO  FROM   EMP
  2  CONNECT BY PRIOR MGR = EMPNO START WITH ENAME = 'SMITH';

     LEVEL ENAME                     DEPTNO
---------- -------------------- ----------
         1   SMITH                       20
         2     FORD                      20
         3       JONES                   20
         4         KING                  10
```

4行が選択されました。

注意点

CONNECT BY句においてPRIORキーワードを子行の値に指定します。さらに、START WITH句に関連性をたどる始点となる行を指定します。LEVEL擬似列は、始点を1として割り当てられています。

DB2の場合

書式

```
WITH 共通表式名 AS (全選択 UNION ALL SELECT ··· FROM 共通表式名[, 表名]
WHERE 親行の列 = 子行の列) SELECT ···
```

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH Recurse (level, empno, ename, deptno, mgr, ancestors) AS (
SELECT 1, empno, ename, deptno, mgr, CAST(CHAR(ename, 10) AS VARCHAR(100))
  FROM EMP
 WHERE ename = 'SMITH'
UNION ALL
SELECT level + 1
     , new.empno, new.ename, new.deptno
     , new.mgr
     , pre.ancestors || CHAR(new.ename, 10)
  FROM Recurse pre
     , EMP     new
 WHERE level < 10
   AND pre.mgr = new.empno
)
SELECT level, SUBSTR('                    ', 1, level*2) || ename as
➥ename, deptno
  FROM Recurse
 ORDER BY
       ancestors;
------------------------------------------------------------------------

LEVEL       ENAME                          DEPTNO
----------- ------------------------------ ------
          1   SMITH                            20
```

```
2      FORD                          20
3        JONES                       20
4          KING                      10
```

4 レコードが選択されました。

1.38 基準日からの経過日数を表示する

経過日数を求めます。

Oracle の場合

書式 **SELECT 日付型列1 - 日付型列2 FROM 表名**

実行例

```
SQL> SELECT TO_DATE('2005-03-31','YYYY-MM-DD')-TO_DATE('2005-01-01','YYYY-
➥MM-DD') FROM DUAL;

TO_DATE('2005-03-31','YYYY-MM-DD')-TO_DATE('2005-01-01','YYYY-MM-DD')
---------------------------------------------------------------------
                                                                   89

1行が選択されました。
```

DB2 の場合

書式 **SELECT DAYS(日付型列1) - DAYS(日付型列2) FROM 表名**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DAYS('2005-03-31') - DAYS('2005-01-01') FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

1
-----------
         89

  1 レコードが選択されました。
```

注意点 日付データは単一引用符（' '）で囲んで表現することができますが、そのまま計算式に指定すると文字列として扱われます。日付型に変換する関数を使用して計算式に含めます。
Oracle では、「日付型1－日付型2」は日数単位で計算されます。

1.39 検索条件に単一引用符（'）を使用する

文字列中の単一引用符（'）を探します。

Oracleの場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 LIKE '%''%'**

実行例

```
SQL> SELECT COL FROM TEST WHERE COL LIKE '%''%';

COL
--------------------
I'm a students

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名 WHERE 列名 LIKE '%''%'**

実行例

```
db2=> SELECT COL FROM TEST WHERE COL LIKE '%''%'

COL
--------------------
I'm a students

  1 レコードが選択されました。
```

注意点 文字列中の単一引用符（'）を検索条件として指定する場合は、単一引用符（'）を2つ続けて記述し、「''」と指定します。

1.40 %を含む文字列を検索する

文字列中のパーセント(%)を探します。アンダースコア(_)も同じように探すことができます。

Oracleの場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 LIKE 'エスケープ文字%' escape ' エスケープ文字'
```

実行例

```
SQL> SELECT COL FROM TEST WHERE COL LIKE '%5¥%' ESCAPE '¥';

COL
--------------------
TAX RATE 5%

1行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 LIKE 'エスケープ文字%' escape ' エスケープ文字'
```

実行例

```
db2=> SELECT COL FROM TEST WHERE COL LIKE '%5¥%' ESCAPE '¥'

COL
--------------------
TAX RATE 5%

  1 レコードが選択されました。
```

注意点

アンダースコア(_)やパーセント(%)は、あいまい検索時のワイルドカードとして使用していますが、データ内の値として「_」や「%」が含まれている場合があります。そこで、ワイルドカードとしてのパーセントなのか、値としてのパーセントなのかを区別する必要があります。この区別には、エスケープ文字を指定します。エスケープ文字の後ろに指定された値はワイルドカードではなく値として処理されるようになりますます。

2

データの集計

2.1 平均値を求める

平均値を求めます。

Oracle の場合

書式 **SELECT AVG(数値型列名) FROM 表名**

実行例

```
SQL> SELECT AVG(SAL) FROM EMP;

  AVG(SAL)
----------
2021.66667

1行が選択されました。
```

DB2 の場合

書式 **SELECT AVG(数値型列名) FROM 表名**

実行例

```
db2=> SELECT AVG(SAL) FROM EMP

1
--------------------------------
  2021.6666666666666666666666666

  1 レコードが選択されました。
```

注意点 AVG 関数の引数には数値型の列を指定します。NULL 値以外の行を対象にして平均値を求めます。

2.2 表内の行数のカウント

表に何行あるのかを調べます。

Oracleの場合

書式 **SELECT COUNT(*) FROM 表名**

実行例

```
SQL> SELECT COUNT(*) FROM EMP;

  COUNT(*)
----------
        15

1行が選択されました。
```

DB2の場合

書式 **SELECT COUNT(*) FROM 表名**

実行例

```
db2=> SELECT COUNT(*) FROM EMP

1
-----------
         15

  1 レコードが選択されました。
```

注意点 AVG（平均）、SUM（合計）、MAX（最大）、MIN（最小）、COUNT（カウント）の各集約関数は、NULL値以外の行を処理対象にします。しかし、COUNT関数の引数にアスタリスク（*）を指定した場合は、表内の全行数（NULL値の行を含む）をカウントします。

2.3 列内の値のカウント

指定した列内にNULL値以外の行が何件あるかを調べます。

Oracleの場合

書式 **SELECT COUNT(列名) FROM 表名**

実行例

```
SQL> SELECT COUNT(JOB),COUNT(COMM) FROM EMP;

COUNT(JOB) COUNT(COMM)
---------- -----------
        15           4

1行が選択されました。
```

DB2の場合

書式 **SELECT COUNT(列名) FROM 表名**

実行例

```
db2=> SELECT COUNT(JOB),COUNT(COMM) FROM EMP

1           2
----------- -----------
         15           4

  1 レコードが選択されました。
```

注意点 | COUNT関数の引数にはどのようなデータ型の列を指定してもかまいません。NULL値以外の行を処理対象にし、行数をカウントします。

2.4 列内の異なる値のカウント

異なる値が何種類あるのかを調べます。

Oracleの場合

書式 **SELECT COUNT(DISTINCT 列名) FROM 表名**

実行例

```
SQL> SELECT COUNT(DISTINCT JOB) FROM EMP;

COUNT(DISTINCTJOB)
------------------
                 5

1行が選択されました。
```

DB2の場合

書式 **SELECT COUNT(DISTINCT 列名) FROM 表名**

実行例

```
db2=> SELECT COUNT(DISTINCT JOB) FROM EMP

1
-----------
          5

  1 レコードが選択されました。
```

注意点 COUNT関数の引数にはどのようなデータ型の列を指定してもかまいません。NULL値以外の行を処理対象にし、行数をカウントします。そこで、DISTINCTキーワードを使用し、重複行を排除してから行数をカウントすると、異なる値が何種類あるのかを調べることができます。

2.5 最大値を求める

最大値を求めます。

Oracle の場合

書式 **SELECT MAX(列名) FROM 表名**

実行例

```
SQL> SELECT MAX(SAL) FROM EMP;

  MAX(SAL)
----------
      5000

1行が選択されました。
```

DB2 の場合

書式 **SELECT MAX(列名) FROM 表名**

実行例

```
db2=> SELECT MAX(SAL) FROM EMP

1
---------
  5000.00

  1 レコードが選択されました。
```

注意点 MAX関数は、NULL値以外の行を処理対象にして最大値を求めます。列の値が文字列であれば文字コードの最も大きい値、列の値が日付であれば最新の日付を最大値として求めます。

2.6 最小値を求める

最小値を求めます。

Oracle の場合

書式 **SELECT MIN(列名) FROM 表名**

実行例

```
SQL> SELECT MIN(SAL) FROM EMP;

  MIN(SAL)
----------
       800

1行が選択されました。
```

DB2 の場合

書式 **SELECT MIN(列名) FROM 表名**

実行例

```
db2=> SELECT MIN(SAL) FROM EMP

1
---------
   800.00

  1 レコードが選択されました。
```

注意点 MIN 関数は、NULL 値以外の行を処理対象にして最小値を求めます。列の値が文字列であれば文字コードの最も小さい値、列の値が日付であれば最も古い（昔の）日付を最小値として求めます。

2.7 合計を求める

合計を求めます。

Oracle の場合

書式 **SELECT SUM(数値型列名) FROM 表名**

実行例

```
SQL> SELECT SUM(SAL) FROM EMP;

  SUM(SAL)
----------
     30325

1行が選択されました。
```

DB2 の場合

書式 **SELECT SUM(数値型列名) FROM 表名**

実行例

```
db2=> SELECT SUM(SAL) FROM EMP

1
--------------------------------
                          30325.00

  1 レコードが選択されました。
```

注意点 SUM 関数の引数には数値型の列を指定します。NULL 値以外の行を処理対象にして合計値を求めます。

2.8 検索結果をグループ化して表示する

同じ値を持つ行を1つのグループとして扱います。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY グループ化する列名
```

実行例

```
SQL> SELECT DEPTNO,SUM(SAL) FROM EMP GROUP BY DEPTNO;

    DEPTNO   SUM(SAL)
---------- ----------
        10       8750
        20      10875
        30       9400
                 1300

4行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY グループ化する列名
```

実行例

```
db2=> SELECT DEPTNO,SUM(SAL) FROM EMP GROUP BY DEPTNO

DEPTNO 2
------ ---------------------------------
   10.                            8750.00
   20.                           10875.00
   30.                            9400.00
     -                            1300.00

  4 レコードが選択されました。
```

注意点 グループ化したい列をGROUP BY句に指定し、グループごとの集約関数の結果を求めることができます。指定した列の同じ値を持つ行を1つのグループとして扱います。NULL値も1つのグループとして扱います。GROUP BY句を指定してしない場合は、表を1つのグループとして扱います。

2.9 集合値に条件を付ける

グループ化して求めた結果に対して検索条件を指定します。

Oracle の場合

書式

SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY グループ化する列名 HAVING 集約関数(列名)を使用した検索条件

実行例

```
SQL> SELECT DEPTNO,SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING SUM(SAL)
➥>=10000;

    DEPTNO   SUM(SAL)
---------- ----------
        20      10875

1行が選択されました。
```

DB2 の場合

書式

SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY グループ化する列名 HAVING 集約関数(列名)を使用した検索条件

実行例

```
db2=> SELECT DEPTNO,SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING SUM(SAL)
➥>=10000

DEPTNO 2
------ ---------------------------------
   20.                          10875.00

  1 レコードが選択されました。
```

注意点 | グループごとの集約関数の結果に対して検索条件を指定することができます。HAVING 句で指定できるのは、集約関数または GROUP BY 句で指定した列のみです。

2.10 重複しているレコードの件数を検索する

同じ値を複数持つ行が何件あるか求めます。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY グループ化する列名 HAVING 集約関数(列名) > 1
```

実行例

```
SQL> SELECT JOB,COUNT(*) FROM EMP GROUP BY JOB HAVING COUNT(*) >1;

JOB          COUNT(*)
--------- ----------
ANALYST             2
CLERK               5
MANAGER             3
SALESMAN            4

4行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY グループ化する列名 HAVING 集約関数(列名) > 1
```

実行例

```
db2=> SELECT JOB,COUNT(*) FROM EMP GROUP BY JOB HAVING COUNT(*) >1

JOB        2
--------- -----------
ANALYST              2
CLERK                5
MANAGER              3
SALESMAN             4

  4 レコードが選択されました。
```

注意点 重複した値がないかどうかを調べたい列をGROUP BY句に指定します。そして、グループに対する行数が1より大きいの場合、値が重複していることがわかります。

2.11 重複したレコードを除いて表示する

ほかに同じ値を持つ行がない行（ユニークな行）を表示します。

Oracleの場合

書式

```
SELECT グループ化する列名 FROM 表名
GROUP BY グループ化する列名 HAVING 集約関数(列名) =1
```

実行例

```
SQL> SELECT JOB FROM EMP GROUP BY JOB HAVING COUNT(*)=1;

JOB
---------
PRESIDENT

1行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名  FROM 表名
GROUP BY グループ化する列名 HAVING  集約関数(列名) =1
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT JOB FROM EMP GROUP BY JOB HAVING COUNT(*) = 1;
--------------------------------------------------------------------------

JOB
---------
PRESIDENT

  1 レコードが選択されました。
```

注意点 重複した値がないかどうか調べたい列をGROUP BY句に指定します。そして、グループに対する行数が1の場合、重複した値を持っていないことになります。

2.12 指定した行だけをグループ化して表示する

指定した条件を満たす行だけグループ化して表示します。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名 WHERE 検索条件
GROUP BY グループ化する列名
```

実行例

```
SQL> SELECT DEPTNO,COUNT(*) FROM EMP WHERE DEPTNO >= 20 GROUP BY DEPTNO;

    DEPTNO   COUNT(*)
---------- ----------
        20          5
        30          6

2行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名 WHERE 検索条件
GROUP BY グループ化する列名
```

実行例

```
db2=> SELECT DEPTNO,COUNT(*) FROM EMP WHERE DEPTNO >= 20 GROUP BY DEPTNO

DEPTNO 2
------ -----------
   20.            5
   30.            6

  2 レコードが選択されました。
```

注意点 グループ処理する対象行をWHERE句で指定します。WHERE句には、FROM句で指定した表内のすべての列を使用することができます。WHERE句で指定した検索条件を満たす行だけにグループ化の処理が行われます。

2.13 小計と総計

グループごとの小計と全行の総計を求めます。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY ROLLUP(グループ化する列名1, グループ化する列名2)
```

実行例

```
SQL> SELECT DEPTNO,JOB,SUM(SAL) FROM EMP WHERE DEPTNO IS NOT NULL GROUP BY
➥ROLLUP(DEPTNO,JOB);

    DEPTNO JOB         SUM(SAL)
---------- --------- ----------
        10 CLERK           1300
        10 MANAGER         2450
        10 PRESIDENT       5000
        10                 8750
        20 CLERK           1900
        20 ANALYST         6000
        20 MANAGER         2975
        20                10875
        30 CLERK            950
        30 MANAGER         2850
        30 SALESMAN        5600
        30                 9400
                          29025

13行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数 AS 列別名] FROM 表名
GROUP BY ROLLUP(グループ化する列名1, グループ化する列名2)
ORDER BY {グループ化する列名 | 集約関数の列別名}
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO,JOB,SUM(SAL)
  FROM EMP
 WHERE DEPTNO IS NOT NULL
 GROUP BY ROLLUP(DEPTNO,JOB)
 ORDER BY DEPTNO, JOB;
------------------------------------------------------------------------------

DEPTNO JOB       3
------ --------- ---------------------------------
    10 CLERK                              1300.00
    10 MANAGER                            2450.00
    10 PRESIDENT                          5000.00
    10 -                                  8750.00
    20 ANALYST                            6000.00
    20 CLERK                              1900.00
    20 MANAGER                            2975.00
    20 -                                 10875.00
    30 CLERK                               950.00
    30 MANAGER                            2850.00
    30 SALESMAN                           5600.00
```

```
30 -                                        9400.00
 - -                                       29025.00

13 レコードが選択されました。
```

注意点

「GROUP BY ROLLUP(DEPTNO, JOB)」では、DEPTNO列とJOB列によりグループ化が行われ、DEPTNOごとの小計と全行の総計が求められます。通常、ORDER BY句を指定しなくても、GROUP BY句で指定した列をもとに昇順で表示されます。しかし、GROUP BY ROLLUP句では、OracleとDB2 UDBのどちらも同じ行が表示されていますが、行の順番が異なっている点に注意してください。特にDB2 UDBは、ORDER BY句を指定しないと順序が保証されません。このため、いつでもORDER BY句を付けるようにしておくとよいでしょう。

2.14 第1グループの小計、第2グループの小計と総計

それぞれのグループごとの小計と総計を求めます。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY CUBE(グループ化する列名1, グループ化する列名2)
```

実行例

```
SQL> SELECT DEPTNO,JOB,SUM(SAL) FROM EMP WHERE DEPTNO IS NOT NULL GROUP BY
➥CUBE(DEPTNO,JOB);

    DEPTNO JOB          SUM(SAL)
---------- --------- ----------
                          29025
           CLERK           4150
           ANALYST         6000
           MANAGER         8275
           SALESMAN        5600
           PRESIDENT       5000
        10                 8750
        10 CLERK           1300
        10 MANAGER         2450
        10 PRESIDENT       5000
        20                10875
        20 CLERK           1900
        20 ANALYST         6000
        20 MANAGER         2975
        30                 9400
        30 CLERK            950
        30 MANAGER         2850
        30 SALESMAN        5600

18行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY CUBE(グループ化する列名1, グループ化する列名2)
ORDER BY {グループ化する列名 | 集約関数の列別名}
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO,JOB,SUM(SAL)
  FROM EMP
 WHERE DEPTNO IS NOT NULL
 GROUP BY CUBE(DEPTNO,JOB)
 ORDER BY COALESCE(DEPTNO, 0), COALESCE(JOB, '');
--------------------------------------------------------------------------

DEPTNO JOB       SUM(SAL)
------ --------- ----------------------------------
     - -                                   29025.00
     - ANALYST                              6000.00
     - CLERK                                4150.00
     - MANAGER                              8275.00
     - PRESIDENT                            5000.00
     - SALESMAN                             5600.00
```

```
10 -                                      8750.00
10 CLERK                                  1300.00
10 MANAGER                                2450.00
10 PRESIDENT                              5000.00
20 -                                     10875.00
20 ANALYST                                6000.00
20 CLERK                                  1900.00
20 MANAGER                                2975.00
30 -                                      9400.00
30 CLERK                                   950.00
30 MANAGER                                2850.00
30 SALESMAN                               5600.00
```

18 レコードが選択されました。

注意点

「GROUP BY CUBE(DEPTNO,JOB)」では、DEPTNO列とJOB列によりグループ化が行われ、DEPTNOごとの小計とJOBごとの小計および全行の総計が求められます。通常、ORDER BY句を指定しなくても、GROUP BY句で指定した列の昇順に行は並んで表示されます。しかし、GROUP BY CUBE句では、OracleとDB2 UDBのどちらも同じ行が表示されていますが、行の順番が異なっている点に注意してください。特にDB2 UDBは、ORDER BY句を指定しないと順序が保証されません。このため、いつでもORDER BY句を付けるようにしておくとよいでしょう。

DB2 UDBでOracleと同じようにNULL値を順序の最初にするには、COALESCE関数を使います。

2.15 複数グループごとの小計

それぞれのグループごとの集計値を求めます。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY GROUPING SETS(グループ化する列名1, グループ化する列名2)
```

実行例

```
SQL> SELECT DEPTNO,JOB,SUM(SAL) FROM EMP WHERE DEPTNO IS NOT NULL GROUP BY
➥GROUPING SETS(DEPTNO,JOB);

    DEPTNO JOB          SUM(SAL)
---------- --------- ----------
        10                 8750
        20                10875
        30                 9400
           ANALYST         6000
           CLERK           4150
           MANAGER         8275
           PRESIDENT       5000
           SALESMAN        5600

8行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY GROUPING SETS(グループ化する列名1, グループ化する列名2)
ORDER BY {グループ化する列名 | 集約関数の列別名}
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO,JOB,SUM(SAL)
  FROM EMP
 WHERE DEPTNO IS NOT NULL
 GROUP BY GROUPING SETS(DEPTNO,JOB)
 ORDER BY DEPTNO,JOB;
--------------------------------------------------------------------------

DEPTNO JOB       3
------ --------- -----------------------------------
    10 -                                    8750.00
    20 -                                   10875.00
    30 -                                    9400.00
     - ANALYST                              6000.00
     - CLERK                                4150.00
     - MANAGER                              8275.00
     - PRESIDENT                            5000.00
     - SALESMAN                             5600.00

  8 レコードが選択されました。
```

注意点

「GROUP BY GROUPING SETS(DEPTNO, JOB)」では、DEPTNO列とJOB列によりグループ化が行われ、DEPTNOごとの小計とJOBごとの小計が求められます。通常、ORDER BY句を指定しなくても、GROUP BY句で指定した列の昇順に行は並んで表示されます。しかし、GROUP BY GROUPING SET句では、OracleとDB2 UDBのどちらも同じ行が表示されていますが、行の順番が異なっている点に注意してください。特にDB2 UDBは、ORDER BY句を指定しないと順序が保証されません。このため、いつでもORDER BY句を付けるようにしておくとよいでしょう。

2.16 年を単位にグルーピングした結果を求める

日付型列の一部の値である「年」単位にグルーピングした結果を求めます。

Oracleの場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名
GROUP BY TO_CHAR(日付型列,'YY')
```

実行例

```
SQL> SELECT TO_CHAR(HIREDATE,'YY'),SUM(SAL) FROM EMP GROUP BY TO_CHAR(
➥HIREDATE,'YY');

TO   SUM(SAL)
-- ----------
80         800
81       22825
82        4300
83        2400

4行が選択されました。
```

DB2の場合

書式

```
SELECT グループ化する列名 [,集約関数] FROM 表名 GROUP BY YEAR(日付型列)
```

実行例

```
db2=> SELECT YEAR(HIREDATE),SUM(SAL) FROM EMP GROUP BY YEAR(HIREDATE)

1           2
----------- ---------------------------------
       1980                            800.00
       1981                          22825.00
       1982                           4300.00
       1983                           2400.00

  4 レコードが選択されました。
```

注意点 日付型は「年月日」を保持しているため、「年」単位にグルーピングするには、日付型列から「年」だけを取り出さなければなりません。GROUP BY句では、関数を使用した列を指定できます。

2.17 指定した列の値が最大値のレコードを検索する

指定した列の値が最大値である行を求めます。

Oracleの場合

書式

```
SELECT 列名1 FROM 表名1
WHERE 列名2 = (SELECT MAX(列名2) FROM 表名1)
```

実行例

```
SQL> SELECT ENAME,SAL FROM EMP WHERE SAL = (SELECT MAX(SAL) FROM EMP);

ENAME                          SAL
-------------------- ----------
KING                         5000

1行が選択されました。
```

DB2の場合

書式

```
SELECT 列名1 FROM 表名1
WHERE 列名2 = (SELECT MAX(列名2) FROM 表名1)
```

実行例

```
db2=> SELECT ENAME,SAL FROM EMP WHERE SAL = (SELECT MAX(SAL) FROM EMP)

ENAME      SAL
---------- ---------
KING         5000.00

  1 レコードが選択されました。
```

注意点 | この結果を求めるためには副問い合わせを使用します。

3

関数の利用方法

3.1 ABS — 絶対値を求める

引数として与えたnの絶対値を戻します。

Oracleの場合

書式 **ABS(n)**

実行例
```
SQL> SELECT ABS(-20) FROM DUAL;

  ABS(-20)
----------
        20
```

DB2の場合

書式 **ABS(n)**

実行例
```
db2=> SELECT ABS(-20) FROM SYSIBM.SYSDUMMY1

1
-----------
         20

  1 レコードが選択されました。
```

Column

サンプルUDF

Oracleの関数に直接対応するDB2の関数がなくても、書き方の工夫で同じ結果を得ることができる場合があります。また、同じ機能を持つUDF（User Definition Function：ユーザー定義関数）を作成できる場合もあります。

一部のOracle関数に対しては、ほぼ同等の機能を持つDB2 UDFのサンプルが提供されている場合があります。ただし、あくまでもサンプルで動作を保証するものではありません。たとえば、次のサイトにそのようなサンプルUDF（SQL）がいくつかあります。また、IBM DB2 Migration Toolkit（MTK）を使用してOracleで開発したSQLを移行するとUDFが生成される場合もあります。

● Sample UDFs for Migration
http://www.ibm.com/developerworks/db2/library/samples/db2/0205udfs/index.html

● MTKの説明
http://www.ibm.com/jp/software/data/developer/products/db2_v7_4.html#migration

● MTKのダウンロード
http://www.ibm.com/software/data/db2/migration/mtk/

3.2 ASCII — 最初の文字の10進表記を求める

文字列の最初の文字の10進表記を戻します。文字コードは、既定のデータベース・キャラクター・セットとなります。

Oracleの場合

書式 **ASCII(文字列)**

実行例

```
SQL> SELECT ASCII('A') FROM DUAL;

ASCII('A')
----------
        65
```

DB2の場合

書式 **ASCII(文字列)**

実行例

```
db2=> SELECT ASCII('A') FROM SYSIBM.SYSDUMMY1

1
-----------
         65

  1 レコードが選択されました。
```

3.3 ASCIISTR — 文字列のASCII表記を求める

任意のキャラクター・セットの文字列を引数として取り、その文字列をASCIIキャラクター・セットで戻します。

Oracleの場合

書式 **ASCIISTR(文字列)**

実行例

```
SQL> SELECT ASCIISTR('1234'),ASCIISTR('。「 」、') FROM DUAL;

ASCI ASCIISTR('。「 」、')
---- --------------------
1234 ¥2160¥2161¥2162¥2163
```

DB2の場合

書式 **対応する関数はありません。**

実行例

注意点 | OracleのASCIISTR関数は、ASCII文字以外のデータを¥xxxxという書式に変換します。このとき、xxxxはUTF-16のコードを表します。

3.4 CHR — 指定された文字コードを文字に変換する

CHRは、データベース・キャラクター・セットまたは各国語キャラクター・セット（USINGNCHAR_CSを指定している場合）の中のnに等しい2進数を持つ文字をVARCHAR2型の値として戻します。

Oracleの場合

書式 **CHR(n)**

実行例

```
SQL> SELECT CHR(65),CHR(66),CHR(67) FROM DUAL;

C C C
- - -
A B C
```

DB2の場合

書式 **CHR(n)**

実行例

```
db2=> SELECT CHR(65),CHR(66),CHR(67) FROM SYSIBM.SYSDUMMY1

1 2 3
- - -
A B C

  1 レコードが選択されました。
```

注意点 | DB2 UDBの場合は、指定されたASCIIコードの文字を戻します。

3.5 NCHR — 指定された文字コードを文字に変換する

各国語キャラクター・セットのnumberと同等のバイナリを持つ文字を戻します。このファンクションは、CHRファンクションにUSING NCHAR_CS句を指定して使用した場合と同じ結果を戻します。

Oracleの場合

書式 **NCHR(n)**

実行例

```
SQL> SELECT NCHR(65),NCHR(66),NCHR(67) FROM DUAL;

NC NC NC
-- -- --
A  B  C
```

DB2の場合

書式 **対応する関数はありません。**

実行例

3.6 COALESCE

その値がNULL値以外の最初の引数を戻します。結果は、すべての引数がNULL値の場合のみNULL値になります。

Oracleの場合

書式 **COALESCE(引数1,引数2,・・・)**

実行例

```
SQL> SELECT COMM,MGR,EMPNO,COALESCE(COMM,MGR,EMPNO) FROM EMP
  2  WHERE DEPTNO IN (10,30);

      COMM         MGR       EMPNO COALESCE(COMM,MGR,EMPNO)
---------- ---------- ---------- ------------------------
       300       7698       7499                      300
       500       7698       7521                      500
      1400       7698       7654                     1400
                 7839       7698                     7839
                 7839       7782                     7839
                            7839                     7839
         0       7698       7844                        0
                 7698       7900                     7698
                 7782       7934                     7782

9行が選択されました。
```

DB2の場合

書式 **COALESCE(引数1,引数2,・・・)**

実行例

```
db2=> SELECT COMM,MGR,EMPNO,COALESCE(COMM,MGR,EMPNO) FROM EMP WHERE
➥DEPTNO IN (10,30)

COMM       MGR    EMPNO  4
---------- ------ ------ ----------
   300.00  7698.  7499.     300.00
   500.00  7698.  7521.     500.00
  1400.00  7698.  7654.    1400.00
         -  7839.  7698.    7839.00
         -  7839.  7782.    7839.00
         -       -  7839.   7839.00
     0.00  7698.  7844.       0.00
         -  7698.  7900.    7698.00
         -  7782.  7934.    7782.00

  9 レコードが選択されました。
```

3.7 CONCAT — 文字列を連結する

文字列1と文字列2を連結した結果を戻します。

Oracleの場合

書式 **CONCAT(文字列1,文字列2)**

実行例

```
SQL> SELECT CONCAT('POST','MAN') FROM DUAL;

CONCAT(
-------
POSTMAN
```

DB2の場合

書式 **CONCAT(文字列1,文字列2)**

実行例

```
db2=> SELECT CONCAT('POST','MAN') FROM SYSIBM.SYSDUMMY1

1
-------
POSTMAN

  1 レコードが選択されました。
```

注意点 通常は、文字連結演算子の || を用いたほうがいいでしょう。特に、3個以上の文字列を連結する場合、CONCAT関数を用いると関数をネストしなければならず、コードが見にくくなります。

3.8 COUNT — 件数をカウントする

問い合わせによって戻されたNULL値以外の行の数を戻します。

Oracleの場合

書式 **COUNT([DISTICT | ALL] 列名または*)**

実行例

```
SQL> SELECT COUNT(*) FROM EMP;

  COUNT(*)
----------
        15

1行が選択されました。
```

DB2の場合

書式 **COUNT([DISTICT | ALL] 列名または*)**

実行例

```
db2=> SELECT COUNT(*) FROM EMP

1
-----------
         15

  1 レコードが選択されました。
```

注意点 AVG（平均）、SUM（合計）、MAX（最大）、MIN（最小）、COUNT（カウント）の各集約関数は、NULL値以外の行を処理対象にします。しかし、COUNT関数の引数にアスタリスク（*）を指定した場合は、表内の全行数（NULL値の行を含む）をカウントします。

3.9 CURRENT_DATE

セッション・タイムゾーンの現在の日付をDATEデータ型のグレゴリオ暦の値で戻します。

Oracleの場合

書式 **CURRENT_DATE**

実行例

```
SQL> SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;

SESSIONTIMEZONE      CURRENT_
-------------------- --------
+09:00               05-03-30
```

DB2の場合

書式 **CURRENT_DATE または CURRENT DATE**

実行例

```
db2=> SELECT CURRENT_TIMEZONE,CURRENT_DATE FROM SYSIBM.SYSDUMMY1

1        2
-------- ----------
  90000. 2005-03-30

  1 レコードが選択されました。
```

3.10 CURRENT_TIMESTAMP

セッション・タイムゾーンの現在の日付および時刻をOracleはTIMESTAMP WITH TIME ZONEデータ型の値で戻します。

Oracleの場合

書式 **CURRENT_TIMESTAMP**

実行例

```
SQL> SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;

SESSIONTIMEZONE      CURRENT_TIMESTAMP
-------------------- ---------------------------------------
+09:00               05-03-30 19:03:09.337000 +09:00
```

DB2の場合

書式 **CURRENT_TIMESTAMP または CURRENT TIMESTAMP**

実行例

```
db2=> SELECT CURRENT_TIMEZONE,CURRENT_TIMESTAMP FROM SYSIBM.SYSDUMMY1

1        2
-------- --------------------------
  90000. 2005-03-30-19.02.12.525001

  1 レコードが選択されました。
```

注意点 | DB2 UDBでは、ZONE付きではなくタイムスタンプ型で値を戻します。

3.11 DBTIMEZONE

セッション・タイムゾーンの現在の日付および時刻をTIMESTAMP WITH TIME ZONEデータ型の値で戻します。

Oracleの場合

書式 **DBTIMEZONE**

実行例

```
SQL> SELECT DBTIMEZONE FROM DUAL;

DBTIME
------
+00:00
```

DB2の場合

書式 **CURRENT TIMEZONE**

実行例

```
db2=> SELECT CURRENT TIMEZONE FROM SYSIBM.SYSDUMMY1

1
--------
  90000.

  1 レコードが選択されました。
```

注意点 Oracleの場合は、データベースのタイムゾーンを戻します。DB2 UDBの場合は、SQLを実行したクライアント（アプリケーション・サーバー）のタイムゾーンを戻します。また、OracleとDB2 UDBとでは戻されるデータ型が違います。

3.12 DECODE

指定している条件と値を1つずつ比較します。条件が値と等しい場合、対応する結果を戻します。一致する値が見つからない場合は、デフォルトを戻します。デフォルトが省略されている場合は、NULL値を戻します。

Oracleの場合

書式 **DECODE(列，条件1，値1，条件2，値2，･･･デフォルト)**

実行例

```
SQL> SELECT DEPTNO,DECODE(DEPTNO,10,'財務',20,'企画','営業')
  2  FROM EMP WHERE JOB = 'MANAGER';

    DEPTNO DECO
---------- ----
        20 企画
        30 営業
        10 財務
```

DB2の場合

書式 **対応する関数はありませんが、CASE式で同等の処理が可能です。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO
     , CASE DEPTNO
       WHEN 10 THEN '財務'
       WHEN 20 THEN '企画'
       ELSE '営業'
       END
  FROM EMP
 WHERE JOB = 'MANAGER';
------------------------------------------------------------------------------

DEPTNO 2
------ ----
    20 企画
    30 営業
    10 財務

  3 レコードが選択されました。
```

注意点 | DB2 UDBではSQL標準のCASE文で書きます。

3.13 DENSE_RANK

順序付けされた行のグループ内の行のランクを計算し、そのランクを数値として戻します。ランクは1から始まる連続した整数です。同じ値が合った場合は同じランクになります。

Oracleの場合

書式

```
DENSE_RANK ( ) OVER ( ORDER BY 列名 {DESC|ASC} )
```

実行例

```
SQL> SELECT empno, ename, sal
     , DENSE_RANK() OVER(ORDER BY sal) drank
  FROM EMP
 WHERE deptno = 30
 ORDER BY drank, empno;
```

DB2の場合

書式

```
DENSE_RANK ( ) OVER ( ORDER BY 列名 {DESC|ASC} )
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT empno, ename, sal
     , DENSE_RANK() OVER(ORDER BY sal) drank
  FROM EMP
 WHERE deptno = 30
 ORDER BY drank, empno;
------------------------------------------------------------------------------

EMPNO  ENAME      SAL       DRANK
------ ---------- --------- --------------------
  7900 JAMES         950.00                    1
  7521 WARD         1250.00                    2
  7654 MARTIN       1250.00                    2
  7844 TURNER       1500.00                    3
  7499 ALLEN        1600.00                    4
  7698 BLAKE        2850.00                    5

  6 レコードが選択されました。
```

3.14 EXTRACT

日時または期間値の式から、指定された日時フィールドの値を抽出します。

Oracle の場合

書式 **EXTRACT({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND | TIMEZONE_HOUR | TIMEZONE_MINUTE | TIMEZONE_REGION | TIMEZONE_ABBR} FROM { 日時型 | インターバル型})**

実行例

```
SQL> SELECT EXTRACT(YEAR FROM HIREDATE) FROM EMP WHERE EMPNO = 7369;

EXTRACT(YEARFROMHIREDATE)
-------------------------
                     1980
```

DB2 の場合

書式 **YEAR | MONTH | DAY | HOUR | MINUTE | SECOND | MICROSECOND(日付型 | 時刻型 | タイムスタンプ型)**

実行例

```
db2=> SELECT YEAR(HIREDATE) FROM EMP WHERE EMPNO = 7369

1
-----------
       1980

  1 レコードが選択されました。
```

注意点 Oracle では、TO_CHAR 関数を使用して日時フィールドの値を抽出することも可能です。
DB2 UDB では、分、マイクロ秒を抽出する関数として、HOUR()、MINUTE()、SECOND() があります。

3.15 FIRST

与えられたソート指定に対して、FIRST またはLAST としてランク付けされた一連の行の一連の値を操作する集約関数および分析関数です。

Oracle の場合

書式

```
aggregate_function KEEP
(DENSE_RANK FIRST ORDER BY expr {DESC|ASC} [NULLS] { FIRST | LAST })
[OVER query_partition_clause]
```

実行例 部署ごとに、最も入社の古い人の内で最も低い給与を表示する

```
SQL> SELECT DEPTNO,
  2  MIN(SAL) KEEP (DENSE_RANK FIRST ORDER BY HIREDATE) "Worst"
  3  FROM EMP
  4  WHERE DEPTNO IS NOT NULL
  5  GROUP BY DEPTNO;

    DEPTNO      Worst
---------- ----------
        10       2450
        20        800
        30       1600
```

DB2 の場合

書式 対応する関数はありませんが、次のようにして同じ結果を得ることができます。

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO
     , MIN(SAL) "Worst"
  FROM (SELECT DEPTNO
             , SAL
             , DENSE_RANK() OVER(PARTITION BY DEPTNO
                                 ORDER BY HIREDATE) drank
          FROM EMP
         WHERE DEPTNO IS NOT NULL
       ) Q
 WHERE drank = 1
 GROUP BY DEPTNO;
------------------------------------------------------------------------------

DEPTNO Worst
------ ---------
    10   2450.00
    20    800.00
    30   1600.00

  3 レコードが選択されました。
```

3.16 FIRST_VALUE

順序付けられた値の集合にある最初の値を戻します。集合内の最初の値がNULLの場合、IGNORE NULLSを指定していないかぎり、この関数はNULLを戻します。

Oracleの場合

書式 **FIRST_VALUE (expr [IGNORE NULLS]) OVER (analytic_clause)**

実行例 部署10に所属する社員の給与と最も少ない給与の人の名前を表示する

```
SQL> SELECT DEPTNO, ENAME, SAL, FIRST_VALUE(ENAME)
  2  OVER (ORDER BY SAL ASC ROWS UNBOUNDED PRECEDING) AS lowest_sal
  3  FROM (SELECT * FROM EMP WHERE DEPTNO = 10 ORDER BY EMPNO);

    DEPTNO ENAME              SAL LOWEST_SAL
---------- ---------- ---------- ----------
        10 MILLER           1300 MILLER
        10 CLARK            2450 MILLER
        10 KING             5000 MILLER
```

DB2の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH Q AS (
SELECT * FROM EMP WHERE DEPTNO = 10
)
SELECT DEPTNO, ENAME, SAL
     , (SELECT ENAME
          FROM (SELECT ENAME
                     , ROWNUMBER() OVER(ORDER BY SAL ASC, EMPNO) rn
                  FROM Q
               ) P
         WHERE rn = 1) AS lowest_sal
  FROM Q
 ORDER BY SAL, EMPNO;
------------------------------------------------------------------------

DEPTNO ENAME      SAL       LOWEST_SAL
------ ---------- --------- ----------
    10 MILLER       1300.00 MILLER
    10 CLARK        2450.00 MILLER
    10 KING         5000.00 MILLER

  3 レコードが選択されました。
```

注意点 | DB2 UDBでは、標準のOLAP関数の組み合わせで書くことができます。

3.17 FROM_TZ

タイムスタンプ値およびタイムゾーンをTIMESTAMP WITH TIME ZONE値に変換します。

Oracleの場合

書式 **FROM_TZ (timestamp_value , time_zone_value)**

実行例

```
SQL> SELECT FROM_TZ(TIMESTAMP '2005-03-28 08:00:00', '3:00')
  2  FROM DUAL;

FROM_TZ(TIMESTAMP'2005-03-2808:00:00','3:00')
---------------------------------------------------------------------------
05-03-28 08:00:00.000000000 +03:00
```

DB2の場合

書式 **TIMESTAMP WITH TIME ZONEデータ型がないため対応する関数はありません。同様の処理を実行するには次のようにします。**

実行例

```
db2=> SELECT CURRENT TIMESTAMP, CURRENT TIMEZONE FROM SYSIBM.SYSDUMMY1

1                          2
-------------------------- --------
2005-11-14-23.03.03.937001   90000.

  1 レコードが選択されました。
```

3.18 GROUP_ID

GROUP BY 指定の結果から、重複するグループを識別します。この関数は、問い合わせ結果から重複グループを除外する場合に有効です。

Oracle の場合

書式 **GROUP_ID()**

実行例

```
SQL> SELECT DEPTNO,JOB,SUM(SAL),GROUP_ID()
  2  FROM EMP WHERE DEPTNO IS NOT NULL
  3  GROUP BY DEPTNO,ROLLUP(DEPTNO,JOB);

    DEPTNO JOB         SUM(SAL) GROUP_ID()
---------- --------- ---------- ----------
        10 CLERK            1300          0
        10 MANAGER          2450          0
        10 PRESIDENT        5000          0
        20 CLERK            1900          0
        20 ANALYST          6000          0
        20 MANAGER          2975          0
        30 CLERK             950          0
        30 MANAGER          2850          0
        30 SALESMAN         5600          0
        10                  8750          0
        20                 10875          0
        30                  9400          0
        10                  8750          1
        20                 10875          1
        30                  9400          1

15行が選択されました。
```

DB2 の場合

書式 **同じ機能の関数はありませんが、次のようにして同じ結果を得ることができます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO1 AS DEPTNO, JOB, DEC(SAL,10,2) SAL, GROUP_ID
  FROM (SELECT DEPTNO1, DEPTNO2, JOB, SUM(SAL) SAL
             , GROUPING(DEPTNO2) GROUP_ID
             , GROUPING(JOB) G_JOB
          FROM (SELECT DEPTNO DEPTNO1, DEPTNO DEPTNO2, JOB, SAL
                  FROM EMP
WHERE DEPTNO IS NOT NULL
               ) Q
         GROUP BY DEPTNO1, ROLLUP(DEPTNO2, JOB)
       ) R
ORDER BY GROUP_ID, G_JOB, DEPTNO1, JOB;
------------------------------------------------------------------------

DEPTNO JOB       SAL          GROUP_ID
------ --------- ------------ -----------
    10 CLERK          1300.00           0
    10 MANAGER        2450.00           0
    10 PRESIDENT      5000.00           0
    20 ANALYST        6000.00           0
```

```
 20 CLERK          1900.00          0
 20 MANAGER        2975.00          0
 30 CLERK           950.00          0
 30 MANAGER        2850.00          0
 30 SALESMAN       5600.00          0
 10 -              8750.00          0
 20 -             10875.00          0
 30 -              9400.00          0
 10 -              8750.00          1
 20 -             10875.00          1
 30 -              9400.00          1
```

15 レコードが選択されました。

3.19 GROUPING

GROUPINGファンクションの引数は、GROUP BY句の式のいずれかと一致する必要があります。行の列名（式）の値がすべての値の集合を表すNULLの場合、このファンクションは1の値を戻します。それ以外の場合は、0（ゼロ）を戻します。
通常のグループ化された行と超集合行を区別します。ROLLUPやCUBEなどのGROUP BY句の拡張機能は、すべての値の集合がNULLで表される超集合行を生成します。
GROUPINGファンクションを使用すると、通常の行に含まれるNULLと超集合行ですべての値の集合を表すNULLを区別できます。

Oracleの場合

書式 **GROUPING(列名)**

実行例

```
SQL> SELECT DECODE(GROUPING(DEPTNO),1,'合計',DEPTNO) AS DEPTNO
  2         ,SUM(SAL)
  3  FROM EMP
  4  WHERE DEPTNO IS NOT NULL
  5  GROUP BY ROLLUP(DEPTNO);

DEPTNO       SUM(SAL)
---------- ----------
10               8750
20              10875
30               9400
合計            29025
```

DB2の場合

書式 **GROUPING(式)**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CASE GROUPING(DEPTNO)
       WHEN 1 THEN '合計'
       ELSE CHAR(DEPTNO)
       END  AS DEPTNO
     , SUM(SAL) "SUM(SAL)"
  FROM EMP
 WHERE DEPTNO IS NOT NULL
 GROUP BY ROLLUP(DEPTNO)
 ORDER BY DEPTNO;
--------------------------------------------------------------------------------

DEPTNO SUM(SAL)
------ ---------------------------------
10                               8750.00
20                              10875.00
30                               9400.00
合計                            29025.00

  4 レコードが選択されました。
```

注意点 | 指定できる式はGROUP BYで指定した式です（式でグループ化できるため）。

3.20 GROUPING_ID

行に関連するGROUPINGビット・ベクトル（1と0を組み合せた文字列）に対応する数値を戻します。

Oracleの場合

書式 **GROUPING_ID()**

実行例

```
SQL> SELECT DEPTNO,JOB,SUM(SAL),GROUPING_ID(DEPTNO,JOB)
  2  FROM EMP
  3  WHERE DEPTNO IS NOT NULL
  4  GROUP BY ROLLUP(DEPTNO,JOB);

    DEPTNO JOB         SUM(SAL) GROUPING_ID(DEPTNO,JOB)
---------- --------- ---------- -----------------------
        10 CLERK           1300                       0
        10 MANAGER         2450                       0
        10 PRESIDENT       5000                       0
        10                 8750                       1
        20 CLERK           1900                       0
        20 ANALYST         6000                       0
        20 MANAGER         2975                       0
        20                10875                       1
        30 CLERK            950                       0
        30 MANAGER         2850                       0
        30 SALESMAN        5600                       0
        30                 9400                       1
                          29025                       3

13行が選択されました。
```

DB2の場合

書式 **GROUP BY ROLLUP(式1, 式2, ... , 式n)ならば、**
GROUPING(式n)+GROUPING(式n-1)*2+ ... + GROUPING(式1)*POWER(2,n-1)

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO,JOB,SUM(SAL)
     , GROUPING(JOB)+GROUPING(DEPTNO)*2 Grouping_ID
  FROM EMP
 WHERE DEPTNO IS NOT NULL
 GROUP BY ROLLUP(DEPTNO,JOB)
 ORDER BY DEPTNO, JOB;
--------------------------------------------------------------------------

DEPTNO JOB       3                                   GROUPING_ID
------ --------- ---------------------------------- -----------
    10 CLERK                                1300.00           0
    10 MANAGER                              2450.00           0
    10 PRESIDENT                            5000.00           0
    10 -                                    8750.00           1
    20 ANALYST                              6000.00           0
    20 CLERK                                1900.00           0
    20 MANAGER                              2975.00           0
    20 -                                   10875.00           1
    30 CLERK                                 950.00           0
    30 MANAGER                              2850.00           0
```

```
30 SALESMAN                    5600.00          0
30 -                           9400.00          1
 - -                          29025.00          3

13 レコードが選択されました。
```

注意点 | DB2 UDB の場合、GROUPING_ID を作成して使用するより、必要な GROUPING 関数だけを組み合わせて式を組み立てたほうが簡単です。

3.21 LAG

その位置より前にある指定された物理オフセットにある行へアクセスします。これは、自己結合せずに、表の1つ以上の行へ同時アクセスを行います

Oracleの場合

書式

```
LAG ( value_expr[, offset ][, default ])
OVER ([query_partition_clause] order_by_clause )
```

実行例

```
SQL> SELECT ENAME,HIREDATE,SAL
  2         ,LAG(SAL,1) OVER (ORDER BY HIREDATE) AS PREV
  3  FROM EMP
  4  WHERE DEPTNO = 10;

ENAME      HIREDATE         SAL       PREV
---------- -------- ---------- ----------
CLARK      81-06-09       2450
KING       81-11-17       5000       2450
MILLER     82-01-23       1300       5000
```

DB2の場合

書式

対応する関数はありませんが、次のようにして同じ結果を得ることができます。

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT ENAME,HIREDATE,SAL
     , MAX(SAL) OVER (ORDER BY HIREDATE
ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING) AS PREV
   FROM EMP
  WHERE DEPTNO = 10;
------------------------------------------------------------------------------

ENAME      HIREDATE   SAL        PREV
---------- ---------- ---------- ----------
CLARK      1981-06-09   2450.00          -
KING       1981-11-17   5000.00    2450.00
MILLER     1982-01-23   1300.00    5000.00

  3 レコードが選択されました。
```

3.22 LAST

与えられたソート指定に対して、FIRSTまたはLASTとしてランク付けされた一連の行の一連の値を操作する集約関数および分析関数です。

Oracleの場合

書式

```
aggregate_function KEEP
(DENSE_RANK LAST ORDER BY  expr {DESC|ASC}[NULLS] {FIRST | LAST})
[OVER query_partition_clause]
```

実行例

```
SQL> SELECT DEPTNO,
  2  MAX(SAL) KEEP (DENSE_RANK LAST ORDER BY HIREDATE) "Best"
  3  FROM EMP
  4  WHERE DEPTNO IS NOT NULL
  5  GROUP BY DEPTNO;

    DEPTNO       Best
---------- ----------
        10       1300
        20       1100
        30        950
```

DB2の場合

書式

対応する関数はありませんが、次のようにして同じ結果を得ることができます。

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO
     , MAX(SAL) "Best"
  FROM (SELECT EMP.*
             , DENSE_RANK() OVER(PARTITION BY DEPTNO
                                 ORDER BY HIREDATE DESC) drank
          FROM EMP) Q
 WHERE DEPTNO IS NOT NULL
   AND drank = 1
 GROUP BY DEPTNO;
------------------------------------------------------------------------------

DEPTNO Best
------ ---------
    10   1300.00
    20   1100.00
    30    950.00

  3 レコードが選択されました。
```

3.23 LAST_VALUE

順序付けられた値の集合にある最後の値を戻します。集合内の最初の値がNULL値の場合、IGNORE NULLSを指定していないかぎり、この関数はNULLを戻します。

Oracleの場合

書式 **LAST_VALUE (expr [IGNORE NULLS]) OVER (analytic_clause)**

実行例

```
SQL>SELECT ename, sal, hiredate, LAST_VALUE(hiredate) OVER
(ORDER BY sal
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS lv
FROM (SELECT * FROM EMP WHERE deptno = 10
ORDER BY hiredate);
```

DB2の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH Q AS (
SELECT * FROM EMP WHERE deptno = 10
)
SELECT ename, sal, hiredate
     , (SELECT hiredate
          FROM (SELECT hiredate
                     , ROWNUMBER()
                         OVER(ORDER BY sal DESC, hiredate DESC) AS rn
                  FROM Q
                ) AS P
         WHERE rn = 1) AS lv
  FROM Q
 ORDER BY sal, hiredate;
------------------------------------------------------------------------

ENAME      SAL       HIREDATE   LV
---------- --------- ---------- ----------
MILLER       1300.00 1982-01-23 1981-11-17
CLARK        2450.00 1981-06-09 1981-11-17
KING         5000.00 1981-11-17 1981-11-17

  3 レコードが選択されました。
```

3.24 LEAD

その位置より後ろにある指定された物理オフセットの行へアクセスします。これは、自己結合せずに、表の1つ以上の行へ同時アクセスを行います

Oracleの場合

書式

```
LEAD ( value_expr[, offset ][, default])
OVER ([query_partition_clause] order_by_clause )
```

実行例

```
SQL> SELECT ENAME,HIREDATE,SAL
  2        ,LEAD(SAL,1) OVER (ORDER BY HIREDATE) AS NEXT
  3  FROM EMP
  4  WHERE DEPTNO = 10;

ENAME      HIREDATE         SAL       NEXT
---------- -------- ---------- ----------
CLARK      81-06-09       2450       5000
KING       81-11-17       5000       1300
MILLER     82-01-23       1300
```

DB2の場合

書式

対応する関数はありませんが、次のようにして同じ結果を得ることができます。

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT ENAME,HIREDATE,SAL
     , MAX(SAL) OVER(ORDER BY HIREDATE
                     ROWS BETWEEN 1 FOLLOWING AND  1 FOLLOWING) AS NEXT
  FROM EMP
 WHERE DEPTNO = 10;
--------------------------------------------------------------------------

ENAME      HIREDATE   SAL        NEXT
---------- ---------- ---------- ----------
CLARK      1981-06-09   2450.00   5000.00
KING       1981-11-17   5000.00   1300.00
MILLER     1982-01-23   1300.00         -

  3 レコードが選択されました。
```

3.25 LENGTH — 文字列の長さを返す

文字を単位として文字列の長さを戻します。

Oracle の場合

書式 **LENGTH(文字列)**

実行例

```
SQL> SELECT LENGTH('ABC'),LENGTH('あいう') FROM DUAL;

LENGTH('ABC') LENGTH('あいう')
------------- ----------------
            3                3
```

DB2 の場合

書式 **対応する関数はありません。**

実行例

注意点

Oracle には、文字列の長さを文字単位で戻す LENGTH 関数とバイト単位で戻す LENGTHB 関数があります。
DB2 UDB にも、LENGTH 関数がありますが、文字単位ではなくバイト単位で結果を戻します。

3.26 LENGTHB — 文字列の長さを返す

バイトを単位として文字列の長さを戻します。

Oracle の場合

書式 **LENGTHB(文字列)**

実行例

```
SQL> SELECT LENGTHB('ABC'),LENGTHB('あいう') FROM DUAL;

LENGTHB('ABC') LENGTHB('あいう')
-------------- -----------------
             3                 6
```

DB2 の場合

書式 **LENGTH(文字列)**

実行例

```
db2=> SELECT LENGTH('ABC'),LENGTH('あいう') FROM SYSIBM.SYSDUMMY1

1           2
----------- -----------
          3           6

  1 レコードが選択されました。
```

注意点 Oracle には、文字列の長さを文字単位で戻す LENGTH 関数とバイト単位で戻す LENGTHB 関数があります。
DB2 UDB にも、LENGTH 関数がありますが、文字単位ではなくバイト単位で結果を戻します。

3.27 LTRIM ― 文字列の先頭の指定された文字列を削除する

文字列1の左端から、文字列2に指定されたすべての文字を削除します。文字列2を指定しない場合、デフォルトでは空白1個を削除します。

Oracleの場合

書式 **LTRIM(文字列1,文字列2)**
文字列1 ―― 対象となる文字列
文字列2 ―― 削除する文字列（省略すると空白）

実行例

```
SQL> SELECT LTRIM('ERASER','ER') FROM DUAL;

LTRI
----
ASER
```

DB2の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
WITH
Recurse (seq, str1) AS (
VALUES (0, 'ERASER')
UNION ALL
SELECT new_seq, SUBSTR(str1,2)
  FROM (SELECT seq+1 AS new_seq, str1
             , LOCATE(SUBSTR(str1,1,1),'ER') AS exist
          FROM Recurse
         WHERE seq < 10000
       ) Q
 WHERE exist > 0
)
SELECT str1
  FROM Recurse
 WHERE seq = (SELECT MAX(seq) FROM Recurse);
------------------------------------------------------------------------

STR1
------
ASER

  1 レコードが選択されました。
```

注意点 DB2 UDBにも、LTRIM関数はありますが、Oracleと完全な互換性はありません。DB2 UDBのLTRIM関数は引数を1つだけとり、文字列の先頭（左側）から空白を削除します。
MTK（70ページ参照）では、ora8.LTRIM関数をサンプルUDFとして提供しています。

●ユーザー定義関数（UDF）の作成例

```
------------------------------ Commands Entered ------------------------------
CREATE FUNCTION MyOra.LTRIM(str1 VARCHAR(4000), srch VARCHAR(4000))
RETURNS VARCHAR(4000)
CONTAINS SQL
```

```
BEGIN ATOMIC
MAIN_LOOP:
FOR ltrim_loop AS
SELECT SEQ + 1 AS pos
  FROM (SELECT N4*1000+N3*100+N2*10+N1 AS SEQ
          FROM (VALUES 0,1,2,3,4,5,6,7,8,9) P1(N1)
             , (VALUES 0,1,2,3,4,5,6,7,8,9) P2(N2)
             , (VALUES 0,1,2,3,4,5,6,7,8,9) P3(N3)
             , (VALUES 0,1,2,3) P4(N4)
       ) Q
 WHERE SEQ < LENGTH(str1)
 ORDER BY
       SEQ
DO
   IF LOCATE(SUBSTR(str1,pos,1), srch) = 0 THEN
      RETURN SUBSTR(str1,pos);
   END IF;
END FOR;
END@
------------------------------------------------------------------------------
DB20000I  The SQL command completed successfully.
```

●ユーザー定義関数（UDF）の使用例

```
------------------------------ Commands Entered ------------------------------
SELECT SUBSTR(MyOra.LTRIM(' xyxXxyLAST WORD',' xy'),1,20) AS "LTRIM example"
  FROM SYSIBM.SYSDUMMY1;
------------------------------------------------------------------------------

LTRIM example
--------------------
XxyLAST WORD

  1 record(s) selected.
```

3.28 LOWER — すべての文字列を小文字にする

すべての文字列を小文字に変換して戻します。

Oracle の場合

書式 **LOWER(文字列)**

実行例

```
SQL> SELECT LOWER('ABC') FROM DUAL;

LOW
---
abc
```

DB2 の場合

書式 **LOWER(文字列) または LCASE(文字列)**

実行例

```
db2=> SELECT LOWER('ABC') FROM SYSIBM.SYSDUMMY1

1
---
abc

  1 レコードが選択されました。
```

または

```
db2=> SELECT LCASE('ABC') FROM SYSIBM.SYSDUMMY1

1
---
abc

  1 レコードが選択されました。
```

3.29 MAX — 最大値を求める

最大値を戻します。

Oracleの場合

書式 **MXA(列名)**

実行例
```
SQL> SELECT MAX(SAL) FROM EMP;

  MAX(SAL)
----------
      5000

1行が選択されました。
```

DB2の場合

書式 **MAX(列名)**

実行例
```
db2=> SELECT MAX(SAL) FROM EMP

1
---------
  5000.00

  1 レコードが選択されました。
```

注意点 MAX関数は、NULL値以外の行を処理対象にして最大値を求めます。列の値が文字列であれば文字コードの最も大きい値、列の値が日付であれば最新の日付を最大値として求めます。

3.30 MIN — 最小値を求める

最小値を戻します。

Oracle の場合

書式 **MIN(列名)**

実行例

```
SQL> SELECT MIN(SAL) FROM EMP;

  MIN(SAL)
----------
       800

1行が選択されました。
```

DB2 の場合

書式 **MIN(列名)**

実行例

```
db2=> SELECT MIN(SAL) FROM EMP

1
---------
   800.00

  1 レコードが選択されました。
```

注意点 MIN 関数は、NULL 値以外の行を処理対象にして最小値を求めます。列の値が文字列であれば文字コードの最も小さい値、列の値が日付であれば最も古い（昔の）日付を最小値として求めます。

3.31 MOD — 余りを求める

mをnで割った余りを戻します。nが0の場合は、mを戻します。

Oracleの場合

書式 **MOD(m,n)**

実行例

```
SQL> SELECT MOD(13,4) FROM DUAL;

 MOD(13,4)
----------
         1
```

DB2の場合

書式 **MOD(m,n)**

実行例

```
db2=> SELECT MOD(13,4) FROM SYSIBM.SYSDUMMY1

1
-----------
          1

  1 レコードが選択されました。
```

注意点 | DB2 UDBでは、nが0だとエラーになります。

3.32 POWER — べき乗を求める

mをn乗した値を戻します。mおよびnは任意の数です。ただし、mが負の場合、nは整数である必要があります。

Oracleの場合

書式 **POWER(m,n)**

実行例

```
SQL> SELECT POWER(4,2) FROM DUAL;

POWER(4,2)
----------
        16
```

DB2の場合

書式 **POWER(m,n)**

実行例

```
db2=> SELECT POWER(4,2) FROM SYSIBM.SYSDUMMY1

1
-----------
         16

  1 レコードが選択されました。
```

3.33 REPLACE

すべて検索文字列を置換文字列に変換して文字列1を戻します。置換文字列を指定しない場合、またはNULL値の場合、すべての検索文字列が削除されます。検索文字列がNULLの場合、文字列1が戻されます。

Oracleの場合

書式 **REPLACE(文字列1,検索文字列,置換文字列)**

実行例

```
SQL> SELECT REPLACE('(株)システム','(株)','株式会社') FROM DUAL;

REPLACE('(株)シ
----------------
株式会社システム
```

DB2の場合

書式 **REPLACE(文字列1,検索文字列,置換文字列)**

実行例

```
db2=> SELECT REPLACE('(株)システム','(株)','株式会社') FROM SYSIBM.SYSDUMY1

1
----------------------
株式会社システム

  1 レコードが選択されました。
```

注意点 REPLACE関数は、単一の文字列から別の単一の文字列への置換、および文字列の削除を実行できます。TRANSLATE関数は、1回の操作で複数の単一文字を1対1で置き換えることができます。

3.34 SIGN

nの符号を戻します。

Oracleの場合

書式

SIGN(n)
NUMBER型の値の場合、符号は次のとおりです。
n<0（ゼロ）の場合、-1
n=0 の場合、0（ゼロ）
n>0 の場合、1

実行例

```
SQL> SELECT SIGN(-20),SIGN(20) FROM DUAL;

 SIGN(-20)   SIGN(20)
---------- ----------
        -1          1
```

DB2の場合

書式

SIGN(n)

実行例

```
db2=> SELECT SIGN(-20),SIGN(20) FROM SYSIBM.SYSDUMMY1

1           2
----------- -----------
         -1           1

  1 レコードが選択されました。
```

3.35 RTRIM

文字列1の右端から、文字列2に指定されたすべての文字を削除します。

Oracleの場合

書式

RTRIM（文字列1,文字列2）
文字列1 ―― 対象となる文字列
文字列2 ―― 削除する文字列(省略すると空白)

実行例

```
SQL> SELECT RTRIM('ERASER','ER') FROM DUAL;

RTRI
----
ERAS
```

DB2の場合

書式

対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。

実行例

```
db2=> WITH
Recurse (seq, str1) AS (
VALUES (0, 'ERASER')
UNION ALL
SELECT new_seq, SUBSTR(str1,1,LENGTH(str1)-1)
        FROM (SELECT seq+1 AS new_seq, str1
                   , LOCATE(SUBSTR(str1,LENGTH(str1),1),'ER') AS exist
                FROM Recurse
               WHERE seq < 10000
             ) Q
       WHERE exist > 0
      )
SELECT str1
        FROM Recurse
      WHERE seq = (SELECT MAX(seq) FROM Recurse);
------------------------------------------------------------------------------

STR1
------
ASER

  1 レコードが選択されました。
```

注意点

DB2 UDBにも、RTRIM関数はありますが、Oracleと完全な互換性はありません。DB2 UDBのRTRIM関数は引数を1つだけとり、文字列の末尾（右側）から空白を削除します。
MTK（70ページ参照）では、ora8.RTRIM関数をサンプルUDFとして提供しています。

●ユーザー定義関数（UDF）の作成例

```
------------------------------ Commands Entered ------------------------------
CREATE FUNCTION MyOra.RTRIM(str1 VARCHAR(4000), srch VARCHAR(4000))
RETURNS VARCHAR(4000)
CONTAINS SQL
BEGIN ATOMIC
MAIN_LOOP:
FOR rtrim_loop AS
```

```
SELECT SEQ + 1 AS pos
  FROM (SELECT N4*1000+N3*100+N2*10+N1 AS SEQ
          FROM (VALUES 0,1,2,3,4,5,6,7,8,9) P1(N1)
             , (VALUES 0,1,2,3,4,5,6,7,8,9) P2(N2)
             , (VALUES 0,1,2,3,4,5,6,7,8,9) P3(N3)
             , (VALUES 0,1,2,3) P4(N4)
       ) Q
 WHERE SEQ < LENGTH(str1)
 ORDER BY
       SEQ DESC
DO
   IF LOCATE(SUBSTR(str1,pos,1), srch) = 0 THEN
      RETURN SUBSTR(str1,1,pos);
   END IF;
END FOR;
END@
------------------------------------------------------------------------------
DB20000I  The SQL command completed successfully.
```

●ユーザー定義関数（UDF）の使用例

```
------------------------------ Commands Entered ------------------------------
SELECT SUBSTR(MyOra.RTRIM('ERASER','ER'),1,20) AS "RTRIM Sample"
  FROM SYSIBM.SYSDUMMY1;
------------------------------------------------------------------------------

RTRIM Sample
--------------------
ERAS

  1 record(s) selected.
```

3.36 SOUNDEX

文字列と同じ音声表現を持つ文字列を戻します。

Oracleの場合

書式 **SOUNDEX(文字列)**

実行例

```
SQL> SELECT ENAME FROM EMP WHERE SOUNDEX(ENAME) = SOUNDEX('ALLAN');

ENAME
----------
ALLEN
```

DB2の場合

書式 **SOUNDEX(文字列)**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT ENAME FROM EMP WHERE SOUNDEX(ENAME) = SOUNDEX('ALLAN');
--------------------------------------------------------------------------

ENAME
----------
ALLEN

  1 レコードが選択されました。
```

3.37 SUBSTR

文字列の文字位置から文字数分の文字列を抜き出して戻します。

Oracleの場合

書式 **SUBSTR(文字列,文字位置,文字数)**

実行例

```
SQL> SELECT SUBSTR('CORPORATE SECTOR',3,4)
  2         ,SUBSTR('けんけんがくがく',3,4) FROM DUAL;

SUBS SUBSTR('
---- --------
RPOR けんがく
```

DB2の場合

書式 **SUBSTR(文字列,文字位置,バイト数)**

実行例

```
db2=> SELECT SUBSTR('CORPORATE SECTOR',3,4),SUBSTR('けんけんがくがく',5,8)
➥FROM SYSIBM.SYSDUMMY1

1    2
---- --------
RPOR けんがく

  1 レコードが選択されました。
```

注意点 どちらのデータベースにもSUBSTR関数はありますが、Oracleは文字単位に処理し、DB2 UDBはバイト単位に処理をします。したがって、SUBSTRB関数と同じ結果を得ることはできますが、SUBSTR関数と同じ結果を得るためには、指定する文字位置および文字数の引数の値は異なります。

3.38 SUBSTRB

文字列の文字（バイト）位置からバイト数分の文字列を抜き出して戻します。

Oracleの場合

書式 **SUBSTRB(文字列,文字位置,バイト数)**

実行例

```
SQL> SELECT SUBSTRB('CORPORATE SECTOR',3,4)
  2         ,SUBSTRB('けんけんがくがく',5,8) FROM DUAL;

SUBS SUBSTRB(
---- --------
RPOR けんがく
```

DB2の場合

書式 **SUBSTR(文字列,文字位置,バイト数)**

実行例

```
db2=> SELECT SUBSTR('CORPORATE SECTOR',3,4),SUBSTR('けんけんがくがく',5,8)
➥FROM SYSIBM.SYSDUMMY1

1    2
---- --------
RPOR けんがく

  1 レコードが選択されました。
```

注意点 どちらのデータベースにもSUBSTR関数はありますが、Oracleは文字単位に処理し、DB2 UDBはバイト単位に処理をします。したがって、SUBSTRB関数と同じ結果を得ることはできますが、SUBSTR関数と同じ結果を得るためには、指定する文字位置および文字数の引数の値は異なります。

3.39 SUM

値の合計を求めます。

Oracle の場合

書式 **SUM(列名)**

実行例

```
SQL> SELECT SUM(SAL) FROM EMP;

  SUM(SAL)
----------
     30325

1行が選択されました。
```

DB2 の場合

書式 **SUM(列名)**

実行例

```
db2=> SELECT SUM(SAL) FROM EMP

1
--------------------------------
                          30325.00

  1 レコードが選択されました。
```

注意点 SUM 関数の引数には数値型の列を指定します。NULL 値以外の行を処理対象にして合計値を求めます。

3.40 TRANSLATE

検索文字内のすべての文字を置換文字内の対応する文字に置換して文字列1を戻します。検索文字内に存在しない文字列内の文字は置換されません。1回の操作で複数の単一文字を1対1で置き換えることができます。

Oracleの場合

書式 **TRANSLATE(文字列,検索文字,置換文字)**

実行例

```
SQL> SELECT TRANSLATE('A.B C/D','. /','-,_') FROM DUAL;

TRANSLA
-------
A-B,C_D
```

DB2の場合

書式 **TRANSLATE(文字列,置換文字,検索文字)**

実行例

```
db2=> SELECT TRANSLATE('A.B C/D','-,_','. /') FROM SYSIBM.SYSDUMMY1

1
-------
A-B,C_D

  1 レコードが選択されました。
```

注意点 REPLACE関数は、単一の文字列から別の単一の文字列への置換、および文字列の削除を実行できます。TRANSLATE関数は、1回の操作で複数の単一文字を1対1で置き換えることができます。
どちらのデータベースにもTRANSLATE関数はありますが、検索文字引数と置換文字引数の指定順序が異なります。同じ結果を求める場合には、引数の指定順序に気をつけてください。

3.41 UPPER

文字を大文字に変換します。

Oracle の場合

書式 **UPPER(文字列)**

実行例

```
SQL> SELECT UPPER('abc') FROM DUAL;

UPP
---
ABC
```

DB2 の場合

書式 **UPPER(文字列) または UCASE(文字列)**

実行例

```
db2=> SELECT UPPER('abc') FROM SYSIBM.SYSDUMMY1

1
---
ABC

  1 レコードが選択されました。
```

または

```
db2=> SELECT UCASE('abc') FROM SYSIBM.SYSDUMMY1

1
---
ABC

  1 レコードが選択されました。
```

3.42 USER

セッション・ユーザー（ログインしているユーザー）の名前を戻します。

Oracleの場合

書式 **USER**

実行例

```
SQL> SELECT USER FROM DUAL;

USER
------------------------------
USER1
```

DB2の場合

書式 **USER**

実行例

```
db2=> SELECT USER FROM SYSIBM.SYSDUMMY1

1
--------------------------------------------------
USER1

  1 レコードが選択されました。
```

3.43 ACOS

nのアーク・コサインを戻します。

Oracleの場合

書式 **ACOS(n)**

実行例
```
SQL> SELECT ACOS(.3) FROM DUAL;

  ACOS(.3)
----------
1.26610367
```

DB2の場合

書式 **ACOS(n)**

実行例
```
db2=> SELECT ACOS(.3) FROM SYSIBM.SYSDUMMY1

1
------------------------
  +1.26610367277950E+000

  1 レコードが選択されました。
```

3.44 ASIN

nのアーク・サインを戻します。

Oracleの場合

書式 **ASIN(n)**

実行例

```
SQL> SELECT ATAN(.3) FROM DUAL;

  ATAN(.3)
----------
.291456794
```

DB2の場合

書式 **ASIN(n)**

実行例

```
db2=> SELECT ATAN(.3) FROM SYSIBM.SYSDUMMY1

1
-------------------------
  +2.91456794477867E-001

  1 レコードが選択されました。
```

3.45 ATAN

nのアーク・タンジェントを戻します。

Oracleの場合

書式 **ATAN(n)**

実行例
```
SQL> SELECT ATAN(.3) FROM DUAL;

  ATAN(.3)
----------
.291456794
```

DB2の場合

書式 **ATAN(n)**

実行例
```
db2=> SELECT ATAN(.3) FROM SYSIBM.SYSDUMMY1

1
------------------------
  +2.91456794477867E-001

  1 レコードが選択されました。
```

3.46 ATAN2

nのアーク・タンジェントを戻します。引数nの範囲に制限はなく、nとmの符号により、関数によって戻される値は$-\pi$～π（ラジアン）の範囲です。ATAN2(n,m)は、ATAN(n/m)と同じです。

Oracleの場合

書式 **ATAN2(n,m)**

実行例

```
SQL> SELECT ATAN2(.3, .2) FROM DUAL;

ATAN2(.3,.2)
------------
  .982793723
```

DB2の場合

書式 **ATAN2(m,n)**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT ATAN2(.2, .3) FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

1
------------------------
  +9.82793723247329E-001

  1 レコードが選択されました。
```

注意点 DB2 UDBでは、Oracleと引数の指定順序が異なります。同じ結果を求める場合には、引数の指定順序に気をつけてください。

3.47 COS

n（ラジアンで表された角度）のコサインを戻します。

Oracleの場合

書式 **COS(n)**

実行例

```
SQL> SELECT COS(180 * 3.14159/180) FROM DUAL;

COS(180*3.14159/180)
--------------------
                  -1
```

DB2の場合

書式 **COS(n)**

実行例

```
db2=> SELECT COS(180 * 3.14159/180) FROM SYSIBM.SYSDUMMY1

1
------------------------
  -9.99999999996479E-001

  1 レコードが選択されました。
```

3.48 COSH

nの双曲線コサインを戻します。

Oracleの場合

書式 **COSH(n)**

実行例
```
SQL> SELECT COSH(0) FROM DUAL;

   COSH(0)
----------
         1
```

DB2の場合

書式 **COSH(n)**

実行例
```
db2=> SELECT COSH(0) FROM SYSIBM.SYSDUMMY1

1
-------------------------
  +1.00000000000000E+000

  1 レコードが選択されました。
```

3.49 SIN

n（ラジアンで表された角度）のサインを戻します。

Oracleの場合

書式 **SIN(n)**

実行例

```
SQL> SELECT SIN(30 * 3.14159/180) FROM DUAL;

SIN(30*3.14159/180)
-------------------
          .499999617
```

DB2の場合

書式 **SIN(n)**

実行例

```
db2=> SELECT SIN(30 * 3.14159/180) FROM SYSIBM.SYSDUMMY1

1
------------------------
  +4.99999616987256E-001

  1 レコードが選択されました。
```

3.50 SINH

nの双曲線サインを戻します。

Oracleの場合

書式 **SINH(n)**

実行例

```
SQL> SELECT SINH(1) FROM DUAL;

   SINH(1)
----------
1.17520119
```

DB2の場合

書式 **SINH(n)**

実行例

```
db2=> SELECT SINH(1) FROM SYSIBM.SYSDUMMY1

1
-------------------------
  +1.17520119364380E+000

  1 レコードが選択されました。
```

3.51 TAN

n（ラジアンで表された角度）のタンジェントを戻します。

Oracleの場合

書式 **TAN(n)**

実行例

```
SQL> SELECT TAN(30 * 3.14159/180) FROM DUAL;

TAN(30*3.14159/180)
-------------------
          .57734968
```

DB2の場合

書式 **TAN(n)**

実行例

```
db2=> SELECT TAN(30 * 3.14159/180) FROM SYSIBM.SYSDUMMY1

1
------------------------
  +5.77349679503156E-001

  1 レコードが選択されました。
```

3.52 TANH

nの双曲線タンジェントを戻します。

Oracleの場合

書式 **TANH(n)**

実行例

```
SQL> SELECT TANH(.5) FROM DUAL;

  TANH(.5)
----------
.462117157
```

DB2の場合

書式 **TANH(n)**

実行例

```
db2=> SELECT TANH(.5) FROM SYSIBM.SYSDUMMY1

1
-------------------------
  +4.62117157260010E-001

  1 レコードが選択されました。
```

3.53 CEIL

n以上の最も小さい整数を戻します。

Oracleの場合

書式 **CEIL(n)**

実行例

```
SQL> SELECT CEIL(13.6) FROM DUAL;

CEIL(13.6)
----------
        14
```

DB2の場合

書式 **CEIL(n)**

実行例

```
db2=> SELECT CEIL(13.6) FROM SYSIBM.SYSDUMMY1

1
-----
  14.

  1 レコードが選択されました。
```

3.54 FLOOR

n以下の最も大きい整数を戻します。

Oracleの場合

書式 **FLOOR(n)**

実行例

```
SQL> SELECT FLOOR(13.6) FROM DUAL;

FLOOR(13.6)
-----------
         13
```

DB2の場合

書式 **FLOOR(n)**

実行例

```
db2=> SELECT FLOOR(13.6) FROM SYSIBM.SYSDUMMY1

1
-----
  13.

  1 レコードが選択されました。
```

3.55 ROUND（数値型：小数点n桁あるいは小数点の左n 桁に四捨五入）

数値型データを小数点以下n桁に丸めた値を戻します。nを指定しない場合、数値型データは小数点以下が丸められます。引数nが負の場合は小数点の左n桁が丸められます。

Oracleの場合

書式 **ROUND(数値,n)**

実行例

```
SQL> SELECT ROUND(123.456,2) FROM DUAL;

ROUND(123.456,2)
----------------
          123.46
```

DB2の場合

書式 **ROUND(数値,n)**

実行例

```
db2=> SELECT ROUND(123.456,2) FROM SYSIBM.SYSDUMMY1

1
---------
  123.460

  1 レコードが選択されました。
```

注意点 どちらのデータベースにもROUND関数はありますが、Oracleの場合、第2引数を省略するとゼロ（0）と解釈され、小数点以下が四捨五入されますが、DB2 UDBでは第2引数を省略できません。また、Oracleは表示桁数を結果の値に合わせますが、DB2 UDBは元の値に合わせます。

3.56 ROUND（日付型：時間、日にち、月を四捨五入）

日付を書式モデル fmt で指定した単位に四捨五入した結果を戻します。
fmt には、以下の値を指定します。

'DD' ： 時間を四捨五入
'MM' ： 日にちを四捨五入
'YY' ： 月を四捨五入

Oracle の場合

書式 **ROUND(日付,fmt)**

実行例

```
SQL> SELECT TO_CHAR(SYSDATE,'YY-MM-DD HH24:MI:SS'),ROUND(SYSDATE,'DD')
  2  FROM DUAL;

TO_CHAR(SYSDATE,' ROUND(SY
----------------- --------
05-03-25 18:12:27 05-03-26
```

DB2 の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT timestamp
     , DATE(timestamp + 12 HOURS)  AS ROUND_DD
  FROM TABLE(VALUES TIMESTAMP('2005-03-25-18.12.27')) Q(timestamp);
------------------------------------------------------------------------

TIMESTAMP                  ROUND_DD
-------------------------- ----------
2005-03-25-18.12.27.000000 2005-03-26

  1 record(s) selected.
```

注意点 Oracle では、ROUND 関数の引数に日付を指定することができますが、DB2 UDB では引数に日付を指定することはできません。
MTK（70 ページ参照）では、ora8.ROUND(date[,format])関数をサンプル UDF として提供しています。

3.57 ROUND（日付型：日にちを四捨五入）

日付を書式モデル fmt で指定した単位に四捨五入した結果を戻します。

Oracle の場合

書式 **ROUND(日付,'MM')**

実行例

```
SQL> SELECT TO_CHAR(SYSDATE,'YY-MM-DD HH24:MI:SS'),ROUND(SYSDATE,'MM')
  2  FROM  DUAL;

TO_CHAR(SYSDATE,' ROUND(SY
----------------- --------
05-03-25 18:16:57 05-04-01
```

DB2 の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT date
     , date - 15 DAYS - (DAY(date - 15 DAYS) - 1) DAYS + 1 MONTH  AS
ROUND_MM
  FROM TABLE(VALUES DATE('2005-02-01')
                  , DATE('2005-02-15')
                  , DATE('2005-02-16')
                  , DATE('2005-02-28')
                  , DATE('2005-03-15')
                  , DATE('2005-03-16')
                  , DATE('2005-03-31')
                  , DATE('2005-12-15')
                  , DATE('2005-12-16')
                  , DATE('2005-12-31')
                  , DATE('2005-01-15')
                  , DATE('2005-01-16')
                  , DATE('2005-01-01')
            ) Q (date);
------------------------------------------------------------------------

DATE       ROUND_MM
---------- ----------
2005-02-01 2005-02-01
2005-02-15 2005-02-01
2005-02-16 2005-03-01
2005-02-28 2005-03-01
2005-03-15 2005-03-01
2005-03-16 2005-04-01
2005-03-31 2005-04-01
2005-12-15 2005-12-01
2005-12-16 2006-01-01
2005-12-31 2006-01-01
2005-01-15 2005-01-01
2005-01-16 2005-02-01
2005-01-01 2005-01-01

  13 record(s) selected.
```

注意点 Oracleでは、ROUND関数の引数に日付を指定することができますが、DB2 UDBでは引数に日付を指定することはできません。
MTK（70ページ参照）では、ora8.ROUND(date[,format])関数をサンプルUDFとして提供しています。

3.58 ROUND（日付型：月を四捨五入）

日付を書式モデル fmt で指定した単位に四捨五入した結果を戻します。

Oracle の場合

書式 **ROUND(日付,'YY')**

実行例

```
SQL> SELECT TO_CHAR(SYSDATE,'YY-MM-DD HH24:MI:SS'),ROUND(SYSDATE,'YY')
  2  FROM DUAL;

TO_CHAR(SYSDATE,' ROUND(SY
----------------- --------
05-03-25 18:21:48 05-01-01
```

DB2 の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
-------------------------------- 入力コマンド --------------------------------
SELECT date
     , date + 7 MONTHS - (DAYOFYEAR(date + 7 MONTHS) -1) DAYS  AS ROUND_YY
  FROM TABLE(VALUES DATE('2004-02-01')
                  , DATE('2005-02-28')
                  , DATE('2004-05-31')
                  , DATE('2004-06-01')
                  , DATE('2005-05-31')
                  , DATE('2005-06-01')
                  , DATE('2004-12-31')
                  , DATE('2005-12-31')
            ) Q (date);
------------------------------------------------------------------------------

DATE       ROUND_YY
---------- ----------
2004-02-01 2004-01-01
2005-02-28 2005-01-01
2004-05-31 2004-01-01
2004-06-01 2005-01-01
2005-05-31 2005-01-01
2005-06-01 2006-01-01
2004-12-31 2005-01-01
2005-12-31 2006-01-01

  8 record(s) selected.
```

注意点

Oracle では、ROUND 関数の引数に日付を指定することができますが、DB2 UDB では引数に日付を指定することはできません。
MTK（70 ページ参照）では、ora8.ROUND(date[,format])関数をサンプル UDF として提供しています。

3.59 TRUNC（数値型：小数点以下m桁あるいは小数点の左m桁を切り捨て）

指定した数値を小数第m位までに切り捨てた値を戻します。mを指定しない場合、指定した数値の小数点以下を切り捨てます。mが負の場合は、小数点の左m桁を切り捨てて0（ゼロ）にします。

Oracleの場合

書式 **TRUNC(数値,m)**

実行例

```
SQL> SELECT TRUNC(123.456,2) FROM DUAL;

TRUNC(123.456,2)
----------------
          123.45
```

DB2の場合

書式 **TRUNC(数値,m)**

実行例

```
db2=> SELECT TRUNC(123.456,2) FROM SYSIBM.SYSDUMMY1

1
--------
 123.450

  1 レコードが選択されました。
```

注意点 どちらのデータベースにもTRUNC関数はありますが、Oracleの場合、第1引数を省略するとゼロ（0）と解釈されて小数点以下が切り捨てられますが、DB2 UDBの場合は第2引数を省略することはできません。また、Oracleは表示桁数を結果の値に合わせ、DB2 UDBは元の値に合わせます。

3.60 TRUNC（日付型：時間、日にち、月の切り捨て）

日付を書式モデルfmtで指定した単位に切り捨てた結果を戻します。
fmtには、以下の値を指定します。

'DD' ： 時間を四捨五入
'MM' ： 日にちを四捨五入
'YY' ： 月を四捨五入

Oracleの場合

書式 **TRUNC(日付,fmt)**

実行例

```
SQL> SELECT TO_CHAR(SYSDATE,'YY-MM-DD HH24:MI:SS'),TRUNC(SYSDATE,'DD')
  2  FROM DUAL;

TO_CHAR(SYSDATE,' TRUNC(SY
----------------- --------
05-03-26 14:26:39 05-03-26
```

DB2の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CURRENT_TIMESTAMP  AS CURRENT_TIMESTAMP
     , TIMESTAMP(DATE(CURRENT_TIMESTAMP),'00.00.00')  AS TRUNC_DD
  FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

CURRENT_TIMESTAMP          TRUNC_DD
-------------------------- --------------------------
2005-11-26-11.12.59.201000 2005-11-26-00.00.00.000000

  1 record(s) selected.
```

注意点 Oracleでは、TRUNC関数の引数に日付を指定することができますが、DB2 UDBでは引数に日付を指定することはできません。
MTK（70ページ参照）では、ora8.TRUNC(date)関数をサンプルUDFとして提供しています。

3.61 TRUNC（日付型：日にちの切り捨て）

日付を書式モデルfmtで指定した単位に切り捨てた結果を戻します。

Oracleの場合

書式 **TRUNC(日付,'MM')**

実行例

```
SQL> SELECT TO_CHAR(SYSDATE,'YY-MM-DD HH24:MI:SS'),TRUNC(SYSDATE,'MM')
  2  FROM DUAL;

TO_CHAR(SYSDATE,' TRUNC(SY
----------------- --------
05-03-26 14:33:08 05-03-01
```

DB2の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT date
     , date - (DAY(date)-1) DAYS  AS TRUNC_MM
  FROM TABLE(VALUES DATE('2005-02-01')
                  , DATE('2005-02-15')
                  , DATE('2005-02-16')
                  , DATE('2005-02-28')
                  , DATE('2005-03-15')
                  , DATE('2005-03-16')
                  , DATE('2005-03-31')
            ) Q (date);
------------------------------------------------------------------------

DATE       TRUNC_MM
---------- ----------
2005-02-01 2005-02-01
2005-02-15 2005-02-01
2005-02-16 2005-02-01
2005-02-28 2005-02-01
2005-03-15 2005-03-01
2005-03-16 2005-03-01
2005-03-31 2005-03-01

  7 record(s) selected.
```

注意点 Oracleでは、TRUNC関数の引数に日付を指定することができますが、DB2 UDBでは引数に日付を指定することはできません。
MTK（70ページ参照）では、ora8.TRUNC(date)関数をサンプルUDFとして提供しています。

3.62 TRUNC（日付型：月の切り捨て）

日付を書式モデル fmt で指定した単位に切り捨てた結果を戻します。

Oracle の場合

書式 **TRUNC(日付,'YY')**

実行例

```
SQL> SELECT TO_CHAR(SYSDATE,'YY-MM-DD HH24:MI:SS'),TRUNC(SYSDATE,'YY')
  2  FROM DUAL;

TO_CHAR(SYSDATE,' TRUNC(SY
----------------- --------
05-03-26 14:36:52 05-01-01
```

DB2 の場合

書式 **対応する関数はありませんが、次のようにして同じ結果を得ることができます。UDF（ユーザー定義関数）として作成しておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT date
     , date - (DAYOFYEAR(date)-1) DAYS  AS TRUNC_YY
  FROM TABLE(VALUES DATE('2004-02-01')
                  , DATE('2005-02-28')
                  , DATE('2004-05-31')
                  , DATE('2004-06-01')
                  , DATE('2005-05-31')
                  , DATE('2005-06-01')
                  , DATE('2004-12-31')
                  , DATE('2005-12-31')
            ) Q (date);
------------------------------------------------------------------------

DATE       TRUNC_YY
---------- ----------
2004-02-01 2004-01-01
2005-02-28 2005-01-01
2004-05-31 2004-01-01
2004-06-01 2004-01-01
2005-05-31 2005-01-01
2005-06-01 2005-01-01
2004-12-31 2004-01-01
2005-12-31 2005-01-01

  8 record(s) selected.
```

注意点 Oracle では、TRUNC 関数の引数に日付を指定することができますが、DB2 UDB では引数に日付を指定することはできません。
MTK（70 ページ参照）では、ora8.TRUNC(date)関数をサンプル UDF として提供しています。

3.63 EXP

EXPは、自然対数のeをn乗した値（e = 2.71828183…）を戻します。

Oracleの場合

書式 **EXP(n)**

実行例
```
SQL> SELECT EXP(2) FROM DUAL;

    EXP(2)
----------
 7.3890561
```

DB2の場合

書式 **EXP(n)**

実行例
```
db2=> SELECT EXP(2) FROM SYSIBM.SYSDUMMY1

1
------------------------
  +7.38905609893065E+000

  1 レコードが選択されました。
```

3.64 LN

nの自然対数を戻します（nは正の数）。

Oracleの場合

書式 **LN(n)**

実行例
```
SQL> SELECT LN(90) FROM DUAL;

    LN(90)
----------
4.49980967
```

DB2の場合

書式 **LN(n)**

実行例
```
db2=> SELECT LN(90) FROM SYSIBM.SYSDUMMY1

1
------------------------
  +4.49980967033027E+000

  1 レコードが選択されました。
```

3.65 LOCALTIMESTAMP

セッションのタイムゾーンの現在の日付および時刻をTIMESTAMPデータ型の値で戻します。

Oracleの場合

書式 **LOCALTIMESTAMP**

実行例

```
SQL> SELECT LOCALTIMESTAMP FROM DUAL;

LOCALTIMESTAMP
---------------------------------------
05-03-30 22:12:33.447000
```

DB2の場合

書式 **CURRENT_TIMESTAMP または CURRENT TIMESTAMP**

実行例

```
db2=> SELECT CURRENT_TIMESTAMP FROM SYSIBM.SYSDUMMY1

1
--------------------------
2005-03-30-22.12.22.337001

  1 レコードが選択されました。
```

3.66 SQRT

nの平方根を戻します。

Oracleの場合

書式 **SQRT(n)**

実行例
```
SQL> SELECT SQRT(25) FROM DUAL;

  SQRT(25)
----------
         5
```

DB2の場合

書式 **SQRT(n)**

実行例
```
------------------------------ 入力コマンド ------------------------------
SELECT SQRT(25) FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

1
------------------------
  +5.00000000000000E+000

  1 レコードが選択されました。
```

3.67 AVG

平均値を求めます。

Oracle の場合

書式 **AVG(数値型列名)**

実行例
```
SQL> SELECT AVG(SAL) FROM EMP;

  AVG(SAL)
----------
2021.66667

1行が選択されました。
```

DB2 の場合

書式 **AVG(数値型列名)**

実行例
```
db2=> SELECT AVG(SAL) FROM EMP

1
---------------------------------
  2021.66666666666666666666666666

  1 レコードが選択されました。
```

注意点 AVG関数の引数には数値型の列を指定します。NULL値以外の行を処理対象にして平均値を求めます。

3.68 INSTR

文字列1の検索文字列を検索します。文字列1の開始位置から検索して出現回数目に一致した位置を示す整数を戻します。INSTR関数は文字単位で処理を実行します。

Oracleの場合

書式 **INSTR(文字列,検索文字列,開始位置,出現回数)**

実行例

```
SQL> SELECT INSTR('CORPORATE SECTOR','OR',1,2)
  2         ,INSTR('けんけんがくがく','ん',1,2) FROM DUAL;

INSTR('CORPORATESECTOR','OR',1,2) INSTR('けんけんがくがく','ん',1,2)
--------------------------------- ---------------------------------
                                5                                 4
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点

指定した検索文字列が最初に出現した位置を求めるには、DB2 UDBのLOCATE関数を使用できます。しかし、OracleのINSTR関数は出現回数を指定できないだけでなく、検索文字列引数と文字列引数の指定位置が異なります。また、Oracleは文字単位に処理し、DB2 UDBはバイト単位に処理をします。したがって、マルチバイト文字列の場合、同じ結果を求めることはできません。

MTK（70ページ参照）では、ora8.INSTR関数をサンプルUDFとして提供しています。

3.69 INSTRB — 出現回数目に一致したものを戻す

文字列1の検索文字列を検索します。文字列1の開始位置から検索して出現回数目に一致した位置を示す整数を戻します。INSTRB関数はバイト単位で処理を実行します。

Oracleの場合

書式 **INSTRB(文字列,検索文字列,開始位置,出現回数)**

実行例

```
SQL> SELECT INSTRB('CORPORATE SECTOR','OR',1,2)
  2         ,INSTRB('けんけんがくがく','ん',1,2) FROM DUAL;

INSTRB('CORPORATESECTOR','OR',1,2) INSTRB('けんけんがくがく','ん',1,2)
---------------------------------- ----------------------------------
                                 5                                  7
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点

指定した検索文字列が最初に出現した位置を求めるためには、DB2 UDBのLOCATE関数を使用することができます。しかし、OracleのINSTR関数およびINSTRB関数とは違って出現回数を指定できないため、上記処理を実現することはできません。また、検索文字列引数と文字列引数の指定位置が異なります。

Sample UDFs for Migration（70ページ参照）では、INSTRBをサンプルUDFとして提供しています。

3.70 INSTRB — 1回目に一致したものを戻す

文字列1の検索文字列を検索します。文字列1の開始位置から検索して1回目に一致した位置を示す整数を戻します。INSTRB関数はバイト単位に処理を実行します。

Oracleの場合

書式 **INSTRB(文字列,検索文字列,開始位置,1)**

実行例

```
SQL> SELECT INSTRB('CORPORATE SECTOR','OR',1,1)
  2         ,INSTRB('けんけんがくがく','ん',1,1) FROM DUAL;

INSTRB('CORPORATESECTOR','OR',1,1) INSTRB('けんけんがくがく','ん',1,1)
---------------------------------- ------------------------------------
                                 2                                    3
```

DB2の場合

書式 **LOCATE(検索文字,文字列,開始位置)**

実行例

```
db2=> SELECT LOCATE('OR','CORPORATE SECTOR',1) ,LOCATE('ん','けんけんがくがく',
➥1) FROM SYSIBM.SYSDUMMY1

1           2
----------- -----------
          2           3

  1 レコードが選択されました。
```

注意点 指定した検索文字列が最初に出現した位置を求めるには、DB2 UDBのLOCATE関数を使用できます。しかし、OracleのINSTR関数およびINSTRB関数とDB2 UDBのLOCATE関数は出現回数を指定できない点と、検索文字列引数と文字列引数の指定位置が異なります。また、OracleのINSTR関数は文字単位に処理し、DB2 UDBのLOCATE関数はバイト単位に処理を実行します。したがって、マルチバイト文字列の場合同じ結果を求めることはできません。DB2 UDBのLOCATE関数は、OracleのINSTRB関数を使用して、指定した検索文字列が最初に出現した位置を求める場合と同等の処理を実行できます。

3.71 LOG

数値1を底とする数値2の対数を戻します。数値1は0（ゼロ）または1以外の任意の正の値で、数値2は任意の正の値です。

Oracleの場合

書式 **LOG(数値1,数値2)**

実行例

```
SQL> SELECT LOG(10,100) FROM DUAL;

LOG(10,100)
-----------
          2
```

DB2の場合

書式 **LOG(数値2)/LOG(数値1)**
底が10の場合は、LOG10(数値)
底がeの場合は、LOG(数値) または LN(数値)

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT LOG(100)/LOG(10) FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

1
------------------------
  +2.00000000000000E+000

  1 レコードが選択されました。
```

底が10の場合は、次も可能。

```
------------------------------ 入力コマンド ------------------------------
SELECT LOG10(100) FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

1
------------------------
  +2.00000000000000E+000

  1 レコードが選択されました。
```

注意点 どちらのデータベースにもLOG関数はありますが、Oracleは底が指定できるのに対して、DB2 UDBではLOGの場合には底がe、LOG10の場合には底が10と決められています。DB2 UDBで任意の底を指定するには、書式の例のように底のLOGで割ります。また、DB2 UDBのLOG関数の引数は、内部で浮動小数点に変換されて計算され結果も浮動小数点で戻されます。

3.72 RAWTOHEX

rawを16進数で表した文字値に変換します。raw引数は、RAWデータ型である必要があります。

Oracleの場合

書式 **RAWTOHEX(raw)**

実行例

DB2の場合

書式 **対応する関数はありませんが、hex関数を使って同様の処理が可能です。**

実行例

```
db2=> db2 select hex('aaa') from table(values(1)) as x

1
------
616161

  1 レコードが選択されました。
```

3.73 NVL

NULLを文字列に置換して問い合わせの結果に含めることができます。値1がNULLの場合、NVLは値2を戻します。値1がNULLでない場合、NVLは値1の値を戻します。

Oracleの場合

書式 **NVL(値1, 値2)**

実行例

```
SQL> SELECT COMM,SAL,NVL(COMM,0),NVL(SAL,COMM) FROM EMP WHERE EMPNO = 7369;

      COMM         SAL NVL(COMM,0) NVL(SAL,COMM)
---------- ---------- ----------- -------------
                  800           0           800
```

DB2の場合

書式 **COALESCE(値1,値2)**

実行例

```
db2=> SELECT COMM,SAL,COALESCE(COMM,0),COALESCE(SAL,COMM) FROM EMP WHERE
➥EMPNO = 7369

COMM       SAL        3               4
---------- ---------- --------------- ----------
         -    800.00            0.00    800.00

  1 レコードが選択されました。
```

3.74 SYSDATE

データベースが存在するオペレーティング・システムの現在の日付と時刻のセットを戻します。戻り値のデータ型はDATE型です。

Oracleの場合

書式 **SYSDATE**

実行例

```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
--------
05-03-30

SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'YY-MM-DD HH24:MI:SS';

セッションが変更されました。

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
-----------------
05-03-30 22:55:42
```

DB2の場合

書式 **CURRENT_DATE または CURRENT DATE**
CURRENT_TIMESTAMP または CURRENT TIMESTAMP

実行例

```
db2=> SELECT CURRENT_DATE FROM SYSIBM.SYSDUMMY1

1
----------
2005-03-30

  1 レコードが選択されました。

db2=> SELECT CURRENT_DATE,CURRENT_TIME FROM SYSIBM.SYSDUMMY1

1          2
---------- --------
2005-03-30 22:56:50

  1 レコードが選択されました。
```

注意点

Oracleの DATE型は日付と時刻を保持しているため、SYSDATE関数は日付と時刻を戻します。ただしデフォルトの書式は 'YY-MMDD' のため、日付だけを受け取ることができます。
DB2 UDBは、日付と時刻を別々に保持できます。日付だけ取得するのであれば、CUURENT_DATE関数を使用し、時刻だけ取得するのであればCURRENT_TIME関数を使用します。日付と時刻を同時に取得するにはCURRENT_TIMESTAMP関数が使えます。
Oracleはセッション単位で日付のデフォルト書式を変更できるため、設定に応じてCURRENT_DATE関数とCURRENT_TIME関数を組み合わせるとよいでしょう。

3.75 TO_MULTI_BYTE

シングルバイト文字を対応するマルチバイト文字に変換して文字列を戻します。

Oracleの場合

書式 **TO_MULTI_BYTE(文字列)**

実行例

```
SQL> SELECT TO_MULTI_BYTE('ABC') FROM DUAL;

TO_MUL
------
ABC
```

DB2の場合

書式 **VARGRAPHIC(文字列)**

実行例

```
db2=> SELECT VARGRAPHIC('ABC') FROM SYSIBM.SYSDUMMY1

1
------
ABC

  1 レコードが選択されました。
```

3.76 VSIZE

引数の内部表現でのバイト数を戻します。引数がNULLの場合、NULLを戻します。

Oracleの場合

書式 **VSIZE(文字列)**

実行例

```
SQL> SELECT VSIZE('ABC'),VSIZE('あいう') FROM DUAL;

VSIZE('ABC') VSIZE('あいう')
------------ ---------------
           3               6
```

DB2の場合

書式 **LENGTH(文字列)**

実行例

```
db2=> SELECT LENGTH('ABC'),LENGTH('あいう') FROM SYSIBM.SYSDUMMY1

1           2
----------- -----------
          3           6

  1 レコードが選択されました。
```

3.77 VARIANCE

列値の標本分散を戻します。

Oracleの場合

書式 **VARIANCE(式)**

実行例

```
SQL> SELECT VARIANCE(SAL) FROM EMP;

VARIANCE(SAL)
-------------
   1338291.67
```

DB2の場合

書式 **VARIANCE(式) * COUNT(*) / (COUNT(*) - 1)**

実行例

```
db2=> SELECT VARIANCE(SAL)*COUNT(*)/(COUNT(*)-1) FROM EMP

1
------------------------
  +1.33829166666667E+006

  1 レコードが選択されました。
```

注意点 どちらのデータベースでも関数名は同じですが、Oracleは標本分散を戻し、DB2 UDBは母分散を戻します。このため、DB2 UDBでOracleと同じ結果を得るには、次の式を用いて補正する必要があります。

```
* COUNT(*) / (COUNT(*) - 1)
```

3.78 BIN_TO_NUM

ビット・ベクトル（1と0を組み合せた文字列）を同等の数値に変換します。この関数の各引数は、ビット・ベクトルのビットを表します。

Oracleの場合

書式 **BIN_TO_NUM(引数1,引数2,・・・引数n)**

実行例

```
SQL> SELECT BIN_TO_NUM(1,0),BIN_TO_NUM(1,0,0) FROM DUAL;

BIN_TO_NUM(1,0) BIN_TO_NUM(1,0,0)
--------------- -----------------
              2                 4
```

DB2の場合

書式 **対応する関数はありません。**

実行例

注意点

OracleのBIN_TO_NUM関数は、引数として任意の数値データ型、または暗黙的にNUMBER型に変換可能な数値以外のデータ型をとります。各引数は、0または1に評価される必要があります。この関数はOracleのNUMBER型の値を戻します。
BIN_TO_NUM関数は、データ・ウェアハウスのアプリケーションで、グルーピング・セットを使用して、マテリアライズド・ビューから対象グループを検索する場合に有効です。

3.79 BITAND

引数1、引数2のビットに対するAND操作の計算を行い、整数を戻します。

Oracleの場合

書式 `BITAND(引数1,引数2)`

実行例

DB2の場合

書式 対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、BITANDをサンプルUDFとして提供しています。

3.80 CAST

組み込みデータ型またはコレクション型の値を、別の組み込みデータ型またはコレクション型の値に変換します。

Oracleの場合

書式 **CAST(引数 AS データ型)**

実行例

```
SQL> SELECT CAST(SYSDATE AS TIMESTAMP WITH LOCAL TIME ZONE) FROM DUAL;

CAST(SYSDATEASTIMESTAMPWITHLOCALTIMEZONE)
---------------------------------------------------------------------------
05-03-31 05:58:42.000000
```

DB2の場合

書式 **CAST(引数 AS データ型)**

実行例

```
db2=> SELECT CAST(CURRENT_TIMESTAMP AS TIMESTAMP) FROM SYSIBM.SYSDUMMY1

1
--------------------------
2005-03-31-05.58.23.978001

  1 レコードが選択されました。
```

3.81 INITCAP

各単語の最初の文字を大文字、残りの文字を小文字にして文字列を戻します。単語は空白または英数字以外の文字で区切ります。

Oracleの場合

書式 **INITCAP(文字列型)**

実行例

```
SQL> SELECT INITCAP('RELATIONAL DATABASE-SYSTEM') FROM DUAL;

INITCAP('RELATIONALDATABAS
--------------------------
Relational Database-System
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、INITCAPをサンプルUDFとして提供しています。

3.82 LPAD

引数1の左側に引数2に指定した文字を連続的に埋め込んで指定桁数にして戻します。

Oracleの場合

書式 **LPAD(引数1,桁数,引数2)**

実行例

```
SQL> SELECT LPAD(SAL,7,'-') FROM EMP WHERE DEPTNO = 20;

LPAD(SAL,7,'-'
--------------
----800
---2975
---3000
---1100
---3000
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、LPADをサンプルUDFとして提供しています。

3.83 RPAD

引数1の右側に引数2に指定した文字を連続的に埋め込んで指定桁数にして戻します。

Oracleの場合

書式 **RPAD(引数1,桁数,引数2)**

実行例

```
SQL> SELECT RPAD(SAL,7,'-') FROM EMP WHERE DEPTNO = 20;

RPAD(SAL,7,'-'
--------------
800----
2975---
3000---
1100---
3000---
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、RPADをサンプルUDFとして提供しています。

3.84 NLS_INITCAP

各単語の最初の文字を大文字、残りの文字を小文字にして文字列を戻します。単語は空白または英数字以外の文字で区切ります。

Oracleの場合

書式 **NLS_INITCAP(文字列,'NLSパラメータ')**

実行例

```
SQL> SELECT NLS_INITCAP('ijsland','NLS_SORT = XDutch') FROM DUAL;

NLS_INI
-------
Ijsland
```

DB2の場合

書式 **対応する関数はありません。**

実行例

3.85 NLS_UPPER

すべての文字を大文字にして文字列を戻します。

Oracleの場合

書式 **NLS_UPPER(文字列)**

実行例

```
SQL> SELECT NLS_UPPER('abc') FROM DUAL;

NLS_UP
------
ABC
```

DB2の場合

書式 **対応する関数はありません。**

実行例

3.86 NLS_LOWER

すべての文字を小文字にした文字列を戻します。

Oracle の場合

書式 **NLS_LOWER(文字列)**

実行例

```
SQL> SELECT NLS_LOWER('ABC') FROM DUAL;

NLS_LO
------
abc
```

DB2 の場合

書式 **対応する関数はありません。**

実行例

3.87 NTILE

これは、順序付けられたデータセットを引数に指定した数のバケットに分割し、適切なバケット番号を各行に割り当てます。バケットには1～引数の番号が付けられます。

Oracleの場合

書式 NTILE(引数)

実行例

```
SQL> SELECT ENAME,SAL,NTILE(2) OVER(ORDER BY SAL)
  2  FROM EMP WHERE DEPTNO = 30;

ENAME              SAL NTILE(2)OVER(ORDERBYSAL)
---------- ---------- ------------------------
JAMES              950                        1
WARD              1250                        1
MARTIN            1250                        1
TURNER            1500                        2
ALLEN             1600                        2
BLAKE             2850                        2

6行が選択されました。
```

DB2の場合

書式 対応する関数はありませんが、次のようにして同様の結果を得ることができます。

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT ENAME, SAL
     , CASE
       WHEN rn <= r*(d+1) THEN
            (rn  -1)/(d+1) +1
       ELSE (rn-r-1)/d     +1
       END  NTILE
  FROM (SELECT EMP.*
             , ROW_NUMBER() OVER(ORDER BY SAL) AS rn
             , COUNT(*)     OVER() / 2         AS d
             , MOD(COUNT(*) OVER() , 2)        AS r
          FROM EMP
         WHERE DEPTNO = 30) AS S;
------------------------------------------------------------------------

ENAME      SAL       NTILE
---------- --------- --------------------
JAMES         950.00                    1
WARD         1250.00                    1
MARTIN       1250.00                    2
TURNER       1500.00                    2
ALLEN        1600.00                    2
BLAKE        2850.00                    2

  6 レコードが選択されました。
```

3.88 CHARTOROWID

CHAR、VARCHAR2、NCHAR、NVARCHAR2の各データ型の値をROWIDデータ型に変換します。

Oracleの場合

書式 **CHARTOROWID(文字列)**

実行例

```
SQL> SELECT EMPNO,ENAME FROM EMP WHERE ROWID = CHARTOROWID('AAAMZYAAEAAAAF
➥kAAA');

     EMPNO ENAME
---------- ----------
      7369 SMITH
```

DB2の場合

書式 **対応する関数はありません。**

実行例

注意点 | DB2 UDBにはROWIDがないため、CHARTROWID関数に対応する関数はありません。

3.89 CONVERT

あるキャラクター・セットから別のキャラクター・セットに変換します。

Oracleの場合

書式 **CONVERT(文字列型,変換後キャラクター・セット,変換前キャラクター・セット)**

実行例

```
SQL> SELECT CONVERT('ABCDE','UTF8','US7ASCII') FROM DUAL;

CONVE
-----
ABCDE
```

DB2の場合

書式 **対応する関数はありません。**

実行例

3.90 ROWIDTOCHAR

ROWIDの値をVARCHAR2データ型に変換します。

Oracleの場合

書式 **ROWIDTOCHAR(ROWID)**

実行例

```
SQL> SELECT ROWIDTOCHAR(ROWID) FROM EMP WHERE EMPNO  = 7369;

ROWIDTOCHAR(ROWID)
------------------
AAAMZYAAEAAAAFkAAA
```

DB2の場合

書式 **対応する関数はありません。**

実行例

注意点 | DB2 UDBにはROWIDがないため、ROWIDを加工する関数はありません。

3.91 TO_DATE

文字列型の文字列をDATE型の値に変換します。書式は、文字列の書式を指定する日時書式モデルです。書式を指定しない場合、文字列はデフォルトの日付書式である必要があります。

Oracleの場合

書式 **TO_DATE(文字列,書式)**

実行例

```
SQL> SELECT TO_DATE('20050101','YYYYMMDD') FROM DUAL;

TO_DATE(
--------
05-01-01
```

DB2の場合

書式 **対応する関数はありません。同名の関数はありますが、書式として 'YYYY-MM-DD HH24:MI:SS' しか指定できません。**

実行例

```
db2=> SELECT DATE(INSERT(INSERT('20050101',5,0,'-'),8,0,'-')) FROM SYSIBM.
➥SYSDUMMY1

1
----------
2005-01-01

  1 レコードが選択されました。
```

注意点 DB2 UDBのCAST関数を使用して型変換は可能ですが、TO_DATE(文字列, 書式)関数が提供するような書式の指定はできません。一般的には、式を工夫して同等の結果を得ます。

3.92 TO_NUMBER

指定した引数をNUMBERデータ型の値に変換します。

Oracleの場合

書式 **TO_NUMBER(引数,書式)**

実行例

```
SQL> SELECT TO_NUMBER('12,345','99,999') FROM DUAL;

TO_NUMBER('12,345','99,999')
----------------------------
                       12345
```

DB2の場合

書式 **対応する関数はありません。**

実行例

注意点 DB2 UDBのCAST関数を使用して型変換は可能ですが、TO_DATE(文字列, 書式)関数が提供するような書式の指定はできません。

3.93 TO_SINGLE_BYTE

マルチバイト文字を対応するシングルバイト文字に変換して文字列を戻します。

Oracleの場合

書式 **TO_SINGLE_BYTE(文字列)**

実行例

```
SQL> SELECT TO_SINGLE_BYTE('ABC') FROM DUAL;

TO_
---
ABC
```

DB2の場合

書式 **対応する関数はありません。**

実行例

3.94 ADD_MONTH

指定した日付に指定したn月数を加えて戻します。

Oracleの場合

書式 **ADD_MONTH(DATE型の列,n)**

実行例

```
SQL> SELECT SYSDATE,ADD_MONTHS(SYSDATE,3) FROM DUAL;

SYSDATE  ADD_MONT
-------- --------
05-03-26 05-06-26
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、ADD_MONTHをサンプルUDFとして提供しています。

3.95 LAST_DAY

指定した日付を含む月の最終日の日付を戻します。

Oracleの場合

書式 **LAST_DAY(日付型)**

実行例

```
SQL> SELECT SYSDATE,LAST_DAY(SYSDATE) FROM DUAL;

SYSDATE  LAST_DAY
-------- --------
05-03-26 05-03-31
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、LAST_DAYをサンプルUDFとして提供しています。

3.96 LEAST

リストされた引数の中から最小値を戻します。

Oracle の場合

書式 **LEAST(引数)**

実行例

```
SQL> SELECT LEAST(100,30,101) FROM DUAL;

LEAST(100,30,101)
-----------------
               30
```

DB2 の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、LEASTをサンプルUDFとして提供しています。

3.97 MONTHS_BETWEEN

日付1と日付2の間の月数を戻します。日付1が日付2より後の日付の場合、結果は正の値になります。日付1が日付2より前の日付の場合、結果は負の値になります。日付1および日付2が、月の同じ日または月の最終日の場合、結果は常に整数になります。それ以外の場合、Oracleデータベースは結果の小数部を1か月31日として計算し、日付1と日付2の差を割り出します。

Oracleの場合

書式 **MONTHS_BETWEEN(日付型1,日付型2)**

実行例

```
SQL> SELECT MONTHS_BETWEEN('05-01-01','04-10-01') FROM DUAL;

MONTHS_BETWEEN('05-01-01','04-10-01')
-------------------------------------
                                    3
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

3.98 NEXT_DAY

weekdayで指定した曜日で、dateよりあとの最初の日付を戻します。

Oracleの場合

書式 **NEXT_DAY(date,weekday)**

実行例

```
SQL> SELECT SYSDATE,NEXT_DAY(SYSDATE,2) FROM DUAL;

SYSDATE  NEXT_DAY
-------- --------
05-03-26 05-03-28
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、NEXT_DAYをサンプルUDFとして提供しています。

3.99 NEW_TIME

タイムゾーンzone1のdateをタイムゾーンzone2の日時を戻します。この関数を使用する前に、24時間で表示されるようにNLS_DATE_FORMATパラメータを設定する必要があります。

Oracleの場合

書式 **NEW_TIME(date,zone1,zone2)**

実行例

```
SQL> SELECT NEW_TIME(TO_DATE('05-03-26 23:14:27','YY-MM-DD HH24:MI:SS'),
➥'AST','GMT') FROM DUAL;

NEW_TIME(TO_DATE(
-----------------
05-03-27 03:14:27
```

DB2の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、NEW_TIMEをサンプルUDFとして提供しています。

3.100 GREATEST

リストされた引数の中から最大値を戻します。

Oracle の場合

書式 **GREATEST(引数)**

実行例

```
SQL> SELECT GREATEST(100,30,101) FROM DUAL;

GREATEST(100,30,101)
--------------------
                 101
```

DB2 の場合

書式 **対応する関数はありませんが、UDF（ユーザー定義関数）として作成しておくことができます。**

実行例

注意点 Sample UDFs for Migration（70ページ参照）では、GREATESTをサンプルUDFとして提供しています。

3.101 NULLIF

NULLIF関数は式1と式2を比較します。同じである場合は、NULLを戻します。異なる場合は、式1を戻します。

Oracleの場合

書式 **NULLIF(式1, 式2)**

実行例

```
SQL> SELECT E.ENAME,E.JOB "Current Job"
  2         ,NULLIF(J.JOB,E.JOB) "Old Job"
  3  FROM EMP E,JOB_HISTORY J
  4  WHERE E.EMPNO = J.EMPNO;

ENAME      Current J Old Job
---------- --------- ----------
SMITH      CLERK     ANALYST
ALLEN      SALESMAN  CLERK
JONES      MANAGER
```

DB2の場合

書式 **NULLIF(式1, 式2)**

実行例

```
db2=> db2 => SELECT E.ENAME,E.JOB "Current Job",NULLIF(J.JOB,E.JOB) "Old
➥Job" FROM EMP E,JOB_HISTORY J WHERE E.EMPNO = J.EMPNO

ENAME      Current Job Old Job
---------- ----------- ----------
SMITH      CLERK       ANALYST
ALLEN      SALESMAN    CLERK
JONES      MANAGER     -

  3 レコードが選択されました。
```

3.102 NUMTODSINTERVAL

nを引数の日から秒を単位として日時に変換します。

Oracleの場合

書式 **NUMTODSINTERVAL(n [,'DAY' | 'HOUR' | 'MINUTE' | 'SECOND'])**

実行例

```
SQL> SELECT NUMTODSINTERVAL(30,'HOUR') FROM DUAL;

NUMTODSINTERVAL(30,'HOUR')
---------------------------------------------------
+000000001 06:00:00.000000000
```

DB2の場合

書式 **対応する関数はありませんが、式で同じ形式を得ることができます。使う機会が多ければ、ユーザー定義関数（UDF）を作っておくこともできます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DIGITS(DAYS(TIMESTAMP('00010101000000') + 30 HOURS) -1)
       || ' ' || SUBSTR(CHAR(TIMESTAMP('00010101000000') + 30
HOURS),12,15)
  FROM SYSIBM.SYSDUMMY1;
------------------------------------------------------------------------------

1
--------------------------
0000000001 06.00.00.000000

  1 レコードが選択されました。
```

3.103 NUMTOYMINTERVAL

nを引数の年か月として年月に変換します。

Oracleの場合

書式 **NUMTOYMINTERVAL(n [,'YEAR' | 'MONTH'])**

実行例

```
SQL> SELECT NUMTOYMINTERVAL(30,'MONTH') FROM DUAL;

NUMTOYMINTERVAL(30,'MONTH')
---------------------------------------------------
+000000002-06
```

DB2の場合

書式 **対応する関数はありませんが、簡単な式で同じ結果を得ることができます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DIGITS(30/12) || '-' || SUBSTR(DIGITS(MOD(30,12)),9,2)
  FROM SYSIBM.SYSDUMMY1;
--------------------------------------------------------------------------

1
-------------
0000000002-06

  1 レコードが選択されました。
```

3.104 NVL2

指定された引数がNULL値かどうかに基づく問い合わせによって戻される値を判断できます。引数1がNULL値でない場合、NVL2は引数2を戻します。式1がNULL値の場合、NVL2は引数3を戻します。

Oracleの場合

書式 **NVL2(引数1,引数2,引数3)**

実行例

```
SQL> SELECT COMM,SAL,NVL2(COMM,SAL+COMM,SAL) FROM EMP WHERE DEPTNO = 30;

      COMM        SAL NVL2(COMM,SAL+COMM,SAL)
---------- ---------- -----------------------
       300       1600                    1900
       500       1250                    1750
      1400       1250                    2650
                 2850                    2850
         0       1500                    1500
                  950                     950

6行が選択されました。
```

DB2の場合

書式 **対応する関数はありませんが、CASE式で同じ結果を得ることができます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT COMM, SAL
     , CASE
       WHEN COMM IS NOT NULL THEN
            SAL+COMM
       ELSE SAL
       END
  FROM EMP WHERE DEPTNO = 30;
--------------------------------------------------------------------------

COMM       SAL        3
---------- ---------- ----------
   300.00   1600.00    1900.00
   500.00   1250.00    1750.00
  1400.00   1250.00    2650.00
        -   2850.00    2850.00
     0.00   1500.00    1500.00
        -    950.00     950.00

  6 レコードが選択されました。
```

3.105 UID

セッション・ユーザー（ログインしているユーザー）を一意に識別する整数を戻します。

Oracleの場合

書式 **UID**

実行例

```
SQL> SELECT UID FROM DUAL;

       UID
----------
        73
```

DB2の場合

書式 **対応する関数はありません。**

実行例

3.106 USERENV

現行のセッションに関する情報を戻します。

Oracle の場合

書式 **USERENV**

実行例

```
SQL> SELECT USERENV('LANGUAGE') FROM DUAL;

USERENV('LANGUAGE')
----------------------------------------------------
JAPANESE_JAPAN.JA16SJIS
```

DB2 の場合

書式 **対応する関数はありません。**

実行例

3.107 MEDIAN

MEDIANは、連続分散モデルを想定する逆分散関数です。この関数は、数値または日時値をとり、その中央値、または値のソート後に中央値となる補間された値を戻します。計算では、NULL値は無視されます。

Oracleの場合

書式 **MEDIAN(列)**

実行例

```
SQL> SELECT MEDIAN(SAL),MIN(SAL),MAX(SAL) FROM EMP WHERE DEPTNO = 10;

MEDIAN(SAL)   MIN(SAL)   MAX(SAL)
----------- ---------- ----------
       2450       1300       5000
```

DB2の場合

書式 **対応する関数はありませんが、次のようにして同様の結果を得ることができます。**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT MAX(fsal) + (MAX(csal) - MAX(fsal)) / 2 AS MEDIAN
     , MIN(sal) AS MIN_SAL
 MAX(sal) AS MAX_SAL
  FROM (SELECT sal
             , CASE rn
               WHEN (cnt + MOD(cnt,2)) / 2 THEN
                    sal
               END  AS fsal
             , CASE rn
               WHEN (cnt - MOD(cnt,2)) / 2 + 1 THEN
                    sal
               END  AS csal
          FROM (SELECT sal
                     , COUNT(sal) OVER() cnt
                     , ROWNUMBER() OVER(ORDER BY sal) rn
                  FROM EMP
                 WHERE DEPTNO = 10
                ) Q
       ) P;
------------------------------------------------------------------------------

MEDIAN                            MIN_SAL   MAX_SAL
--------------------------------- --------- ---------
   2450.000000000000000000000000   1300.00   5000.00

  1 レコードが選択されました。
```

注意点 DB2 UDBで「MAX(fsal)+(MAX(csal) - MAX(fsal))/2」を「(MAX(fsal)+MAX(csal))/2」としなかったのは、日付データにも同じような形式が使えるからです。日付の中央値を計算する場合は、次のように加算の2番目の結果をDEC(8,0)にCASTする必要があります。

```
MAX(fhiredate) + CAST((MAX(chiredate) - MAX(fhiredate)) / 2 AS
DEC(8,0)) AS MEDIAN
```

4

複雑な問い合わせ

4.1 WHERE句に使用する問い合わせ

ACCOUNTINGに所属している人を求めます。表に対して直接的な検索条件が指定できない場合、ほかの表に対する検索結果を求めたい表に対する検索条件として使用します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 = (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM DEPT
➥WHERE DNAME = 'ACCOUNTING');

 DEPTNO ENAME
------- ----------
     10 CLARK
     10 KING
     10 MILLER

3行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 = (SELECT 列名 FROM 表名 WHERE 検索条件)
```

または

```
SELECT 表別名1.列名 FROM 表名1 表別名1, 表名2 表別名2
WHERE 表別名1.列名 = 表別名2.列名 AND 表2に対する検索条件)
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT DEPTNO,ENAME FROM EMP
 WHERE DEPTNO = (SELECT DEPTNO FROM DEPT WHERE DNAME = 'ACCOUNTING');
--------------------------------------------------------------------------

DEPTNO ENAME
------ -----
    10 CLARK
    10 KING
    10 MILLE

  3 レコードが選択されました。
```

別実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT E.DEPTNO, ENAME
  FROM EMP  E
     , DEPT D
 WHERE E.DEPTNO = D.DEPTNO
   AND D.DNAME  = 'ACCOUNTING';
--------------------------------------------------------------------------

DEPTNO ENAME
------ -----
    10 CLARK
    10 KING
    10 MILLE
```

3 レコードが選択されました。

注意点 | SELECT文のWHERE句に別の問い合わせの結果を比較条件として使用できます。

4.2 1行のみ戻す副問い合わせ

社長が所属している部署名を求めます。副問い合わせの検索結果として1行だけ戻されることが明らかな場合の検索です。

Oracleの場合

書式

```
SELECT列名 FROM 表名1
WHERE 列名 = (SELECT 列名 FROM 表名WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,DNAME FROM DEPT WHERE DEPTNO = (SELECT DEPTNO FROM EMP
➥WHERE JOB = 'PRESIDENT');

 DEPTNO DNAME
------- --------------
     10 ACCOUNTING

1行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 = (SELECT 列名 FROM 表名 WHERE 検索条件)
```

または、

```
SELECT 表別名1.列名 FROM 表名1 表別名1, 表名2 表別名2
WHERE 表別名1.列名 = 表別名2 列名 AND 表2に対する検索条件)
```

実行例

```
db2=> SELECT DEPTNO,DNAME FROM DEPT WHERE DEPTNO = (SELECT DEPTNO FROM EMP
➥WHERE JOB = 'PRESIDENT')

DEPTNO DNAME
------ --------------
   10. ACCOUNTING

  1 レコードが選択されました。
```

注意点

副問い合わせから戻される結果が1行だけであることが明確な場合は、主問い合わせの比較条件に比較演算子（=、<>、>、<、>=、<=）を使用することができます。
DB2 UDBでは、副問い合わせを副照会と呼んでいます。

4.3 複数行戻す副問い合わせ

部門番号30に所属している人と同じ職種の人を求めます。副問い合わせの検索結果として複数行戻される可能性がある場合の検索です。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 IN (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT JOB,ENAME FROM EMP WHERE JOB IN (SELECT JOB FROM EMP WHERE DEP
➥TNO = 30);

JOB       ENAME
--------- ----------
SALESMAN  TURNER
SALESMAN  MARTIN
SALESMAN  WARD
SALESMAN  ALLEN
MANAGER   CLARK
MANAGER   BLAKE
MANAGER   JONES
CLERK     MARY
CLERK     MILLER
CLERK     JAMES
CLERK     ADAMS
CLERK     SMITH

12行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 IN (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT JOB,ENAME FROM EMP WHERE JOB IN (SELECT JOB FROM EMP WHERE DE
➥PTNO = 30)

JOB       ENAME
--------- ----------
CLERK     SMITH
SALESMAN  ALLEN
SALESMAN  WARD
MANAGER   JONES
SALESMAN  MARTIN
MANAGER   BLAKE
MANAGER   CLARK
SALESMAN  TURNER
CLERK     ADAMS
CLERK     JAMES
CLERK     MILLER
CLERK     MARY

  12 レコードが選択されました。
```

注意点

副問い合わせから戻される結果が複数行の可能性がある場合は、主問い合わせの比較条件に比較演算子（=、<>、>、<、>=、<=）を使用することができません。述語のIN、NOT IN、ANY、ALLを使用する必要があります。
一般に、INとEXISTSは互いに書き換えができます。以下に、書き換えの例を示します。

（1）比較しやすいように、フォーマットを変更し、表に別名を付けます。

```
SELECT JOB, ENAME
  FROM EMP A
 WHERE A.JOB IN
    (SELECT B.JOB
       FROM EMP.B
      WHERE B.DEPTNO = 30
    )
;
```

（2）EXISTSを使うと、次のように書き換えられます。

```
SELECT JOB, ENAME
  FROM EMP A
 WHERE EXISTS
    (SELECT *
       FROM EMP.B
      WHERE A.JOB = B.JOB
        AND B.DEPTNO = 30
    )
;
```

4.4 副問い合わせの結果のいずれかに一致する行を表示する

2月入社の人たちのいずれかの給与と等しい人を探します。副問い合わせから戻される複数の値のうちのいずれかの値と等しい行を検索します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 =ANY (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL =ANY (SELECT SAL
➥FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');

 DEPTNO ENAME      HIREDATE     SAL
------- ---------- -------- -------
     30 ALLEN      81-02-20    1600
     30 MARTIN     81-09-28    1250
     30 WARD       81-02-22    1250

3行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 =ANY (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL =ANY (SELECT SAL
➥FROM EMP WHERE MONTH(HIREDATE) = 2)

DEPTNO ENAME      HIREDATE   SAL
------ ---------- ---------- ---------
   30. ALLEN      1981-02-20   1600.00
   30. WARD       1981-02-22   1250.00
   30. MARTIN     1981-09-28   1250.00

  3 レコードが選択されました。
```

注意点 副問い合わせから戻される複数の値のうちのいずれかの値と等しい行を検索する場合には、IN述語または「=ANY」を使用します。

4.5 副問い合わせの結果のすべてに一致する行を表示する

2月入社の人たちのすべての給与と等しい人を探します。副問い合わせから戻される複数の値のすべての値と等しい行を検索します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 =ALL (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL =ALL (SELECT SAL
➥FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');

レコードが選択されませんでした。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 =ALL (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL =ALL (SELECT SAL
➥FROM EMP WHERE MONTH(HIREDATE) = 2)

DEPTNO ENAME      HIREDATE   SAL
------ ---------- ---------- ---------

  0 レコードが選択されました。
```

注意点 「=ALL」は、副問い合わせから戻される複数の値のすべての値と等しいことを評価する比較述部です。しかし、異なる複数の値すべてと等しい値は存在しないため、結果はゼロ件となります。

4.6 副問い合わせの結果の最小値より大きい行を表示する

2月入社の人たちの給与の中で最小金額より大きい金額の人を探します。副問い合わせから戻される複数の値のいずれかより大きい人を求めることにより、副問い合わせの結果の最小値より大きい行を求めることができます。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 >ANY (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL >=ANY (SELECT SAL
➥FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');

 DEPTNO ENAME      HIREDATE      SAL
------- ---------- --------- -------
     10 KING       81-11-17     5000
     20 SCOTT      82-12-09     3000
     20 FORD       81-12-03     3000
     20 JONES      81-04-02     2975
     30 BLAKE      81-05-01     2850
     10 CLARK      81-06-09     2450
     30 ALLEN      81-02-20     1600
     30 TURNER     81-09-08     1500
     10 MILLER     82-01-23     1300
        MARY       83-04-01     1300
     30 WARD       81-02-22     1250
     30 MARTIN     81-09-28     1250

12行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
WHERE 列名 >ANY (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL >=ANY (SELECT
➥SAL FROM EMP WHERE MONTH(HIREDATE) = 2)

DEPTNO ENAME      HIREDATE   SAL
------ ---------- ---------- ---------
   30. ALLEN      1981-02-20   1600.00
   30. WARD       1981-02-22   1250.00
   20. JONES      1981-04-02   2975.00
   30. MARTIN     1981-09-28   1250.00
   30. BLAKE      1981-05-01   2850.00
   10. CLARK      1981-06-09   2450.00
   20. SCOTT      1982-12-09   3000.00
   10. KING       1981-11-17   5000.00
   30. TURNER     1981-09-08   1500.00
   20. FORD       1981-12-03   3000.00
   10. MILLER     1982-01-23   1300.00
     - MARY       1983-04-01   1300.00

  12 レコードが選択されました。
```

注意点

「>ANY」は、副問い合わせから戻される複数の値のいずれかより大きい行を求めます。いずれかより大きければよいということは、最も小さい値より大きければよいということであるため、副問い合わせの結果の最小値より大きい行を求めることができます。

Oracleの場合、次のSQL文でも同じ結果が求められますが、例に示したSQL文のほうがグループ化する負荷が減り効率的です。

```
SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL >=
(SELECT MIN(SAL) FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');
```

4.7 副問い合わせの結果の最大値より大きい行を表示する

2月入社の人たちの給与の中で最大金額より大きい金額の人を探します。副問い合わせから戻される複数のすべての値より大きい人を求めることにより、副問い合わせの結果の最大値より大きい行を求めることができます。

Oracleの場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 >=ALL (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL >=ALL (SELECT SAL
➥FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');

 DEPTNO ENAME      HIREDATE     SAL
------- ---------- -------- -------
     30 ALLEN      81-02-20    1600
     20 JONES      81-04-02    2975
     30 BLAKE      81-05-01    2850
     10 CLARK      81-06-09    2450
     20 SCOTT      82-12-09    3000
     10 KING       81-11-17    5000
     20 FORD       81-12-03    3000

7行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 >=ALL (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL >=ALL (SELECT
➥SAL FROM EMP WHERE MONTH(HIREDATE) = 2)

DEPTNO ENAME      HIREDATE   SAL
------ ---------- ---------- ---------
   30. ALLEN      1981-02-20   1600.00
   20. JONES      1981-04-02   2975.00
   30. BLAKE      1981-05-01   2850.00
   10. CLARK      1981-06-09   2450.00
   20. SCOTT      1982-12-09   3000.00
   10. KING       1981-11-17   5000.00
   20. FORD       1981-12-03   3000.00

  7 レコードが選択されました。
```

注意点

「>ALL」は、副問い合わせから戻されるすべての値より大きい行を求めます。すべての値より大きければよいということは、最も大きい値より大きければよいということであるため、副問い合わせの結果の最大値より大きい行を求めることができます。

Oracleの場合、次のSQL文でも同じ結果が求められますが、例に示したSQL文のほうがグループ化する負荷が減り効率的です。

```
SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL >=
(SELECT MAX(SAL) FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');
```

4.8 副問い合わせの結果の最小値より小さい行を表示する

2月入社の人たちの給与の中で最小金額より小さい金額の人を探します。副問い合わせから戻される複数のすべて値のより小さい人を求めることにより、副問い合わせの結果の最小値より小さい行を求めることができます。

Oracleの場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 <=ALL (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL <=ALL (SELECT SAL
➥FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');

 DEPTNO ENAME      HIREDATE      SAL
------- ---------- -------- -------
     20 SMITH      80-12-17      800
     30 WARD       81-02-22     1250
     30 MARTIN     81-09-28     1250
     20 ADAMS      83-01-12     1100
     30 JAMES      81-12-03      950

5行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 <=ALL (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL <=ALL (SELECT
➥SAL FROM EMP WHERE MONTH(HIREDATE) = 2)

DEPTNO ENAME      HIREDATE   SAL
------ ---------- ---------- ---------
   20. SMITH      1980-12-17    800.00
   30. WARD       1981-02-22   1250.00
   30. MARTIN     1981-09-28   1250.00
   20. ADAMS      1983-01-12   1100.00
   30. JAMES      1981-12-03    950.00

  5 レコードが選択されました。
```

注意点

「<ALL」は、副問い合わせから戻されるすべての値より小さい行を求めます。すべての値より小さければよいということは、最も小さい値より小さければよいということであるため、副問い合わせの結果の最小値より小さい行を求めることができます。

Oracleの場合、次のSQL文でも同じ結果が求められますが、例に示したSQL文のほうがグループ化する負荷が減り効率的です。

```
SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL <=
(SELECT MIX(SAL) FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');
```

4.9 副問い合わせの結果の最大値より小さい行を表示する

2月入社の人たちのいずれかの給与より小さい金額の人を探します。副問い合わせから戻される複数のいずれかの値のより小さい人を求めることにより、副問い合わせの結果の最大値より小さい行を求めることができます。

Oracleの場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 <=ANY (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL <=ANY (SELECT SAL
➥FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');

 DEPTNO ENAME      HIREDATE     SAL
------- ---------- -------- -------
     20 SMITH      80-12-17     800
     30 JAMES      81-12-03     950
     20 ADAMS      83-01-12    1100
     30 WARD       81-02-22    1250
     30 MARTIN     81-09-28    1250
     10 MILLER     82-01-23    1300
        MARY       83-04-01    1300
     30 TURNER     81-09-08    1500
     30 ALLEN      81-02-20    1600

9行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名 <=ANY (SELECT 列名 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL <=ANY (SELECT
➥SAL FROM EMP WHERE MONTH(HIREDATE) = 2)

DEPTNO ENAME      HIREDATE   SAL
------ ---------- ---------- ---------
   20. SMITH      1980-12-17    800.00
   30. ALLEN      1981-02-20   1600.00
   30. WARD       1981-02-22   1250.00
   30. MARTIN     1981-09-28   1250.00
   30. TURNER     1981-09-08   1500.00
   20. ADAMS      1983-01-12   1100.00
   30. JAMES      1981-12-03    950.00
   10. MILLER     1982-01-23   1300.00
     - MARY       1983-04-01   1300.00

  9 レコードが選択されました。
```

注意点

「<ANY」は、副問い合わせから戻される行のいずれかより小さい行を求めます。いずれかより小さければよいということは、最も大きい値より小さければよいということであるため、副問い合わせの結果の最大値より小さい行を求めることができます。これは、次のSQL文でも同じ結果が求められますが、例に示したSQL文のほうがグループ化する負荷が減り効率的です。

```
SELECT DEPTNO,ENAME,HIREDATE,SAL FROM EMP WHERE SAL <=
(SELECT MAX(SAL) FROM EMP WHERE TO_CHAR(HIREDATE,'MM') = '02');
```

DB2 UDBでは、副問い合わせから戻される複数の値とそれぞれ比較するより、副問い合わせの結果の最大値と1回だけ比較したほうが効率が良いと考えて、例に示したSQL文でも副問い合わせの結果の最大値と比較するようにSQLを書き換える場合があります。

4.10 EXISTSを使用した副問い合わせ

ほかの表に存在する値を表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1 WHERE 列名 EXISTS
(SELECT 列名 FROM 表名2 WHERE 表名1.列名 = 表名2.列名)
```

実行例

```
SQL> SELECT * FROM DEPT WHERE EXISTS (SELECT * FROM EMP WHERE EMP.DEPTNO =
➥DEPT.DEPTNO);

 DEPTNO DNAME          LOC
------- -------------- -------------
     20 RESEARCH       DALLAS
     30 SALES          CHICAGO
     10 ACCOUNTING     NEW YORK

3行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1 WHERE 列名 EXISTS
(SELECT 列名 FROM 表名2 WHERE 表名1.列名 = 表名2.列名)
```

実行例

```
db2=> SELECT * FROM DEPT WHERE EXISTS (SELECT * FROM EMP WHERE EMP.DEPTNO
➥= DEPT.DEPTNO)

DEPTNO DNAME          LOC
------ -------------- -------------
   10. ACCOUNTING     NEW YORK
   20. RESEARCH       DALLAS
   30. SALES          CHICAGO

  3 レコードが選択されました。
```

注意点 EXISTSは、副問い合わせの中で1つ以上の行が戻される場合に真（TRUE）を返します。
INを使った副問い合わせを使用するよりも効率的に処理されます。

4.11 NOT EXISTSを使用した副問い合わせ

ほかの表には存在しない値を表示する。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1 WHERE 列名 NOT EXISTS
(SELECT 列名 FROM 表名2 WHERE 表名1.列名 = 表名2.列名)
```

実行例

```
SQL> SELECT * FROM DEPT WHERE NOT EXISTS (SELECT * FROM EMP WHERE EMP.DEPT
➥NO = DEPT.DEPTNO);

 DEPTNO DNAME          LOC
------- -------------- -------------
     40 OPERATIONS     BOSTON

1行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1 WHERE 列名 NOT EXISTS
(SELECT 列名 FROM 表名2 WHERE 表名1.列名 = 表名2.列名)
```

実行例

```
db2=> SELECT * FROM DEPT WHERE NOT EXISTS (SELECT * FROM EMP WHERE EMP.DEP
➥TNO = DEPT.DEPTNO)

DEPTNO DNAME          LOC
------ -------------- -------------
   40. OPERATIONS     BOSTON

  1 レコードが選択されました。
```

注意点

NOT EXISTSは、副問い合わせから1つも行が戻されない場合に真（TRUE）を返します。
NOT INを使った副問い合わせを使用するよりも効率的に処理されます。

4.12 複数列を使用した副問い合わせ

注文番号2368と同じ商品番号と数量の組合せと一致する注文を探します。副問い合わせと比較する列が1つではなく、複数の列の組み合わせた値とセットで一致する行を検索します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名 WHERE (列名1,列2) 比較演算子
(SELECT 列名1,列名2 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT ORDER_ID,PRODUCT_ID,QUANTITY FROM ORDER_ITEMS
  2  WHERE (PRODUCT_ID,QUANTITY) IN (SELECT PRODUCT_ID,QUANTITY FROM ORDER
➥_ITEMS WHERE ORDER_ID = 2368);

ORDER_ID PRODUCT_ID QUANTITY
-------- ---------- --------
    2368       3129       72
    2368       3143       75
    2368       3155       75
    2368       3106      150
    2396       3106      150
    2419       3106      150
    2368       3110       60
    2368       3117       62
    2368       3123       70
    2368       3127       70

10行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名 WHERE (列名1,列2) 比較演算子
(SELECT 列名1,列名2 FROM 表名 WHERE 検索条件)
```

実行例

```
db2=> SELECT ORDER_ID,PRODUCT_ID,QUANTITY FROM ORDER_ITEMS WHERE (PRODUCT_
➥ID,QUANTITY) IN (SELECT PRODUCT_ID,QUANTITY FROM ORDER_ITEMS WHERE ORDER_
➥ID = 2368)

ORDER_ID       PRODUCT_ID QUANTITY
-------------- ---------- ----------
         2368.      3129.        72.
         2368.      3143.        75.
         2368.      3155.        75.
         2368.      3106.       150.
         2396.      3106.       150.
         2419.      3106.       150.
         2368.      3110.        60.
         2368.      3117.        62.
         2368.      3123.        70.
         2368.      3127.        70.

  10 レコードが選択されました。
```

注意点 複数の列の組み合わせと一致する行を求める場合には、複数列を括弧で囲んで比較します。実行例の場合、比較する値は、次の組み合わせと一致しなければなりません。

（3129,72）（3129,72 ）（3143,75）（3155,75）（3106,150）（3106,150）（3106,150）
(3110,60)（3117,62）(3123,70）(3127,70)

4.13 複数の副問い合わせを使用した問い合わせ

注文番号2368と同じ商品、注文番号2368と同じ数量の注文を探します。比較する列は複数あるが、それぞれの列がそれぞれの副問い合わせと一致するものがあればよいという行を検索します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名
WHERE 列名1 比較演算子(SELECT 列名1 FROM 表名 WHERE 検索条件)
AND 列名2 比較演算子(SELECT 列名2 FROM 表名 WHERE 検索条件)
```

実行例

```
SQL> SELECT ORDER_ID,PRODUCT_ID,QUANTITY FROM ORDER_ITEMS
  2  WHERE PRODUCT_ID IN (SELECT PRODUCT_ID FROM ORDER_ITEMS WHERE ORDER_
➥ID = 2368)
  3  AND   QUANTITY   IN (SELECT QUANTITY   FROM ORDER_ITEMS WHERE ORDER_
➥ID = 2368);

ORDER_ID PRODUCT_ID QUANTITY
-------- ---------- --------
    2368       3129       72
    2368       3143       75
    2368       3155       75
    2413       3155       62
    2419       3155       72
    2368       3106      150
    2396       3106      150
    2419       3106      150
    2368       3110       60
    2368       3117       62
    2412       3127       72
    2368       3123       70
    2368       3127       70

13行が選択されました。
```

DB2の場合

書式

```
SELECT列名 FROM 表名
WHERE 列名1 比較演算子(SELECT 列名1 FROM 表名WHERE 検索条件)
AND 列名2 比較演算子(SELECT 列名2 FROM 表名WHERE 検索条件)
```

実行例

```
db2=> SELECT ORDER_ID,PRODUCT_ID,QUANTITY FROM ORDER_ITEMS WHERE PRODUCT_
➥ID IN (SELECT PRODUCT_ID FROM ORDER_ITEMS WHERE ORDER_ID = 2368) AND   QU
➥ANTITY   IN (SELECT QUANTITY   FROM ORDER_ITEMS WHERE ORDER_ID = 2368)

ORDER_ID       PRODUCT_ID QUANTITY
-------------- ---------- ----------
         2368.      3129.        72.
         2368.      3143.        75.
         2368.      3155.        75.
         2413.      3155.        62.
         2419.      3155.        72.
         2368.      3106.       150.
         2396.      3106.       150.
         2419.      3106.       150.
         2368.      3110.        60.
         2368.      3117.        62.
```

```
2412.       3127.         72.
2368.       3123.         70.
2368.       3127.         70.
```

13 レコードが選択されました。

注意点 WHERE句で指定する列それぞれを副問い合わせの値と比較することができます。上の実行例の場合、PRODUCT_ID列の値は、(3129,3143,3155,3106,3110,3117,3123,3127)のいずれかに一致する必要があります。また、QUANTITY列の値は、(72,75,150,60,62,70)と一致する必要があります。したがって、求める行は、(3129,3143,3155,3106,3110,3117,3123,3127)と(72,75,150,60,62,70)の組み合わせと一致すればよいため、複数列を指定した副問い合わせとは結果が異なります。

4.14 相関副問い合わせ

自分の所属する部門の平均給与以上の人を探します。まず、自分の所属する部門の平均給与を求める必要があるため、自分の所属する部門番号を副問い合わせの検索条件として渡す必要があります。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1 WHERE 列名 比較演算子
(SELECT 列名 FROM 表名2 WHERE 表名2.列名 = 表名1.列名)
```

実行例

```
SQL> SELECT E1.ENAME,E1.SAL FROM EMP E1 WHERE E1.SAL >=(SELECT AVG(SAL) FR
➥OM EMP E2 WHERE E2.DEPTNO = E1.DEPTNO);

ENAME           SAL
---------- --------
ALLEN          1600
JONES          2975
BLAKE          2850
SCOTT          3000
KING           5000
FORD           3000

6行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1 WHERE 列名 比較演算子
(SELECT 列名 FROM 表名2 WHERE 表名2.列名 = 表名1.列名)
```

実行例

```
db2=> SELECT E1.ENAME,E1.SAL FROM EMP E1 WHERE E1.SAL >=(SELECT AVG(SAL)
➥FROM EMP E2 WHERE E2.DEPTNO = E1.DEPTNO)

ENAME      SAL
---------- ---------
ALLEN        1600.00
JONES        2975.00
BLAKE        2850.00
SCOTT        3000.00
KING         5000.00
FORD         3000.00

  6 レコードが選択されました。
```

注意点 主問い合わせの行の値を副問い合わせの検索条件として渡す処理を相関副問い合わせと呼びます。主問い合わせの表名に別名を付け、副問い合わせで条件値として指定することが可能になり、主問い合わせの行ごとに副問い合わせの検索条件を評価します。

4.15 SET句（UPDATE文）に使用する問い合わせ

給与を社長の給与の9割に更新する。まず社長の給与を求める必要があるため、SET句に副問い合わせを使用します。

Oracleの場合

書式

```
UPDATE 表名1
SET 列名 = (SELECT 列名 FROM 表名2)
```

実行例

```
SQL> UPDATE EMP SET SAL = (SELECT SAL*0.9 FROM EMP WHERE JOB = 'PRESIDENT
➥');

15行が更新されました。
```

DB2の場合

書式

```
UPDATE 表名1
SET 列名 = (SELECT 列名 FROM 表名2)
```

実行例

```
db2=> UPDATE EMP SET SAL = (SELECT SAL*0.9 FROM EMP WHERE JOB = 'PRESIDEN
➥T')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 副問い合わせは、WHERE句だけではなくSET句でも使用できます。「更新後の値を求める処理」と「更新する」という2つの処理を1つのSQL文で行うことができます。

4.16 相関副問い合わせを使用したUPDATE

給与を自分が所属する部門の平均給与に更新します。まず、自分の所属する部門の平均給与を求める必要があるため、自分の所属する部門番号を副問い合わせの検索条件として渡す必要があります。

Oracleの場合

書式

```
UPDATE 表名1
SET 列名 =  (SELECT 列名 FROM 表名2 WHERE 表名2.列名 = 表名1.列名)
```

実行例

```
SQL> UPDATE EMP E1 SET E1.SAL = (SELECT AVG(E2.SAL) FROM EMP E2 WHERE E2.
➥DEPTNO = E1.DEPTNO);

15行が更新されました。
```

DB2の場合

書式

```
UPDATE 表名1
SET 列名 =  (SELECT 列名 FROM 表名2 WHERE 表名2.列名 = 表名1.列名)
```

実行例

```
db2=> UPDATE EMP E1 SET E1.SAL = (SELECT AVG(E2.SAL) FROM EMP E2 WHERE E2.
➥DEPTNO = E1.DEPTNO)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

副問い合わせは、WHERE句だけではなくSET句でも使用できます。また、主問い合わせの行の値を副問い合わせの検索条件として渡す処理を相関副問い合わせと呼びます。主問い合わせの表名に別名を付け、副問い合わせで条件値として指定することが可能になり、主問い合わせの行ごとに副問い合わせの検索条件を評価します。上の例では、各社員が所属する部門の平均給与は異なりますが、相関副問い合わせを使用することにより、主問い合わせで更新する行の部門番号が副問い合わせに渡されるため、それぞれの行に一致した部門の平均給与を求めることができます。

4.17 FROM句に使用する問い合わせ

部門ごとの集計結果と部門名を結合して表示します。

Oracleの場合

書式 **SELECT 列名 FROM (SELECT 列名 FROM 表名)**

実行例

```
SQL> SELECT D.DEPTNO,DNAME,SUM_SAL,MAX_SAL
  2  FROM   DEPT D,(SELECT DEPTNO,SUM(SAL) AS SUM_SAL,MAX(SAL) AS MAX_SAL
➥FROM EMP GROUP BY DEPTNO) SUMMARY
  3  WHERE  D.DEPTNO = SUMMARY.DEPTNO;

 DEPTNO DNAME          SUM_SAL MAX_SAL
------- -------------- ------- -------
     10 ACCOUNTING        8750    5000
     20 RESEARCH         10875    3000
     30 SALES             9400    2850

3行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM (SELECT 列名 FROM 表名) [AS] 表別名**

実行例

```
db2=> SELECT D.DEPTNO,DNAME,SUM_SAL,MAX_SAL FROM   DEPT D,(SELECT DEPTNO,
➥SUM(SAL) AS SUM_SAL,MAX(SAL) AS MAX_SAL FROM EMP GROUP BY DEPTNO) SUMMARY
➥WHERE  D.DEPTNO = SUMMARY.DEPTNO

DEPTNO DNAME          SUM_SAL                              MAX_SAL
------ -------------- ---------------------------------- ---------
   10. ACCOUNTING                                 8750.00   5000.00
   20. RESEARCH                                  10875.00   3000.00
   30. SALES                                      9400.00   2850.00

  3 レコードが選択されました。
```

注意点 副問い合わせは、WHERE句だけではなくFROM句でも使用できます。表をグループ化した結果と結合をしたい場合など、グループ化する問い合わせをFROM句副問い合わせにして扱うと便利です。
DB2 UDBでは、FROM句における副問い合わせには必ず表別名を付けます。

4.18 ソート済み結果への連番表示

給与の小さい順に表示番号を付けて検索します。

Oracleの場合

書式

```
SELECT ROWNUM,列名 FROM (SELECT 列名 FROM 表名 ORDER BY 列名)
```

実行例

```
SQL> SELECT ROWNUM,EMPNO,SAL,ENAME FROM (SELECT EMPNO,SAL,ENAME FROM EMP
➥ORDER BY SAL);

 ROWNUM    EMPNO      SAL ENAME
------- -------- -------- ----------
      1     7369      800 SMITH
      2     7900      950 JAMES
      3     7876     1100 ADAMS
      4     7521     1250 WARD
      5     7654     1250 MARTIN
      6     7934     1300 MILLER
      7     7950     1300 MARY
      8     7844     1500 TURNER
      9     7499     1600 ALLEN
     10     7782     2450 CLARK
     11     7698     2850 BLAKE
     12     7566     2975 JONES
     13     7788     3000 SCOTT
     14     7902     3000 FORD
     15     7839     5000 KING

15行が選択されました。
```

DB2の場合

書式

```
SELECT ROW_NUMBER OVER(),列名
FROM (SELECT 列名 FROM 表名 ORDER BY 列名)
```

実行例

```
db2=> SELECT row_number() over(),EMPNO,SAL,ENAME FROM (SELECT EMPNO,SAL,EN
➥AME FROM EMP ORDER BY SAL) TBL1

1                    EMPNO  SAL       ENAME
-------------------- ------ --------- ----------
                   1  7369.    800.00 SMITH
                   2  7900.    950.00 JAMES
                   3  7876.   1100.00 ADAMS
                   4  7521.   1250.00 WARD
                   5  7654.   1250.00 MARTIN
                   6  7934.   1300.00 MILLER
                   7  7950.   1300.00 MARY
                   8  7844.   1500.00 TURNER
                   9  7499.   1600.00 ALLEN
                  10  7782.   2450.00 CLARK
                  11  7698.   2850.00 BLAKE
                  12  7566.   2975.00 JONES
                  13  7788.   3000.00 SCOTT
                  14  7902.   3000.00 FORD
                  15  7839.   5000.00 KING
```

15 レコードが選択されました。

注意点 Oracleでは、行番号を表すROWNUM擬似列を使用します。
DB2 UDBでは、ROW_NUMBER関数を使用します。いずれも先に、給与の小さい順に行を並べ替えておく必要があるため、FROM句副問い合わせを使用します。なお、DB2 UDBではFROM句副問い合わせに表別名を定義する必要があります。

4.19 上位５件までのデータを表示する

給与額の多い人から5人表示します。上位（または下位）からn件を求める検索です。

Oracleの場合

書式

```
SELECT 列名 FROM (SELECT 列名 FROM 表名 ORDER BY 列名)
WHERE ROWNUM <= n
```

実行例

```
SQL> SELECT EMPNO,SAL,ENAME FROM (SELECT EMPNO,SAL,ENAME FROM EMP ORDER BY
➥SAL DESC) WHERE ROWNUM <= 5;

  EMPNO      SAL ENAME
------- -------- ----------
   7839     5000 KING
   7788     3000 SCOTT
   7902     3000 FORD
   7566     2975 JONES
   7698     2850 BLAKE

5行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名 ORDER BY 列名 FETCH FIRST n ROWS ONLY
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT EMPNO,SAL,ENAME FROM EMP ORDER BY SAL DESC
FETCH FIRST 5 ROWS ONLY;
------------------------------------------------------------------------

EMPNO  SAL        ENAME
------ ---------- -----
  7839    5000.00 KING
  7788    3000.00 SCOTT
  7902    3000.00 FORD
  7566    2975.00 JONES
  7698    2850.00 BLAKE

  5 レコードが選択されました。
```

注意点

n件だけ取り出すために、Oracleでは行番号を表すROWNUM擬似列を使用します。
DB2 UDBでは、「FETCH FIRST n ROWS ONLY」を使用します。
いずれも先に、給与の多い順に行を並べ替えておく必要があるため、OracleではORDER BY句を含んだFROM句副問い合わせを使用します。
なお、DB2 UDBではFROM句副問い合わせにする必要はありません。

5

表の結合

5.1 共通な列を持つ表の結合（列名、データ型が一致）

結合する表の互いの結合列が、同じ列名でなおかつ同じデータ型の場合の検索です。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 NATURAL JOIN 表名2**

実行例

```
SQL> SELECT DEPTNO,DNAME,ENAME
  2  FROM DEPT
  3  NATURAL JOIN EMP;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     20 RESEARCH       SMITH
     30 SALES          ALLEN
     30 SALES          WARD
     20 RESEARCH       JONES
     30 SALES          MARTIN
     30 SALES          BLAKE
     10 ACCOUNTING     CLARK
     20 RESEARCH       SCOTT
     10 ACCOUNTING     KING
     30 SALES          TURNER
     20 RESEARCH       ADAMS
     30 SALES          JAMES
     20 RESEARCH       FORD
     10 ACCOUNTING     MILLER

14行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 INNER JOIN 表名2 ON 表名1.列名 = 表名2.列名**

実行例

```
db2=> SELECT DEPT.DEPTNO,DNAME,ENAME FROM DEPT INNER JOIN EMP ON DEPT.DEP
➥TNO = EMP.DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     CLARK
   10. ACCOUNTING     MILLER
   10. ACCOUNTING     KING
   20. RESEARCH       SMITH
   20. RESEARCH       FORD
   20. RESEARCH       ADAMS
   20. RESEARCH       SCOTT
   20. RESEARCH       JONES
   30. SALES          ALLEN
   30. SALES          JAMES
   30. SALES          TURNER
   30. SALES          BLAKE
   30. SALES          MARTIN
   30. SALES          WARD

  14 レコードが選択されました。
```

注意点

Oracleは、結合列が同じ列名でなおかつ同じデータ型の場合は、NATURAL JOIN句を使用できます。NATURAL JOIN句を使用した場合、結合条件を指定する必要はありません。
DB2 UDBは、NATURAL JOIN句は使用しません。INNER JOIN句を使用し、結合条件はON句に指定します。

5.2 共通な列を持つ表の結合（列名のみ一致）

結合する表の互いの結合列が同じ列名だけれども、データ型は異なる場合の検索です。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 JOIN 表名2 USING(列名)**

実行例

```
SQL> SELECT DEPTNO,DNAME,ENAME
  2  FROM DEPT
  3  JOIN EMP
  4  USING(DEPTNO);

 DEPTNO DNAME          ENAME
------- -------------- ----------
     20 RESEARCH       SMITH
     30 SALES          ALLEN
     30 SALES          WARD
     20 RESEARCH       JONES
     30 SALES          MARTIN
     30 SALES          BLAKE
     10 ACCOUNTING     CLARK
     20 RESEARCH       SCOTT
     10 ACCOUNTING     KING
     30 SALES          TURNER
     20 RESEARCH       ADAMS
     30 SALES          JAMES
     20 RESEARCH       FORD
     10 ACCOUNTING     MILLER

14行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 INNER JOIN 表名2 ON 表名1.列名 = 表名2.列名**

実行例

```
db2=> SELECT DEPT.DEPTNO,DNAME,ENAME FROM DEPT INNER JOIN EMP ON DEPT.DEP
➥TNO = EMP.DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     CLARK
   10. ACCOUNTING     MILLER
   10. ACCOUNTING     KING
   20. RESEARCH       SMITH
   20. RESEARCH       FORD
   20. RESEARCH       ADAMS
   20. RESEARCH       SCOTT
   20. RESEARCH       JONES
   30. SALES          ALLEN
   30. SALES          JAMES
   30. SALES          TURNER
   30. SALES          BLAKE
   30. SALES          MARTIN
   30. SALES          WARD

  14 レコードが選択されました。
```

注意点

Oracleは、結合列が同じ列名でデータ型は異なる場合はUSING句を使用できます。USING句を使用した場合、結合条件を指定する必要はありません。結合条件として使用する列名をUSING句に指定します。

DB2 UDBは、USING句を使用しません。INNER JOIN句を使用し、結合条件はON句に指定します。

5.3 一部の列を使用した表の結合

結合する表に同じ列名でなおかつ同じデータ型の列が複数あり、そのうちの一部の列のみが結合列である場合の検索です。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 JOIN 表名2 USING(列名)**

実行例

```
SQL> SELECT DEPTNO,DNAME,ENAME
  2  FROM DEPT
  3  JOIN EMP
  4  USING(DEPTNO);

 DEPTNO DNAME          ENAME
------- -------------- ----------
     20 RESEARCH       SMITH
     30 SALES          ALLEN
     30 SALES          WARD
     20 RESEARCH       JONES
     30 SALES          MARTIN
     30 SALES          BLAKE
     10 ACCOUNTING     CLARK
     20 RESEARCH       SCOTT
     10 ACCOUNTING     KING
     30 SALES          TURNER
     20 RESEARCH       ADAMS
     30 SALES          JAMES
     20 RESEARCH       FORD
     10 ACCOUNTING     MILLER

14行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 INNER JOIN 表名2 ON 表名1.列名 = 表名2.列名**

実行例

```
db2=> SELECT DEPT.DEPTNO,DNAME,ENAME FROM DEPT INNER JOIN EMP ON DEPT.DEP
➥TNO = EMP.DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     CLARK
   10. ACCOUNTING     MILLER
   10. ACCOUNTING     KING
   20. RESEARCH       SMITH
   20. RESEARCH       FORD
   20. RESEARCH       ADAMS
   20. RESEARCH       SCOTT
   20. RESEARCH       JONES
   30. SALES          ALLEN
   30. SALES          JAMES
   30. SALES          TURNER
   30. SALES          BLAKE
   30. SALES          MARTIN
   30. SALES          WARD
```

14 レコードが選択されました。

注意点

Oracleは、同じ列名かつ同じデータ型の列が存在し、そのうちの一部の列のみが結合列である場合は、USING句を使用できます。USING句を使用した場合、結合条件を指定する必要はありません。結合条件として使用する列名をUSING句に指定します。
DB2 UDBは、USING句を使用しません。INNER JOIN句を使用し、結合条件はON句に指定します。

5.4 等価条件以外での結合

給与が下限額と上限額の間にあれば、そのランクを求めます。互いの表を結合する条件が、=比較演算子以外の場合です。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1 JOIN 表名2
ON 表名1.列名 BETWEEN 表名2.列名 AND 表名2.列名
```

実行例

```
SQL> SELECT ENAME,SAL,GRADE,LOSAL,HISAL
  2  FROM EMP
  3  JOIN SALGRADE
  4  ON SAL BETWEEN LOSAL AND HISAL;

ENAME           SAL   GRADE   LOSAL   HISAL
---------- ------- ------- ------- -------
SMITH           800       1     700    1200
JAMES           950       1     700    1200
ADAMS          1100       1     700    1200
WARD           1250       2    1201    1400
MARTIN         1250       2    1201    1400
MILLER         1300       2    1201    1400
MARY           1300       2    1201    1400
TURNER         1500       3    1401    2000
ALLEN          1600       3    1401    2000
CLARK          2450       4    2001    3000
BLAKE          2850       4    2001    3000
JONES          2975       4    2001    3000
SCOTT          3000       4    2001    3000
FORD           3000       4    2001    3000
KING           5000       5    3001    9999

15行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1 [INNER] JOIN 表名2
ON 表名1.列名 BETWEEN 表名2.列名 AND 表名2.列名
```

実行例

```
db2=> SELECT ENAME,SAL,GRADE,LOSAL,HISAL FROM EMP JOIN SALGRADE ON SAL
➥BETWEEN LOSAL AND HISAL

ENAME      SAL        GRADE   LOSAL   HISAL
---------- ---------- ------- ------- -------
SMITH         800.00      1.    700.   1200.
ALLEN        1600.00      3.   1401.   2000.
WARD         1250.00      2.   1201.   1400.
JONES        2975.00      4.   2001.   3000.
MARTIN       1250.00      2.   1201.   1400.
BLAKE        2850.00      4.   2001.   3000.
CLARK        2450.00      4.   2001.   3000.
SCOTT        3000.00      4.   2001.   3000.
KING         5000.00      5.   3001.   9999.
TURNER       1500.00      3.   1401.   2000.
ADAMS        1100.00      1.    700.   1200.
JAMES         950.00      1.    700.   1200.
```

```
FORD          3000.00       4.    2001.    3000.
MILLER        1300.00       2.    1201.    1400.
MARY          1300.00       2.    1201.    1400.

15 レコードが選択されました。
```

注意点

Oracleでは、非等価（=演算子以外）条件を使用した結合は、ON句を使用します。自然結合ではないため、NATURAL JOIN句は使用しません。INNER JOIN句を使用します。
DB2 UDBも同様にINNER JOIN句を使用し、結合条件をON句に指定します。JOIN句でのデフォルトはINNERなので、INNERは省略可能です。

5.5 表の異なる行同士での結合

直属の上司を表す列は、同じ表の社員番号を参照しています。そこで、直属の上司名とその部下の名前を結合して求めます。同じ表の異なる行を結合する検索です。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1 表別名1 JOIN 表名2 表別名2
ON 表名1.列名 = 表名2.列名
```

実行例

```
SQL> SELECT M.EMPNO AS MGR_NO,M.ENAME AS MGR_NAME,E.EMPNO AS EMP_NO,E.ENAM
➥E AS EMP_NAME
  2  FROM EMP M JOIN EMP E
  3  ON M.EMPNO = E.MGR;

 MGR_NO MGR_NAME    EMP_NO EMP_NAME
------- ---------- ------- ----------
   7902 FORD          7369 SMITH
   7698 BLAKE         7499 ALLEN
   7698 BLAKE         7521 WARD
   7839 KING          7566 JONES
   7698 BLAKE         7654 MARTIN
   7839 KING          7698 BLAKE
   7839 KING          7782 CLARK
   7566 JONES         7788 SCOTT
   7698 BLAKE         7844 TURNER
   7788 SCOTT         7876 ADAMS
   7698 BLAKE         7900 JAMES
   7566 JONES         7902 FORD
   7782 CLARK         7934 MILLER

13行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1 表別名1 JOIN 表名2 表別名2
ON 表名1.列名 = 表名2.列名
```

実行例

```
db2=> SELECT M.EMPNO AS MGR_NO,M.ENAME AS MGR_NAME,E.EMPNO AS EMP_NO,E.ENA
➥ME AS EMP_NAME FROM EMP M JOIN EMP E ON M.EMPNO = E.MGR

MGR_NO MGR_NAME   EMP_NO EMP_NAME
------ ---------- ------ ----------
 7902. FORD        7369. SMITH
 7698. BLAKE       7499. ALLEN
 7698. BLAKE       7521. WARD
 7839. KING        7566. JONES
 7698. BLAKE       7654. MARTIN
 7839. KING        7698. BLAKE
 7839. KING        7782. CLARK
 7566. JONES       7788. SCOTT
 7698. BLAKE       7844. TURNER
 7788. SCOTT       7876. ADAMS
 7698. BLAKE       7900. JAMES
 7566. JONES       7902. FORD
 7782. CLARK       7934. MILLER
```

13 レコードが選択されました。

注意点 同じ表の異なる行を結合することを自己結合と呼びます。結合するためには複数の表が必要です。そこで、表に別名を定義し、上司データの入った社員表（表別名： M）と部下データの入った社員表（表別名： E）が存在するかのごとく扱います。OracleもDB2 UDBもJOIN句とON句を使用します。

5.6 3つ以上の表の結合

3つ以上の表を結合します。

Oracleの場合

書式

SELECT 列名 FROM 表名1 JOIN 表名2 ON 表名1と表名2の結合条件
JOIN 表名3 ON 表名2と表名3の結合条件

実行例

```
SQL> SELECT D.DEPTNO,D.DNAME,E.ENAME,S.GRADE
  2  FROM DEPT D
  3  JOIN EMP E
  4  ON D.DEPTNO = E.DEPTNO
  5  JOIN SALGRADE S
  6  ON SAL BETWEEN LOSAL AND HISAL;

 DEPTNO DNAME          ENAME         GRADE
------- -------------- ---------- -------
     10 ACCOUNTING     KING              5
     10 ACCOUNTING     CLARK             4
     10 ACCOUNTING     MILLER            2
     20 RESEARCH       FORD              4
     20 RESEARCH       SCOTT             4
     20 RESEARCH       JONES             4
     20 RESEARCH       ADAMS             1
     20 RESEARCH       SMITH             1
     30 SALES          BLAKE             4
     30 SALES          ALLEN             3
     30 SALES          TURNER            3
     30 SALES          MARTIN            2
     30 SALES          WARD              2
     30 SALES          JAMES             1

14行が選択されました。
```

DB2の場合

書式

SELECT 列名 FROM 表名1 JOIN 表名2 ON 表名1と表名2の結合条件
JOIN 表名3 ON 表名2と表名3の結合条件

実行例

```
db2=> SELECT D.DEPTNO,D.DNAME,E.ENAME,S.GRADE FROM DEPT D JOIN EMP E ON
➥D.DEPTNO = E.DEPTNO JOIN SALGRADE S ON SAL BETWEEN LOSAL AND HISAL

DEPTNO DNAME          ENAME      GRADE
------ -------------- ---------- -------
   20. RESEARCH       SMITH           1.
   30. SALES          ALLEN           3.
   30. SALES          WARD            2.
   20. RESEARCH       JONES           4.
   30. SALES          MARTIN          2.
   30. SALES          BLAKE           4.
   10. ACCOUNTING     CLARK           4.
   20. RESEARCH       SCOTT           4.
   10. ACCOUNTING     KING            5.
   30. SALES          TURNER          3.
   20. RESEARCH       ADAMS           1.
   30. SALES          JAMES           1.
```

```
20. RESEARCH        FORD            4.
10. ACCOUNTING      MILLER          2.

14 レコードが選択されました。
```

注意点

「JOIN 表名 ON 結合条件」を繰り返すことにより、3つ以上の表の結合を行うことができます。
Oracleは、NATURAL JOIN句やUSING句を使った結合が含まれていてもかまいません。
DB2 UDBは、[INNER] JOIN、LEFT [OUTER] JOIN、RIGHT [OUTER] JOIN、FULL [OUTER] JOIN句を使用した結合が含まれていてもかまいません。

5.7 一致する行に加えてJOIN句の左側の表にしかない行（一致しない行）も表示する外部結合

配属先が決まっている社員に加えて、誰も配属していない部署であるOPERATIONSの行も表示します。結合条件に一致しない行も表示します。FROM句に指定した表には存在し、結合する相手となる行がJOIN句に指定した表に存在しない場合、FROM句のすべての行を表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
LEFT OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
```

実行例

```
SQL> SELECT D.DEPTNO,D.DNAME,E.ENAME
  2  FROM DEPT D
  3  LEFT OUTER JOIN EMP E
  4  ON D.DEPTNO = E.DEPTNO;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     20 RESEARCH       SMITH
     30 SALES          ALLEN
     30 SALES          WARD
     20 RESEARCH       JONES
     30 SALES          MARTIN
     30 SALES          BLAKE
     10 ACCOUNTING     CLARK
     20 RESEARCH       SCOTT
     10 ACCOUNTING     KING
     30 SALES          TURNER
     20 RESEARCH       ADAMS
     30 SALES          JAMES
     20 RESEARCH       FORD
     10 ACCOUNTING     MILLER
     40 OPERATIONS

15行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
LEFT OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
```

実行例

```
db2=> SELECT D.DEPTNO,D.DNAME,E.ENAME FROM DEPT D LEFT OUTER JOIN EMP E
➥ON D.DEPTNO = E.DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     CLARK
   10. ACCOUNTING     MILLER
   10. ACCOUNTING     KING
   20. RESEARCH       SMITH
   20. RESEARCH       FORD
   20. RESEARCH       ADAMS
   20. RESEARCH       SCOTT
   20. RESEARCH       JONES
   30. SALES          ALLEN
   30. SALES          JAMES
```

```
30. SALES           TURNER
30. SALES           BLAKE
30. SALES           MARTIN
30. SALES           WARD
40. OPERATIONS      -
```

15 レコードが選択されました。

注意点 結合条件に一致しない行も表示する結合を外部結合（OUTER JOIN）と呼びます。FROM句に指定した表には存在するけれども、結合する相手となる行がJOIN句に指定した表に存在しない場合、FROM句のすべての行を表示します。
OracleもDB2 UDBもLEFT［OUTER］JOIN、RIGHT［OUTER］JOIN、FULL［OUTER］JOIN句を使用します。

5.8 一致する行に加えてJOIN句の右側の表にしない行（一致しない行）も表示する外部結合

配属先が決まっている社員に加えて、まだ配属先が決まっていない従業員MARYの行も表示します。JOIN句に指定した表には存在するが、結合する相手となる行がFROM句に指定した表に存在しない場合、JOIN句のすべての行を表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
RIGHT OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
```

実行例

```
SQL> SELECT D.DEPTNO,D.DNAME,E.ENAME
  2  FROM DEPT D
  3  RIGHT OUTER JOIN EMP E
  4  ON D.DEPTNO = E.DEPTNO;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     10 ACCOUNTING     MILLER
     10 ACCOUNTING     KING
     10 ACCOUNTING     CLARK
     20 RESEARCH       FORD
     20 RESEARCH       ADAMS
     20 RESEARCH       SCOTT
     20 RESEARCH       JONES
     20 RESEARCH       SMITH
     30 SALES          JAMES
     30 SALES          TURNER
     30 SALES          BLAKE
     30 SALES          MARTIN
     30 SALES          WARD
     30 SALES          ALLEN
                       MARY

15行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
RIGHT OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
```

実行例

```
db2=> SELECT D.DEPTNO,D.DNAME,E.ENAME FROM DEPT D RIGHT OUTER JOIN EMP E
➥ON D.DEPTNO = E.DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   20. RESEARCH       SMITH
   30. SALES          ALLEN
   30. SALES          WARD
   20. RESEARCH       JONES
   30. SALES          MARTIN
   30. SALES          BLAKE
   10. ACCOUNTING     CLARK
   20. RESEARCH       SCOTT
   10. ACCOUNTING     KING
   30. SALES          TURNER
```

```
20. RESEARCH        ADAMS
30. SALES           JAMES
20. RESEARCH        FORD
10. ACCOUNTING      MILLER
  - -               MARY

15 レコードが選択されました。
```

注意点

結合条件に一致しない行も表示する結合を外部結合（OUTER JOIN）と呼びます。JOIN句に指定した表には存在するけれども、結合する相手となる行がFROM句に指定した表に存在しない場合、JOIN句のすべての行を表示します。
OracleもDB2 UDBもRIGHT OUTER JOIN句を使用します。

5.9 一致する行に加えて互いに一致しない行を表示する外部結合

配属先が決まっている社員に加えて、誰も配属していない部署OPERATIONSとまだ配属先が決まっていない従業員MARYの行も表示します。FROM句にもJOIN句にも互いに結合する相手となる行が一方の表に存在しない場合、FROM句およびJOIN句のすべての行を表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
FULL OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
```

実行例

```
SQL> SELECT D.DEPTNO,D.DNAME,E.ENAME
  2  FROM DEPT D
  3  FULL OUTER JOIN EMP E
  4  ON D.DEPTNO = E.DEPTNO;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     20 RESEARCH       SMITH
     30 SALES          ALLEN
     30 SALES          WARD
     20 RESEARCH       JONES
     30 SALES          MARTIN
     30 SALES          BLAKE
     10 ACCOUNTING     CLARK
     20 RESEARCH       SCOTT
     10 ACCOUNTING     KING
     30 SALES          TURNER
     20 RESEARCH       ADAMS
     30 SALES          JAMES
     20 RESEARCH       FORD
     10 ACCOUNTING     MILLER
     40 OPERATIONS
                       MARY

16行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
FULL OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
```

実行例

```
db2=> SELECT D.DEPTNO,D.DNAME,E.ENAME FROM DEPT D FULL OUTER JOIN EMP E
➥ON D.DEPTNO = E.DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   20. RESEARCH       SMITH
   30. SALES          ALLEN
   30. SALES          WARD
   20. RESEARCH       JONES
   30. SALES          MARTIN
   30. SALES          BLAKE
   10. ACCOUNTING     CLARK
   20. RESEARCH       SCOTT
   10. ACCOUNTING     KING
```

```
30. SALES           TURNER
20. RESEARCH        ADAMS
30. SALES           JAMES
20. RESEARCH        FORD
10. ACCOUNTING      MILLER
  - -               MARY
40. OPERATIONS      -
```

16 レコードが選択されました。

注意点 結合条件に一致しない行も表示する結合を外部結合（OUTER JOIN）と呼びます。FROM句にもJOIN句にも互いに結合する相手となる行が一方の表に存在しない場合、FROM句およびJOIN句のすべての行を表示します。
OracleもDB2 UDBもFULL OUTER JOIN句を使用します。

5.10 一致しない行のみ表示する結合

誰も配属していない部署OPERATIONSの行だけを表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
LEFT OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
WHERE 表名2.列名 IS NULL
```

実行例

```
SQL> SELECT D.DEPTNO,D.DNAME,E.ENAME
  2  FROM DEPT D
  3  LEFT OUTER JOIN EMP E
  4  ON D.DEPTNO = E.DEPTNO
  5  WHERE E.EMPNO IS NULL;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     40 OPERATIONS

1行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
LEFT OUTER JOIN 表名2 ON 表名1.列名 = 表名2.列名
WHERE 表名2.列名 IS NULL
```

実行例

```
db2=> SELECT D.DEPTNO,D.DNAME,E.ENAME FROM DEPT D LEFT OUTER JOIN EMP E
➥ON D.DEPTNO = E.DEPTNO WHERE E.EMPNO IS NULL

DEPTNO DNAME          ENAME
------ -------------- ----------
   40. OPERATIONS     -

  1 レコードが選択されました。
```

注意点

結合条件に一致しない行も表示する結合を外部結合（OUTER JOIN）と呼びます。FROM句に指定した表には存在するけれども、結合する相手となる行がJOIN句に指定した表に存在しない場合、FROM句のすべての行を表示します。

OracleもDB2 UDBもLEFT OUTER JOIN句を使用します。このとき、一致する相手がいない行のJOIN句の表の列はNULLのため、「JOIN句の表.列名 IS NULL」と指定することで、一致しない行だけを表示できます。

表名2の列はないので、表示してもNULL以外は表示されません。したがって、次のようなSQL文と同等になります。

```
SELECT 列名 FROM 表名1
WHERE NOT EXISTS
(SELECT * FROM 表名2 WHERE 表名1.列名 = 表名2.列名)
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT D.DEPTNO, D.DNAME
  FROM DEPT D
 WHERE NOT EXISTS
```

```
       (SELECT * FROM EMP E
         WHERE D.DEPTNO = E.DEPTNO);
--------------------------------------------------------------------------------

DEPTNO DNAME
------ --------------
    40 OPERATIONS

  1 レコードが選択されました。
```

5.11 検索条件を付加した結合

部署番号10で職種がマネジャー（MANAGER）の行を部署表と結合します。結合条件のほかに検索条件を指定した結合です。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
JOIN 表名2 ON 表名1.列名 = 表名2.列名
WHERE 検索条件
```

実行例

```
SQL> SELECT DEPTNO,DNAME,ENAME
  2  FROM DEPT
  3  NATURAL JOIN EMP
  4  WHERE DEPTNO = 10 AND JOB = 'MANAGER';

 DEPTNO DNAME          ENAME
------- -------------- ----------
     10 ACCOUNTING     CLARK

1行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
[INNER] JOIN 表名2 ON 表名1.列名 = 表名2.列名
WHERE 検索条件
```

実行例

```
db2=> SELECT DEPT.DEPTNO,DNAME,ENAME FROM DEPT INNER JOIN EMP ON DEPT.DEPT
➥NO = EMP.DEPTNO WHERE DEPT.DEPTNO = 10 AND JOB = 'MANAGER'

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     CLARK

  1 レコードが選択されました。
```

注意点 ｜ 結合条件はON句で指定します。検索条件は、単一表の検索と同様にWHERE句で指定します。

5.12 ORDER BY句を付加した結合

部署表と社員表を結合し、部署番号の昇順に並べ替えて表示します。

Oracleの場合

書式

```
SELECT 列名 FROM 表名1
JOIN 表名2 ON 表名1.列名 = 表名2.列名
ORDER BY 列名
```

実行例

```
SQL> SELECT DEPTNO,DNAME,ENAMEi
  2  FROM DEPT
  3  NATURAL JOIN EMP
  4  ORDER BY DEPTNO;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     10 ACCOUNTING     CLARK
     10 ACCOUNTING     KING
     10 ACCOUNTING     MILLER
     20 RESEARCH       SMITH
     20 RESEARCH       ADAMS
     20 RESEARCH       FORD
     20 RESEARCH       SCOTT
     20 RESEARCH       JONES
     30 SALES          ALLEN
     30 SALES          BLAKE
     30 SALES          MARTIN
     30 SALES          JAMES
     30 SALES          TURNER
     30 SALES          WARD

14行が選択されました。
```

DB2の場合

書式

```
SELECT 列名 FROM 表名1
INNER JOIN 表名2 ON 表名1.列名 = 表名2.列名
ORDER BY 列名
```

実行例

```
db2=> SELECT DEPT.DEPTNO,DNAME,ENAME FROM DEPT INNER JOIN EMP ON DEPT.DEPT
➥NO = EMP.DEPTNO ORDER BY DEPTNO

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     CLARK
   10. ACCOUNTING     KING
   10. ACCOUNTING     MILLER
   20. RESEARCH       SMITH
   20. RESEARCH       JONES
   20. RESEARCH       SCOTT
   20. RESEARCH       ADAMS
   20. RESEARCH       FORD
   30. SALES          ALLEN
   30. SALES          WARD
   30. SALES          MARTIN
   30. SALES          BLAKE
```

```
 30. SALES           TURNER
 30. SALES           JAMES

14 レコードが選択されました。
```

注意点 | ORDER BY句は単一表の検索と同様に、SQL文の最後に1つだけ指定します。

5.13 複数の問い合わせから戻される行を表示する（重複行を排除）

部署表の部署番号と社員表の部署番号を合わせて表示します。同じ値を持つ行は排除して一意な値のみを表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 UNION SELECT 列名 FROM 表名2**

実行例

```
SQL> SELECT DEPTNO FROM DEPT
  2  UNION
  3  SELECT DEPTNO FROM EMP;

 DEPTNO
-------
     10
     20
     30
     40

4行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 UNION SELECT 列名 FROM 表名2**

実行例

```
db2=> SELECT DEPTNO FROM DEPT UNION SELECT DEPTNO FROM EMP

DEPTNO
------
   10.
   20.
   30.
   40.
     -

  5 レコードが選択されました。
```

注意点 最初の問い合わせ結果と次（2番目以降）の問い合わせ結果を合わせて同じ列で表示する場合、集合演算子のUNION（和集合）を使用します。このとき重複する値を持つ行は排除され、一意な値が昇順並べ替えられて表示されます。

5.14 複数の問い合わせから戻される行を表示する（重複行を含む）

部署表の部署番号と社員表の部署番号を合わせて表示します。同じ値を持つ行は排除せずに表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 UNION ALL SELECT 列名 FROM 表名2**

実行例

```
SQL> SELECT DEPTNO FROM DEPT
  2  UNION ALL
  3  SELECT DEPTNO FROM EMP;

 DEPTNO
-------
     10
     20
     30
     40
     20
     30
     30
     20
     30
     30
     10
     20
     10
     30
     20
     30
     20
     10

19行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 UNION ALL SELECT 列名 FROM 表名2**

実行例

```
db2=> SELECT DEPTNO FROM DEPT UNION ALL SELECT DEPTNO FROM EMP

DEPTNO
------
   20.
   30.
   30.
   20.
   30.
   30.
   10.
   20.
   10.
   30.
   20.
   30.
   20.
```

```
10.
  -
10.
20.
30.
40.
```

19 レコードが選択されました。

注意点 最初の問い合わせ結果と次（2番目以降）の問い合わせ結果を合わせて同じ列で表示する場合、集合演算子のUNION ALL（全体集合）を使用します。このとき、重複する値を持つ行は排除されません。

5.15 最初の問い合わせから戻される行のみを表示する

部署表には存在するが社員表には存在しない部署番号を表示します。最初の問い合わせからしか戻されない行を表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 MINUS SELECT 列名 FROM 表名2**

実行例

```
SQL> SELECT DEPTNO FROM DEPT
  2  MINUS
  3  SELECT DEPTNO FROM EMP;

 DEPTNO
-------
     40

1行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 EXCEPT SELECT 列名 FROM 表名2**

実行例

```
db2=> SELECT DEPTNO FROM DEPT EXCEPT SELECT DEPTNO FROM EMP

DEPTNO
------
   40.

  1 レコードが選択されました。
```

注意点

最初の問い合わせ結果には存在するけれども、次（2番目以降）の問い合わせ結果には存在しない行を表示する問い合わせです。行は昇順に並べ替えられて表示されます。
Oracleは、集合演算子のMINUS（差集合）を使用します。
DB2 UDBは、EXCEPT演算子を使用します。

5.16 複数の問い合わせから戻される共通の行のみを表示する

部署表にも社員表にも存在する部署番号を表示します。すべての問い合わせから戻る共通の行を表示します。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 INTERSECT SELECT 列名 FROM 表名2**

実行例

```
SQL> SELECT DEPTNO FROM DEPT
  2  INTERSECT
  3  SELECT DEPTNO FROM EMP;

 DEPTNO
-------
     10
     20
     30

7行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM 表名1 INTERSECT SELECT 列名 FROM 表名2**

実行例

```
db2=> SELECT DEPTNO FROM DEPT INTERSECT SELECT DEPTNO FROM EMP

DEPTNO
------
   10.
   20.
   30.

  3 レコードが選択されました。
```

注意点 最初の問い合わせ結果にも次（2番目以降）の問い合わせ結果にも存在する行を表示する場合、集合演算子のINTERSECT（積集合）を使用します。行は昇順に並べ替えられて表示されます。

5.17 2つの表を総当たりで結合する

2つ表のすべての行を結合します。

Oracleの場合

書式 **SELECT 列名 FROM 表名1 CROSS JOIN 表名2**

実行例

```
SQL> SELECT DEPT.DEPTNO,DNAME,ENAME
  2  FROM DEPT
  3  CROSS JOIN EMP;

 DEPTNO DNAME          ENAME
------- -------------- ----------
     10 ACCOUNTING     SMITH
     10 ACCOUNTING     ALLEN
     10 ACCOUNTING     WARD
     10 ACCOUNTING     JONES
     10 ACCOUNTING     MARTIN
     10 ACCOUNTING     BLAKE
     10 ACCOUNTING     CLARK
     10 ACCOUNTING     SCOTT
     10 ACCOUNTING     KING
     10 ACCOUNTING     TURNER
     10 ACCOUNTING     ADAMS
     10 ACCOUNTING     JAMES
     10 ACCOUNTING     FORD
     10 ACCOUNTING     MILLER
     10 ACCOUNTING     MARY
     20 RESEARCH       SMITH
     20 RESEARCH       ALLEN
     20 RESEARCH       WARD
     20 RESEARCH       JONES
     20 RESEARCH       MARTIN
     20 RESEARCH       BLAKE

 DEPTNO DNAME          ENAME
------- -------------- ----------
     20 RESEARCH       CLARK
     20 RESEARCH       SCOTT
     20 RESEARCH       KING
     20 RESEARCH       TURNER
     20 RESEARCH       ADAMS
     20 RESEARCH       JAMES
     20 RESEARCH       FORD
     20 RESEARCH       MILLER
     20 RESEARCH       MARY
     30 SALES          SMITH
     30 SALES          ALLEN
     30 SALES          WARD
     30 SALES          JONES
     30 SALES          MARTIN
     30 SALES          BLAKE
     30 SALES          CLARK
     30 SALES          SCOTT
     30 SALES          KING
     30 SALES          TURNER
```

```
     30 SALES          ADAMS
     30 SALES          JAMES

 DEPTNO DNAME          ENAME
------- -------------- ----------
     30 SALES          FORD
     30 SALES          MILLER
     30 SALES          MARY
     40 OPERATIONS     SMITH
     40 OPERATIONS     ALLEN
     40 OPERATIONS     WARD
     40 OPERATIONS     JONES
     40 OPERATIONS     MARTIN
     40 OPERATIONS     BLAKE
     40 OPERATIONS     CLARK
     40 OPERATIONS     SCOTT
     40 OPERATIONS     KING
     40 OPERATIONS     TURNER
     40 OPERATIONS     ADAMS
     40 OPERATIONS     JAMES
     40 OPERATIONS     FORD
     40 OPERATIONS     MILLER
     40 OPERATIONS     MARY

60行が選択されました。
```

DB2の場合

書式

SELECT 列名 FROM 表名1,表名2

または、

SELECT 列名 FROM 表名1 INNER JOIN 表名2 ON 0=0

実行例

```
db2=> SELECT DEPT.DEPTNO,DNAME,ENAME FROM DEPT,EMP

DEPTNO DNAME          ENAME
------ -------------- ----------
   10. ACCOUNTING     SMITH
   20. RESEARCH       SMITH
   30. SALES          SMITH
   40. OPERATIONS     SMITH
   10. ACCOUNTING     ALLEN
   20. RESEARCH       ALLEN
   30. SALES          ALLEN
   40. OPERATIONS     ALLEN
   10. ACCOUNTING     WARD
   20. RESEARCH       WARD
   30. SALES          WARD
   40. OPERATIONS     WARD
   10. ACCOUNTING     JONES
   20. RESEARCH       JONES
   30. SALES          JONES
   40. OPERATIONS     JONES
   10. ACCOUNTING     MARTIN
   20. RESEARCH       MARTIN
   30. SALES          MARTIN
   40. OPERATIONS     MARTIN
   10. ACCOUNTING     BLAKE
   20. RESEARCH       BLAKE
   30. SALES          BLAKE
   40. OPERATIONS     BLAKE
   10. ACCOUNTING     CLARK
```

```
20. RESEARCH        CLARK
30. SALES           CLARK
40. OPERATIONS      CLARK
10. ACCOUNTING      SCOTT
20. RESEARCH        SCOTT
30. SALES           SCOTT
40. OPERATIONS      SCOTT
10. ACCOUNTING      KING
20. RESEARCH        KING
30. SALES           KING
40. OPERATIONS      KING
10. ACCOUNTING      TURNER
20. RESEARCH        TURNER
30. SALES           TURNER
40. OPERATIONS      TURNER
10. ACCOUNTING      ADAMS
20. RESEARCH        ADAMS
30. SALES           ADAMS
40. OPERATIONS      ADAMS
10. ACCOUNTING      JAMES
20. RESEARCH        JAMES
30. SALES           JAMES
40. OPERATIONS      JAMES
10. ACCOUNTING      FORD
20. RESEARCH        FORD
30. SALES           FORD
40. OPERATIONS      FORD
10. ACCOUNTING      MILLER
20. RESEARCH        MILLER
30. SALES           MILLER
40. OPERATIONS      MILLER
10. ACCOUNTING      MARY
20. RESEARCH        MARY
30. SALES           MARY
40. OPERATIONS      MARY

60 レコードが選択されました。
```

注意点

2つの表のすべての行を結合させることを直積と呼びます。通常、この結果が意味を持つことはありません。

Oracleは、CORSS JOIN句または、DB2 UDBと同様FROM句に2つの表名を指定し、結合条件を記述しなければ直積を求めることができます。

DB2 UDBは、CROSS JOIN句はサポートしていませんが、ON句に常に真となる条件（たとえば「0=0」）を指定することでCROSS JOIN句と同等の処理を実現できます。

6

データベース管理

6.1 表一覧の取得

既存の表名を検索します。

Oracle の場合

書式

```
SELECT TABLE_NAME FROM USER_TABLES;
```

実行例

```
SQL> SELECT TABLE_NAME FROM USER_TABLES;

TABLE_NAME
------------------------------
DUMMY
SALGRADE
BONUS
DEPT
EMP
```

DB2 の場合

書式

```
SELECT TABNAME FROM SYSCAT.TABLES
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT SUBSTR(TABNAME,1,30) AS TABNAME
  FROM SYSCAT.TABLES
 WHERE TABSCHEMA = 'IS3';
------------------------------------------------------------------------

TABNAME
------------------------------
ANIMALS_PRIV_INDEXES
BASE_INDEX_SIMULATE
TABLE_VAR_DEFN

  3 レコードが選択されました。
```

注意点

Oracle では、USER_TABLES データディクショナリ・ビューのほかに「TAB」または「CAT」データディクショナリを検索して、既存表の一覧を検索することもできます。
DB2 では、DB2 LIST TABLES コマンドを使います。この場合接続ユーザーのスキーマ名（デフォルトは接続ユーザー名）の表一覧が表示されます。
全スキーマについては DB2 LIST TABLES FOR ALL コマンドを使います。
DB2 UDB では、SYSCAT.TABLES ビューを使って既存表の一覧を検索することもできます。

6.2 新規表の作成

新しく表を作成します。

Oracleの場合

書式 **CREATE TABLE 表名 (列名1 データ型(有効桁数),[,列名2 データ型(有効桁数),・・・])**

実行例

```
SQL> CREATE TABLE TEST1
  2  (COL1 NUMBER,
  3   COL2 VARCHAR2(10),
  4   COL3 DATE);

表が作成されました。
```

DB2の場合

書式 **CREATE TABLE 表名 (列名1 データ型(有効桁数),[,列名2 データ型(有効桁数),・・・])**

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE TABLE TEST1
(COL1 FLOAT
,COL2 VARCHAR(10)
,COL3 TIMESTAMP);
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

OracleもDB2 UDBも表を作成するSQLコマンドは同じですが、使用可能なデータ型や有効桁数は異なります。主なデータ型は次のとおりです。

	Oracle	DB2 UDB
数値（10進数）	NUMBER(p,s)	DECIMAL(p,s)
数値（固定小数点数）	NUMBER(p)	SMALLINT INTEGER BIGINT
数値（浮動小数点数）	NUMBER	FLOAT
固定長文字列	CHAR	CHAR
可変長文字列	VARCHAR2	VARCHAR
日付（日付、時刻）	DATE	TIMESTAMP
日付（小数秒含む）	TIMESTAMP	TIMESTAMP
日付	なし	DATE
時刻	なし	TIME

6.3 表作成時に文字列をデフォルト値として設定する

表作成時に文字列型の列にデフォルト値を設定します。デフォルト値は行挿入時に値の指定が省略された場合に適用されます。

Oracleの場合

書式

```
CREATE TABLE 表名
(列名 {CHAR | VARCHAR2}(有効桁数) DEFAULT 'デフォルト値')
```

実行例

```
SQL> CREATE TABLE TEST2
  2  (COL1 NUMBER,
  3   COL2 VARCHAR2(10) DEFAULT 'Hello',
  4   COL3 DATE);

表が作成されました。
```

DB2の場合

書式

```
CREATE TABLE 表名
(列名 {CHAR | VARCHAR}(有効桁数) DEFAULT 'デフォルト値')
```

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE TABLE TEST2
(COL1 FLOAT
,COL2 VARCHAR(10) DEFAULT 'Hello'
,COL3 TIMESTAMP);
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

6.4 表作成時に今日の日付をデフォルト値として設定する

表作成時に日付型の列にデフォルト値として今日の日付（挿入時の日付）を設定します。デフォルト値は行挿入時に値の指定が省略された場合に適用されます。

Oracleの場合

書式

```
CREATE TABLE 表名
(列名 DATE DEFAULT SYSDATE)
```

実行例

```
SQL> CREATE TABLE TEST3
  2  (COL1 NUMBER,
  3   COL2 VARCHAR2(10),
  4   COL3 DATE DEFAULT SYSDATE);

表が作成されました。
```

DB2の場合

書式

```
CREATE TABLE 表名
(列名 TIMESTAMP DEFAULT CURRENT_TIMESTAMP)
```

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE TABLE TEST3
(COL1 FLOAT
,COL2 VARCHAR(10)
,COL3 TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

6.5 ほかの表を元に新規表を作成する

ほかの表を元に新規表を作成します。表定義および行をコピーすることができます。

Oracle の場合

書式 **CREATE TABLE 表名 AS SELECT * FROM 表名**

実行例

```
SQL> CREATE TABLE EMP1
  2  AS SELECT * FROM EMP;

表が作成されました。

SQL> SELECT COUNT(*) FROM EMP1;

  COUNT(*)
----------
        15
```

DB2 の場合

書式 **対応するコマンドはありません。表定義と行コピーを分けて行います。**

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE TABLE EMP1 LIKE EMP;
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。

------------------------------ 入力コマンド ------------------------------
INSERT INTO EMP1
SELECT * FROM EMP;
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。

------------------------------ 入力コマンド ------------------------------
SELECT COUNT(*) FROM EMP1;
------------------------------------------------------------------------

1
-----------
         15

  1 レコードが選択されました。
```

注意点

DB2 UDB では既存表の IXF 形式の EXPORT ファイルを作成し、このファイルを IMPORT ユーティティの入力として使い、新規表の作成、および表定義と行のコピーを実行できます。

```
EXPORT TO ファイル名 OF IXF SELECT * FROM EMP;
IMPORT FROMファイル名 OF IXF CREATE INTO EMP1 in 表スペース index in 表スペース;
```

また、コントロール・センターから表をクリックし、メニューから [コピー] を選択するというように、GUI から行うこともできます。

6.6 ほかの表の特定列を元に新規表を作成する

ほかの表を元に新規表を作成します。すべての列ではなくSELECT句で指定した列だけを元に行をコピーすることができます。

Oracleの場合

書式 **CREATE TABLE 表名1 AS SELECT 列名1[,列名2, ...] FROM 表名2**

実行例

```
SQL> CREATE TABLE EMP2
  2  AS SELECT DEPTNO,EMPNO,ENAME FROM EMP;

表が作成されました。
```

DB2の場合

書式 **対応するコマンドはありませんが、「6.5 ほかの表を元に新規表を作成する」で紹介したようにEXPORT/IMPORTユーティリティを利用します。**

実行例

```
db2=> EXPORT TO ファイル名 OF IXF SELECT DEPTNO,EMPNO,ENAME FROM EMP

db2=> IMPORT FROM ファイル名 OF IXF CREATE INTO EMP2
```

6.7 ほかの表を元に新規表を作成する際に列名を変更する

ほかの表をもとに新規表を作成します。元の表の列名ではなく新規表を作成する際に列名を変更します。

Oracleの場合

書式 **CREATE TABLE 表名（列名）AS SELECT 列名 FROM 表名**

実行例

```
SQL> CREATE TABLE EMP3
  2  (DEPT_ID,EMP_ID,EMP_NAME)
  3  AS SELECT DEPTNO,EMPNO,ENAME FROM EMP;

表が作成されました。
```

DB2の場合

書式 **対応するコマンドはありませんが、「6.5 ほかの表を元に新規表を作成する」で紹介したようにEXPORT/IMPORTユーティリティを利用します。**

実行例

```
db2=> EXPORT TO ファイル名 OF IXF SELECT DEPTNO,EMPNO,ENAME FROM EMP

db2=> IMPORT FROM ファイル名 OF IXF CREATE INTO EMP3(DEPT_ID,EMP_ID,EMP_NAME)
```

6.8 ほかの表の特定行を元に新規表を作成する

ほかの表を元に新規表を作成します。SELECT句で指定した行だけをコピーできます。

Oracleの場合

書式 **CREATE TABLE 表名 AS SELECT 列名 FROM 表名 WHERE 検索条件**

実行例

```
SQL> CREATE TABLE EMP4
  2  AS SELECT * FROM EMP
  3  WHERE DEPTNO = 10;

表が作成されました。
```

DB2の場合

書式 **対応するコマンドはありません。表定義と行コピーを分けて行うので、行コピー時にコピーしたい行を指定します。**

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE TABLE EMP4 LIKE EMP;
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。

------------------------------ 入力コマンド ------------------------------
INSERT INTO EMP4
SELECT * FROM EMP
 WHERE DEPTNO = 10;
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 「6.5 ほかの表を元に新規表を作成する」で紹介したようにEXPORT/IMPORTユーティリティを使うこともできます。

```
db2=> EXPORT TO ファイル名 OF IXF SELECT * FROM EMP WHERE DEPTNO=10
db2=> IMPORT FROM ファイル名 OF IXF CREATE INTO EMP4
```

6.9 ほかの表の定義だけ複写して新規表を作成する

ほかの表を元に新規表を作成します。表定義のみをコピーし、行はコピーしません。

Oracleの場合

書式 **CREATE TABLE 表名1 AS SELECT 列名 FROM 表名2 WHERE 1=2**

実行例

```
SQL> CREATE TABLE EMP5
  2  AS SELECT * FROM EMP
  3  WHERE 1=2;

表が作成されました。

SQL> SELECT COUNT(*) FROM EMP5;

  COUNT(*)
----------
         0
```

DB2の場合

書式 **CREATE TABLE 表名1 LIKE 表名2**

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE TABLE EMP5 LIKE EMP;
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。

------------------------------ 入力コマンド ------------------------------
SELECT COUNT(*) FROM EMP5;
------------------------------------------------------------------------

1
-----------
          0

  1 レコードが選択されました。
```

注意点 | DB2 UDBでは、CREATE TABLE LIKE文を使います。

6.10 表を削除する

表を削除します。

Oracleの場合

書式 **DROP TABLE 表名**

実行例
```
SQL> DROP TABLE EMP1;

表が削除されました。
```

DB2の場合

書式 **DROP TABLE 表名**

実行例
```
------------------------------ 入力コマンド ------------------------------
DROP TABLE EMP1
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

6.11 表削除時に参照整合性制約を無効にする

表を削除します。ほかの表で定義されている参照整合性制約も一緒に削除します。

Oracle の場合

書式 **DROP TABLE 表名 CASCADE CONSTRAINTS**

実行例

```
SQL> DROP TABLE DEPT CASCADE CONSTRAINTS;

表が削除されました。
```

DB2 の場合

書式 **DROP TABLE 表名**

実行例

```
------------------------------ 入力コマンド ------------------------------
DROP TABLE DEPT;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDB では、「CASCADE CONSTRAINTS」は不要です。親表の削除と同時にほかの表から参照されている参照整合性制約も削除されます。

6.12 表名を変更する

表の名前を変更します。

Oracleの場合

書式 **RENAME 旧表名 TO 新表名**

実行例
```
SQL> RENAME SALGRADE TO GRADE;

表名が変更されました。
```

DB2の場合

書式 **RENAME [TABLE] 旧表名 TO 新表名**

実行例
```
------------------------------ 入力コマンド ------------------------------
RENAME TABLE SALGRADE TO GRADE;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDBでは、変更する表がビュー、マテリアライズ照会表、トリガー、SQL関数、SQLメソッド、チェック制約または参照制約で参照されている場合には名前を変更することができません。

6.13 ほかのスキーマの表名を変更する

ほかのスキーマの表の名前を変更します。

Oracle の場合

書式 **ALTER TABLE スキーマ.旧表名 RENAME TO 新表名**

実行例

```
SQL> ALTER TABLE SCOTT.SALGRADE RENAME TO GRADE;

表が変更されました。
```

DB2 の場合

書式 **RENAME TABLE スキーマ.旧表名 TO 新表名**

実行例

```
------------------------------ 入力コマンド ------------------------------
RENAME TABLE SCOTT.SALGRADE TO GRADE;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | DB2 UDB で RENAME を実行する場合、表の CONTROL 特権、スキーマへの ALTERIN 特権、または SYSADM か DBADM 権限が必要です。

6.14 列を追加する

表に列を追加します。

Oracleの場合

書式 **ALTER TABLE 表名 ADD 列名 データ型(有効桁数)**

実行例
```
SQL> ALTER TABLE EMP ADD EMAIL VARCHAR2(30);

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 ADD 列名 データ型(有効桁数)**

実行例
```
------------------------------ 入力コマンド ------------------------------
ALTER TABLE EMP ADD EMAIL VARCHAR(30);
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました
```

6.15 列を削除する

表の列を削除します。

Oracle の場合

書式 **ALTER TABLE 表名 DROP COLUMN 列名**

実行例

```
SQL> ALTER TABLE EMP DROP COLUMN EMAIL;

表が変更されました。
```

DB2 の場合

書式 **該当するコマンドはありませんが、コントロール・センターで列を削除できます。**

実行例

注意点 | DB2 UDB ではコントロール・センターから GUI で行います。あるいは、ALTOBJ ストアード・プロシージャーで行います。

6.16 列に未使用マークを付ける

列に未使用マークを付けます。DROP 文との違いは、データが物理的にデータベースに残っていることです。

Oracle の場合

書式 **ALTER TABLE 表名 SET UNUSED COLUMN 列名**

実行例

```
SQL> ALTER TABLE EMP SET UNUSED COLUMN EMAIL;

表が変更されました。
```

DB2 の場合

書式 対応する機能はありません。

実行例

注意点 Oracle では、SET UNUSED で使用できなくなった列は、対応する SET USED コマンドがないので再び使用できるようにはなりません。

6.17 列名を変更する

列名を変更します。

Oracleの場合

書式 **ALTER TABLE 表名 RENAME COLUMN 旧列名 TO 新列名**

実行例

```
SQL> ALTER TABLE EMP RENAME COLUMN COMM TO COMMISSION;
```

表が変更されました。

DB2の場合

書式 **対応するコマンドはありませんが、コントロール・センターで列名を変更できます。**

実行例

注意点 | DB2 UDBではコントロール・センターからGUIで行います。または、ALTOBJストアード・プロシージャーで行います。

6.18 列の長さを大きくする

列の有効桁数を大きくします。

Oracleの場合

書式 **ALTER TABLE 表名 MODIFY 列名 データ型(有効桁数)**

実行例

```
SQL> ALTER TABLE EMP MODIFY ENAME VARCHAR2(30);

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 ALTER 列名 SET DATA TYPE データ型(有効桁数)**

実行例

```
------------------------------ 入力コマンド ------------------------------
ALTER TABLE EMP ALTER ENAME SET DATA TYPE VARCHAR(30);
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

6.19 列の長さを小さくする

列の有効桁数を小さくします。

Oracle の場合

書式 **ALTER TABLE 表名 MODIFY 列名 データ型(有効桁数)**

実行例

```
SQL> ALTER TABLE EMP MODIFY ENAME VARCHAR2(10);

表が変更されました。
```

DB2 の場合

書式 **対応するコマンドはありませんが、コントロール・センターで列の有効桁数を小さくできます。**

実行例

注意点 DB2 UDB では、すでに値が格納されている列の有効桁数を小さくする場合、データが切り捨てられる可能性があるため、変更過程の途中で統計情報の変更などが(コントロール・センターから)求められる場合があります。DB2 UDB ではコ ALTOBJ ストアード・プロシージャーで行うことも可能です。

6.20 列のデータ型を変更する

列のデータ型を変更します。

Oracleの場合

書式 **ALTE TABLE 表名 MODIFY 列名 データ型(有効桁数)**

実行例

```
SQL> ALTER TABLE EMP MODIFY ENAME CHAR(30);

表が変更されました。
```

DB2の場合

書式 **該当するコマンドはありませんが、コントロール・センターで列のデータ型を小さくできます。**

実行例

注意点

Oracleにおけるデータ型の変更は、次の条件を満たしている必要があります。

- 表に1件も行が格納されていないか
- 該当列の全行がNULL
- データ型に互換性がある

DB2 UDBにおけるデータ型の変更は、次の条件を満たしている必要があります。

- データ型に互換性があるか
- 変換式が定義できる

6.21 表の列情報の取得

表を構成している列名およびデータ型の情報を検索します。

Oracle の場合

書式

```
SELECT COLUMN_NAME,DATA_TYPE FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = '表名'
```

実行例

```
SQL> SELECT COLUMN_NAME,DATA_TYPE FROM USER_TAB_COLUMNS
  2  WHERE  TABLE_NAME = 'EMP2';

COLUMN_NAME          DATA_TYPE
-------------------- --------------------
DEPTNO               NUMBER
EMPNO                NUMBER
ENAME                VARCHAR2
```

DB2 の場合

書式

```
SELECT CHAR(COLNAME,18) COLUMN_NAME, CHAR(TYPENAME,10) DATA_TYPE
FROM SYSCAT.COLUMNS WHERE TABNAME = '表名'
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CHAR(COLNAME,18) COLUMN_NAME, CHAR(TYPENAME,10) DATA_TYPE
  FROM SYSCAT.COLUMNS
 WHERE TABNAME = 'EMP2';
--------------------------------------------------------------------------

COLUMN_NAME        DATA_TYPE
------------------ ----------
EMPNO              SMALLINT
ENAME              VARCHAR
DEPTNO             SMALLINT

  3 レコードが選択されました。
```

注意点

DB2 UDB のシステム・カタログ表では、列が長い場合が多いので、CHAR 関数などを使って適当に短くすると見やすくなります。「6.24 表の定義を確認する」で説明している DESCRIBE コマンドを使うと簡単に列情報を取得できます。

6.22 表を移動する

表が格納されている場所を変更します。

Oracleの場合

書式 **ALTER TABLE 表名 MOVE TABLESPACE 表領域名**

実行例

```
SQL> ALTER TABLE EMP MOVE TABLESPACE USERS;

表が変更されました。
```

DB2の場合

書式 **該当する機能はありません。**

実行例

注意点 DB2 UDBでは、表をカーソルLOADでコピーし、元の表を削除し、コピーした表の名前を変更することで、格納されている場所を変更できます。

```
DECLARE カーソル名 CURSOR FOR SELECT * FROM EMP
LOAD FROM カーソル名 OF CURSOR INSERT INTO EMP2
```

6.23 表の再編成を行う

表の断片化を解消します。

Oracle の場合

書式 **ALTER TABLE 表名 MOVE TABLESPACE 表領域名**

実行例
```
SQL> ALTER TABLE EMP MOVE TABLESPACE USERS;

表が変更されました。
```

DB2 の場合

書式 **REORG TABLE 表名 オプション [オプション・・・]**

実行例
```
------------------------------ 入力コマンド ------------------------------
REORG TABLE EMP;
--------------------------------------------------------------------------
DB20000I  REORG コマンドが正常に終了しました。
```

6.24 表の定義を確認する

表の列名、必須か否かおよびデータ型と有効桁数を表示することができます。

Oracleの場合

書式 **DESC 表名**

実行例

```
SQL> DESC EMP2
 名前                  NULL?    型
 ----------------- -------- ------------
 DEPTNO                     NUMBER(2)
 EMPNO              NOT NULL NUMBER(4)
 ENAME                      VARCHAR2(10)
```

DB2の場合

書式 **DESCRIBE TABLE 表名**

実行例

```
------------------------------ 入力コマンド ------------------------------
DESCRIBE TABLE EMP2;
------------------------------------------------------------------------

                              タイプ・
列名                          スキーマ   タイプ名             長さ        位取り  NULL
----------------------- --------- ------------------ --------- ------ ------
EMPNO                         SYSIBM     SMALLINT                2      0 いいえ
ENAME                         SYSIBM     VARCHAR                30      0 はい
DEPTNO                        SYSIBM     SMALLINT                2      0 はい

  3 レコードが選択されました。
```

6.25 ほかのユーザーに表に対する問い合わせの権限を与える

ほかのユーザーに表に対する問い合わせの権限を与えます。

Oracle の場合

書式 **GRANT SELECT ON 表名 TO ユーザー名**

実行例

```
SQL> GRANT SELECT ON EMP TO USER2;

権限付与が成功しました。
```

DB2 の場合

書式 **GRANT SELECT ON 表名 TO ユーザー名**

実行例

```
------------------------------ 入力コマンド ------------------------------
GRANT SELECT ON EMP TO USER2;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

6.26 ほかのユーザーに表に対する挿入、削除、更新の権限を与える

ほかのユーザーに表に対して、挿入、削除、更新の権限を与えます。

Oracleの場合

書式 **GRANT INSERT,DELETE,UPDATE ON表名 TO ユーザー名**

実行例

```
SQL> GRANT INSERT,UPDATE,DELETE ON EMP TO USER2;

権限付与が成功しました。
```

DB2の場合

書式 **GRANT INSERT,DELETE,UPDATE ON表名 TO ユーザー名**

実行例

```
------------------------------ 入力コマンド ------------------------------
GRANT INSERT,UPDATE,DELETE ON EMP TO USER2;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

6.27 ほかのユーザーに表の特定の列に対する挿入の権限を与える

表の特定の列に対する挿入の権限をほかのユーザーに与えます。

Oracle の場合

書式 **GRANT INSERT(列名) ON 表名 TO ユーザー名**

実行例

```
SQL> GRANT INSERT(EMPNO,ENAME,HIREDATE,DEPTNO) ON EMP TO USER2;

権限付与が成功しました。
```

DB2 の場合

書式 **対応する機能はありませんが、ビューを使うことにより同様の機能を実現できます。**
CREATE VIEW ビュー名 AS SELECT 列名 FROM 表名
GRANT INSERT ON ビュー名 TO ユーザー名

実行例

```
db2=> CREATE VIEW EMP_VIEW AS SELECT EMPNO,ENAME,HIREDATE,DEPTNO FROM EMP
DB20000I  SQL コマンドが正常に終了しました。

db2=> GRANT INSERT ON EMP_VIEW TO USER2
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDB は、UPDATE と REFERENCES 権限は列を指定することができますが、INSERT に対しては列を指定することができません。

6.28 ほかのユーザーに表の特定の列に対する更新の権限を与える

表の特定の列に対する更新の権限をほかのユーザーに与えます。

Oracleの場合

書式 **GRANT UPDATE(列名) ON 表名 TO ユーザー名**

実行例

```
SQL> GRANT UPDATE(ENAME,JOB),DELETE ON EMP TO USER2;

権限付与が成功しました。
```

DB2の場合

書式 **GRANT UPDATE(列名) ON 表名 TO ユーザー名**

実行例

```
db2=> GRANT UPDATE(ENAME,JOB),DELETE ON EMP TO USER2
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDBは、UPDATEとREFERENCES権限は列を指定することができますが、INSERTに対しては列を指定することができません。

6.29 ほかのユーザーに表の特定の列に対する問い合わせの権限を与える

表の特定の列に対する問い合わせの権限をほかのユーザーに与えます。

Oracle の場合

書式

```
CREATE VIEW ビュー名 AS SELECT 列名 FROM 表名
GRANT SELECT ON ビュー名 TO ユーザー名
```

実行例

```
SQL> CREATE OR REPLACE VIEW EMP_VIEW AS SELECT DEPTNO,EMPNO,ENAME FROM EMP;

ビューが作成されました。

SQL> GRANT SELECT ON EMP_VIEW TO USER2;

権限付与が成功しました。
```

DB2 の場合

書式

```
CREATE VIEW ビュー名 AS SELECT 列名 FROM 表名
GRANT SELECT ON ビュー名 TO ユーザー名
```

実行例

```
db2=> CREATE VIEW EMP_VIEW AS SELECT DEPTNO,EMPNO,ENAME FROM EMP
DB20000I  SQL コマンドが正常に終了しました。

db2=> GRANT SELECT ON EMP_VIEW TO USER2
DB20000I  SQL コマンドが正常に終了しました。
```

6.30 ほかのユーザーに権限付与操作の許可を与える

表に対する操作権限を付与するだけでなく、権限をほかのユーザーに付与する操作も許可します。

Oracleの場合

書式 **GRANT 権限 ON 表名 TO ユーザー名 WITH GRANT OPTION**

実行例

```
SQL> GRANT ALL ON EMP TO USER2 WITH GRANT OPTION;

権限付与が成功しました。
```

DB2の場合

書式 **GRANT 権限 ON 表名 TO ユーザー名 WITH GRANT OPTION**

実行例

```
db2=> GRANT ALL ON EMP TO USER2 WITH GRANT OPTION
DB20000I  SQL コマンドが正常に終了しました。
```

6.31 与えたオブジェクト権限を取り消す

ほかのユーザーに与えたオブジェクト権限を取り消します。

Oracle の場合

書式 **REVOKE 権限名 ON 表名 FROM ユーザー名**

実行例

```
SQL> REVOKE ALL ON EMP FROM USER2;

取消しが成功しました。
```

DB2 の場合

書式 **REVOKE 権限名 ON 表名 FROM ユーザー名**

実行例

```
db2=> REVOKE ALL ON EMP FROM USER2
DB20000I  SQL コマンドが正常に終了しました。
```

6.32 与えられている権限を一覧する

自分に操作権限を与えられている表名を一覧表示します。

Oracleの場合

書式

```
SELECT TABLE_NAME FROM USER_TAB_PRIVS_RECD;
```

実行例

DB2の場合

書式

```
SELECT 列名 [,列名, 列名 ・・・] FROM SYSCAT.TABAUTH
WHERE GRANTEE = '名前';
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CHAR(GRANTOR,10) AS GRANTOR, CHAR(TABNAME,30) AS TABLE_NAME
       , CONTROLAUTH || ALTERAUTH ||INDEXAUTH AS CAI
       , SELECTAUTH||INSERTAUTH||DELETEAUTH||UPDATEAUTH AS SIDU
       , REFAUTH AS REF
  FROM SYSCAT.TABAUTH
 WHERE GRANTEE = 'DB2ADMIN';
------------------------------------------------------------------------------

GRANTOR    TABLE_NAME                     CAI SIDU REF
---------- ------------------------------ --- ---- ---
SYSIBM     VC_9                           YGG GGGG G
SYSIBM     ADDRESS                        YGG GGGG G
SYSIBM     CL_SCHED                       YGG GGGG G
SYSIBM     CODES                          YGG GGGG G
SYSIBM     CONTAINS_TYPEA_1               YGG GGGG G

(省略)

SYSIBM     ANIMALS_PRIV_INDEXES           YGG GGGG G
SYSIBM     BASE_INDEX_SIMULATE            YGG GGGG G
SYSIBM     TABLE_VAR_DEFN                 YGG GGGG G
SYSIBM     GRADE                          YGG GGGG G
SYSIBM     T_FILTER_RESULTS               YGG GGGG G

  33 レコードが選択されました。
```

6.33 スキーマにアクセスできるユーザーを一覧する

スキーマ（自分が所有するオブジェクト）にアクセス許可したユーザーの一覧を表示します。

Oracle の場合

書式

```
SELECT GRANTEE,TABLE_NAME,PRIVILEGE
FROM USER_TAB_PRIVS_MADE
```

実行例

```
SQL> SELECT GRANTEE,TABLE_NAME,PRIVILEGE FROM USER_TAB_PRIVS_MADE;

GRANTEE    TABLE_NAME           PRIVILEGE
---------- -------------------- --------------------
USER2      EMP_VIEW             DELETE
USER2      EMP_VIEW             INSERT
USER2      EMP_VIEW             SELECT
USER2      EMP_VIEW             UPDATE
USER2      EMP_VIEW             REFERENCES
USER2      EMP_VIEW             ON COMMIT REFRESH
USER2      EMP_VIEW             QUERY REWRITE
USER2      EMP_VIEW             DEBUG
USER2      EMP_VIEW             FLASHBACK

9行が選択されました。
```

DB2 の場合

書式

```
SELECT GRANTEE, 列名 [,列名, 列名 ･･･] FROM SYSCAT.TABAUTH
WHERE GRANTOR = '名前';
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CHAR(GRANTEE, 10) AS GRANTEE, CHAR(TABNAME,30) AS TABLE_NAME
     , CONTROLAUTH || ALTERAUTH || INDEXAUTH AS CAI
     , SELECTAUTH || INSERTAUTH || DELETEAUTH || UPDATEAUTH AS SIDU
     , REFAUTH AS REF
FROM SYSCAT.TABAUTH
WHERE GRANTOR = 'DB2ADMIN';
------------------------------------------------------------------------------

GRANTEE    TABLE_NAME                     CAI SIDU REF
---------- ------------------------------ --- ---- ---
USER2      EMP_VIEW                       NNN YYYY N

  1 レコードが選択されました。
```

6.34 表にコメントを定義する

表にコメントを定義します。

Oracleの場合

書式 **COMMENT ON TABLE 表名 IS 'コメント'**

実行例
```
SQL> COMMENT ON TABLE EMP IS 'Employee Information';

コメントが作成されました。
```

DB2の場合

書式 **COMMENT ON TABLE 表名 IS 'コメント'**

実行例
```
db2=> COMMENT ON TABLE EMP IS 'Employee Information'
DB20000I  SQL コマンドが正常に終了しました。
```

6.35 列にコメントを定義する

列にコメントを定義します。

Oracleの場合

書式 **COMMENT ON COLUMN 表名.列名 IS 'コメント'**

実行例
```
SQL> COMMENT ON COLUMN EMP.ENAME IS 'Employee Name';

コメントが作成されました。
```

DB2の場合

書式 **COMMENT ON COLUMN 表名.列名 IS 'コメント'**

実行例
```
db2=> COMMENT ON COLUMN EMP.ENAME IS 'Employee Name'
DB20000I  SQL コマンドが正常に終了しました。
```

6.36 表作成時に列制約を宣言する

表作成時に列制約を定義します。

Oracleの場合

書式

```
CREATE TABLE 表名
(列名 データ型 CONSTRAINT 制約名 {NOT NULL| PRIMARY KEY | UNIQUE
 | CHECK(評価条件) | REFERENCES 表名(列名)
 [, 列名 ・・・] )
```

実行例

```
SQL>CREATE TABLE EMP
  2  (EMPNO NUMBER(4)  CONSTRAINT PK_EMP_EMPNO  PRIMARY KEY,
  3   ENAME VARCHAR2(10) CONSTRAINT NN_EMP_ENAME  NOT NULL,
  4   JOB VARCHAR2(9),
  5   MGR NUMBER(4)    CONSTRAINT FK_EMP_MGR   REFERENCES EMP(EMPNO),
  6   HIREDATE DATE,
  7   SAL NUMBER(7, 2)         CONSTRAINT CK_EMP_SAL   CHECK(SAL >= 0),
  8   COMM NUMBER(7, 2),
  9   DEPTNO NUMBER(2)         CONSTRAINT FK_EMP_DEPTNO REFERENCES
DEPT(DEPTNO));

表が作成されました。
```

DB2の場合

書式

```
CREATE TABLE 表名
(列名 データ型 {NOT NUL} CONSTRAINT 制約名 {PRIMARY KEY | UNIQUE
 | CHECK(評価条件) | REFERENCES 表名(列名)
 [, 列名 ・・・] )
```

実行例

```
db2=> CREATE TABLE EMP (EMPNO  NUMERIC(4)  NOT NULL CONSTRAINT PK_EMP_EMPN
➥O  PRIMARY KEY, ENAME  VARCHAR(10)  CONSTRAINT NN_EMP_ENAME  NOT NULL, JO
➥B  VARCHAR(9), MGR  NUMERIC(4)  CONSTRAINT FK_EMP_MGR  REFERENCES EMP(EMP
➥NO), HIRETIMESTAMP TIMESTAMP, SAL  NUMERIC(7, 2) CONSTRAINT CK_EMP_SAL  C
➥HECK(SAL >= 0), COMM  NUMERIC(7, 2), DEPTNO  NUMERIC(2)  CONSTRAINT FK_EM
➥P_DEPTNO REFERENCES DEPT(DEPTNO))
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

DB2 UDBでは、NOT NULLは制約ではなく列の属性です。したがって、制約名を付ける必要はありません。また、主キーおよび一意制約を構成する列はあらかじめNOT NULL制約が設定されている必要があります。

6.37 表作成時に表制約を宣言する

表作成時に表制約を定義します。

Oracleの場合

書式

```
CREATE TABLE 表名 (列名 データ型 [ NOT NULL]
 [, 列名データ型 [ NOT NULL], ・・・],
 CONSTRAINT 制約名 { PRIMARY KEY(列名)| UNIQUE(列名)
 | CHECK(評価条件)| FOREIGN KEY(列名) REFERENCES 表名(列名) } )
```

実行例

```
SQL> CREATE TABLE EMP
  2  (EMPNO NUMBER(4),
  3   ENAME VARCHAR2(10) CONSTRAINT NN_EMP_ENAME  NOT NULL,
  4   JOB VARCHAR2(9),
  5   MGR NUMBER(4),
  6   HIREDATE DATE,
  7   SAL NUMBER(7, 2),
  8   COMM NUMBER(7, 2),
  9   DEPTNO NUMBER(2),
 10    CONSTRAINT PK_EMP_EMPNO       PRIMARY KEY(EMPNO),
 11    CONSTRAINT FK_EMP_MGR  FOREIGN KEY(MGR) REFERENCES EMP(EMPNO),
 12    CONSTRAINT CK_EMP_SAL  CHECK(SAL >= 0),
 13    CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REFERENCES
DEPT(DEPTNO));

表が作成されました。
```

DB2の場合

書式

```
CREATE TABLE 表名 (列名 データ型 [ NOT NULL]
 [, 列名データ型 [ NOT NULL], ・・・],
 CONSTRAINT 制約名 { PRIMARY KEY(列名)| UNIQUE(列名)
 | CHECK(評価条件)| FOREIGN KEY(列名) REFERENCES 表名(列名) } )
```

実行例

```
db2=> CREATE TABLE EMP (EMPNO NUMERIC(4) NOT NULL, ENAME VARCHAR(10) NOT N
➥ULL, JOB VARCHAR(9), MGR NUMERIC(4), HIRETIMESTAMP TIMESTAMP, SAL NUMERIC
➥(7, 2), COMM NUMERIC(7, 2), DEPTNO NUMERIC(2), CONSTRAINT PK_EMP_EMPNO PR
➥IMARY KEY(EMPNO), CONSTRAINT FK_EMP_MGR FOREIGN KEY(MGR) REFERENCES EMP (
➥EMPNO), CONSTRAINT CK_EMP_SAL  CHECK(SAL >= 0), CONSTRAINT FK_EMP_DEPTNO
➥FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO))
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

DB2 UDBでは、NOT NULLは制約ではなく列の属性です。したがって、制約名を付ける必要はありません。また、主キーおよび一意制約を構成する列はあらかじめNOT NULL制約が設定されている必要があります。

6.38 主キー制約を削除するときに参照整合性制約も削除する

主キー制約を削除するときに、従属表の参照整合性制約も一緒に削除します。

Oracleの場合

書式 **ALTER TABLE 表名 DROP [CONSTRAINT 制約名 | PRIMARY KEY] CASCADE**

実行例
```
SQL> ALTER TABLE EMP DROP PRIMARY KEY CASCADE;

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 DROP [CONSTRAINT 制約名 | PRIMARY KEY]**

実行例
```
db2=> ALTER TABLE EMP DROP PRIMARY KEY
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | DB2 UDBでは、CASCADEオプションを付けなくても参照整合性制約は削除されます。

6.39 主キー制約を追加する

主キー制約を追加します。

Oracle の場合

書式 **ALTER TABLE 表名 ADD CONSTRAINT 制約名 PRIMARY KEY(列名)**

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT PK_EMP_EMPNO PRIMARY KEY(EMPNO);

表が変更されました。
```

DB2 の場合

書式 **ALTER TABLE 表名 ADD CONSTRAINT 制約名 PRIMARY KEY(列名)**

実行例

```
db2=> ALTER TABLE EMP ADD CONSTRAINT PK_EMP_EMPNO PRIMARY KEY(EMPNO)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDB では、主キーを構成する列はあらかじめ NOT NULL 制約が設定されている必要があります。

6.40 主キー制約を削除する

主キー制約を削除します。

Oracleの場合

書式 **ALTER TABLE 表名 DROP [CONSTRAINT 制約名 | PRIMARY KEY]**

実行例
```
SQL> ALTER TABLE EMP DROP PRIMARY KEY;

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 DROP [CONSTRAINT 制約名 | PRIMARY KEY]**

実行例
```
db2=> ALTER TABLE EMP DROP PRIMARY KEY
DB20000I  SQL コマンドが正常に終了しました。
```

6.41 一意制約を追加する

一意制約を追加します。

Oracle の場合

書式 **ALTER TABLE 表名 ADD CONSTRAINT 制約名 UNIQUE(列名)**

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT U_EMP_ENAME UNIQUE(ENAME);

表が変更されました。
```

DB2 の場合

書式 **ALTER TABLE 表名 ADD CONSTRAINT 制約名 UNIQUE(列名)**

実行例

```
db2=> ALTER TABLE EMP ADD CONSTRAINT U_EMP_ENAME UNIQUE(ENAME)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDB では、一意制約を構成する列にはあらかじめ NOT NULL 制約が設定されている必要があります。

6.42 一意制約を削除する

一意制約を削除します。

Oracleの場合

書式 **ALTER TABLE 表名 DROP CONSTRAINT 制約名**

実行例
```
SQL> ALTER TABLE EMP DROP CONSTRAINT U_EMP_ENAME;

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 DROP CONSTRAINT 制約名**

実行例
```
db2=> ALTER TABLE EMP DROP CONSTRAINT U_EMP_ENAME
DB20000I  SQL コマンドが正常に終了しました。
```

6.43 CHECK制約を追加する

CHECK制約を追加します。

Oracleの場合

書式

```
ALTER TABLE 表名 ADD CONSTRAINT 制約名 CHECK(評価条件)
REFERENCES 表名(列名)
```

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT CK_EMP_SAL CHECK(SAL >=0);

表が変更されました。
```

DB2の場合

書式

```
ALTER TABLE 表名 ADD CONSTRAINT 制約名 CHECK(評価条件)
REFERENCES 表名(列名)
```

実行例

```
db2=> ALTER TABLE EMP ADD CONSTRAINT CK_EMP_SAL CHECK(SAL >=0)
DB20000I  SQL コマンドが正常に終了しました。
```

6.44 CHECK制約を削除する

CHECK制約を削除します。

Oracleの場合

書式 **ALTER TABLE 表名 DROP CONSTRAINT 制約名**

実行例
```
SQL> ALTER TABLE EMP DROP CONSTRAINT CK_EMP_SAL ;

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 DROP CONSTRAINT 制約名**

実行例
```
db2=> ALTER TABLE EMP DROP CONSTRAINT CK_EMP_SAL
DB20000I  SQL コマンドが正常に終了しました。
```

6.45 外部キーを追加する

参照整合性制約（外部キー）を追加します。

Oracle の場合

書式

```
ALTER TABLE 表名 ADD CONSTRAINT 制約名 FOREIGN KEY(列名)
REFERENCES 表名(列名)
```

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_MGR FOREIGN KEY(MGR) REFERENCES
➥EMP(EMPNO);

表が変更されました。
```

DB2 の場合

書式

```
ALTER TABLE 表名 ADD CONSTRAINT 制約名 FOREIGN KEY(列名)
REFERENCES 表名(列名)
```

実行例

```
db2=> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_MGR FOREIGN KEY(MGR) REFERENC
➥ES EMP(EMPNO)
DB20000I  SQL コマンドが正常に終了しました。
```

6.46 外部キーを削除する

参照整合性制約（外部キー）を削除します。

Oracleの場合

書式 **ALTER TABLE 表名 DROP CONSTRAINT 制約名**

実行例
```
SQL> ALTER TABLE EMP DROP CONSTRAINT FK_EMP_MGR;

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE 表名 DROP CONSTRAINT 制約名**

実行例
```
db2=> ALTER TABLE EMP DROP CONSTRAINT FK_EMP_MGR
DB20000I  SQL コマンドが正常に終了しました。
```

6.47 NOT NULLを削除する

NOT NULL制約を削除します。

Oracleの場合

書式 **ALTER TABLE 表名 MODIFY 列名 NULL**

実行例

```
SQL> ALTER TABLE EMP MODIFY ENAME NULL;

表が変更されました。
```

DB2の場合

書式 **対応するコマンドはありませんが、コントロール・センターでNOT NULLを削除できます。**

実行例

注意点 | DB2 UDBではコントロール・センターからGUIで行います。あるいは、ALTOBJストアード・プロシージャーで行います。

6.48 NOT NULLを追加する

NOT NULL制約を追加します。

Oracleの場合

書式 **ALTER TABLE 表名 MODIFY 列名 NOT NULL**

実行例

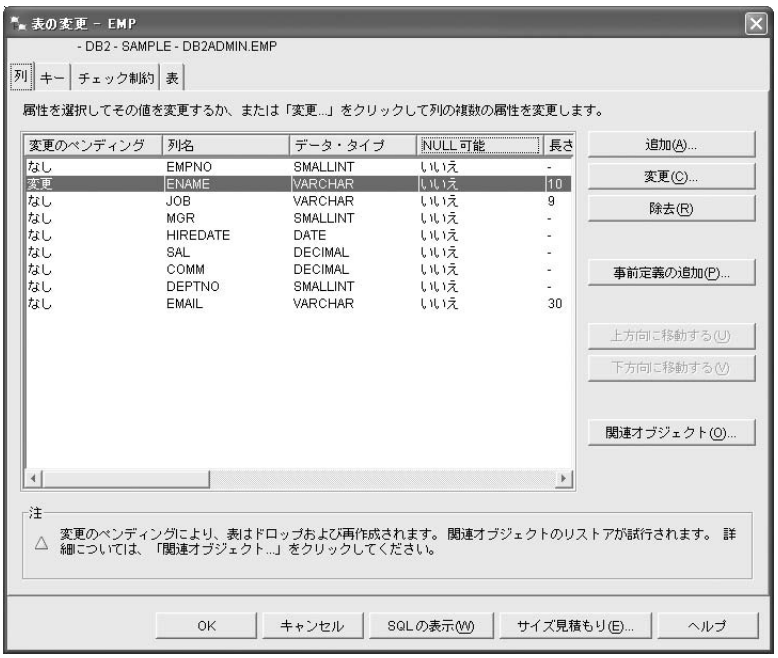
```
SQL> ALTER TABLE EMP MODIFY ENAME NOT NULL;

表が変更されました。
```

DB2の場合

書式 **対応するコマンドはありませんが、コントロール・センターでNOT NULL制約を追加できます。**

実行例



6.49 遅延制約を定義する

遅延制約（コミット時に制約を評価する）を設定する。

Oracle の場合

書式 **ALTER TABLE 表名 ADD CONSTRAINT 制約名 制約内容 DEFERRABLE INITIALLY DEFERRED**

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT CK_EMP_SAL CHECK(SAL >=0) DEFERRABLE
➥INITIALLY DEFERRED;

表が変更されました。
```

DB2 の場合

書式 **対応する機能はありません。**

実行例

注意点 | DB2 UDB では、ステートメントの最後に制約を評価します。

6.50 即時制約を定義する

即時制約（SQL実行直後に制約を評価する）を設定します。

Oracleの場合

書式 **ALTER TABLE 表名 ADD CONSTRAINT 制約名 制約内容 NOT DEFERRABLE**

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT CK_EMP_SAL CHECK(SAL >=0) NOT DEFERR
➥ABLE;

表が変更されました。
```

DB2の場合

書式 **対応する機能はありません。**

実行例

注意点 | DB2 UDBでは、ステートメントの最後に制約を評価するので即時制約として機能します。

6.51 制約を無効にする

制約を無効にします。

Oracle の場合

書式 **ALTER TABLE 表名 DISABLE CONSTRAINT 制約名**

実行例

```
SQL> ALTER TABLE EMP DISABLE CONSTRAINT CK_EMP_SAL;

表が変更されました。
```

DB2 の場合

書式 **ALTER TABLE表名 [CHECK | FOREIGN KEY] 制約名 NOT ENFORCED**

実行例

注意点 | どちらのデータベースでも、制約を削除し、再定義することができます。

6.52 制約を有効にする

制約を有効にします。

Oracleの場合

書式 **ALTER TABLE 表名 ENABLE CONSTRAINT 制約名**

実行例

```
SQL> ALTER TABLE EMP ENABLE CONSTRAINT CK_EMP_SAL;

表が変更されました。
```

DB2の場合

書式 **ALTER TABLE表名 [CHECK | FOREIGN KEY] 制約名 ENFORCED**

実行例

注意点 | どちらのデータベースでも、制約を削除し、再定義することができます。

6.53 データ削除時に子表行を削除する参照整合性制約

親表の行を削除するときに従属表（子表）の該当行も削除します。

Oracleの場合

書式

```
ALTER TABLE 表名 ADD [CONSTRAINT 制約名] FOREIGN KEY(列名)
REFERENCES 表名(列名) ON DELETE CASCADE
```

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REFE
➥RENCES DEPT(DEPTNO) ON DELETE CASCADE;

表が変更されました。
```

DB2の場合

書式

```
ALTER TABLE 表名 ADD [CONSTRAINT 制約名] FOREIGN KEY(列名)
REFERENCES 表名(列名) ON DELETE CASCADE
```

実行例

```
db2=> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REF
➥ERENCES DEPT(DEPTNO) ON DELETE CASCADE
DB20000I  SQL コマンドが正常に終了しました。
```

6.54 データ削除時に子表行をNULL値に更新する参照整合性制約

親表の行を削除するときに従属表（子表）の該当行をNULL値に更新します。

Oracleの場合

書式

```
ALTER TABLE 表名 ADD [ CONSTRAINT 制約名] FOREIGN KEY(列名)
REFERENCES 表名(列名) ON DELETE SET NULL
```

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REFE
➥RENCES DEPT(DEPTNO) ON DELETE SET NULL;

表が変更されました。
```

DB2の場合

書式

```
ALTER TABLE 表名 ADD [CONSTRAINT 制約名] FOREIGN KEY(列名)
REFERENCES 表名(列名) ON DELETE SET NULL
```

実行例

```
db2=> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REF
➥ERENCES DEPT(DEPTNO) ON DELETE SET NULL
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 DB2 UDBでは、DELETE CASCADE, SET NULLのほかに、子表があったら削除をしない（エラーSQL0532Nになる）RESTRICTがあります。

6.55 ビューの作成

ビューを作成します。

Oracle の場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT {列名 | *} FROM 表名 [WHERE 検索条件]
```

実行例

```
SQL> CREATE VIEW EMP_VIEW
  2  AS SELECT * FROM EMP;

ビューが作成されました。
```

DB2 の場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS [WITH 共通表式] SELECT {列名 | *} FROM 表名 [WHERE 検索条件]
```

実行例

```
db2=> CREATE VIEW EMP_VIEW AS SELECT * FROM EMP
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 問い合わせ結果を元にビューを作成できます。ビューを使えば、結合や集計など複雑なSQL処理が可能になります。

6.56 特定列を指定したビューの作成

指定した列を元にビューを作成します。列に対するセキュリティを実現します。

Oracleの場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT 列名[,・・・] FROM 表名 [WHERE 検索条件]
```

実行例

```
SQL> CREATE VIEW EMP_VIEW2 AS SELECT EMPNO,ENAME,JOB FROM EMP;

ビューが作成されました。
```

DB2の場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT 列名[,・・・] FROM 表名 [WHERE 検索条件]
```

実行例

```
db2=> CREATE VIEW EMP_VIEW2 AS SELECT EMPNO,ENAME,JOB FROM EMP
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 特定の列だけを指定してビューを作成することにより、列に対するセキュリティを強化できます。

6.57 特定行を指定したビューの作成

条件に一致した行だけのビューを作成します。行に対するセキュリティを実現します。

Oracle の場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT {列名 | *} FROM 表名 [WHERE 検索条件]
```

実行例

```
SQL> CREATE VIEW EMP_VIEW3 AS SELECT * FROM EMP WHERE DEPTNO = 30;

ビューが作成されました。
```

DB2 の場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT {列名 | *} FROM 表名 [WHERE 検索条件]
```

実行例

```
db2=> CREATE VIEW EMP_VIEW3 AS SELECT * FROM EMP WHERE DEPTNO = 30
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

WHERE 句を指定し、特定の行だけアクセスできるビューを作成することにより、行に対するセキュリティを実現することができます。

6.58 複数の表を使用したビューの作成

結合を使用した（複数の表を使用した）ビューを作成します。正規化され、複雑な構造になっている表関係を隠すことができます。

Oracleの場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT {列名 | *} FROM 表名 JOIN 表名 ON 結合条件
```

実行例

```
SQL> CREATE VIEW EMP_DEPT
  2  AS SELECT D.DEPTNO,D.DNAME,E.EMPNO,E.ENAME
  3     FROM   DEPT D JOIN EMP E
  4     ON     D.DEPTNO = E.DEPTNO;

ビューが作成されました。
```

DB2の場合

書式

```
CREATE VIEW ビュー名[ (列名) ]
AS SELECT {列名 | *} FROM 表名 JOIN 表名 ON 結合条件
```

実行例

```
------------------------------ 入力コマンド ------------------------------
CREATE VIEW EMP_DEPT AS
SELECT D.DEPTNO,D.DNAME,E.EMPNO,E.ENAME
 FROM EMP E
JOIN
 DEPT D
 ON E.DEPTNO = D.DEPTNO;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 正規化された複雑な表関係を、結合を使用した（複数の表を使用した）ビューを作成することで、隠蔽することができます。

6.59 ビューに対する問い合わせ

ビューに対して問い合わせを行います。

Oracle の場合

書式 **SELECT * FROM ビュー名**

実行例

```
SQL> SELECT * FROM EMP_VIEW3;

  EMPNO ENAME      JOB            MGR HIREDATE      SAL     COMM  DEPTNO
------- ---------- --------- ------- -------- ------- -------- -------
   7499 ALLEN      SALESMAN      7698 81-02-20    1600      300       30
   7521 WARD       SALESMAN      7698 81-02-22    1250      500       30
   7654 MARTIN     SALESMAN      7698 81-09-28    1250     1400       30
   7698 BLAKE      MANAGER       7839 81-05-01    2850                30
   7844 TURNER     SALESMAN      7698 81-09-08    1500        0       30
   7900 JAMES      CLERK         7698 81-12-03     950                30

6行が選択されました。
```

DB2 の場合

書式 **SELECT * FROM ビュー名**

実行例

```
db2=> SELECT * FROM EMP_VIEW3

EMPNO  ENAME      JOB       MGR    HIREDATE   SAL        COMM       DEPTNO
------ ---------- --------- ------ ---------- --------- --------- ------
 7499. ALLEN      SALESMAN   7698. 1981-02-20   1600.00    300.00    30.
 7521. WARD       SALESMAN   7698. 1981-02-22   1250.00    500.00    30.
 7654. MARTIN     SALESMAN   7698. 1981-09-28   1250.00   1400.00    30.
 7698. BLAKE      MANAGER    7839. 1981-05-01   2850.00          -    30.
 7844. TURNER     SALESMAN   7698. 1981-09-08   1500.00       0.00    30.
 7900. JAMES      CLERK      7698. 1981-12-03    950.00          -    30.

  6 レコードが選択されました。
```

注意点 ビューは仮想表です。表と同じように使用することができます。

6.60 ビューから行を挿入する

ビューから行を挿入します。

Oracleの場合

書式 **INSERT INTO ビュー名 VALUES(値)**

実行例

```
SQL> INSERT INTO EMP_VIEW (EMPNO,ENAME,HIREDATE) VALUES(8001,'BOB',SYSDATE
➥);

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO ビュー名 VALUES(値)**

実行例

```
db2=> INSERT INTO EMP_VIEW (EMPNO,ENAME,HIREDATE) VALUES(8001,'BOB',CURREN
➥T_DATE)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | ビューは仮想表です。表と同じように使用することができます。

6.61 ビューから列を更新する

ビューから列を更新します。

Oracle の場合

書式 **UPDATE ビュー名 SET 列名 = 値 WHERE 検索条件**

実行例

```
SQL> UPDATE EMP_VIEW SET SAL = 1200 WHERE EMPNO = 8001;

1行が更新されました。
```

DB2 の場合

書式 **UPDATE ビュー名 SET 列名 = 値 WHERE 検索条件**

実行例

```
db2=> UPDATE EMP_VIEW SET SAL = 1200 WHERE EMPNO = 8001
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 ビューは仮想表です。表と同じように使用することができます。

6.62 ビューから行を削除する

ビューから行を削除します。

Oracleの場合

書式 **DELETE FROM ビュー名 [WHERE 検索条件]**

実行例

```
SQL> DELETE FROM EMP_VIEW WHERE EMPNO = 8001;

1行が削除されました。
```

DB2の場合

書式 **DELETE FROM ビュー名 [WHERE 検索条件]**

実行例

```
db2=> DELETE FROM EMP_VIEW WHERE EMPNO = 8001
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | ビューは仮想表です。表と同じように使用することができます。

6.63 ビューの変更

ビューを変更したり、ビューを再作成します。

Oracle の場合

書式

```
CREATE OR REPLACE ビュー名 [ (列名) ]
AS SELECT {列名 | *} FROM 表名 [ WHERE 検索条件 ]
```

実行例

```
SQL> CREATE OR REPLACE VIEW EMP_VIEW
  2  AS SELECT EMPNO,ENAME FROM EMP WHERE DEPTNO = 10;

ビューが作成されました。
```

DB2 の場合

書式

対応するコマンドはありませんが、コントロール・センターを使って変更ができます。

実行例

注意点

OR REPLACE キーワードをつけてビューを作成することにより、すでに同じ名前のビューが存在していた場合は再作成されます。ビューの定義変更を行うことができます。

6.64 ビューの削除

ビューを削除します。

Oracleの場合

書式 **DROP VIEW ビュー名**

実行例
```
SQL> DROP VIEW EMP_VIEW2;

ビューが削除されました。
```

DB2の場合

書式 **DROP VIEW ビュー名**

実行例
```
db2=> DROP VIEW EMP_VIEW2
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | ビューを削除しても元の表に影響はありません。

6.65 ビュー同士を結合する

ビュー同士を結合して結果を求めます。

Oracleの場合

書式 **SELECT 列名 FROM ビュー名 JOIN ビュー名 ON 結合条件**

実行例

```
SQL> SELECT DV.DEPTNO,DV.DNAME,EV.EMPNO,EV.ENAME
  2  FROM DEPT_VIEW DV JOIN EMP_VIEW EV
  3  ON   DV.DEPTNO = EV.DEPTNO;

    DEPTNO DNAME              EMPNO ENAME
---------- -------------- ---------- ----------
        10 ACCOUNTING           7782 CLARK
        10 ACCOUNTING           7839 KING
        10 ACCOUNTING           7934 MILLER
        20 RESEARCH             7369 SMITH
        20 RESEARCH             7566 JONES
        20 RESEARCH             7788 SCOTT
        20 RESEARCH             7876 ADAMS
        20 RESEARCH             7902 FORD
        30 SALES                7499 ALLEN
        30 SALES                7521 WARD
        30 SALES                7654 MARTIN
        30 SALES                7698 BLAKE
        30 SALES                7844 TURNER
        30 SALES                7900 JAMES

14行が選択されました。
```

DB2の場合

書式 **SELECT 列名 FROM ビュー名 JOIN ビュー名 ON 結合条件**

実行例

```
db2=> SELECT DV.DEPTNO,DV.DNAME,EV.EMPNO,EV.ENAME FROM DEPT_VIEW DV JOIN
➥EMP_VIEW EV ON DV.DEPTNO = EV.DEPTNO

DEPTNO DNAME          EMPNO  ENAME
------ -------------- ------ ----------
   20. RESEARCH         7369. SMITH
   30. SALES            7499. ALLEN
   30. SALES            7521. WARD
   20. RESEARCH         7566. JONES
   30. SALES            7654. MARTIN
   30. SALES            7698. BLAKE
   10. ACCOUNTING       7782. CLARK
   20. RESEARCH         7788. SCOTT
   10. ACCOUNTING       7839. KING
   30. SALES            7844. TURNER
   20. RESEARCH         7876. ADAMS
   30. SALES            7900. JAMES
   20. RESEARCH         7902. FORD
   10. ACCOUNTING       7934. MILLER

  14 レコードが選択されました。
```

注意点 ビューは仮想表です。表と同じように使用することができます。ビューと表を結合することもできます。

6.66 一意索引を作成する

一意索引を作成します。一意な値しか持たない列に索引を作成します。

Oracle の場合

書式 **CREATE UNIQUE INDEX 索引名 ON 表名(列名 [,･･･])**

実行例

```
SQL> CREATE UNIQUE INDEX I_EMP_ENAME ON EMP(ENAME);

索引が作成されました。
```

DB2 の場合

書式 **CREATE UNIQUE INDEX 索引名 ON 表名(列名 [,･･･])**

実行例

```
db2=> CREATE UNIQUE INDEX I_EMP_ENAME ON EMP(ENAME)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

一意索引を作成する場合は、CREATE UNIQUE INDEX文を使用します。表にまだ1件も行が存在しない場合、または索引を定義する列に重複した値が存在しない場合、一意索引は作成できません。索引を定義する列に、すでに重複した値を持つ行が存在する場合は作成できません。DB2 UDBでは、表にまだ1件も行が存在しない場合でも一意索引は作成できます。
また、一意索引の列名にNULL可能な列を含めることができます。表に対する一意制約はNULL可能な列に対して定義できません。違いに注意してください。

6.67 非一意索引を作成する

非一意索引を作成します。重複した値を持つ列に索引を定義します。

Oracleの場合

書式 **CREATE INDEX 索引名 ON 表名(列名 [,･･･])**

実行例

```
SQL> --非一意索引を作成する
SQL> CREATE INDEX I_EMP_DEPTNO ON EMP(DEPTNO);

索引が作成されました。
```

DB2の場合

書式 **CREATE INDEX 索引名 ON 表名(列名 [,･･･])**

実行例

```
db2=> CREATE INDEX I_EMP_DEPTNO ON EMP(DEPTNO)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 非一意索引を作成する場合は、CREATE INDEX文を使用します。表に行が格納されていてもいなくても非一意索引を作成ですることができます。

6.68 複数列索引を作成する

複数の列を使用して1つの索引を定義します。WHERE句で指定される複数の列に対して1つの索引を定義します。

Oracleの場合

書式 **CREATE INDEX 索引名 ON 表名(列名,列名, ・・・)**

実行例

```
SQL> CREATE INDEX I_EMP_JOB_SAL ON EMP(JOB,SAL);

索引が作成されました。
```

DB2の場合

書式 **CREATE INDEX 索引名 ON 表名(列名,列名, ・・・)**

実行例

```
db2=> CREATE INDEX I_EMP_JOB_SAL ON EMP(JOB,SAL)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 複数の列を組み合わせて1つの索引を作成することができます。多くのSQLの検索条件に使用される列または値の種類の多い列を先頭に指定するとパフォーマンスを向上できます。

6.69 索引を削除する

索引を削除します。

Oracleの場合

書式 **DROP INDEX 索引名**

実行例
```
SQL> DROP INDEX I_EMP_JOB_SAL;
```

DB2の場合

書式 **DROP INDEX 索引名**

実行例
```
db2=> DROP INDEX I_EMP_JOB_SAL
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 索引を削除します。索引は検索のパフォーマンスを向上させますが、データ操作時（INSERT、UPDATE、DELETE）には書き込みの負荷をかけるため、使用されていない索引や効果のない索引は削除することをお勧めします。

6.70 索引を再構築する

索引を再作成します。索引の断片化を解消します。

Oracleの場合

書式 **ALTER INDEX 索引名 REBUILD [ONLINE]**

実行例

```
SQL> ALTER TABLE EMP ADD CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REFE
➥RENCES DEPT(DEPTNO) ON DELETE SET NULL;

表が変更されました。
```

DB2の場合

書式 **REORG INDEXES ALL FOR 表名**

実行例

```
db2=> REORG INDEXES ALL FOR EMP
```

注意点

索引を再構築します。索引の断片化はパフォーマンスを劣化させるため、断片化を解消をする必要があります。本コマンドは索引全体が断片化している場合、過剰に索引の階層が深くなっている場合に使用します。
DB2 UDBでは索引の再編成と呼び、REORG INDEXESコマンドは索引を完全に再作成します。

6.71 索引を再編成する

索引を再編成します。索引の断片化を解消します。

Oracleの場合

書式 **ALTER INDEX 索引名 COALESCE**

実行例
```
SQL> ALTER INDEX I_EMP_ENAME COALESCE;

索引が変更されました。
```

DB2の場合

書式 **REORG INDEXES ALL FOR 表名 CLEANUP ONLY**

実行例
```
db2=> REORG INDEXES ALL FOR EMP CLEANUP ONLY
```

注意点 索引の断片化はパフォーマンスを劣化させるため、断片化を解消をする必要があります。本コマンドは隣接したリーフ・ブロックが断片化している場合に使用します。
DB2 UDBでCLEANUP ONLYオプションを指定した場合は、索引の再作成は行われません。既存の索引スペース内で再編成します。

6.72 索引の使用状況を監視する

索引が一度でも使用されたかどうかを監視します。

Oracle の場合

書式 **ALTER INDEX 索引名 MONITORING USAGE**

実行例

```
SQL> ALTER INDEX I_EMP_ENAME MONITORING USAGE;

索引が変更されました。
```

DB2 の場合

書式 **対応する機能はありませんが、設計アドバイザーを使用して、無駄な索引を削除することができます。**

実行例

注意点 索引が使用されるかどうかを監視します。索引は検索のパフォーマンスを向上させますが、データ操作時（INSERT、UPDATE、DELETE）には書き込みの負荷をかけるため、使用されていない索引や効果のない索引は削除することをお勧めします。

6.73 索引の一覧を表示する

表に定義されている索引を確認します。

Oracleの場合

書式 **SELECT INDEX_NAME,TABLE_NAME FROM USER_INDEXES**

実行例

```
SQL> SELECT INDEX_NAME,TABLE_NAME FROM USER_INDEXES;

INDEX_NAME                     TABLE_NAME
------------------------------ ------------------------------
I_EMP_DEPTNO                   EMP
I_EMP_ENAME                    EMP
I_EMP_JOB_SAL                  EMP
```

DB2の場合

書式 **SELECT 列名 [, 列名 ...] FROM SYSCAT.INDEXES [検索条件]**

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CHAR(INDNAME,25) INDEX_NAME, CHAR(TABNAME,30) TABLE_NAME
  FROM SYSCAT.INDEXES
 WHERE DEFINER = 'DB2ADMIN'
   AND TABNAME LIKE 'E%'
ORDER BY TABLE_NAME;
------------------------------------------------------------------------

INDEX_NAME                TABLE_NAME
------------------------- ------------------------------
SQL051026075053530        EMP_PHOTO
SQL051026075056120        EMP_RESUME
E3_A                      EMP3
SQL051026075051820        EMPLOYEE

  4 レコードが選択されました。
```

注意点 Oracleでは、表に定義されている索引を確認する場合は、USER_INDEXESデータディクショナリ・ビューを使用します。DB2 UDBでは、DESCRIBE INDEXES FOR TABLEコマンドを使います。

6.74 複数列索引の列の並び順を調べる

列に定義されている索引および複数列索引の列の並び順を確認します。

Oracleの場合

書式

```
SELECT 列名[, 列名 ･･･]
FROM USER_IND_COLUMNS
WHERE TABLE_NAME = '表名' ORDER BY 1,2,3
```

実行例

```
SQL> SELECT INDEX_NAME,COLUMN_NAME,COLUMN_POSITION FROM USER_IND_COLUMNS
  2  WHERE TABLE_NAME = 'EMP' ORDER BY 1,2,3;

INDEX_NAME           COLUMN_NAME          COLUMN_POSITION
-------------------- -------------------- ---------------
I_EMP_DEPTNO         DEPTNO                             1
I_EMP_ENAME          ENAME                              1
I_EMP_JOB_SAL        JOB                                1
I_EMP_JOB_SAL        SAL                                2
```

DB2の場合

書式

```
SELECT 列名[, 列名 ･･･] FROM SYSCAT.INDEXCOLUSE C, SYSCAT.INDEXES I
WHERE IC.INDNAME=I.INDNAME AND I.TABNAME = '表名'
```

実行例

```
------------------------------ 入力コマンド ------------------------------
SELECT CHAR(TABNAME,20) TABLE_NAME, CHAR(I.INDNAME,25) INDEX_NAME
     , CHAR(COLNAME,15) COLUMN_NAME, COLSEQ
  FROM SYSCAT.INDEXCOLUSE C
     , SYSCAT.INDEXES     I
 WHERE C.INDNAME = I.INDNAME
   AND I.TABNAME LIKE 'EMP%'
 ORDER BY 1, 2, 4;
------------------------------------------------------------------------

TABLE_NAME           INDEX_NAME                COLUMN_NAME     COLSEQ
-------------------- ------------------------- --------------- ------
EMP_PHOTO            SQL051026075053530        EMPNO                1
EMP_PHOTO            SQL051026075053530        PHOTO_FORMAT         2
EMP_RESUME           SQL051026075056120        EMPNO                1
EMP_RESUME           SQL051026075056120        RESUME_FORMAT        2
EMP3                 E3_A                      EMPNO                1
EMPLOYEE             SQL051026075051820        EMPNO                1

  6 レコードが選択されました。
```

注意点

Oracleでは、列に定義されている索引および複数列索引の列の並び順を確認する場合、USER_IND_COLUMNSデータディクショナリ･ビューを使用します。複数列索引において、列の並び順はパフォーマンスに影響を与えるため重要です。

6.75 順序を作成する

順序（シーケンス）を作成します。

Oracleの場合

書式

```
CREATE SEQUENCE 順序名
[START WITH 開始値] [INCREMENT BY 増分値] [CYCLE | NOCYCLE]
[MINVALUE 値] [MAXVALUE 値] [CACHE 値 | NOCACHE]
```

実行例

```
SQL> CREATE SEQUENCE SEQ_DEPTNO START WITH 50 INCREMENT BY 10;

順序が作成されました。
```

DB2の場合

書式

```
CREATE SEQUENCE 順序名
[START WITH 開始値] [INCREMENT BY 増分値] [CYCLE | NOCYCLE]
[MINVALUE 値] [MAXVALUE 値]
```

実行例

```
db2=> CREATE SEQUENCE SEQ_DEPTNO START WITH 50 INCREMENT BY 10
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 一定の間隔で連続した値を生成する必要がある場合は、順序（シーケンス）を使用すると便利です。

6.76 順序を変更する

既存の順序を変更します。

Oracle の場合

書式

```
ALTER SEQUENCE 順序名
[INCREMENT BY 増分値] [CYCLE | NOCYCLE]
[MINVALUE 値] [MAXVALUE 値] [CACHE 値 | NOCACHE]
```

実行例

```
ALTER SEQUENCE SEQ_DEPTNO INCREMENT BY 5;

順序が変更されました。
```

DB2 の場合

書式

```
ALTER SEQUENCE 順序名
[INCREMENT BY 増分値] [CYCLE | NOCYCLE] [MINVALUE 値]
[MAXVALUE 値] [CACHE 値 | NOCACHE] [ORDER| NOORDER]
```

実行例

```
db2=> ALTER SEQUENCE SEQ_DEPTNO INCREMENT BY 5
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | 既存の順序を変更することができます。変更後の値は、次の発番から有効になります。

6.77 順序を採番する

順序から新しい値を発番します。

Oracle の場合

書式 **順序名.NEXTVAL**

実行例

```
SQL> SELECT SEQ_DEPTNO.NEXTVAL FROM DUAL;

   NEXTVAL
----------
        50
```

DB2 の場合

書式 **NEXT VALUE**
または
NEXTVAL FOR 順序名

実行例

```
db2=> SELECT NEXTVAL FOR SEQ_DEPTNO FROM DUMMY

1
-----------
         50

  1 レコードが選択されました。
```

注意点 順序から新しい値を発番します。SELECT文、INSERT文、UPDATE文のSET句で使用できます。
Oracleでは、NEXTVAL擬似列を使用します。
DB2 UDBでは、NEXT VALUEまたはNEXTVAL FORを使用します。

6.78 最新の順序値を確認する

最新の順序値（現在の順序値）を確認します。

Oracleの場合

書式 **順序名.CURRVAL**

実行例

```
SQL> SELECT SEQ_DEPTNO.CURRVAL FROM DUAL;

   CURRVAL
----------
        50
```

DB2の場合

書式

PREVIOUS VALUE
または
PREVVAL FOR 順序名

実行例

```
db2=> SELECT PREVVAL FOR SEQ_DEPTNO FROM DUMMY

1
-----------
         50

  1 レコードが選択されました。
```

注意点

セッション内で最後に発番した順序値を求めます。SELECT文、INSERT文、UPDATE文のSET句で使用できます。
Oracleでは、CURRVAL擬似列を使用します。
DB2 UDBでは、PREVIOUS VALUEまたはPREVVAL FORを使用します。
いずれも、セッション内で順序を生成していなければCURRVALは使用できません。

6.79 行挿入時に順序を使用する

行挿入時に順序を使用します。

Oracleの場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(順序名.NEXTVAL [, ･･･])**

実行例

```
SQL> INSERT INTO DEPT VALUES(SEQ_DEPTNO.NEXTVAL,'FINACE','TOKYO');

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(NEXT VALUE | NEXTVAL FOR 順序名 [, ･･･])**

実行例

```
db2=> INSERT INTO DEPT VALUES(NEXTVAL FOR SEQ_DEPTNO,'FINACE','TOKYO')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | INSERTのVALUES句に順序を使用することができます。

6.80 列更新時に順序を使用する

列の更新時に順序を使用します。

Oracleの場合

書式 **UPDATE 表名 SET 列名 = 順序名.NEXTVAL [WHERE 検索条件]**

実行例

```
SQL> UPDATE DEPT SET DEPTNO = SEQ_DEPTNO.NEXTVAL WHERE DEPTNO = 40;

1行が更新されました。
```

DB2の場合

書式 **UPDATE 表名 SET 列名 = NEXT VALUE | NEXTVAL FOR 順序名 [WHERE 検索条件]**

実行例

```
db2=> UPDATE DEPT SET DEPTNO = NEXTVAL FOR SEQ_DEPTNO WHERE DEPTNO = 40
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | UPDATEのSET句に順序を使用することができます。

6.81 順序を削除する

順序を削除します。

Oracleの場合

書式 **DROP SEQUENCE 順序名**

実行例
```
SQL> DROP SEQUENCE SEQ_DEPTNO;

順序が削除されました。
```

DB2の場合

書式 **DROP SEQUENCE 順序名 [RESTRICT]**

実行例
```
db2=> DROP SEQUENCE SEQ_DEPTNO
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | 順序を削除しても、すでに発番した順序に影響はありません。

6.82 シノニムを作成する

オブジェクトの別名（シノニム、エイリアス）を定義します。

Oracle の場合

書式　**CREATE SYNONYM シノニム名 FOR オブジェクト名**

実行例

```
SQL> CREATE SYNONYM SEMP FOR EMP;
```

シノニムが作成されました。

DB2 の場合

書式　**CREATE ALIAS エイリアス名 FOR {表名 | ビュー名 | ニックネーム}**
または
CREATE SYNONYM シノニム名 FOR {表名 | ビュー名 | ニックネーム}

実行例

```
db2=> CREATE SYNONYM SEMP FOR EMP
DB20000I  SQL コマンドが正常に終了しました。
```

または

```
db2=> CREATE ALIAS SEMP FOR EMP
DB20000I  SQL コマンドが正常に終了しました。
```

注意点　長い名前のオブジェクトやほかのスキーマのオブジェクトに別名（シノニム、エイリアス）を定義し、オブジェクトの扱いを容易にすることができます。
DB2 UDB で CREATE SYNONYM コマンドは、互換性のために残っています。推奨されている方法は CREATE ALIAS コマンドです。

6.83 パブリック・シノニムを作成する

データベース全体で使用できるオブジェクトの別名（シノニム、エイリアス）を定義します。
パブリック・シノニムを定義します。

Oracleの場合

書式 **CREATE PUBLIC SYNONYM シノニム名 FOR オブジェクト名**

実行例

```
SQL> CREATE PUBLIC SYNONYM SEMP FOR USER1.EMP;

シノニムが作成されました。
```

DB2の場合

書式 **対応するコマンドはありません。**

実行例

注意点

長い名前のオブジェクトやほかのスキーマのオブジェクトに別名（シノニム、エイリアス）を定義し、オブジェクトの扱いを容易にすることができます。
パブリック・シノニムは、スキーマ内のシノニムより優先されます。シノニムの元となるオブジェクトに対してアクセス権限を持っているすべてのデータベース・ユーザーが利用できます。
DB2 UDBではスキーマ内で管理するエイリアスまたはシノニムを作成することはできますが、パブリック・シノニムを作成することはできません。DB2 UDBのエイリアスは、エイリアスの元となるオブジェクトに対するアクセス権限を持っているすべてのユーザーが利用できます。

6.84 シノニムを削除する

別名（シノニム、エイリアス）を削除します。

Oracle の場合

書式

```
DROP SYNONYM シノニム名
```

実行例

```
SQL> DROP SYNONYM SEMP;

シノニムが削除されました。
```

DB2 の場合

書式

```
DROP  SYNONYM シノニム名
または
DROP  ALIAS エイリアス名
```

実行例

```
db2=> DROP SYNONYM SEMP
DB20000I  SQL コマンドが正常に終了しました。

または

db2=> DROP ALIAS SEMP
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 ｜ 別名（シノニム、エイリアス）を削除しても、元のオブジェクトに影響はありません。

6.85 全行を高速に削除する（TRUNCATE）

表内の全行を高速に削除します。表の行を切り捨てます。削除する行の更新前の値（UNDOデータ）は作成しません。

Oracleの場合

書式 **TRUNCATE TABLE 表名**

実行例

```
SQL> TRUNCATE TABLE EMP;

表が切り捨てられました。

SQL> SELECT COUNT(*) FROM EMP;

COUNT(*)
--------
       0
```

DB2の場合

書式 **ALTER TABLE EMP ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE**

または

●UNIX

IMPORT FROM /dev/null OF DEL REPLACE INTO TBL名

●Windows

IMPORT FROM NUL OF DEL REPLACE INTO TBL名

実行例

```
db2=> ALTER TABLE EMP ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
DB20000I  SQL コマンドが正常に終了しました。

db2=> SELECT COUNT(*) FROM EMP

1
-----------
          0

  1 レコードが選択されました。
```

注意点 TRUNCATE文はDDL（データ定義言語）のひとつで、表内の行を切り捨てます。削除する行の更新前の値（UNDOデータ）は作成しないため、高速に行を削除できます。
DB2 UDBにTRUNCATE文はありませんが、IMPORT～REPLACEコマンドを使う方法が一般的です。

7

レコードの操作

7.1 行の挿入

表に行を1行追加します。

Oracleの場合

書式 **INSERT INTO 表名 (列名[, ･･･]) VALUES(値[, ･･･])**

実行例

```
SQL> INSERT INTO DEPT (DEPTNO,DNAME,LOC) VALUES(50,'EDUCATION','TOKYO');

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (列名[, ･･･]) VALUES(値[, ･･･])**

実行例

```
db2=> INSERT INTO DEPT (DEPTNO,DNAME,LOC) VALUES(50,'EDUCATION','TOKYO')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

INSERTを使用して、表に行を追加します。
Oracleで1行ずつ行を挿入する場合は、VALUES句を使用します。
DB2 UDBでは、VALUES句を使用して1行以上挿入していくことができます。
挿入する値に、文字および日付を指定する場合は、値を単一引用符(')で囲む必要があります。
Oracleは、暗黙型変換を行いますが、DB2 UDBは暗黙型変換を行わないため、数値を単一引用符で囲んではいけません。

7.2 挿入列を省略した行の挿入

列名の指定を省略して、表に行を追加します。

Oracleの場合

書式 **INSERT INTO 表名 VALUES(値[, ･･･])**

実行例

```
SQL> INSERT INTO DEPT VALUES(50,'EDUCATION','TOKYO');

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 VALUES(値[, ･･･])**

実行例

```
db2=> INSERT INTO DEPT VALUES(50,'EDUCATION','TOKYO')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 列名を省略すると、表を定義した際の列の順に従い、VALUES句に値を指定します。すべての列に値を指定する必要があります。このとき、NULL値やデフォルト値を使用することも可能です。

7.3 特定の列を指定した行の挿入

表のすべての列ではなく、特定の列だけを指定して行を追加します。

Oracle の場合

書式 **INSERT INTO 表名 (列名[, ･･･]) VALUES(値[, ･･･])**

実行例

```
SQL> INSERT INTO DEPT (DEPTNO,LOC) VALUES(50,'TOKYO');

1行が作成されました。
```

DB2 の場合

書式 **INSERT INTO 表名 (列名[, ･･･]) VALUES(値[, ･･･])**

実行例

```
db2=> INSERT INTO DEPT (DEPTNO,LOC) VALUES(50,'TOKYO')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 特定の列だけを対象にし、行を挿入する場合は、表名の後ろに列名を指定します。NOT NULL制約が定義してある列は、デフォルト値が設定してあれば省略することができますが、設定していない場合は、必ず指定する必要があります。

7.4 ほかの表から既存表へ行を挿入

ほかの表の行を挿入します（コピーします）。

Oracleの場合

書式 **INSERT INTO 表名 SELECT文**

実行例

```
SQL> INSERT INTO DEPT_COPY SELECT * FROM DEPT;

4行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 SELECT文**

実行例

```
db2=> INSERT INTO DEPT_COPY SELECT * FROM DEPT
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | INSERTに問い合わせを指定することで、ほかの表の行を挿入（コピー）できます。

7.5 ほか表からの既存表に列を挿入

ほかの表の特定の列だけを挿入（コピー）する。

Oracleの場合

書式 **INSERT INTO 表名（列名 [, ･･･]）SELECT文**

実行例

```
SQL> INSERT INTO DEPT_COPY (DEPTNO,DNAME) SELECT DEPTNO,DNAME FROM DEPT;

4行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名（列名 [, ･･･]）SELECT文**

実行例

```
db2=> INSERT INTO DEPT_COPY (DEPTNO,DNAME) SELECT DEPTNO,DNAME FROM DEPT
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 ｜ SELECT文に特定の列だけを指定し、ほかの表の一部（特定の列）を挿入（コピー）できます。

7.6 数値データの挿入

数値データを挿入します。

Oracleの場合

書式 **INSERT INTO 表名 (数値型列名 [, ･･･]) VALUES(数値[, ･･･])**

実行例

```
SQL> INSERT INTO EMP (EMPNO) VALUES(8000);

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (数値型列名 [, ･･･]) VALUES(数値[, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO) VALUES(8000)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

数値データは単一引用符 (') で囲む必要はありません。
Oracleは、暗黙型変換を行うため、単一引用符で囲んでも正しく処理されます。
DB2 UDBは暗黙型変換を行わないため、数値を単一引用符で囲んではいけません。

7.7 文字データの挿入

文字データを挿入します。

Oracle の場合

書式 **INSERT INTO 表名 (文字列型列名 [, ･･･]) VALUES(文字列[, ･･･])**

実行例

```
SQL> INSERT INTO EMP (EMPNO,ENAME) VALUES(8000,'ALICE');

1行が作成されました。
```

DB2 の場合

書式 **INSERT INTO 表名 (文字列型列名 [, ･･･]) VALUES(文字列[, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO,ENAME) VALUES(8000,'ALICE')
DB20000I  SQL コマンドが正常に終了しました
```

注意点 | 挿入する値に文字および日付を指定する場合は、値を単一引用符 (') で囲む必要があります。

7.8 日付データの挿入

日付データを挿入します。

Oracleの場合

書式 **INSERT INTO 表名 (日付型列名 [, ･･･]) VALUES(日付[, ･･･])**

実行例

```
SQL> INSERT INTO EMP (EMPNO,ENAME,HIREDATE) VALUES(8000,'ALICE','05-04-01
➥');

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (日付型列名 [, ･･･]) VALUES(日付[, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO,ENAME,HIREDATE) VALUES(8000,'ALICE','2005-04-
➥01')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 挿入する値に、文字および日付を指定する場合は、値を単一引用符 (') で囲む必要があります。
また、日付は指定された書式に従わなければいけません。
使える書式に、'yyyy-mm-dd'、'mm/dd/yyyy'、'dd.mm.yyyy' があります。

7.9 NULLデータの挿入

明示的にNULL値を挿入します

Oracleの場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(NULL [, ･･･])**

実行例

```
SQL> INSERT INTO EMP (EMPNO,ENAME,JOB) VALUES(8000,'ALICE',NULL);

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(NULL [, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO,ENAME,JOB) VALUES(8000,'ALICE',NULL)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 行挿入時にNULLを明示的に指定する場合は、VALUES句の該当箇所に「NULL」と指定します。あるいは、挿入対象に指定されなかった列は、暗黙的にNULL値が設定されます。その列にデフォルト値が定義されている場合は、暗黙的にデフォルト値が設定されます。

7.10 デフォルト値を使用した挿入

明示的にデフォルト値を挿入します。

Oracleの場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(DEFAULT [, ･･･])**

実行例

```
SQL> SQL> INSERT INTO EMP (EMPNO,ENAME,LOC) VALUES(8000,'ALICE',DEFAULT);

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(DEFAULT [, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO,ENAME,SAL) VALUES(8000,'ALICE',DEFAULT)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 行挿入時に明示的にデフォルト値を指定する場合は、VALUES句の該当箇所に「DEFAULT」と指定します。あるいは、デフォルト値が定義されている列が、挿入対象に指定されなかった場合は、暗黙的にデフォルト値が設定されます。

7.11 今日の日付の挿入

今日の日付を挿入します。

Oracleの場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(SYSDATE [, ･･･])**

実行例

```
SQL> INSERT INTO EMP (EMPNO,ENAME,HIREDATE) VALUES(8000,'ALICE',SYSDATE);

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(CURRENT_DATE [, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO,ENAME,HIREDATE) VALUES(8000,'ALICE',CURRENT
➥_DATE)
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

行挿入時に今日の日付を指定する場合は、今日の日付（時刻）を戻す関数を使用すると便利です。Oracleでは、SYSDATE関数を使用します。なお、SYSDATE関数は日付と時刻を戻します。DB2 UDBでは、日付だけを求めるのであれば、CURRENT_DATE特殊レジスターを使用します。

7.12 書式を指定した日付の挿入

デフォルト書式以外の書式を使用して、日付データを挿入します。

Oracleの場合

書式 **INSERT INTO 表名 (列名 [, ･･･]) VALUES(TO_DATE('日付','日付書式'))**

実行例

```
SQL> INSERT INTO EMP (EMPNO,ENAME,HIREDATE) VALUES(8000,'ALICE',TO_DATE('
➥20050401','YYYYMMDD'));

1行が作成されました。
```

DB2の場合

書式 **対応する機能はありません。ただし、日付のフォーマットとして、'yyyy-mm-dd'、'mm/dd/yyyy'、'dd.mm.yyyy'が使えます。通常は、式によってデフォルトの日付フォーマットに変換します。**

実行例

```
------------------------------ 入力コマンド ------------------------------
INSERT INTO EMP (EMPNO, ENAME, HIREDATE)
VALUES (8000, 'ALICE', TRANSLATE('ABCD-EF-GH','20050401','ABCDEFGH'));
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。

------------------------------ 入力コマンド ------------------------------
SELECT EMPNO, ENAME, HIREDATE
  FROM EMP
 WHERE EMPNO = 8000;
------------------------------------------------------------------------

EMPNO  ENAME HIREDATE
------ ----- ----------
  8000 ALICE 2005-04-01

  1 レコードが選択されました。
```

注意点

Oracleでは、セッション単位に日付書式を変更できます。SQL文ごとに指定する場合は、TO_DATE関数を使用します。

DB2 UDBは、日付データの挿入はできますが、デフォルト以外の書式のデータは式によりデフォルト形式に変換します。よく使う書式があれば、デフォルト形式に変換するユーザー定義関数(UDF)を作成しておくとよいでしょう。

7.13 単一引用符（'）をデータとして挿入する

文字データの一部に単一引用符を含んだ値を挿入する。

Oracleの場合

書式 **INSERT INTO 表名 (列 [, ･･･]) VALUES('文字''文字' [, ･･･])**

実行例

```
SQL> INSERT INTO EMP (EMPNO,ENAME) VALUES(8000,'AL''ICE');

1行が作成されました。
```

DB2の場合

書式 **INSERT INTO 表名 (列 [, ･･･]) VALUES('文字''文字' [, ･･･])**

実行例

```
db2=> INSERT INTO EMP (EMPNO,ENAME) VALUES(8000,'AL''ICE')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 文字データを値として挿入する場合は、文字データ全体を単一引用符（'）で囲む必要があります。文字データ中に単一引用符を含む場合は、文字としての単一引用符の前に単一引用符を1つ指定します。

7.14 複数行の挿入

1つのINSERT文で複数行を挿入します。

Oracleの場合

書式
```
INSERT ALL INTO 表名 VALUES(値 [, ･･･])
           INTO 表名 VALUES(値 [, ･･･])
           [, ･･･]
SELECT 'X' FROM DUAL;
```

実行例
```
SQL> INSERT ALL
  2  INTO DEPT VALUES(50,'EDUCATION','TOKYO')
  3  INTO DEPT VALUES(60,'MARKETING','TOKYO')
  4  SELECT 'X' FROM DUAL;

2行が作成されました。
```

DB2の場合

書式
```
INSERT INTO 表名 VALUES(値 [, ･･･] ) , (値 [, ･･･] ) [, ･･･]
```

実行例
```
db2=> INSERT INTO DEPT VALUES(50,'EDUCATION','TOKYO'),(60,'MARKETING','TOK
➥YO')
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 Oracleは、1つのINSERT文で複数行を挿入する場合、問い合わせ文を使用します。ほかの表の値を挿入値に使用する場合はVALUES句に列名を指定し、必要がなければ値を指定します。DB2 UDBは、VALUES句にカンマで区切って複数の挿入値を指定できます。

7.15 行をほかの表にコピーし、列の値によって、コピーする先を変更する

1つのINSERT文を使用し、表の行を複数の表に挿入します。また、挿入する先の表は列の値によって決定します。

Oracleの場合

書式

```
INSERT ALL WHEN 列名 = 値 THEN INTO 表名 VALUES(値 [, ···])
           WHEN 列名 = 値 THEN INTO 表名 VALUES(値 [, ···])
           [, ···]
           ELSE INTO 表名 VALUES(値2 [, ···])
SELECT文
```

実行例

```
SQL> INSERT ALL
  2  WHEN PAYMENT_TYPE = 1 THEN INTO CREDIT VALUES(ORDER_ID,ORDER_DATE,
➥CUST_ID,TOTAL)
  3  WHEN PAYMENT_TYPE = 2 THEN INTO CASH   VALUES(ORDER_ID,ORDER_DATE,
➥CUST_ID,TOTAL)
  4  ELSE     INTO OTHERS VALUES(ORDER_ID,ORDER_DATE,CUST_ID,TOTAL)
  5  SELECT ORDER_ID,ORDER_DATE,CUST_ID,TOTAL,PAYMENT_TYPE FROM ORDERS;

5行が作成されました。
```

DB2の場合

書式

```
WITH S AS (SELECT文)
,CASE1 (N) AS (SELECT 1 FROM FINAL TABLE(
INSERT INTO 表名 SELECT 値 [, ···] FROM S WHERE列名 = 値))
,CASE2 (N) AS (SELECT 2 FROM FINAL TABLE(
INSERT INTO 表名 SELECT 値 [, ···] FROM S WHERE列名 = 値))
···
SELECT 'X' FROM SYSIBM.SYSDUMMY1
```

実行例

注意点

OracleのINSERT ALLでは、WHEN句を使用して挿入する条件を指定できます。
DB2 UDBでは、WHERE句を使用して挿入する条件を指定できます。このとき、すべての条件を評価します。つまり、最初の条件に一致しても、2番目以降の条件も評価し、条件に一致するすべての挿入を行います。

7.16 1行の更新

列値を更新します。

Oracleの場合

書式 **UPDATE 表名 SET 列名 = 値 [WHERE 検索条件]**

実行例

```
SQL> UPDATE EMP SET DEPTNO = 30
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2の場合

書式 **UPDATE 表名 SET 列名 = 値 [WHERE 検索条件]**

実行例

```
db2=> UPDATE EMP SET DEPTNO = 30 WHERE  EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | 既存の値を変更する場合はUPDATE文を使用します。

7.17 特定行の更新

特定の列値だけを更新します。

Oracle の場合

書式 **UPDATE 表名 SET 列名 = 値 [WHERE 検索条件]**

実行例

```
SQL> UPDATE EMP SET SAL = SAL*1.5
  2  WHERE  SAL <= 1000;

2行が更新されました。
```

DB2 の場合

書式 **UPDATE 表名 SET 列名 = 値 [WHERE 検索条件]**

実行例

```
db2=> UPDATE EMP SET SAL = SAL*1.5 WHERE  SAL <= 1000
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 SET句に更新対象となる列を指定します。複数の列を更新対象とする場合は、列をカンマ(,)で区切ります。指定する列の順番は任意です。指定されなかった列の値は更新されません。

7.18 全行の更新

表のすべての行の値を更新します。

Oracleの場合

書式 **UPDATE 表名 SET 列名 = 値**

実行例
```
SQL> UPDATE EMP SET SAL = SAL*1.1;

15行が更新されました。
```

DB2の場合

書式 **UPDATE 表名 SET 列名 = 値**

実行例
```
db2=> UPDATE EMP SET SAL = SAL*1.1
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 UPDATE文では、WHERE句の指定は任意です。WHERE句を省略するとすべての行が更新されます。

7.19 更新対象の行が存在しなかった場合の処理

更新対象となる行が表に存在しなかった場合の処理です。

Oracle の場合

書式 **UPDATE 表名 SET 列名 = 値 WHERE 検索条件**

実行例

```
SQL> UPDATE EMP SET SAL = SAL*1.5 WHERE EMPNO = 9999;

0行が更新されました。
```

DB2 の場合

書式 **UPDATE 表名 SET 列名 = 値 WHERE 検索条件**

実行例

```
db2=> UPDATE EMP SET SAL = SAL*1.5 WHERE EMPNO = 9999
SQL0100W  FETCH、UPDATE または DELETE
の対象となる行がないか、または照会の結果が空の表です。  SQLSTATE=02000
```

注意点 Oracle は、0件更新したというメッセージが戻されるだけで、エラー扱いではありません。DB2 UDB では、対象行が存在しなかった場合または空の表に対して操作した場合は、エラーが戻されます。

7.20 文字データの更新

文字列型の列を更新します。

Oracleの場合

書式 **UPDATE 表名 SET 文字列型の列名 = 値 [WHERE 検索条件]**

実行例

```
SQL> UPDATE EMP SET JOB = 'SE'
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2の場合

書式 **UPDATE 表名 SET 文字列型の列名 = 値 [WHERE 検索条件]**

実行例

```
db2=> UPDATE EMP SET JOB = 'SE' WHERE  EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | 文字列型の列を更新する場合は、値を単一引用符 (') で囲んで指定します。

7.21 数値データの更新

数値型の列を更新します。

Oracle の場合

書式 **UPDATE 表名 SET 数値型の列名 = 値 [WHERE 検索条件]**

実行例
```
SQL> UPDATE EMP SET SAL = 1000
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2 の場合

書式 **UPDATE 表名 SET 数値型の列名 = 値 [WHERE 検索条件]**

実行例
```
db2=> UPDATE EMP SET SAL = 1000 WHERE  EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

数値型の列を更新する場合は、値を単一引用符 (') で囲んでいけません。
Oracle は、単一引用符で囲んでも暗黙的に型変換を行いますが、負荷を軽減するために無駄な型変換が行われないように指定してください。
DB2 UDB は、暗黙型変換しません。

7.22 計算式を使用した数値データの更新

計算した結果を更新値として使用する。

Oracleの場合

書式 **UPDATE 表名 SET 列名 = 式 [WHERE 検索条件]**

実行例

```
SQL> UPDATE EMP SET SAL = SAL*1.2
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2の場合

書式 **UPDATE 表名 SET 列名 = 式 [WHERE 検索条件]**

実行例

```
db2=> UPDATE EMP SET SAL = SAL*1.2 WHERE  EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | SET句の更新する値には、リテラルのほかに、式や列名を指定することができます。

7.23 日付データの更新

日付型の列を更新します。

Oracle の場合

書式 **UPDATE 表名 SET 日付型列名 = 式 [WHERE 検索条件]**

実行例

```
SQL> UPDATE EMP SET HIREDATE = '05-04-01'
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2 の場合

書式 **UPDATE 表名 SET 日付型列名 = 式 [WHERE 検索条件]**

実行例

```
db2=> UPDATE EMP SET HIREDATE = '2005-04-01' WHERE EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

日付型の列を更新する場合は、値を単一引用符 (') で囲み、デフォルトの日付書式に従わなければいけません。
DB2 UDB では、デフォルトの日付のフォーマットとして、'yyyy-mm-dd'、'mm/dd/yyyy'、'dd.mm.yyyy' が使えます。

7.24 書式を指定した日付の更新

デフォルト以外の書式を使用して日付データを更新します。

Oracleの場合

書式

```
UPDATE 表名 SET 日付型列名 = TO_DATE('値','日付書式') [ WHERE 検索条件 ]
```

実行例

```
SQL> UPDATE EMP SET HIREDATE = TO_DATE('05/04/01','YY/MM/DD')
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2の場合

書式

対応する機能はありませんが、式でデフォルトのフォーマットに変換できます。

実行例

```
------------------------------ 入力コマンド ------------------------------
UPDATE EMP
   SET HIREDATE = '20'||TRANSLATE('05/04/01','-','/')
 WHERE EMPNO = 7369;
--------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

Oracleでは、セッション単位に日付書式を変更することができます。SQL文ごとに指定する場合は、TO_DATE関数を使用します。
DB2 UDBは、日付データの更新はできますが、デフォルト以外の書式のデータは式を用いてデフォルト形式に変換します。よく使う書式があれば、デフォルト形式に変換するユーザー定義関数（UDF）を作成しておくとよいでしょう。

7.25 複数列の更新

複数の列を更新します。

Oracle の場合

書式 **UPDATE 表名 SET 列名 = 値, 列名 = 値 [, ･･･] [WHERE 検索条件]**

実行例

```
SQL> UPDATE EMP SET DEPTNO = 30,JOB = 'SE'
  2  WHERE  EMPNO = 7369;

1行が更新されました。
```

DB2 の場合

書式 **UPDATE 表名 SET 列名 = 値, 列名 = 値 [, ･･･] [WHERE 検索条件]**

または

UPDATE 表名 SET (列名[, ･･･]) = (値,[, ･･･]) [WHERE 検索条件]

実行例

```
db2=> UPDATE EMP SET DEPTNO = 30,JOB = 'SE' WHERE  EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 SET句に更新対象となる列を指定します。複数の列を更新対象とする場合は、列をカンマ(,)で区切ります。指定する列の順番は任意です。指定されなかった列の値は更新されません。

7.26 ほかの表の値を元に更新する

ほかの表の値を元に更新します。SET句に副問い合わせを指定します。

Oracleの場合

書式 **UPDATE 表名 SET 列名 = (SELECT 列名 FROM 表名 [WHERE 検索条件])**

実行例

```
SQL> UPDATE EMP SET DEPTNO = (SELECT DEPTNO FROM DEPT WHERE LOC = 'NEW YOR
➥K')
  2  WHERE EMPNO =7369;

1行が更新されました。
```

DB2の場合

書式 **UPDATE 表名 SET 列名[, ･･･] =**
(SELECT 列名[, ･･･] FROM 表名 [WHERE 検索条件])

実行例

```
db2=> UPDATE EMP SET DEPTNO = (SELECT DEPTNO FROM DEPT WHERE LOC = 'NEW YO
➥RK') WHERE EMPNO =7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 SET句に副問い合わせを指定することができます。副問い合わせからは1行だけ戻す必要があります。

7.27 行の削除

表から1行削除します。

Oracleの場合

書式 **DELETE FROM 表名 WHERE 検索条件**

実行例

```
SQL> DELETE FROM EMP WHERE EMPNO = 7369;

1行が削除されました。
```

DB2の場合

書式 **DELETE FROM 表名 WHERE 検索条件**

実行例

```
db2=> DELETE FROM EMP WHERE EMPNO = 7369
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | 行の削除はDELETE文を使用します。WHERE句で削除対象となる行の条件を指定します。

7.28 特定行の削除

特定の条件に一致する複数の行を削除します。

Oracleの場合

書式 **DELETE FROM 表名 WHERE 検索条件**

実行例
```
SQL> DELETE FROM EMP WHERE HIREDATE <= TO_DATE('1981/12/31','YYYY/MM/DD');

11行が削除されました。
```

DB2の場合

書式 **DELETE FROM 表名 WHERE 検索条件**

実行例
```
db2=> DELETE FROM EMP WHERE HIREDATE <= '1981-12-31'
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | 行の削除はDELETE文を使用します。WHERE句で削除対象となる行の条件を指定します。

7.29 すべてのレコードを一括して削除する

表のすべての行を削除する。

Oracle の場合

書式 **DELETE FROM 表名**

実行例

```
SQL> DELETE FROM EMP;

15行が削除されました。
```

DB2 の場合

書式 **DELETE FROM 表名**

実行例

```
db2=> DELETE FROM EMP
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | WHERE 句を指定せずに DELETE 文を実行すると、表のすべての行が削除されます。

7.30 全行を高速に削除する

表のすべての行を高速に削除する。

Oracleの場合

書式 **TRUNCATE TABLE 表名**

実行例

```
SQL> TRUNCATE TABLE EMP;

表が切り捨てられました。
```

DB2の場合

書式 **ALTER TABLE 表名 ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE**
または

●UNIX
IMPORT FROM /dev/null OF DEL REPLACE INTO TBL名
●Windows
IMPORT FROM NUL OF DEL REPLACE INTO TBL名

実行例

```
db2=> ALTER TABLE EMP ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
DB20000I  SQL コマンドが正常に終了しました。

db2=> SELECT COUNT(*) FROM EMP

1
-----------
          0

  1 レコードが選択されました。
```

注意点

Oracleは、TRUNACTEを使用してすべての行を削除することができます。TRUNCATEはUNDO(ROLLBACK) データを作成しないため、DELETEを使用した全行の削除よりも高速に処理されます。
UDB DB2には、TRUNCATE TABLE文はありませんが、IMPORT～REPLACEコマンドを使う方法が一般的です。

7.31 重複する行を削除する

重複した行を削除します。

Oracleの場合

書式

```
DELETE FROM 表名1 表別名1 WHERE ROWID >
(SELECT MIN(ROWID) FROM 表名2 表別名2
WHERE 表別名1.列名 = 表別名2.列名
```

実行例

```
SQL> DELETE FROM DEPT D1
  2  WHERE  ROWID > (SELECT MIN(ROWID) FROM DEPT D2 WHERE  D2.DEPTNO =
➥D1.DEPTNO);

1行が削除されました。
```

DB2の場合

書式

```
DELETE FROM
(SELECT ROWNUMBER() OVER(PARTITION BY キー列)
FROM 表名) AS 表別名(RN)
WHERE RN > 1
```

実行例

7.32 削除対象の行が存在しなかった

削除対象となる行が表に存在しなかった場合の処理です。

Oracleの場合

書式 **DELETE FROM 表名 WHERE 検索条件**

実行例

```
SQL> DELETE FROM EMP WHERE EMPNO = 9999;

0行が削除されました。
```

DB2の場合

書式 **DELETE DEOM 表名 WHERE 検索条件**

実行例

```
db2=> DELETE FROM EMP WHERE EMPNO = 9999
SQL0100W  FETCH、UPDATE または DELETE
の対象となる行がないか、または照会の結果が空の表です。  SQLSTATE=02000
```

注意点 Oracleは、0件削除したというメッセージが戻されるだけで、エラー扱いではありません。
DB2 UDBでは、対象行が存在しなかった場合または空の表に対して操作した場合は、エラーが戻されます。

7.33 トランザクションを確定する

トランザクションを確定（コミット）します。

Oracleの場合

書式 **COMMIT**

実行例

```
SQL> COMMIT;

コミットが完了しました。
```

DB2の場合

書式 **COMMIT**

実行例

```
db2=> COMMIT
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 | COMMITを使用し、トランザクションを明示的に確定終了させることができます。

7.34 トランザクションを破棄する

トランザクションを破棄（ロールバック）します。

Oracleの場合

書式 **ROLLBACK**

実行例
```
SQL> ROLLBACK;

ロールバックが完了しました。
```

DB2の場合

書式 **ROLLBACK**

実行例
```
db2=> ROLLBACK
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 ROLLBACKを使用し、トランザクションを明示的に取り消し（破棄）終了させることができます。

7.35 トランザクションのセーブポイントを設定する

トランザクションのセーブポイントを指定します。

Oracle の場合

書式 **SAVEPOINT セーブポイント名**

実行例

```
SQL> SAVEPOINT S1;

セーブ・ポイントが作成されました。
```

DB2 の場合

書式 **SAVEPOINT セーブポイント名 ON ROLLBACK RETAIN CURSORS**

実行例

```
db2=> SAVEPOINT S1 ON ROLLBACK RETAIN CURSORS
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 セーブポイントを指定します。
DB2 UDB では、ON ROLLBACK RETAIN CURSORS を指定する必要があります。

7.36 トランザクションに複数のセーブポイントを設定する

トランザクション内に複数のセーブポイントを指定します。

Oracleの場合

書式 **SAVEPOINT セーブポイント名**

実行例

```
SQL> INSERT INTO DEPT VALUES(50,'EDUCATION','TOKYO');

1行が作成されました。

SQL> SAVEPOINT S1;

セーブ・ポイントが作成されました。

SQL> INSERT INTO DEPT VALUES(60,'MARKETING','TOKYO');

1行が作成されました。

SQL> SAVEPOINT S2;

セーブ・ポイントが作成されました。
```

DB2の場合

書式 **SAVEPOINT セーブポイント名 ON ROLLBACK RETAIN CURSORS**

実行例

```
db2=> SAVEPOINT S1 ON ROLLBACK RETAIN CURSORS
DB20000I  SQL コマンドが正常に終了しました。
```

注意点 Oracleはトランザクション内で複数のセーブポイントを指定できます。ただし、同じ名前のセーブポイントを指定すると後から指定したセーブポイントのみが有効になります。
DB2 UDBでは、トランザクション内で複数のセーブポイントを指定することはできません。

7.37 トランザクションを指定されたセーブポイントまで戻す

トランザクションを指定したセーブポイントまで破棄（取消）します

Oracleの場合

書式 **ROLLBACK TO セーブポイント名**

実行例

```
SQL> ROLLBACK TO S1;

ロールバックが完了しました。
```

DB2の場合

書式 **ROLLBACK TO SAVEPOINT セーブポイント名**

実行例

```
db2=> ROLLBACK TO SAVEPOINT S1
```

7.38 2つの表データをマージする

表の既存データ行を更新／削除または新規行を追加します。

Oracleの場合

書式

```
MERGE INTO 対象表 USING ソース表 ON 対象表とソース表の関係
WHEN MATCHED THEN UPDATE SET 対象表.列名 = ソース表.列名
                             DELETE WHERE (ソース表.列名 = 値)
WHEN NOT MATCHED THEN INSERT [ (対象表.列名 [, ･･･]) ]
                               VALUES(ソース表.列名 [, ･･･])
```

実行例

```
SQL> MERGE INTO ORDERS O
  2  USING TODAYS_ORDER T ON (O.ORDER_ID = T.ORDER_ID)
  3  WHEN MATCHED THEN
  4   UPDATE SET O.PAYMENT_TYPE=T.PAYMENT_TYPE,O.TOTAL=T.TOTAL
  5   DELETE WHERE (T.TOTAL = 0)
  6  WHEN NOT MATCHED THEN
  7   INSERT (O.ORDER_ID,O.ORDER_DATE,O.CUST_ID,O.PAYMENT_TYPE,O.TOTAL)
  8   VALUES  (T.ORDER_ID,T.ORDER_DATE,T.CUST_ID,T.PAYMENT_TYPE,T.TOTAL);

4行がマージされました。
```

DB2の場合

書式

```
MERGE INTO 対象表
USING ソース表 ON 対象表とソース表の関係
WHEN MATCHED [AND 条件] THEN
  UPDATE SET 対象表.列名 = ソース表.列名
| DELETE
WHEN NOT MATCHED [AND 条件] THEN
 INSERT [ (対象表.列名[, ･･･]) ]
 VALUES(ソース表.列名[, ･･･])
```

実行例

```
------------------------------ 入力コマンド ------------------------------
MERGE INTO ORDERS O
USING TODAYS_ORDER T ON (O.ORDER_ID = T.ORDER_ID)
 WHEN MATCHED AND (T.TOTAL <> 0) THEN
      UPDATE SET O.PAYMENT_TYPE=T.PAYMENT_TYPE,O.TOTAL=T.TOTAL
 WHEN MATCHED AND (T.TOTAL = 0) THEN
      DELETE
 WHEN NOT MATCHED THEN
      INSERT (O.ORDER_ID,O.ORDER_DATE,O.CUST_ID,O.PAYMENT_TYPE,O.TOTAL)
      VALUES (T.ORDER_ID,T.ORDER_DATE,T.CUST_ID,T.PAYMENT_TYPE,T.TOTAL);
------------------------------------------------------------------------
DB20000I  SQL コマンドが正常に終了しました。
```

注意点

MERGE INTOのあとに更新対象となる表を指定します。USING句の次に追加または更新を行う元のソースを記述します。ON句の次に対象となる表とソース表の関係を記述します。WHEN MATCHED THEN以下には、条件式が真であった場合の更新（削除）処理を記述します。WHEN NOT MATCED THEN以下には、条件式が偽であった場合の追加条件を指定します。

機能索引

｜英数字｜

1 文字だけ不明な文字列を検索する 15
2 つの表を総当たりで結合する 240
EXISTS を使用した副問い合わせ 197
FROM 句に使用する問い合わせ 206
NOT EXISTS を使用した副問い合わせ 198
NULL 以外の値を検索する 18
NULL 値か否かを比較する 178
NULL 値を検索する 17
NULL 値を置換する 147
ORDER BY 句を使って結合する 233
ROWID 型に文字列のデータ型を変換する 162
ROWID の値を VARCHAR2 型に変換する 164
WHERE 句で副問い合わせを使う 184

｜あ｜

値の集合から指定した値を求める 85, 94
値の種類数を求める 51
値リストから NULL 値以外の引数を見つける 75
値をグループ化する 55
値を比較する 175
余りを求める 103

｜い｜

いずれかの値に一致する行を検索する 13
いずれかの条件を満たした行を検索する 9

｜お｜

大文字に変換する 114, 158
音声表現に対応する文字列を取得する 109

｜か｜

階層関係にあるデータを検索する 35, 38, 40, 42
掛け算を行う 21
下限値を指定して検索する 11

｜き｜

キャラクター・セットを変換する 163
逆分散関数を求める 181
行数を求める 49, 50
行にバケット番号を割り当てる 161
行番号を表示する 207
行を更新する 347-357
行を削除する 329, 358-363
行を挿入する 332-344
切り捨てる（数値） 133
切り捨てる（日付） 134-136

｜く｜

グループ化された行と超集合行を区別する 89

｜け｜

経過日数を求める 44
結合する（一致しない行も表示） 224, 226, 228, 230
結合する（検索条件も付加する） 232
月末日を求める 169
権限を与える 268-273
権限を一覧する 275
権限を取り消す 274
現在のタイムゾーン日付を取得する 78-80, 139
現在の日時を取得する 148
件数を集計する 77

｜こ｜

合計を求める 54, 112
降順に表示する 30
コメントをつける（表） 277
コメントをつける（列） 278
小文字に変換する 100, 160

｜さ｜

最小値を求める 53, 102
最初の文字だけ明確なあいまい検索 14
最大値を求める 52, 101
索引を監視する 316
索引を削除する 313
索引を作成する 310-312
索引を調査する 317, 318
索引を変更する 314, 315
三角関数の値を求める 116-125
参照整合性制約を活用する 296, 297

｜し｜

自己結合する 220
四捨五入する（数値） 128
四捨五入する（日付） 129, 130, 132
システム日付を取得する 148
自然対数の n 乗を求める 137
自然対数を求める 138
指定した行のみグループ化 59

指定した件数だけ表示する 24
指定した列の値が最大値である 67
シノニムを削除する 328
シノニムを作成する 326, 327
集約関数で求めた結果から検索する 56
順序を削除する 325
順序を作成する 319
順序を使用する 321-324
順序を変更する 320
上位5件までのデータを表示する 209
小計と総計を求める 60, 62
小計を求める 64
上限値を指定して検索する 12
条件と値を指定して比較する 81
昇順に表示する 29
書式を変換する 165, 166
シングルバイトに変換する 167
シングルバイトをマルチバイトに変換する 149

| す |

数値データを検索条件に指定する 5
数値に変換する 166
スキーマにアクセスできるユーザーを一覧する 276
すべての行を削除する 329, 360, 361
すべての列を表示する 2

| せ |

制約を削除する 281, 283, 285, 287, 289, 290
制約を追加する 282, 284, 286, 288, 291
制約を定義する 292-294, 295
制約を定義する（表制約） 280
制約を定義する（列制約） 279
セッション情報を取得する 180
セッション・ユーザー名を取得する 179
絶対値を求める 70

| そ |

相関副問い合わせを使用する 203-205

| た |

対数を求める 145
タイムゾーンを変換する 86, 173
足し算を行う 19
単一行副問い合わせを実行する 186
単一引用符（'）を検索する 45
単語の先頭を大文字にする 155

| ち |

重複した行を取り除いて表示する 58
重複している行の数を求める 57
重複するものを除外する 87
重複を排除して表示する 25

| つ |

月数を求める 171
月単位の足し算を行う 168
次の指定曜日の日付を求める 172

| て |

データ型を変換する 154, 162
データ型を変更する 164
デフォルト値を設定する 246, 247

| と |

問い合わせ結果を縦に結合する 235, 236, 238, 239
等価結合を行う 212, 214, 216
特定の列だけ表示する 3
トランザクションのセーブポイントを設定する 366, 367
トランザクションを確定する 364
トランザクションを指定されたセーブポイントまで戻す 368
トランザクションを破棄する 365

| ね |

年次単位の集計を行う 66

| は |

範囲を指定して検索する 10

| ひ |

引き算を行う 20
引数リストの最小値を求める 170
引数リストの最大値を求める 174
日付形式を変換する 176, 177
日付データを検索する 4, 83
日付に変換する 165
ビットに対するAND演算を行う 153
ビット・ベクトルを数値に変換する 152
ビット・ベクトルを戻す 90
否定演算子を使って検索する 7
非等価結合を行う 218
ビューから行を削除する 305
ビューから列を更新する 304

ビューを結合する 308
ビューを検索する 302
ビューを再作成する 306
ビューを削除する 307
ビューを作成する 298-301
ビューを挿入する 303
標準分散を求める 151
表の一部を別の表と結合する 222
表の一覧を取得する 244
表のコピーをする 248-251
表の再編成を行う 266
表の定義だけコピーする 252
表の定義を確認する 267
表の名前を変更する 255, 256
表の別名を使用する 26
表の列情報を取得する 264
表を移動する 265
表を削除する 253, 254
表を作成する 245
表をマージする 369

ふ

複数行を戻す副問い合わせを行う 187
複数行を挿入する 345, 346
複数列を指定して並べ替える 31, 32
複数列を指定して副問い合わせを行う 199, 201
複数の条件を指定して検索する 8
副問い合わせを行う 184, 186-191, 193-195, 197-199, 201
符合を求める 106
物理オフセットを使って行にアクセスする 92, 95

へ

平方根を求める 140
平均を求める 48, 141
べき乗を求める 104

め

命名規則に違反した列名を使う 28

も

文字コードを文字に変換する 73, 74
文字データを検索条件に指定する 6
文字の10進表記を求める 71
文字のASCII表記を求める 72
文字列中のパーセント(%)を検索する 46
文字列の位置を求める 142-144
文字列のデータ型を変換する 162, 165
文字列の長さを求める 96, 97, 150
文字列を連結する 23, 76
文字列を埋める 156, 157
文字列を抜き出す 110, 111
文字列を変換する 100, 158-160
文字を切り捨てる 98, 107
文字を置換する 105, 113
最も小さい整数を求める 126
最も大きい整数を求める 127

ゆ

ユーザーを一覧する 276
ユーザー名を取得する 115
ユニークな行を表示する 58

よ

予約語を使った列名を使う 28

ら

ランク付けされたデータを操作する 84, 93
ランクを求める 82

れ

列に未使用マークを付ける 259
列の桁数を変更する 261, 262
列の情報を取得する 264
列のデータ型を変更する 263
列番号を使用して並べ替える 33
列別名を使用する 27, 34
列名を変更する 260
列を削除する 258
列を追加する 257

わ

ワイルドカードを使用して検索する 14-16
割り算を行う 22

キーワード索引

記号

– (引き算) 20
% (パーセント) 46
% (ワイルドカード) 14, 16
* 2
* (掛け算) 21
/ (割り算) 22
+ (足し算) 19
' (単一引用符) 45
" (二重引用符) 28
_ (ワイルドカード) 15
|| 23

A

ABS 70
ACOS 116
ADD CONSTRAINT 282, 284, 286, 288, 292, 293
ADD_MONTH 168
ALL 190, 193, 194
ALTER INDEX 314, 315
ALTER SEQUENCE 320
ALTER TABLE 256-263, 265, 266, 281-297
AND 8
ANY 189, 191, 195
AS 27, 28
ASCII 71
ASCIISTR 72
ASIN 117
ATAN 118
ATAN2 119
AVG 48, 141

B

BETWEEN～AND 10, 218
BITAND 153

C

CASCADE CONSTRAINTS 254
CAST 154
CEIL 126
CHARTOROWID 162
CHECK 286
CHR 73
COALESCE 75, 147, 315
COMMENT ON COLUMN 278
COMMENT ON TABLE 277
COMMIT 364
CONCAT 76
CONNECT BY 35, 38, 40, 42
CONSTRAINT 281, 283
CONVERT 163
COS 120
COSH 121
COUNT 49, 50, 51, 77
CREATE ALIAS 326
CREATE INDEX 311, 312
CREATE OR REPLACE 306
CREATE PUBLIC SYNONYM 327
CREATE SEQUENCE 319
CREATE SYNONYM 326
CREATE TABLE 245, 246, 279, 280
CREATE TABLE AS SELECT 248-252
CREATE UNIQUE INDEX 310
CREATE VIEW 298-301
CROSS JOIN 240
CUBE 62
CURRENT_DATE 78
CURRENT_TIMESTAMP 79
CURRVAL 322

D

DBTIMEZONE 80
DECODE 81
DEFAULT 246, 247
DELETE 269, 296, 358-360, 362, 363
DENSE_RANK 82
DESC 30, 267
DISABLE CONSTRAINT 294
DISTINCT 25, 51
DROP COLUMN 258
DROP CONSTRAINT 285, 287, 289
DROP INDEX 313
DROP PRIMARY KEY 281, 283
DROP SEQUENCE 325
DROP SYNONYM 328
DROP TABLE 253, 254
DROP VIEW 307
DUAL 44

E

ENABLE CONSTRAINT 295
ESCAPE 46
EXISTS 197
EXP 137
EXTRACT 83

| F |

FETCH FIRST ROWS ONLY 24, 209
FIRST 84
FIRST_VALUE 85
FLOOR 127
FOREIGN 288
FROM_TZ 86
FROM句 206
FULL OUTER JOIN 228

| G |

GRANT 268-273
GREATEST 174
GROUP BY 55, 59, 66
GROUP_ID 87
GROUPING 89
GROUPING SETS 64
GROUPING_ID 90

| H |

HAVING 56-58

| I |

IN 13, 187, 197
INITCAP 155
INSERT 269, 270, 303, 323, 332-344, 369
INSERT ALL 345, 346
INSTR 142
INSTRB 143, 144
INTERSECT 239
IS NOT NULL 18
IS NULL 17

| J |

JOIN 212, 214, 220, 222, 232

| L |

LAG 92
LAST 93
LAST_DAY 169
LAST_VALUE 94
LEAD 95
LEAST 170
LEFT OUTER JOIN 224, 230
LENGTH 96
LENGTHB 97
LIKE 14-16, 45, 46
LN 138
LOCALTIMESTAMP 139
LOG 145
LOWER 100
LPAD 156
LTRIM 98

| M |

MAX 52, 67, 101
MEDIAN 181
MERGE 369
MIN 53, 102
MINUS 238
MOD 103
MODIFY 261-263, 290
MONITORING USAGE 316
MONTHS_BETWEEN 171
MOVE TABLESPACE 265, 266
MTK 70

| N |

NATURAL JOIN 212
NCHR 74
NEW_TIME 173
NEXT_DAY 172
NEXTVAL 321, 323, 324
NLS_INITCAP 158
NLS_LOWER 160
NLS_SORT 158
NLS_UPPER 159
NOT 7
NOT DEFERRABLE 293
NOT EXISTS 198
NOT NULL 290
NTILE 161
NULL 17, 18, 340
NULLIF 175
NUMTODSINTERVAL 176
NUMTOYMINTERVAL 177
NVL 147
NVL2 178

| O |

ON DELETE SET NULL 297
OR 9
ORDER BY 29-34, 233

| P |

POWER 104
PRIMARY KEY 279, 280

PUBLIC SYNONYM 327

| R |

RAWTOHEX 146
REBUILD 314
REFERENCES 296
RENAME 255
RENAME COLUMN 260
REPLACE 105
REVOKE 274
RIGHT OUTER JOIN 226
ROLLBACK 365, 368
ROLLUP 60
ROUND 128-130, 132
row_number() over() 207
ROWID 362
ROWIDTOCHAR 164
ROWNUM 24, 207, 209
RPAD 157
RTRIM 107

| S |

SAVEPOINT 366-368
SELECT 2-35, 38, 40, 42, 44-46
SET 204
SET UNUSED COLUMN 259
SIGN 106
SIN 122
SINH 123
SOUNDEX 109
SQRT 140
SUBSTR 110
SUBSTRB 111
SUM 54, 112
SYNONYM 326, 328
SYSDATE 148, 247
SYSIBM.SYSDUMMY1 44

| T |

TAN 124
TANH 125
TO_DATE 165, 355
TO_MULTI_BYTE 149
TO_NUMBER 166
TO_SINGLE_BYTE 167
TRANSLATE 113
TRUNC 133-136
TRUNCATE 361
TRUNCATE TABLE 329

| U |

UID 179
UNION 235
UNION ALL 236
UNIQUE 284
UPDATE ... 204, 205, 269, 271, 304, 324, 347-357, 369
UPPER 114
USER 115
USER_IND_COLUMNS 318
USER_INDEXES 317
USER_TAB_COLUMNS 264
USER_TAB_PRIVS_RECD 275
USER_TABLES 244
USERENV 180
USING 214, 216

| V |

VARIANCE 151
VIEW 272
VSIZE 150

| W |

WHERE 184

| い |

一意索引 310

| え |

エイリアス 326
エスケープ文字 46
演算 19-22

| お |

オブジェクト権限 268-274

| か |

カウント 49-51
外部結合 224, 226, 228, 230

| き |

切り捨て 329

| く |

グループ化 55

｜け｜

結合 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 233, 240, 308
検索 2-12, 14-18

｜こ｜

合計 54

｜さ｜

最大値 52
最小値 53
索引 310-312, 317, 318
索引の監視 316
索引の再構築 314
索引の再編成 315
索引の削除 313
サンプルUDF 70

｜し｜

シーケンス　→順序
集合値 56
小計と総計 60
シノニム 326, 328
順序 319-325

｜せ｜

制約 279-297
絶対値 70

｜そ｜

上位n件 209
相関副問い合わせ 203-205
連番表示 207

｜た｜

第1グループの小計 62
第2グループの小計 62
単一引用符 45

｜ち｜

直積 242

｜と｜

トランザクション 364-368

｜な｜

並び替え 29, 30-34

｜は｜

パブリック・シノニム 327

｜ひ｜

非一意索引 311
日付計算 44
ビュー 298-308

｜ふ｜

副問い合わせ 186-191, 193-195, 197-199, 201, 203
複数グループごとの小計 64
複数列索引 312, 318
分散 151, 181

｜へ｜

平均値 48, 141
別名 26, 27

｜よ｜

予約語 28

｜れ｜

連結 23, 76

Oracle開発者のためのDB2 UDB SQLリファレンス

今すぐ使える実例集

2006年1月5日発行