DB2® ユニバーサル・データベース

SQL 概説

バージョン 7

SC88-8539-00
(英文原典：SC09-2973-00)

DB2® ユニバーサル・データベース

SQL 概説

バージョン 7

SC88-8539-00
(英文原典：SC09-2973-00)

ご注意!

本書、および本書がサポートする製品をご使用になる前に、 123ページの『付録C. 特記事項』にある一般的な情報を必ずお読みください。

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミング、またはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミング、またはサービスを、日本で発表する意図があることを必ずしも示すものではありません。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

原 典:	SC09-2973-00 IBM® DB2® Universal Database SQL Getting Started Version 7
発 行:	日本アイ・ビー・エム株式会社
担 当:	ナショナル・ランゲージ・サポート

第1刷 2000.6

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

序文

このマニュアルでは、構造化照会言語 (SQL) と関係データベースについて紹介します。内容は、次のとおりです。

- DB2 製品で使用する SQL の基本概念。
- データベース操作のさまざまな作業を実行する方法。
- さまざまな作業の簡単な例。

システム管理者としてこのマニュアルに載せられている例を実行するためには、次のことが必要です。

- オペレーティング・システム別の *概説およびインストール* の説明に従って、サーバーをインストールおよび構成してください。さらに、″First Steps″ オプションを使用して、サンプル・データベース (SAMPLE) を作成します。このサンプル・データベースは、コマンド行プロンプトから作成することもできます。詳細については、*SQL 解説書* を参照してください。注: DB2 サンプル・データベース (SAMPLE) の中には、独自のデータを入れないようにしてください。
- *概説およびインストール* の説明に従って、DB2 の管理担当者のユーザー ID を作成してください。

システム管理者でない場合は、有効なユーザー ID と、サンプル・データベースにアクセスするために必要な権限を取得しておいてください。

関連資料

次のマニュアルも役に立つかもしれません。

概説およびインストール	データベース・マネージャーをインストールして使用するために必要な情報が載せられています。
SQL 解説書	SQL についての参照情報が載せられています。
管理の手引き	ローカルに、またはクライアント / サーバー環境でアクセスするデータベースの設計、実装、運用のために必要な情報が載せられています。
アプリケーション開発の手引き	アプリケーション開発のプロセスについて説明し、組み込み SQL を使ってデータベースにアクセスするアプリケーション・プログラム、または SQL プロシージャー言語を使用して DB2 ストアード・プロシージャーとして実行するアプリケーション・プログラムをコーディング、コンパイル、および実行する方法について解説します。

強調表示の表記規則

このマニュアルでは、次のような表記規則を使っています。

ボールド (Bold)	例の中で、システムによってあらかじめ定義されているコマンドとキーワードを示します。
イタリック (Italics)	次のいずれかです。 • 新しい用語が登場した場合 • 参照先として別の資料を示す場合
英大文字 (UPPERCASE)	次のいずれかです。 • システムによってあらかじめ定義されているコマンドとキーワード • 特定のデータ値または列名の例

第1章 関係データベースと SQL

関係データベース では、データが*表* の形で保管されます。表は、*行* と*列* の集まりです。表の例については、4ページの図1 を参照してください。その図では、列 (縦) と行 (横) が図示されています。列や表を指定したり、それらの間のさまざまな関係を指定したりすることによってデータを取り出したり更新したりするには、*構造化照会言語 (SQL)* を使います。

SQL は、関係データベースに含まれているデータを定義したり操作したりするために使う標準化された言語です。 SQL のステートメントは、データベース・マネージャー によって実行されます。データベース・マネージャーは、データを管理するためのコンピューター・プログラムです。

区分 関係データベースは、データを複数の区分 (*ノード* ともいう) に分けて管理する関係データベースです。 それぞれの区分を物理的なコンピューターだと考えれば、分かりやすいかもしれません。このマニュアルでは、主として単一区分データベースを取り上げます。

サンプル・データベースにアクセスしたり、このマニュアルに含まれている例を試したりするために、コマンド行プロセッサー (CLP) やコマンド・センター (CC) のようなインターフェースによる*対話式* SQL を使う方法もあります。

第2章 データの編成

この章では、*表*、*視点*、およびスキーマ に関する重要な概念を説明します。これは、関係データベースを構成するさまざまな要素の間の結び付きを示す全体概要です。最後の部分では、いくつかの重要な、そしてよく使われるデータ・タイプについて簡単に説明します。

表

表は、決まった数の*列* と可変数の*行* とで構成される論理的な構造です。列は、同じデータ・タイプの値の集合です。行は、表の中の 1 つのレコードを構成する値の集合です。 1 つの表の中で、行は必ずしも一定の順序に並んではいません。結果セットを一定の順序にするには、表からデータを選択する SQL ステートメントの中で、順序を明示的に指定する必要があります。任意の列と任意の行の交差位置は、特定のデータ項目であり、*値* と呼ばれます。 4ページの図1 に示す表の中で、'Sanders' は値の一例です。

基本表 はユーザー・データを入れるものであり、 CREATE TABLE ステートメントによって作成します。*結果表* とは、照会への応答としてデータベース・マネージャーが、1 つまたは複数の基本表から選択または生成した行の集合のことです。

4ページの図1 に、表の一部の例を示します。どれが列でどれが行かが示されています。

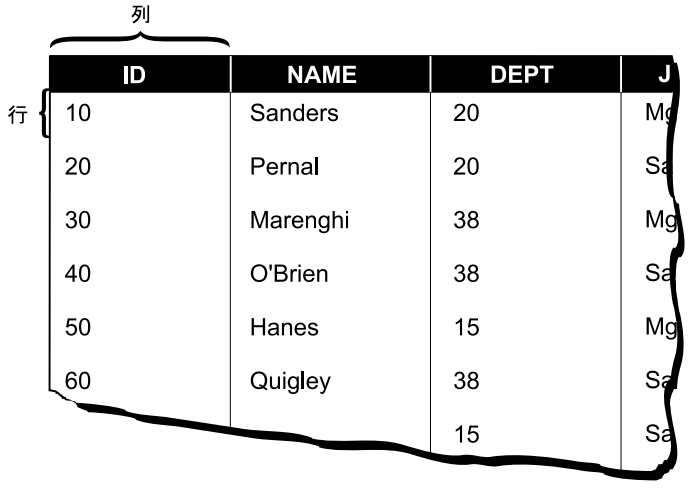


図 1. 表を図示したもの

視点

視点 は、 1 つまたは複数の表のデータを調べるための代替手段です。これは、表を見るための動的な窓口です。

視点を使うと、同じデータでも、ユーザーごとに違う仕方で表示されるようにすることができます。たとえば、従業員に関するデータを含む表に、何人ものユーザーがアクセスするかもしれません。マネージャーは自分の部下のデータを見ますが、別の部門の従業員のデータは見ません。人事部長はすべての従業員の採用日付を見ますが、給与は見ません。それに対して、経理部長は採用日付ではなく給与を見ます。それらのユーザーは、それぞれ実際の表から派生した視点を使って作業をします。どの視点も、表によく似ており、独自の名前が付けられています。

視点を使うことの 1 つの利点は、機密データへのアクセスを制御できるというところにあります。データのうちアクセスできる列や行を、ユーザーごとに変えることができます。

スキーマ

スキーマ は、名前の付けられたオブジェクト (表や視点など) の集まりであり、データベースの中のオブジェクトを論理的に分類するものです。

スキーマは、表や視点など、名前の付いたオブジェクトを作成するときに、暗黙的に作成されます。あるいは、CREATE SCHEMA ステートメントを使って明示的に作成することもできます。

名前の付いたオブジェクトを作成するとき、オブジェクトの名前を特定のスキーマの名前で*修飾* することができます。言い換えれば、オブジェクトの名前とスキーマの名前を関連付けるということです。オブジェクトの名前は、2 つの部分からなっています。その第 1 の部分は、そのオブジェクトの割り当てられるスキーマ名です。スキーマ名を指定しない場合、オブジェクトは、デフォルトのスキーマに割り当てられます。 (デフォルト・スキーマの名前は、ステートメントを実行するユーザーの*許可 ID* になります。)

対話式 SQL でこのマニュアルに載せられている例を実行する場合、許可 ID は、CONNECT ステートメントで指定するユーザー ID です。たとえば、表の名前が STAFF で、ユーザー ID として USERXYZ を指定した場合、表の修飾名は USERXYZ.STAFF になります。 CONNECT ステートメントについては、 18ページの『データベースへの接続』を参照してください。

スキーマ名の中には、予約済みのものがあります。たとえば、*組み込み関数* は SYSIBM スキーマのものであり、あらかじめインストールされているユーザー*定義関数* は SYSFUN スキーマのものです。 CREATE SCHEMA ステートメントについては、 *SQL 解説書* を参照してください。

データ・タイプ

データ・タイプは、定数、列、ホスト変数、関数、式、および特殊レジスターの値として受け入れ可能なものを定義します。ここでは、例の中で参照されているデータ・タイプについて説明します。その他のすべてのデータ・タイプのリストと詳しい説明については、 *SQL 解説書* を参照してください。

文字ストリング

文字ストリング は、バイトの列です。ストリングの長さは、その列の中のバイト数です。長さが 0 なら、その値は空ストリング と呼ばれます。

固定長文字ストリング

CHAR(x) は、固定長ストリングです。長さ属性 x は、1 以上 254 以下でなければなりません。

可変長文字ストリング

可変長文字ストリングには、VARCHAR、LONG VARCHAR、および CLOB の 3 種類があります。

VARCHAR(x) データ・タイプは可変長ストリングなので、長さ 9 のストリングを VARCHAR(15) に挿入しても、そのストリング長はやはり 9 です。

CLOB については、 73ページの『ラージ・オブジェクト (LOB)』を参照してください。

グラフィック・ストリング

グラフィック・ストリング は、2 バイト文字の列です。

固定長グラフィック・ストリング

GRAPHIC(x) は、固定長ストリングです。長さ属性 x は、1 以上 127 以下でなければなりません。

可変長グラフィック・ストリング

可変長グラフィック・ストリングには、VARGRAPHIC、LONG VARGRAPHIC、および DBCLOB の 3 種類があります。DBCLOB については、 73ページの『ラージ・オブジェクト (LOB)』を参照してください。

バイナリー・ストリング

バイナリー・ストリング は、バイトの列です。これは、画像データなど、従来のタイプに当てはまらないデータを入れるのに使います。バイナリー・ラージ・オブジェクト (BLOB) は、バイナリー・ストリングです。詳しい情報については、 73ページの『ラージ・オブジェクト (LOB)』を参照してください。

数値

数値には、すべて符号と*精度* があります。精度とは、ビット数または桁数のことです (符号を含む)。

SMALLINT
SMALLINT (短整数) は、精度が 5 桁の 2 バイト整数です。

INTEGER
INTEGER (長整数) は、精度が 10 桁の 4 バイト整数です。

BIGINT
BIGINT (大整数) は、精度が 19 桁の 8 バイト整数です。

REAL *REAL (単精度浮動小数点数)* は、実数の 32 ビット近似値です。

DOUBLE

DOUBLE (倍精度浮動小数点数) は、実数の 64 ビット近似値です。 DOUBLE は、FLOAT と書かれることもあります。

DECIMAL(p,s)

DECIMAL は、10 進数です。 10 進小数点の位置は、数値の*精度 (p)* と*位取り (s)* によって決まります。精度は、総桁数であり、32 未満でなければなりません。位取りは、小数点以下の桁数であり、常に精度値以下でなければなりません。精度と位取りを指定しない場合の 10 進数値のデフォルトは、精度 5、位取り 0 です。

日付 / 時刻値

日付 / 時刻値は、日付、時刻、およびタイム・スタンプ (*yyyyxxddhhmmss* という形式で有効な日付と時刻を表す 14 桁の文字ストリング) を表します。日付 / 時刻値は、特定の算術演算やストリング操作で使うことができ、特定のストリングとの互換性がありますが、日付 / 時刻値はストリングでも数値でもありません。[1]

Date *DATE (日付)* は、年、月、および日の 3 つの部分で構成される値です。

Time *TIME (時刻)* は、時、分、秒の 3 つの部分で構成されており、 24 時間制によって表される時刻を指します。

Timestamp

TIMESTAMP (タイム・スタンプ) は、年、月、日、時、分、秒、およびマイクロ秒の 7 つの部分で構成される値であり、1 つの日時を指定します。

NULL 値

NULL 値は、 NULL 以外のすべての値とは区別された特殊な値です。これは、その行のその列に、その他の値がないことを意味します。NULL 値は、どのデータ・タイプについても存在します。

次の表に、例で使用されるデータ・タイプの特性をまとめておきます。すべての数値データ・タイプは、特定の範囲内のものとして定義されています。この表には、数値データ・タイプの範囲も示されています。この表は、データ・タイプの正しい使い方について調べるのに使うことができます。

1. このマニュアルでは、日付 / 時刻値の ISO 表記を使います。

データ・タイプ	種類	特性	例または範囲
CHAR(15)	固定長文字ストリング	最大長 254	'Sunny day'
VARCHAR(15)	可変長文字ストリング	最大長 32672	'Sunny day'
SMALLINT	数値	長さ 2 バイト、精度 5 桁	範囲は -32768～32767
INTEGER	数値	長さ 4 バイト、精度 10 桁	範囲は -2147483648～2147483647
BIGINT	数値	長さ 8 バイト、精度 19 桁	範囲は -9223372036854775808～9223372036854775807
REAL	数値	単精度浮動小数点 32 ビット近似値	範囲は -3.402E+38～-1.175E-37、または 1.175E-37～-3.402E+38、またはゼロ
DOUBLE	数値	倍精度浮動小数点 64 ビット近似値	範囲は -1.79769E+308～-2.225E-307、または 2.225E-307～1.79769E+308、またはゼロ
DECIMAL(5,2)	数値	精度 5、位取り 2	範囲は -10**31+1～10**31-1
DATE	日付 / 時刻	3 つの部分で構成される値	1991-10-27
TIME	日付 / 時刻	3 つの部分で構成される値	13.30.05
TIMESTAMP	日付 / 時刻	7 つの部分で構成される値	1991-10-27-13.30.05.000000

詳細については、 *SQL 解説書* の中のデータ・タイプ互換性の表を参照してください。

第3章 表と視点の作成

この章では、DB2 ユニバーサル・データベースの表と視点を作成したり操作したりする方法について説明します。また、表と視点の間の関係を図と例によって示します。

この章の内容は、次のとおりです。

- 表の作成と視点の作成
- データの挿入
- データの変更
- データの削除
- 視点を使用したデータ操作

表の作成

独自の表を作成するには、CREATE TABLE ステートメントを使い、列の名前とタイプ、そして*制約* を指定します。制約については、 55ページの『制約とトリガーによる業務規則の適用』を参照してください。

次のステートメントでは、PERS という表が作成されます。これは STAFF 表によく似ていますが、誕生日の列 (BIRTH_DATE) が追加されています。

```
CREATE TABLE          PERS
    ( ID              SMALLINT        NOT NULL,
      NAME            VARCHAR(9),
      DEPT            SMALLINT WITH DEFAULT 10,
      JOB             CHAR(5),
      YEARS           SMALLINT,
      SALARY          DECIMAL(7,2),
      COMM            DECIMAL(7,2),
      BIRTH_DATE      DATE)
```

このステートメントでは、データが何も含まれていない表が作成されます。新しい表の中にデータを挿入する方法については、次の節で説明します。

この例に示されているように、列ごとにそれぞれ名前とデータ・タイプの両方を指定します。データ・タイプについては、 5ページの『データ・タイプ』を参照してください。 NOT NULL はオプションであり、列の値が NULL であってはならないことを指定します。省略時値 (DEFAULT) もオプションです。

CREATE TABLE に指定できるオプションには、ほかにもたくさんあります。その中には、*固有制約* や*参照制約* が含まれます。すべてのオプションについては、 *SQL 解説書* の中の CREATE TABLE ステートメントの説明を参照してください。

データの挿入

新しい表を作成した時点では、その表には何もデータが含まれていません。表に新しい行を挿入するには、INSERT ステートメントを使います。このステートメントには、次の 2 種類の形式があります。

- 1 つの形式は、VALUES 文節を使って、 1 つまたは複数の行の列の値を指定するものです。この後の 3 つの例では、この形式を使って表にデータを挿入しています。
- もう 1 つの形式は、VALUES を使わず、*全選択* を指定して、他の表や視点に含まれている行から選んだ列を指定します。

全選択とは、 INSERT または CREATE VIEW ステートメントの中で、または述部の後で使われている SELECT ステートメントのことです。括弧で囲まれている全選択は、一般に*副照会* と呼ばれます。

表作成時に選んだデフォルト・オプションに応じて、挿入するどの行についても、列ごとに値を指定するか省略時値を受け入れるかのどちらかになります。さまざまなデータ・タイプの省略時値については、 *SQL 解説書* を参照してください。

次のステートメントでは、VALUES 文節を使って、PERS 表に 1 行分のデータを挿入しています。

```
INSERT INTO PERS
   VALUES (12, 'Harris', 20, 'Sales', 5, 18000, 1000, '1950-1-1')
```

次のステートメントでは、VALUES 文節を使って、ID、NAME、および JOB しかわかっていない 3 行を PERS 表に挿入しています。列が NOT NULL として定義されており、その列に省略時値がないなら、その列には必ず値を指定しなければなりません。

CREATE TABLE ステートメントの列定義の中の NOT NULL 文節には、WITH DEFAULT 句を指定できます。列が NOT NULL WITH DEFAULT として定義されているか、または WITH DEFAULT 10 のように定数の省略時値が指定されている場合、列リストの中にその列を指定しないなら、挿入する行の

その列には省略時値が挿入されます。たとえば、CREATE TABLE ステートメントにおいて、DEPT 列についてのみ省略時値が指定されており、それが 10 だったとします。その場合、部署番号 (DEPT) は 10 に設定され、明示的に値を指定するその他のすべての列は NULL に設定されます。

```
INSERT INTO PERS (NAME, JOB, ID)
   VALUES ('Swagerman', 'Prgmr', 500),
          ('Limoges', 'Prgmr', 510),
          ('Li', 'Prgmr', 520)
```

ここまでの挿入結果は、次のステートメントによって調べることができます。

```
SELECT *
   FROM PERS

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM      BIRTH_DATE
------ --------- ------ ----- ------ --------- --------- ----------
    12 Harris        20 Sales      5  18000.00   1000.00 01/01/1950
   500 Swagerman     10 Prgmr      -         -         - -
   510 Limoges       10 Prgmr      -         -         - -
   520 Li            10 Prgmr      -         -         - -
```

この場合は、すべての列に値が指定されたわけではありませんでした。 NULL 値はダッシュ (-) として表示されています。これが機能するためには、列名のリストが、順序の点でもデータ・タイプの点でも、VALUES 文節に指定されている値に対応していなければなりません。最初の例の場合のように列名のリストを省略する場合、 VALUES の後のデータ値のリストの順序は挿入先の表の列の順序と同じでなければならず、値数は表の列数と同じでなければなりません。

どの値も、挿入先の列のデータ・タイプとそれぞれ互換性のあるものでなければなりません。 NULL 可能として定義されている列の値が指定されていないなら、挿入される行のその列の値は NULL になります。

次の例では、行の中の YEARS、COMM、および BIRTH_DATE 列に値が指定されていないため、それらの列に NULL 値が挿入されます。

```
INSERT INTO PERS (ID, NAME, JOB, DEPT, SALARY)
   VALUES (410, 'Perna', 'Sales', 20, 20000)
```

INSERT ステートメントの第 2 の形式は、別の表の行の値を使って表にデータを入れる場合にたいへん便利なものです。前述のように、この場合には VALUES を使わず、全選択を指定して、他の表や視点に含まれている行から選んだ列を指定します。

次の例では、 STAFF 表から部署 38 のメンバーのデータを選択し、それを PERS 表に挿入しています。

```
INSERT INTO PERS (ID, NAME, DEPT, JOB, YEARS, SALARY)
   SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
      FROM STAFF
      WHERE DEPT = 38
```

この挿入操作を実行した後で次の SELECT ステートメントを実行すると、その結果は INSERT ステートメントの全選択と同じになります。

```
SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
   FROM PERS
   WHERE DEPT = 38
```

結果は、次のとおりです。

```
ID     NAME      DEPT   JOB   YEARS  SALARY
------ --------- ------ ----- ------ ---------
    30 Marenghi      38 Mgr        5  17506.75
    40 O'Brien       38 Sales      6  18006.00
    60 Quigley       38 Sales      -  16808.30
   120 Naughton      38 Clerk      -  12954.75
   180 Abrahams      38 Clerk      3  12009.75
```

データの変更

表の中のデータを変更するには、UPDATE ステートメントを使います。このステートメントを使うことによって、 WHERE 文節の検索条件を満たす行ごとに 1 つまたは複数の列の値を変更できます。

次の例では、ID が 410 の従業員についての情報を更新しています。

```
UPDATE PERS
   SET JOB='Prgmr', SALARY = SALARY + 300
   WHERE ID = 410
```

SET 文節は、更新の対象となる列とその値を指定します。

WHERE 文節はオプションであり、更新の対象となる行を指定します。
WHERE 文節を省略した場合、データベース・マネージャーは、表または視点の中のすべての行を、指定した値に更新します。

この例の場合、まず表 (PERS) を指定し、次に更新の対象となる行を決めるための条件を指定しています。従業員番号 (ID) 410 に対応する情報を変更します。その従業員の肩書き (JOB) を 'Prgmr' に変更し、その給与 (SALARY) を $300 アップします。

複数の行に適用される WHERE 文節を含めることによって、複数の行を一度に変更することができます。次の例では、すべての営業マン (Sales) の給与を 15% アップしています。

```
UPDATE PERS
   SET SALARY = SALARY * 1.15
   WHERE JOB = 'Sales'
```

データの削除

WHERE 文節で指定した検索条件に基づいて表からデータ行を削除するには、DELETE ステートメントを使います。次の例では、従業員 ID が 120 の行を削除しています。

```
DELETE FROM PERS
   WHERE ID = 120
```

WHERE 文節はオプションであり、削除の対象となる行を指定します。WHERE 文節を省略した場合、データベース・マネージャーは、表または視点の中のすべての行を削除します。

DELETE ステートメントを使って、複数の行を一度に削除できます。次の例では、DEPT が 20 の従業員の行をすべて削除しています。

```
DELETE FROM PERS
   WHERE DEPT = 20
```

行を削除すると、特定の列値ではなくその行全体が削除されます。

表の内容を削除するだけでなくその定義も削除する場合は、 DROP TABLE ステートメントを使います。これについては、 *SQL 解説書* を参照してください。

視点の作成

4ページの『視点』で説明したとおり、視点は、1 つまたは複数の表のデータを調べるための代替手段です。視点を作成するなら、さまざまなユーザーから見える情報をユーザーごとに制限することができます。次の図に、視点と表の間の関係を示します。

14ページの図2 において、 View_A は、Table_A のうち AC1 と AC2 の 2 つの列しかアクセスできないようにしたものです。

View_AB は、Table_A の AC3 と Table_B の BC2 にアクセスできるようにしたものです。

View_A を作成することによって、 TABLE_A へのユーザー・アクセスを制限できます。また VIEW_AB を作成することによって、アクセスを両方の表の特定の列だけに制限できます。

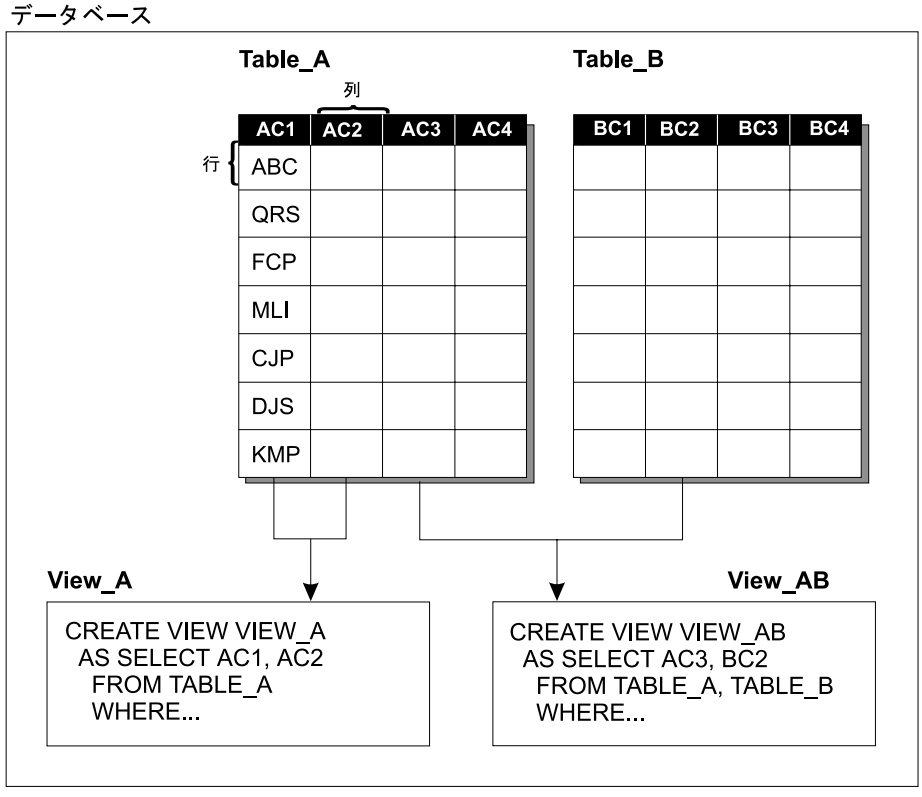


図 2. 表と視点の間の関係

次のステートメントでは、STAFF 表のうち、部署 (DEPT) 20 の中で、管理職 (Mgr) でない人のデータのうち、基本表に含まれる給与 (SALARY) と歩合 (COMM) を除く残りのデータの視点を作成しています。

```
CREATE VIEW STAFF_ONLY
   AS SELECT ID, NAME, DEPT, JOB, YEARS
         FROM STAFF
         WHERE JOB <> 'Mgr' AND DEPT=20
```

この視点を作成した後、次のステートメントによって視点の内容を確認できます。

```
SELECT *
   FROM STAFF_ONLY
```

このステートメントの結果は、次のとおりです。

```
ID     NAME      DEPT   JOB   YEARS
------ --------- ------ ----- ------
    20 Pernal        20 Sales      8
    80 James         20 Clerk      -
   190 Sneider       20 Clerk      8
```

さらに例を挙げれば、 STAFF 表と ORG 表を使用することによって、各部署の名前とそれぞれの部長の名前をリストアップした視点を作成できます。その視点を作成するためのステートメントは、次のとおりです。

```
CREATE VIEW DEPARTMENT_MGRS
   AS SELECT NAME, DEPTNAME
         FROM STAFF, ORG
         WHERE MANAGER = ID
```

視点作成時に WITH CHECK OPTION 文節を使うと、その視点によって、表の挿入や更新に対する付加的な制約を設けることができます。その文節を使うと、データベース・マネージャーは、その視点に対する更新または挿入の操作が視点の定義に違反していないかどうかを確認し、もし違反しているならその操作を拒否します。その文節を省略すると、挿入操作と更新操作が視点定義に違反していないかどうかの検査は実行されません。 WITH CHECK OPTION については、 *SQL 解説書* の中の CREATE VIEW ステートメントの説明を参照してください。

視点を使用したデータ操作

SELECT ステートメントと同じように、 INSERT、DELETE、および UPDATE ステートメントは、あたかも表に対するようにして視点に適用されます。それらのステートメントは、元となっている基本表のデータを操作することになります。したがって、次に視点にアクセスすると、最新の基本表を使った評価になります。 WITH CHECK OPTION 文節を使わない場合に視点を使ってデータを変更すると、そのデータは元の視点定義に違反している可能性があるため、その視点を繰り返してアクセスしても変更後のデータにはなりません。

視点 FIXED_INCOME に更新を適用する例を以下に示します。

```
CREATE VIEW FIXED_INCOME (LNAME, DEPART, JOBTITLE, NEWSALARY)
   AS SELECT NAME, DEPT, JOB, SALARY
         FROM PERS
         WHERE JOB <> 'Sales' WITH CHECK OPTION

UPDATE FIXED_INCOME
   SET NEWSALARY = SALARY * 1.10
   WHERE LNAME = 'Li'
```

この視点での更新操作は、基本表 PERS での次のような更新操作に相当します (チェック・オプションを除く)。

```
UPDATE PERS
   SET SALARY = SALARY * 1.10
   WHERE NAME = 'Li'
     AND JOB <> 'Sales'
```

視点作成時に、CREATE VIEW FIXED_INCOME の中で JOB <> 'Sales' 制約に WITH CHECK OPTION を使っているため、 Limoges を営業 (Sales) に移すための次の更新操作は実行できません。

```
UPDATE FIXED_INCOME
   SET JOBTITLE = 'Sales'
   WHERE LNAME = 'Limoges'
```

SALARY + COMM や SALARY * 1.25 などの式で定義されている列は更新できません。視点の定義にそのような列が含まれているなら、その所有者にはそれらの列に対する UPDATE 特権が与えられません。そのような列を含む視点に対する INSERT ステートメントは実行できませんが、DELETE ステートメントは可能です。

PERS 表のどの列に対しても NOT NULL が定義されていないとしましょう。その場合、基礎表 PERS の列のうち ID、YEARS、COMM、または BIRTH_DATE は FIXED_INCOME に含まれていませんが、それでも、FIXED_INCOME 視点を使って PERS 表に行を挿入できることになります。視点では見えない列は、NULL または省略時値のいずれか該当するものに設定されます。

しかし、PERS 表の列 ID は、NOT NULL として定義されていません。FIXED_INCOME 視点によって行を挿入しようとすると、システムは、PERS の列のうちその視点では 『見えない』 すべての列に NULL 値を挿入しようとします。 ID 列は視点に含まれておらず、しかも NULL 値が禁止されているため、その視点への挿入は許可されません。

視点の変更に関する規則と制限事項については、 *SQL 解説書* の中の CREATE VIEW ステートメントの説明を参照してください。

第4章 SQL ステートメントによるデータ・アクセス

ここでは、SQL ステートメントを使って、データベースに接続したりデータを取り出したりする方法について説明します。

ここに示す例では、入力するステートメントが示されています。また多くの例については、そのステートメントをサンプル・データベースに対して発行した場合の結果も示します。ステートメントはここでは大文字になっていますが、実際に入力する場合は大文字小文字を混合させて入力できます (単一引用符 (') または二重引用符 ('') で囲まれている箇所を除く)。

DB2 ユニバーサル・データベースに含まれているサンプル・データベース (SAMPLE) は、 79ページの『付録A. サンプル・データベース表』に載せられているいくつかの表で構成されています。サンプル・データベースは、 ″First Steps″ インストール・ランチパッドを使用して作成します。コマンド行からサンプル・データベースを作成することもできます。 *SQL 解説書* を参照してください。

ほかにも DB2 ユニバーサル・データベースには、データ・ウェアハウス・センターと OLAP スターター・キットの機能を示すためのサンプル・データベースが含まれています。このマニュアルの例では、汎用サンプル・データベース (SAMPLE) だけを使っています。

データベースのセットアップ方法によっては、使用する表名の前にスキーマ名とピリオドを付けて修飾する必要があるかもしれません。このマニュアルの中では、デフォルト・スキーマが USERID であるとしています。その場合、表 ORG は、USERID.ORG として参照することになります。そのような修飾が必要かどうかについては、管理担当者にお問い合わせください。

この章の内容は、次のとおりです。

- データベースへの接続
- エラーの調査
- 列の選択と行の選択
- 行のソートと重複行の除去
- 操作の順序
- 式による値の計算
- 式に名前を付ける

- 複数の表からのデータ選択
- 副照会の使用
- 関数の使用
- グループ化

データベースへの接続

SQL ステートメントを使ってデータベースを照会したり操作したりするには、その前にそのデータベースに接続する必要があります。 CONNECT ステートメントは、データベース接続とユーザー名とを関連付けます。

たとえば、SAMPLE データベースに接続するには、DB2 コマンド行プロセッサーで次のように入力します。

```
CONNECT TO SAMPLE USER USERID USING PASSWORD
```

(ユーザー ID とパスワードの値には、サーバー・システムで有効なものを使ってください。)

この例では、USER は USERID であり、USING は PASSWORD です。

正常に接続されると、次のようなメッセージが表示されます。

```
Database Connection Information

   Database product       = DB2/NT 7.1.0
   SQL authorization ID   = USERID
   Local database alias   = SAMPLE
```

接続が確立されたなら、データベースの操作を開始できます。接続方法の詳細については、*SQL 解説書* の中の CONNECT ステートメントの説明を参照してください。

エラーの調査

例を入力する際に入力ミスがあったり、SQL ステートメントの実行中にエラーが発生したりすると、データベース・マネージャーからエラー・メッセージが戻されます。エラー・メッセージは、メッセージ ID、簡単な説明、および SQLSTATE で構成されています。

SQLSTATE エラーは、 DB2 プロダクト・ファミリーに共通のエラー・コードです。 SQLSTATE エラーは、ISO/ANSI SQL92 規格に準拠しています。

たとえば、CONNECT ステートメントの中でユーザー ID またはパスワードが間違っているなら、データベース・マネージャーはメッセージ ID として SQL1403N、 SQLSTATE として 08004 を戻します。そのメッセージは、次のとおりです。

```
SQL1403N The username and/or password supplied is
         incorrect.   SQLSTATE=08004
```

疑問符 (?) の後にメッセージ ID または SQLSTATE を入力すると、エラー・メッセージに関するさらに詳しい情報が表示されます。

```
       ? SQL1403N
または
       ? SQL1403
または
       ? 08004
```

エラー SQL1403N の説明の第 2 行 (最後の行) は、SQLCODE が -1403 であることを示しています。 SQLCODE は、プロダクト特有のエラー・コードです。 N (Notification、通知) または C (Critical、重大) で終わるメッセージ ID は、エラーの発生を表しており、SQLCODE は負の値です。 W (警告) で終わるメッセージ ID は警告であり、その SQLCODE は正の値です。

列の選択

表から特定の列を選択するには、SELECT ステートメントを使います。このステートメントには、列の名前をコンマで区切ったリストを指定します。このリストのことを*選択リスト* といいます。

次のステートメントでは、SAMPLE データベースの ORG 表から部署名 (DEPTNAME) と部署番号 (DEPTNUMB) を選択しています。

```
SELECT DEPTNAME, DEPTNUMB
   FROM ORG
```

このステートメントの結果は、次のとおりです。

```
DEPTNAME       DEPTNUMB
-------------- --------
Head Office          10
New England          15
Mid Atlantic         20
South Atlantic       38
Great Lakes          42
Plains               51
Pacific              66
Mountain             84
```

アスタリスク (*) を使うと、表のすべての列を選択できます。次の例は、ORG 表からそのすべての列と行を取り出しています。

```
SELECT *
  FROM ORG
```

このステートメントの結果は、次のとおりです。

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

行の選択

表から特定の行を選択するには、SELECT ステートメントの後に WHERE 文節を使って、行が選択されるために満たすべき条件を指定します。表から行を選択する基準のことを、*検索条件* といいます。

検索条件は、1 つまたは複数の*述部* で構成されます。述部は、ある行に関して真か偽 (または不明) となる条件を指定します。 WHERE 文節の中で条件を指定するには、次の基本的な述部を使います。

述部	機能
x = y	x は y に等しい
x <> y	x は y に等しくない
x < y	x は y より小さい
x > y	x は y より大きい
x <= y	x は y 以下である
x >= y	x は y 以上である
IS NULL/IS NOT NULL	NULL 値かどうかの検査

検索条件を作成する場合、数値データ・タイプ以外には算術計算を実行しないようにし、互換性のあるデータ・タイプの間以外では比較を行わないように注意してください。たとえば、テキスト・ストリングと数値は比較できません。

文字値に基づいて行を選択する場合、その値は単一引用符で囲む必要があり (たとえば WHERE JOB = 'Clerk')、各文字値はデータベース中に存在しているものと正確に一致するように入力する必要があります。データベースの中でデータ値が小文字になっているのに、大文字として入力すると、行は選択されません。数値に基づいて行を選択する場合、その値は引用符で囲まないようにしてください (たとえば WHERE DEPT = 20)。

次の例では、STAFF 表から部署 20 の行だけを選択しています。

```
SELECT DEPT, NAME, JOB
   FROM STAFF
   WHERE DEPT = 20
```

このステートメントの結果は、次のとおりです。

```
 DEPT   NAME      JOB
 ------ --------- -----
     20 Sanders   Mgr
     20 Pernal    Sales
     20 James     Clerk
     20 Sneider   Clerk
```

次の例では、AND を使って複数の条件を指定しています。条件は、必要なだけ何個でも指定できます。この例では、STAFF 表から部署 20 の事務員 (Clerk) を選択しています。

```
SELECT DEPT, NAME, JOB
   FROM STAFF
   WHERE JOB = 'Clerk'
   AND DEPT = 20
```

このステートメントの結果は、次のとおりです。

```
DEPT   NAME      JOB
------ --------- -----
    20 James     Clerk
    20 Sneider   Clerk
```

省略時値がサポートされていない列に値が入力されていないなら、NULL 値になります。また、値が特に NULL として設定されている場合もあります。NULL 値になるのは、NULL 値をサポートするよう定義されている列だけです。表の中の NULL 値の定義とサポートについては、 9ページの『表の作成』を参照してください。

NULL 値かどうかを調べるには、IS NULL 述部および IS NOT NULL 述部を使います。

次のステートメントは、歩合が該当しない従業員のリストを出力します。

```
SELECT ID, NAME
   FROM STAFF
   WHERE COMM IS NULL
```

このステートメントの結果は、次のとおりです。

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
    50 Hanes
   100 Plotz
   140 Fraye
   160 Molinare
   210 Lu
   240 Daniels
   260 Jones
   270 Lea
   290 Quill
```

値 0 (ゼロ) は、NULL 値と同じではありません。次のステートメントでは、表に含まれている従業員のうち、歩合が 0 の人を選択しています。

```
SELECT ID, NAME
   FROM STAFF
   WHERE COMM = 0
```

サンプル表の中に、COMM 列の値が 0 のものはないため、戻される結果セットは空です。

次の例では、 STAFF 表の中で YEARS の値が 9 より大きい行をすべて選択しています。

```
SELECT NAME, SALARY, YEARS
   FROM STAFF
   WHERE YEARS > 9
```

このステートメントの結果は、次のとおりです。

```
NAME      SALARY    YEARS
--------- --------- ------
Hanes      20659.80     10
Lu         20010.00     10
Jones      21234.00     12
Quill      19818.00     10
Graham     21000.00     13
```

行のソート

特定の順序で情報が戻されるようにしたい場合があるかもしれません。ORDER BY 文節を使うと、1 つまたは複数の列の値に基づいて情報をソートすることができます。

次のステートメントでは、部署 84 の従業員が就業年数 (YEARS) でソートして表示されます。

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY YEARS
```

このステートメントの結果は、次のとおりです。

```
NAME      JOB   YEARS
--------- ----- ------      Davis     Sales      5
Gafney    Clerk      5
Edwards   Sales      7
Quill     Mgr       10
```

ORDER BY 文節は、SELECT ステートメント全体の最後の文節として指定してください。この文節に指定する列名は、式または表の列です。 ORDER BY 文節の列名は、選択リストに指定したものである必要はありません。

ORDER BY 文節に ASC または DESC を明示的に指定すると、行を昇順または降順にすることができます。そのどちらも指定しないなら、行は自動的に昇順になります。次のステートメントでは、部署 84 の従業員が就業年数 (YEARS) の降順で表示されます。

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY YEARS DESC
```

このステートメントの結果は、次のとおりです。

```
NAME      JOB   YEARS
--------- ----- ------
Quill     Mgr       10
Edwards   Sales      7
Davis     Sales      5
Gafney    Clerk      5
```

行の順序は、文字値によるものも数値によるものも可能です。次のステートメントでは、部署 84 の従業員が名前のアルファベット順序で表示されます。

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY NAME
```

このステートメントの結果は、次のとおりです。

```
NAME      JOB   YEARS
--------- ----- ------
Davis     Sales      5
Edwards   Sales      7
Gafney    Clerk      5
Quill     Mgr       10
```

重複行の除去

SELECT ステートメントを使う場合に、情報が重複して戻されることがないようにしたいことがあります。たとえば、STAFF の DEPT 列ではいくつかの部署番号が何度も出現しますし、JOB 列ではいくつかの肩書きが何度も出現します。

重複した行をなくすには、SELECT 文節に DISTINCT オプションを使います。たとえば、ステートメントに DISTINCT を挿入すると、 1 つの部署 (DEPT) の中で肩書き (JOB) は 1 度しか出力されなくなります。

```
SELECT DISTINCT DEPT, JOB
   FROM STAFF
   WHERE DEPT < 30
   ORDER BY DEPT, JOB
```

このステートメントの結果は、次のとおりです。

```
DEPT   JOB
------ -----
    10 Mgr
    15 Clerk
    15 Mgr
    15 Sales
    20 Clerk
    20 Mgr
    20 Sales
```

DISTINCT を指定したため、 SELECT ステートメントの中で指定されている列集合の中の重複データを含む行はすべて除去されています。

操作の順序

操作の順序を考慮に入れることは、たいへん重要です。下記のリストに従って、1 つの文節の出力が次の文節の入力となります。操作の順序が重要になる例については、 26ページの『式に名前を付ける』を参照してください。

以下の操作順序は、 DB2 コード内で実際に操作が実行される順序であるとは限りません。ここでの簡単な説明は、単に照会の順序を直感的に考える方法を示すためのものです。操作の順序は、次のとおりです。

1. FROM 文節
2. WHERE 文節
3. GROUP BY 文節
4. HAVING 文節
5. SELECT 文節
6. ORDER BY 文節

式による値の計算

式 は、ステートメントに含める計算または関数です。次のステートメントでは、部署 38 の従業員がそれぞれ $500 のボーナスを受け取った場合に、その給与が合計でいくらになるかを計算しています。

```
SELECT DEPT, NAME, SALARY + 500
   FROM STAFF
   WHERE DEPT = 38
   ORDER BY 3
```

結果は、次のとおりです。

```
DEPT   NAME      3
------ --------- ----------------
    38 Abrahams          12509.75
    38 Naughton          13454.75
    38 Quigley           17308.30
    38 Marenghi          18006.75
    38 O'Brien           18506.00
```

第 3 列の列名が番号になっていることに注意してください。これは、SALARY+500 は列名を指定するものではないためシステムが生成した番号です。後で ORDER BY 文節において、第 3 列を参照するのにこの番号を使っています。式に意味のある名前を付ける方法については、 26ページの『式に名前を付ける』を参照してください。

算術式は、加算 (+)、減算 (−)、乗算 (*)、および除算 (/) の基本的な演算子を使って構成できます。

演算子の被演算数値としては、さまざまなタイプのオペランドが可能です。たとえば、
- 列名 (RATE * HOURS など)
- 定数値 (RATE * 1.07 など)
- スカラー関数 (LENGTH(NAME) + 1 など)

式に名前を付ける

オプションの AS 文節を使うと、式に意味のある名前を割り当てておき、後でそれを参照するのに使うことができます。 AS 文節によって、選択リストの中のどの項目にも名前を付けることができます。

次のステートメントでは、給与と歩合の合計額が $13,000 より小さい従業員が表示されます。式 SALARY + COMM には、PAY という名前が付けられています。

```
SELECT NAME, JOB, SALARY + COMM AS PAY
   FROM STAFF
   WHERE (SALARY + COMM) < 13000
   ORDER BY PAY
```

このステートメントの結果は、次のとおりです。

```
NAME      JOB   PAY
--------- ----- ----------
Yamaguchi Clerk   10581.50
Burke     Clerk   11043.50
Scoutten  Clerk   11592.80
Abrahams  Clerk   12246.25
Kermisch  Clerk   12368.60
Ngan      Clerk   12714.80
```

AS 文節を使うことによって、 ORDER BY 文節の中でシステムの生成した番号ではなく、特定の列名を参照することができます。この例の WHERE 文節の中では、 PAY ではなく (SALARY + COMM) を 13000 と比較しています。これは、操作の順序を考慮した結果です。 WHERE 文節は、(SALARY + COMM) に PAY という名前が付けられる前に評価されます。 SELECT 文節が WHERE 文節の後に実行されるからです。したがって、その述部には PAY を使えません。

複数の表からのデータ選択

SELECT ステートメントを使って、複数の表から取られた情報を含むレポートを作成できます。これは、一般に結合 と呼ばれています。たとえば、STAFF 表と ORG 表のデータを結合して新しい表にすることができます。 2 つの表を結合するには、SELECT 文節に列名、FROM 文節に表名、そして WHERE 文節に検索条件を指定します。 WHERE 文節はオプションです。

次の例では、管理職 (MANAGER) の人の名前と部署名とを関連付けています。従業員情報 (STAFF 表) と部署情報 (ORG 表) は別々に保管されているため、情報を 2 つの表から選択する必要があります。下記の照会では、STAFF 表および ORG 表からそれぞれ NAME 列および DEPTNAME 列を選択しています。検索条件によって、MANAGER 列の値が ID 列の値と同じである行だけが選択されるよう限定しています。

```
SELECT DEPTNAME, NAME
   FROM ORG, STAFF
   WHERE MANAGER = ID
```

図3 に、2 つの異なる表の列を比較する方法を示します。四角で囲んだ値は、検索条件が満たされているものを示しています。

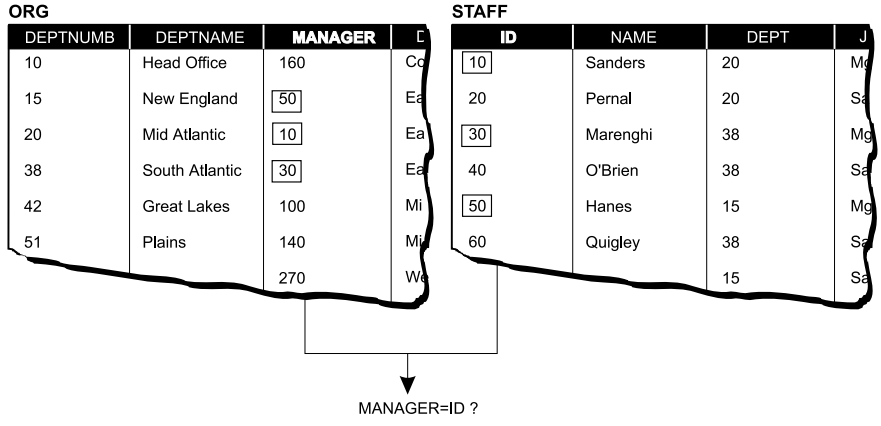


図3. STAFF 表と ORG 表からの選択

この SELECT ステートメントの結果は、次のとおりです。

```
DEPTNAME       NAME
-------------- ---------
Mid Atlantic   Sanders
South Atlantic Marenghi
New England    Hanes
Great Lakes    Plotz
```

```
Plains          Fraye
Head Office     Molinare
Pacific         Lea
Mountain        Quill
```

結果には、管理職の人の名前と、その部署とが示されています。

副照会の使用

SQL の SELECT ステートメントを作成する場合、 WHERE 文節の中に追加の SELECT ステートメントを入れることができます。追加の SELECT によって、副照会が開始されます。

副照会の中にさらに別の副照会を含めて、元の副照会の WHERE 文節が別の副照会の結果で置き換えられるようにすることができます。さらに、WHERE 文節の複数の検索条件の中にも副照会を含めることができます。副照会では、メインの照会で使う表や列とは違う表や列を参照できます。

次のステートメントでは、 STAFF 表の中で ID が 280 の従業員に関して、ORG 表の中から地域 (DIVISION) と場所 (LOCATION) を選択しています。

```
SELECT DIVISION, LOCATION
   FROM ORG
   WHERE DEPTNUMB = (SELECT DEPT
                        FROM STAFF
                        WHERE ID = 280)
```

ステートメントの処理では、まず副照会の結果が調べられます。 ID が 280 の従業員の部署は 66 なので、この例の副照会の結果は 66 になります。次に、ORG 表の中で、 DEPTNUMB 列の値が 66 になっている行から最終的な結果が取り出されます。最終的な結果は、次のとおりです。

```
DIVISION   LOCATION
---------- -------------
Western    San Francisco
```

副照会を使用すると、データベース・マネージャーはそれを評価し、WHERE 文節をその結果で直接置き換えます。

副照会の詳細については、41ページの『相関副照会』を参照してください。

関数の使用

ここでは、このマニュアルの例の中で使う関数について簡単に説明します。データベース*関数* とは、入力データ値と結果値の間の関係付けです。

関数には、*組み込み* 関数とユーザー*定義* 関数があります。 DB2 ユニバーサル・データベースには、たくさんの組み込み関数と、事前インストール済みユーザー定義関数が付属しています。

組み込み関数は SYSIBM スキーマの中にあり、事前インストール済みユーザー定義関数は SYSFUN スキーマの中にあります。 SYSIBM と SYSFUN は予約スキーマです。

組み込み関数と事前インストール済みユーザー定義関数だけですべてのユーザー要件を満たすことは到底できません。したがって、アプリケーション開発者は、開発するアプリケーションに特化した関数のセットを作成することが必要になる場合があります。これは、ユーザー定義関数によって可能になります。ユーザー定義関数を活用するなら、 DB2 ユニバーサル・データベースの適用範囲を拡張して、カスタマイズされた業務用関数または科学計算用関数を含めることができます。これについては、72ページの『ユーザー定義関数』を参照してください。

列関数

列関数 は、 1 つの列の一群の値から単一の結果値を計算します。列関数の例としては、下記のものがあります。これらは列関数のほんの一部分です。完全なリストについては、 *SQL 解説書* を参照してください。

AVG ある集合内の値の和を、その集合内の値の数で割った結果を戻します。

COUNT 行または値の集合内の行数または値数を戻します。

MAX 値の集合の中の最大値を戻します。

MIN 値の集合の中の最小値を戻します。

次のステートメントでは、STAFF 表から給与の最大値を選択しています。

```
SELECT MAX(SALARY)
   FROM STAFF
```

このステートメントは、STAFF サンプル表から 22959.20 の値を戻します。

次の例では、収入が平均収入より多く、就業年数が平均就業年数より短い従業員の名前と給与を選択しています。

```
SELECT NAME, SALARY
   FROM STAFF
   WHERE SALARY > (SELECT AVG(SALARY) FROM STAFF)
   AND YEARS < (SELECT AVG(YEARS) FROM STAFF)
```

このステートメントの結果は、次のとおりです。

```
 NAME      SALARY
--------- ---------
Marenghi   17506.75
Daniels    19260.25
Gonzales   16858.20
```

この例の WHERE 文節では、列関数が直接的な書き方 (例: WHERE SALARY > AVG(SALARY)) ではなく、*副照会* の形で書かれています。 WHERE 文節の中に列関数をコーディングすることはできません。 これは操作の順序のためです。 WHERE 文節は、SELECT 文節より前に評価されるものと考えてください。そのため、WHERE 文節が評価される時点では、列関数によって値の集合にアクセスすることができません。この値の集合は、もっと後で SELECT 文節によって選択されます。

列関数の引き数の一部として DISTINCT 要素を使用するなら、関数の適用前に値の重複をなくすことができます。 COUNT(DISTINCT WORKDEPT) とすると、異なる部署の数が計算されます。

スカラー関数

スカラー関数 は、 1 つの値に対してなんらかの操作を実行して、別の 1 つの値を戻します。 DB2 ユニバーサル・データベースに用意されているスカラー関数の例としては、下記のものがあります。これらはほんの一部分にすぎません。

ABS 数値の絶対値を戻します。

HEX 値を 16 進数にしたものを戻します。

LENGTH 引き数の文字数を戻します (グラフィック・ストリングの場合、2 バイト文字の数になります)。

YEAR 日付 / 時刻値のうち年の部分を抽出します。

スカラー関数のリストとその説明については、 *SQL 解説書* を参照してください。

次のステートメントでは、 ORG 表から部署名と、その名前の長さを戻します。

```
SELECT DEPTNAME, LENGTH(DEPTNAME)
   FROM ORG
```

このステートメントの結果は、次のとおりです。

```
DEPTNAME       2
-------------- -----------
Head Office             11
New England             11
Mid Atlantic            12
South Atlantic          14
Great Lakes             11
Plains                   6
Pacific                  7
Mountain                 8
```

注: LENGTH(DEPTNAME) に意味のある名前を付けるための AS 文節を使っていないため、システムの生成した番号が第 2 欄に表示されています。

表関数

表関数 は、表の列を戻します。表の列は、単純な CREATE TABLE ステートメントで作成する表そのものとよく似ています。

表関数は、 SQL ステートメントの FROM 文節でのみ使用できます。

今のところ、 DB2 ユニバーサル・データベースでサポートされている唯一の表関数は SQLCACHE_SNAPSHOT です。

SQLCACHE_SNAPSHOT

DB2 動的 SQL ステートメント・キャッシュのスナップショットの結果を表の形で戻します。

グループ化

DB2 ユニバーサル・データベースには、表の中の特定の列に基づいてデータを分析する機能があります。

GROUP BY 文節でグループ化構造を定義すると、そのグループ化構造にしたがって行を分類できます。最も単純な形式の場合、*グループ* は、 ″GROUP BY″ 列に同じ値が入っている行の集合です。 SELECT 文節に指定する列名は、グループ化列か列関数のいずれかでなければなりません。列関数は、GROUP BY 文節で定義される各グループごとに値を 1 つずつ戻します。各グループは、結果セットの 1 つの行によって表されます。次の例では、各部署番号ごとの給与の最大値のリストを生成しています。

```
SELECT DEPT, MAX(SALARY) AS MAXIMUM
   FROM STAFF
   GROUP BY DEPT
```

このステートメントの結果は、次のとおりです。

```
DEPT   MAXIMUM
------ ---------
    10  22959.20
    15  20659.80
    20  18357.50
    38  18006.00
    42  18352.80
    51  21150.00
    66  21000.00
    84  19818.00
```

MAX(SALARY) は、会社全体ではなく、 GROUP BY 文節によって定義される各グループ、つまり各部署ごとに計算されることに注意してください。

WHERE 文節と GROUP BY 文節の併用

グループ化照会には、グループが作成されて列関数が計算される前に行をふるいにかけるための標準的な WHERE 文節を含めることができます。 WHERE 文節は、GROUP BY 文節より*前に* 指定する必要があります。たとえば、

```
SELECT WORKDEPT, EDLEVEL, MAX(SALARY) AS MAXIMUM
   FROM EMPLOYEE
   WHERE HIREDATE > '1979-01-01'
   GROUP BY WORKDEPT, EDLEVEL
   ORDER BY WORKDEPT, EDLEVEL
```

結果は、次のとおりです。

```
WORKDEPT EDLEVEL MAXIMUM
-------- ------- -----------
D11           17    18270.00
D21           15    27380.00
D21           16    36170.00
D21           17    28760.00
E11           12    15340.00
E21           14    26150.00
```

SELECT ステートメントに指定する列名はすべて、GROUP BY 文節の中にも登場しています。列名がどちらか一方だけにしか使われていないなら、エラーになります。この GROUP BY 文節は、WORKDEPT と EDLEVEL の固有の各組み合わせごとに 1 つの行を戻します。

GROUP BY 文節の後に HAVING 文節を使う

ある条件を満たすグループについてのみ結果が戻されるようにするため、グループに対してフィルター条件を適用することができます。そのためには、GROUP BY 文節の後に HAVING 文節を指定します。 HAVING 文節には、述部を 1 つ含めたり、複数の述部を AND や OR で結合したものを含めたりすることができます。各述部では、グループの 1 つの特性 (AVG(SALARY) など) を、次のうちのいずれかと比較します。

- そのグループの別の特性

  たとえば、

  ```
  HAVING AVG(SALARY) > 2 * MIN(SALARY)
  ```

- 定数値

  たとえば、

  ```
  HAVING AVG(SALARY) > 20000
  ```

たとえば、次の照会では、従業員が 4 人以上いる部署の給与の最大値と最小値を調べています。

```
SELECT WORKDEPT, MAX(SALARY) AS MAXIMUM, MIN(SALARY) AS MINIMUM
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   HAVING COUNT(*) > 4
   ORDER BY WORKDEPT
```

このステートメントの結果は、次のとおりです。

```
WORKDEPT MAXIMUM     MINIMUM
-------- ----------- -----------
D11         32250.00    18270.00
D21         36170.00    17250.00
E11         29750.00    15340.00
```

HAVING 文節はあるが GROUP BY 文節のない照会も、あまり普通ではありませんが、可能な照会です。その場合、DB2 は表全体を 1 つのグループとして扱います。表が単一のグループとして扱われるため、結果の行は多くても 1 行です。その HAVING 条件が表全体として真であるなら、選択結果が戻されます (結果は列関数だけで構成されるものでなければなりません)。そうでないなら、行は戻されません。

第5章 式と副照会

DB2 には、照会を表現するためのいろいろな方法が用意されています。この章では、複雑な照会を表現するためのいくつかの重要な方法について説明します。

この章の内容は、次のとおりです。

- スカラー全選択
- データ・タイプのキャスト
- CASE 式
- 表式
- 相関名

スカラー全選択

全選択は、 SQL ステートメントで使用できる照会の 1 つの形態です。スカラー全選択は、 1 つの値しか含まない行を 1 つだけ戻す全選択です。スカラー全選択は、データベースからデータ値を取り出して、それを式の中で使う場合に便利です。

- 次の例では、全従業員の平均給与より高い給与の従業員の名前のリストが表示されます。括弧の中の SELECT ステートメントが照会内のスカラー全選択です。

```
SELECT LASTNAME, FIRSTNME
   FROM EMPLOYEE
   WHERE  SALARY > (SELECT AVG(SALARY)
                    FROM EMPLOYEE)
```

- この例では、 STAFF 表と EMPLOYEE 表で従業員の平均給与を調べています。

```
SELECT AVG(SALARY) AS "Average_Employee",
   (SELECT AVG(SALARY) AS "Average_Staff" FROM STAFF)
      FROM EMPLOYEE
```

データ・タイプのキャスト

値をあるデータ・タイプから別のデータ・タイプに (たとえば数値から文字ストリングに) 変換することが必要になる場合がよくあります。値のタイプを変換するには、CAST 指定を使います。

CAST 指定の別の用途としては、非常に長い文字ストリングを切り捨てることがあります。 EMP_RESUME 表の列 RESUME は CLOB(5K) です。そのうち、個人情報が含まれている最初の 370 バイトだけを表示させたいということがあるかもしれません。 EMP_RESUME 表の RESUME の最初の 370 バイトを ASCII 形式で表示させるには、次の照会を発行します。

```
SELECT EMPNO, CAST(RESUME AS VARCHAR(370))
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

370 バイトより長い値が切り捨てられることを示す警告が出されます。

NULL 値を、照会の中で操作しやすい他のデータ・タイプにキャストすることもできます。この目的でキャストを使う例については、 39ページの『共通表式』を参照してください。

CASE 式

SQL ステートメントの中で CASE 式を使うことによって、表のデータの表現を簡単に操作できます。それによって、いくつかのプログラミング言語の CASE ステートメントと同じように、強力な条件式の機能を実現できます。

- ORG 表の DEPTNAME 列に含まれる部署番号を、意味のある言葉にするには、次のような照会を入力します。

```
SELECT DEPTNAME,
   CASE DEPTNUMB
     WHEN 10 THEN 'Marketing'
     WHEN 15 THEN 'Research'
     WHEN 20 THEN 'Development'
     WHEN 38 THEN 'Accounting'
     ELSE 'Sales'
   END  AS FUNCTION
   FROM ORG
```

結果は、次のとおりです。

```
DEPTNAME       FUNCTION
-------------- -----------
Head Office    Marketing
New England    Research
Mid Atlantic   Development
```

```
South Atlantic Accounting
Great Lakes    Sales
Plains         Sales
Pacific        Sales
Mountain       Sales
```

- CASE 式を使うことによって、ゼロ除算などの例外が発生しないよう保護できます。次の例では、ボーナスも歩合もない従業員については、ステートメントの条件式で除算操作をしないようにすることによりエラーを回避しています。

```
SELECT LASTNAME, WORKDEPT FROM EMPLOYEE
   WHERE(CASE
            WHEN BONUS+COMM=0 THEN NULL
            ELSE SALARY/(BONUS+COMM)
         END ) > 10
```

- ある列のすべての値の総和に対するその列の値のサブセットの和の比率を計算するのに、 CASE 式を使うことができます。この比率は、CASE 式を使う 1 つのステートメントに含めることができるので、データ操作の 1 回のパスだけで済みます。 CASE 式を使わないとすると、同じ計算を実行するのに最低 2 回のパスが必要になります。

  次の例では、CASE 式を使うことによって、すべての給与の総和に対する部署 20 の給与の和の比率を計算しています。

```
SELECT CAST(CAST (SUM(CASE
                        WHEN DEPT = 20 THEN SALARY
                        ELSE 0
                      END) AS DECIMAL(7,2))/
                   SUM(SALARY) AS DECIMAL (3,2))
   FROM STAFF
```

  結果は 0.11 になります。 CAST 関数を使うことによって、結果の精度が保たれるようにしています。

- CASE 式を使うことによって、関数を呼び出して余分なオーバーヘッドをもたらすことなく、簡単な関数を評価できます。たとえば、

```
CASE
   WHEN X<0 THEN -1
   WHEN X=0 THEN 0
   WHEN X>0 THEN 1
END
```

  この式の結果は、SYSFUN スキーマの中の SIGN ユーザー定義関数によるものと同じです。

表式

単一の照会のためだけに視点の定義が必要となる場合には、表式 を使うことができます。

表式は一時的なものであり、それが有効なのはその SQL ステートメントの存続期間だけです。表式は視点と違って共有できませんが、視点よりも柔軟性があります。

ここでは、照会の中での共通表式およびネストされた表式の使い方について説明します。

ネストされた表式

ネストされた表式は、メインの照会の FROM 文節の中で定義がネストされている (直接定義されている) 一時視点です。

次の照会では、ネストされた表式を使うことによって、教育レベル (EDLEVEL) が 16 を超える人について、所得合計 (TOTAL_PAY) の平均、教育レベル、および採用年 (HIREYEAR) を調べています。

```
SELECT EDLEVEL, HIREYEAR, DECIMAL(AVG(TOTAL_PAY),7,2)
   FROM (SELECT EDLEVEL, YEAR(HIREDATE) AS HIREYEAR,
                SALARY+BONUS+COMM AS TOTAL_PAY
            FROM EMPLOYEE
            WHERE EDLEVEL > 16)  AS PAY_LEVEL
   GROUP BY EDLEVEL, HIREYEAR
   ORDER BY EDLEVEL, HIREYEAR
```

結果は、次のとおりです。

```
EDLEVEL HIREYEAR    3
------- ----------- ---------
     17        1967  28850.00
     17        1973  23547.00
     17        1977  24430.00
     17        1979  25896.50
     18        1965  57970.00
     18        1968  32827.00
     18        1973  45350.00
     18        1976  31294.00
     19        1958  51120.00
     20        1975  42110.00
```

この照会では、ネストした表式を使うことによって、まず HIREDATE 列から採用年を抽出し、次にそれを GROUP BY 文節で使うようにしています。EDLEVEL の値をいろいろに変えて同様の照会を実行しようとする場合、これを視点として作成しないほうがよいでしょう。

この例では、スカラー組み込み関数 DECIMAL が使われています。DECIMAL は、数値または文字ストリングを 10 進数として戻します。関数については、 *SQL 解説書* を参照してください。

共通表式

共通表式 は、複雑な照会で一貫して使用するために作成する表式です。照会の最初に、WITH 文節を使って定義し、名前を付けます。共通表式を繰り返して参照した場合、同じ結果セットが使われます。これに対し、もしネストされた表式や視点を使うとすれば、いちいち結果セットを生成し直すことになり、生成するたびにその結果が違うという可能性も出てきます。

次の例では、全社員の中で、教育レベルが 16 を超える人のうち、同じ年に採用された同じ教育レベルの人の平均所得を下回っている人のリストを出力します。この照会の各部分について、この後さらに詳しく説明します。

1

```
WITH
      PAYLEVEL AS
         (SELECT EMPNO, EDLEVEL, YEAR(HIREDATE) AS HIREYEAR,
                 SALARY+BONUS+COMM AS TOTAL_PAY
             FROM EMPLOYEE
             WHERE EDLEVEL > 16),
```

2

```
      PAYBYED (EDUC_LEVEL, YEAR_OF_HIRE, AVG_TOTAL_PAY) AS
         (SELECT EDLEVEL, HIREYEAR, AVG(TOTAL_PAY)
             FROM PAYLEVEL
             GROUP BY EDLEVEL, HIREYEAR)
```

3

```
SELECT EMPNO, EDLEVEL, YEAR_OF_HIRE, TOTAL_PAY, DECIMAL(AVG_TOTAL_PAY,7,2)
   FROM PAYLEVEL, PAYBYED
   WHERE EDLEVEL = EDUC_LEVEL
     AND HIREYEAR = YEAR_OF_HIRE
     AND TOTAL_PAY < AVG_TOTAL_PAY
```

1 これは、PAYLEVEL という名前の共通表式です。この結果表には、従業員番号、従業員の採用年、その従業員の総所得、およびその教育レベルが含まれています。その中に含まれるのは、教育レベルが 16 を超える従業員の行だけです。

2 これは、PAYBYED (PAY BY EDucation、教育別所得) という名前の共通表式です。この中では、その前の共通表式で作成された PAYLEVEL 表を使って、各教育レベルごと、また同じ年に採用された人ごとに、従業員の教育レベル、採用年、および平均所得を調べていま

す。この表で戻される列には、選択リストで使われている列名とは違う名前が付けられています (たとえば EDUC_LEVEL)。これによって生成される結果セット (PAYBYED) は、ネストされた表式の例で生成された結果と同じものです。

3 最後は、希望する結果を生成するための実際の照会です。 2 つの表 (PAYLEVEL、PAYBYED) を結合することにより、総所得が、同じ年に採用された人の中での平均所得を下回っている人を調べます。 PAYBYED は PAYLEVEL に基づいていることに注意してください。そのため、ステートメント全体を通じて PAYLEVEL に 2 回アクセスすることにより、効率のよいものとなっています。 2 回とも、照会の評価には同じ行集合が使われます。

最終的な結果は次のとおりです。

```
EMPNO  EDLEVEL YEAR_OF_HIRE TOTAL_PAY     5
------ ------- ------------ ------------- ---------
000210      17         1979      20132.00  25896.50
```

相関名

相関名 は、1 つのオブジェクトを何度も使う場合に、複数の使用を互いに区別するために使う識別子です。相関名は、照会の FROM 文節の中で定義したり、 UPDATE または DELETE ステートメントの最初の文節の中で定義したりできます。表、視点、またはネストした表式に関連付けることができますが、それが定義されている文脈以外では使用できません。

たとえば、FROM STAFF S、ORG O という文節では、S および O がそれぞれ STAFF および ORG の相関名として確立されます。

```
SELECT NAME, DEPTNAME
   FROM STAFF S, ORG O
   WHERE O.MANAGER = S.ID
```

相関名を定義したなら、オブジェクトの修飾にはその相関名しか 使えません。たとえば、上の例の場合、 ORG.MANAGER=STAFF.ID とすると、ステートメントは失敗します。

また相関名は、データベース・オブジェクトを参照するための短縮名として使うこともできます。 STAFF と入力するよりは、S だけを入力するほうが簡単です。

相関名を使うなら、 1 つのオブジェクトの**複製**を作成できます。これは、ある表の全体をそれ自体と比較する場合に便利です。次の例では、表 EMPLOYEE

をそれ自体の別のインスタンスと比較することによって、全従業員の上司 (MANAGER) を調べています。設計者以外の従業員の名前、その上司の名前、そして部署番号 (WORKDEPT) を表示します。

```
SELECT E2.FIRSTNME, E2.LASTNAME, E2.JOB, E1.FIRSTNME AS MGR_FIRSTNAME,
                    E1.LASTNAME AS MGR_LASTNAME, E1.WORKDEPT
   FROM EMPLOYEE E1, EMPLOYEE E2
   WHERE E1.WORKDEPT = E2.WORKDEPT
     AND E1.JOB = 'MANAGER'
     AND E2.JOB <> 'MANAGER'
     AND E2.JOB <> 'DESIGNER'
```

このステートメントの結果は、次のとおりです。

```
FIRSTNME     LASTNAME   JOB      MGR_FIRSTNAME MGR_LASTNAME    WORKDEPT
------------ ---------- -------- ------------- --------------- --------
DOLORES      QUINTANA   ANALYST  SALLY         KWAN            C01
HEATHER      NICHOLLS   ANALYST  SALLY         KWAN            C01
JAMES        JEFFERSON  CLERK    EVA           PULASKI         D21
SALVATORE    MARINO     CLERK    EVA           PULASKI         D21
DANIEL       SMITH      CLERK    EVA           PULASKI         D21
SYBIL        JOHNSON    CLERK    EVA           PULASKI         D21
MARIA        PEREZ      CLERK    EVA           PULASKI         D21
ETHEL        SCHNEIDER  OPERATOR EILEEN        HENDERSON       E11
JOHN         PARKER     OPERATOR EILEEN        HENDERSON       E11
PHILIP       SMITH      OPERATOR EILEEN        HENDERSON       E11
MAUDE        SETRIGHT   OPERATOR EILEEN        HENDERSON       E11
RAMLAL       MEHTA      FIELDREP THEODORE      SPENSER         E21
WING         LEE        FIELDREP THEODORE      SPENSER         E21
JASON        GOUNOT     FIELDREP THEODORE      SPENSER         E21
```

相関副照会

それ以前に言及された表のいずれかを参照できる副照会のことを*相関副照会* といいます。また、その副照会にはメインの照会の表への*相関参照* があるともいいます。

次の例では、相関副照会ではない副照会を使って、部署 'A00' の従業員のうち、その部署の平均給与より高い給与を受け取る従業員の従業員番号と名前を出力しています。

```
SELECT EMPNO, LASTNAME
   FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
     AND SALARY > (SELECT AVG(SALARY)
                   FROM EMPLOYEE
                   WHERE WORKDEPT = 'A00')
```

このステートメントの結果は、次のとおりです。

```
EMPNO  LASTNAME
------ ---------------
000010 HAAS
000110 LUCCHESSI
```

すべての部署の平均給与を調べたい場合、すべての部署ごとにこのような副照会を評価することが必要になります。それには、SQL の相関機能を使います。その場合、外側の照会で使用されている表の各行ごとに 1 回ずつ繰り返して実行される副照会を 1 つ作成することができます。

次の例では、相関副照会を使って、給与がその部署の平均給与より高い従業員をすべて表示しています。

```
SELECT E1.EMPNO, E1.LASTNAME, E1.WORKDEPT
   FROM EMPLOYEE E1
   WHERE SALARY > (SELECT AVG(SALARY)
                      FROM EMPLOYEE E2
                      WHERE E2.WORKDEPT = E1.WORKDEPT)
   ORDER BY E1.WORKDEPT
```

この照会では、すべての部署についてそれぞれ 1 回ずつ評価されます。結果は、次のとおりです。

```
EMPNO  LASTNAME        WORKDEPT
------ --------------- --------
000010 HAAS            A00
000110 LUCCHESSI       A00
000030 KWAN            C01
000060 STERN           D11
000150 ADAMSON         D11
000170 YOSHIMURA       D11
000200 BROWN           D11
000220 LUTZ            D11
000070 PULASKI         D21
000240 MARINO          D21
000270 PEREZ           D21
000090 HENDERSON       E11
000280 SCHNEIDER       E11
000100 SPENSER         E21
000330 LEE             E21
000340 GOUNOT          E21
```

相関副照会を含む照会を作成するには、副照会を含む外側の照会の通常の形式と同じ基本的な形式を使います。しかし、外側の照会の FROM 文節の中で、表名の直後に相関名を指定します。副照会には、その相関名によって修飾した列参照を含めることができます。たとえば、E1 が相関名なら、E1.WORKDEPT は、外側の照会の中での表の現在行の WORKDEPT の値ということです。副照会は、(概念的には) 外側の照会の表の各行ごとに再評価できます。

相関副照会を使うことによって、システムに作業をまかせて、アプリケーションの中でのコーディング量を少なくすることができます。

DB2 では、修飾なしの相関参照が可能です。たとえば、表 EMPLOYEE には LASTNAME という列があるのに対して、表 SALES には SALES_PERSON という列はあっても LASTNAME という列がないとします。

```
SELECT LASTNAME, FIRSTNME, COMM
   FROM EMPLOYEE
   WHERE 3 > (SELECT AVG(SALES)
                FROM SALES
                WHERE LASTNAME = SALES_PERSON)
```

この例では、システムは一番内側の FROM 文節で LASTNAME 列があるかどうかを調べます。それが見つからないなら、その外側の FROM 文節 (この例では一番外側の FROM 文節) を調べます。必ずしも常に必要ではありませんが、照会を読みやすくし、目的の結果を間違いなく得るようにするため、できるだけ相関参照を修飾するようにしてください。

相関副照会のインプリメント

相関副照会を使うのは、どんな場合でしょうか? 1 つの手掛かりとなるのは列関数が使われるかどうかです。

ここで、教育レベルが部署平均より高い従業員のリストを作成するとしましょう。

まず、選択リスト項目を決定する必要があります。ここでの課題では「従業員のリストを作成する」ということです。したがって、EMPLOYEE 表の LASTNAME によって、従業員が一意に確定することになります。また、この課題では教育レベル (EDLEVEL) と従業員の部署 (WORKDEPT) も条件の中に含まれています。この課題には明示的にそれらの列を表示することが述べられていませんが、それらを選択リストに含めるならわかりやすくなるでしょう。ここまでで照会の一部分を次のように構成できます。

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE
```

次に、検索条件 (WHERE 文節) が必要です。ここでの課題では「教育レベルが部署平均より高い」となっています。つまり、表に含まれているすべての従業員について、その従業員の属する部署の平均教育レベルを計算する必要があります。このことは、相関副照会の説明に当てはまっています。なんらかの未知の特性値 (現在の従業員の属する部署の教育レベルの平均) を各行ごとに計算する、ということです。 EMPLOYEE 表のための相関名が必要となります。

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE E1
```

必要な副照会は簡単なものです。各部署ごとに教育レベルの平均を計算します。 SQL ステートメントは、全体として次のようになります。

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE E1
   WHERE EDLEVEL > (SELECT AVG(EDLEVEL)
                    FROM EMPLOYEE  E2
                    WHERE E2.WORKDEPT = E1.WORKDEPT)
```

結果は、次のとおりです。

```
LASTNAME        WORKDEPT EDLEVEL
--------------- -------- -------
HAAS            A00           18
KWAN            C01           20
PULASKI         D21           16
HENDERSON       E11           16
LUCCHESSI       A00           19
PIANKA          D11           17
SCOUTTEN        D11           17
JONES           D11           17
LUTZ            D11           18
MARINO          D21           17
JOHNSON         D21           16
SCHNEIDER       E11           17
MEHTA           E21           16
GOUNOT          E21           16
```

従業員の部署番号を出力するのではなく部署名を出力する場合を考えてみましょう。必要な情報 (DEPTNAME) は別の表 (DEPARTMENT) に含まれています。相関変数を定義する一番外側の照会は、結合照会にすることも可能です (27ページの『複数の表からのデータ選択』を参照)。

外側の照会で結合を使う場合、結合する表のリストを FROM 文節に指定し、該当表名の後に相関名を指定します。

前述の照会を、部署の番号ではなく部署の名前を出力するよう変更するには、選択リストの中の WORKDEPT を DEPTNAME に置き換えます。 FROM 文節には DEPARTMENT 表も含めることが必要であり、WHERE 文節には該当する結合条件を指定する必要があります。

変更後の照会は次のようになります。

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
   FROM EMPLOYEE E1, DEPARTMENT
   WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
```

```
   AND EDLEVEL > (SELECT AVG(EDLEVEL)
                  FROM EMPLOYEE E2
                  WHERE E2.WORKDEPT = E1.WORKDEPT)
```

このステートメントの結果は、次のとおりです。

```
LASTNAME        DEPTNAME                    EDLEVEL
--------------- --------------------------- -------
HAAS            SPIFFY COMPUTER SERVICE DIV.      18
LUCCHESSI       SPIFFY COMPUTER SERVICE DIV.      19
KWAN            INFORMATION CENTER                20
PIANKA          MANUFACTURING SYSTEMS             17
SCOUTTEN        MANUFACTURING SYSTEMS             17
JONES           MANUFACTURING SYSTEMS             17
LUTZ            MANUFACTURING SYSTEMS             18
PULASKI         ADMINISTRATION SYSTEMS            16
MARINO          ADMINISTRATION SYSTEMS            17
JOHNSON         ADMINISTRATION SYSTEMS            16
HENDERSON       OPERATIONS                        16
SCHNEIDER       OPERATIONS                        17
MEHTA           SOFTWARE SUPPORT                  16
GOUNOT          SOFTWARE SUPPORT                  16
```

上記の例は、副照会で使われる相関名は、その相関副照会を含む照会の FROM 文節で定義しなければならないことを示しています。しかし、そのような包含関係には複数のレベルのネストが関係することがあります。

一部の部署は従業員がごく少数である場合には、そのためその平均教育レベルは正しい結果を導くものとはならないかもしれません。平均教育レベルが従業員の比較の対象として意味のある数値となるようにするため、部署には最低 5 人の従業員が含まれていなければならない、ということにしてみます。それで、5 人以上の従業員が所属している部署だけを考慮しつつ、従業員のうち所属する部署の平均教育レベルよりも高い教育レベルの従業員のリストを出力することになります。

一番外側の照会の中の各従業員ごとに、その従業員の所属する部署の従業員数を数える必要があるため、さらにもう 1 つの副照会が関係してきます。

```
SELECT COUNT(*)
   FROM EMPLOYEE E3
   WHERE E3.WORKDEPT = E1.WORKDEPT
```

カウント結果が 5 以上の場合に限って平均を計算するようにします。

```
SELECT AVG(EDLEVEL)
   FROM EMPLOYEE E2
   WHERE E2.WORKDEPT = E1.WORKDEPT
   AND 5 <= (SELECT COUNT(*)
             FROM EMPLOYEE  E3
             WHERE E3.WORKDEPT = E1.WORKDEPT)
```

最後に、教育レベルが、その部署の平均より高い従業員だけを含めます。

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
   FROM EMPLOYEE E1, DEPARTMENT
   WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
   AND EDLEVEL >
   (SELECT AVG(EDLEVEL)
       FROM EMPLOYEE E2
       WHERE E2.WORKDEPT = E1.WORKDEPT
       AND 5 <=
       (SELECT COUNT(*)
           FROM EMPLOYEE E3
           WHERE E3.WORKDEPT = E1.WORKDEPT))
```

このステートメントの結果は、次のとおりです。

```
LASTNAME        DEPTNAME                    EDLEVEL
--------------- --------------------------- -------
PIANKA          MANUFACTURING SYSTEMS            17
SCOUTTEN        MANUFACTURING SYSTEMS            17
JONES           MANUFACTURING SYSTEMS            17
LUTZ            MANUFACTURING SYSTEMS            18
PULASKI         ADMINISTRATION SYSTEMS           16
MARINO          ADMINISTRATION SYSTEMS           17
JOHNSON         ADMINISTRATION SYSTEMS           16
HENDERSON       OPERATIONS                       16
SCHNEIDER       OPERATIONS                       17
```

第6章 照会の中での演算子と述部の使用

DB2 ユニバーサル・データベースでは、さまざまな*集合演算子* を使って複数の照会を組み合わせたり、*比較述部* を含む複雑な条件ステートメントを構成したりできます。

この章では、次のことについて説明します。

- UNION、EXCEPT、および INTERSECT 集合演算子を使って異なる表を組み合わせる方法。
- 比較述部を使うことによって、照会のための複雑な条件を構成する方法。基本的な述部に関する簡単な説明は、 20ページの『行の選択』にあります。

集合演算子によって複数の照会を組み合わせる

UNION、EXCEPT、および INTERSECT 集合演算子を使うと、単一の照会の中に複数の外側レベルの照会を組み合わせることができます。それらの集合演算子で結合した照会がそれぞれ実行され、それぞれ別個の結果が組み合わされます。各演算子は、それぞれ違った結果を生成します。

UNION 演算子

UNION 演算子を使うと、他の 2 つの表 (たとえば TABLE1 と TABLE2) を組み合わせ、その中の行の重複を除去することによって結果表が生成されます。 UNION と共に ALL を使った場合 (UNION ALL)、行の重複は除去されません。いずれにしても、生成される表の各行は、TABLE1 か TABLE2 のいずれかから取られた行です。

次の例は、UNION 演算子を使うことによって、給与が $21,000 より多い人、または管理職 (MANAGER) で就業年数 (YEARS) が 8 年未満の人の名前を戻す照会の例です。

1

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
UNION
```

2

```
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
ORDER BY ID
```

個々の照会の結果は、次のとおりです。

1

```
ID     NAME
------ ---------
   140 Fraye
   160 Molinare
   260 Jones
```

2

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
   100 Plotz
   140 Fraye
   160 Molinare
   240 Daniels
```

データベース・マネージャーは、2 つの照会の結果を組み合わせ、重複を除去してから、最終結果を昇順で戻します。

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
   100 Plotz
   140 Fraye
   160 Molinare
   240 Daniels
   260 Jones
```

ORDER BY 文節をいずれかの集合演算子と共に使う場合、それは最後の照会より後でなければなりません。 順序は、組み合わせた後の結果セットに適用されます。

2 つの表の中の列名が違っているなら、組み合わされた結果の表には、対応する列の名前が含まれません。列は、それが出現する順序で番号が付けられます。そのため、結果表の順序を指定するのであれば、その列番号を ORDER BY 文節に指定する必要があります。

EXCEPT 演算子

EXCEPT 演算子は、TABLE1 には含まれているが TABLE2 には含まれていない行を取り出し、行の重複をすべて除去することによって結果表を生成します。 EXCEPT と共に ALL を使った場合 (EXCEPT ALL)、行の重複は除去されません。

次の例は、EXCEPT 演算子を使うことによって、給与が $21,000 より多いが、管理職 (MANAGER) ではなく就業年数 (YEARS) が 8 年以上の人の名前を戻す照会の例です。

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
EXCEPT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

個々の照会の結果は、UNION の説明で示したものと同じです。このステートメントの結果は、次のとおりです。

```
ID     NAME
------ ---------
   260 Jones
```

INTERSECT 演算子

INTERSECT 演算子は、 TABLE1 にも TABLE2 にも含まれている行を取り出し、行の重複をすべて除去することによって結果表を生成します。
INTERSECT と共に ALL を使った場合 (INTERSECT ALL)、重複した行は削除されません。

次の例は、INTERSECT 演算子を使うことによって、給与が $21,000 より多く、かつ管理職 (MANAGER) で就業年数 (YEARS) が 8 年未満の従業員の名前と ID を戻す照会の例です。

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
INTERSECT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

個々の照会の結果は、UNION の説明で示したものと同じです。 2 つの照会を INTERSECT で結合した結果は、次のとおりです。

```
ID     NAME
------ ---------
   140 Fraye
   160 Molinare
```

UNION、EXCEPT、および INTERSECT 演算子を使う場合、次のことに注意してください。

- 演算子で結合した各照会の選択リストの中の対応するすべての項目は、互換性のあるものでなければなりません。さらに詳しい情報については、*SQL 解説書* の中のデータ・タイプ互換性の表を参照してください。
- ORDER BY 文節を使用する場合、それは集合演算子で結合した最後の照会より後になければなりません。どの演算子の場合でも、列名を ORDER BY 文節で使うことができるのは、照会の選択リスト中の対応する項目の列名が同じ場合だけです。
- 同じデータ・タイプで同じ長さの列の間での操作では、そのデータ・タイプおよび長さの列が生成されます。 UNION、EXCEPT、および INTERSECT 集合演算子の結果については、 *SQL 解説書* の中の結果データ・タイプの規則の部分を参照してください。

述部

述部は、条件を構成して、それらの条件を満たす行だけが処理されるようにするために使います。基本的な述部については、20ページの『行の選択』で説明されています。ここでは、IN、BETWEEN、LIKE、EXISTS、および比較述部について説明します。

IN 述部の使用

IN 述部は、ある値を他のいくつかの値と比較するのに使います。たとえば、

```
SELECT NAME
   FROM STAFF
   WHERE DEPT IN (20, 15)
```

この例は、次のものと同じです。

```
SELECT NAME
   FROM STAFF
   WHERE DEPT = 20  OR DEPT = 15
```

副照会が値の集合を戻す場合には、IN および NOT IN 演算子を使えます。たとえば、次の照会では、プロジェクト MA2100 と OP2012 を担当する従業員の姓を出力します。

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE EMPNO IN
      (SELECT RESPEMP
          FROM PROJECT
          WHERE PROJNO = 'MA2100'
          OR PROJNO = 'OP2012')
```

副照会は一度評価され、外側の照会の対応する位置がその結果によって置き換えられます。たとえば、上記の副照会で、従業員番号が 10 と 330 の従業員が選択されると、外側の照会は WHERE 文節が次のものになったかのようにして評価されます。

```
WHERE EMPNO IN (10, 330)
```

副照会から戻される値のリストには、0 個、1 個、あるいは複数個の値が含まれる可能性があります。

BETWEEN 述部の使用

BETWEEN 述部は、 1 つの値と一定範囲の値 (BETWEEN 述部で指定) を比較します。

次の例では、給与が $10, 000 以上、 $20, 000 以下の従業員について調べています。

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY BETWEEN 10000 AND 20000
```

これは、次のものと同じです。

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY >= 10000 AND SALARY <= 20000
```

次の例では、給与が $10, 000 未満か、または $20, 000 を超える従業員について調べています。

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY NOT BETWEEN 10000 AND 20000
```

LIKE 述部の使用

LIKE 述部は、特定のパターンにマッチするストリングを検索するのに使います。パターンは、パーセント記号と下線記号を使って指定します。

- 下線文字 (_) は、任意の単一の文字を表します。
- パーセント記号 (%) は、0 個以上の文字ストリングを表します。
- その他の文字は、その文字自体を表します。

次の例では、文字 ’S’ で始まる長さが 7 文字の従業員名を選択しています。

```
SELECT NAME
   FROM STAFF
   WHERE NAME LIKE 'S _ _ _ _ _ _'
```

次の例では、最初の文字が文字 'S' ではない従業員名を選択しています。

```
SELECT NAME
   FROM STAFF
   WHERE NAME  NOT LIKE 'S%'
```

EXISTS 述部の使用

副照会を使うことによって、なんらかの条件を満たす行の*存在* を検査することができます。その場合、その副照会は EXISTS または NOT EXISTS 述部によって外側レベルの照会に結び付けられます。

EXISTS 述部によって副照会を外側照会に結び付ける場合、その副照会から値は戻されません。この場合、副照会の結果セットに 1 行以上の行が含まれているなら EXISTS 述部は真になり、行が含まれていないなら偽になります。

EXISTS 述部は、相関副照会で使われることがよくあります。下の例は、現在、PROJECT 表の中に項目がない部署のリストを出力します。

```
SELECT DEPTNO, DEPTNAME
   FROM DEPARTMENT X
   WHERE NOT EXISTS
            (SELECT *
                FROM PROJECT
                WHERE DEPTNO = X.DEPTNO)
   ORDER BY DEPTNO
```

EXISTS および NOT EXISTS 述部を他の述部と結合するには、外側レベルの照会の WHERE 文節の中で AND や OR を使います。

比較述部

比較述部は、ある値を値の集合と比較します。全選択が複数の値を戻す場合、述部の中の比較演算子に接尾部 ALL、ANY、または SOME を付けるよう変更する必要があります。それらの接尾部は、戻される値セットが外側レベルの述部の中でどのように処理されるかを決めるものです。ここでは、一例として > 比較演算子を使っています (以下の説明はその他の演算子にも適用されます)。

式 > ALL (全選択)

この述部は、式が全選択から戻されるどの値よりも大きい場合に真になります。全選択から何も値が戻されない場合、述部は真です。指定された関係が少なくとも 1 つの値について偽なら、結果は偽になります。 <>ALL という比較述部は、NOT IN 述部と同じことになります。

次の例は、副照会と > ALL 比較を使って、どの管理職よりも給与の多い従業員の姓 (LASTNAME) とその肩書き (JOB) を出力します。

```
SELECT LASTNAME, JOB
   FROM EMPLOYEE
   WHERE SALARY > ALL
   (SELECT SALARY
       FROM EMPLOYEE
       WHERE JOB='MANAGER')
```

式 > ANY (全選択)

この述部は、式が全選択から戻される少なくとも 1 つの値より大きい場合に真になります。全選択から何も値が戻されない場合、述部は偽です。

=ANY は IN 述部と同じことになります。

式 > SOME (全選択)

SOME は ANY の同義語です。

述部と演算子については、 *SQL 解説書* を参照してください。

第7章 高度な SQL

この章では、個々の必要に応じて照会をカスタマイズし、しかも効率のよいものに設計するための DB2 ユニバーサル・データベースの機能について説明します。この章の内容は、このマニュアルのここまでの内容に関する理解に基づいています。

この章の内容は、次のとおりです。

- 制約とトリガーによる業務規則の適用
- 結合
- ROLLUP および CUBE 照会と再帰照会
- OLAP 関数

制約とトリガーによる業務規則の適用

実務では、特定の規則をどんな場合にも適用する必要が生じます。たとえば、あるプロジェクトを担当する従業員は、給与支払い名簿に載っていなければなりません。あるいは、特定の処置を自動的に実行する必要があるかもしれません。たとえば、営業担当者が 1 件契約するごとに、その歩合給に所定の額を加算する必要があります。

DB2 ユニバーサル・データベースには、この目的のために役立つ手段が用意されています。

- *固有制約* は、表の 1 つまたは複数の列での値の重複を禁止します。
- *参照保全制約* は、指定したいくつかの表の相互におけるデータの一貫性を保証するものです。
- *表検査制約* は、列に入れる値を制限するための規則です。列に割り当てる値がその列の検査制約に合わない場合は、挿入も更新も失敗します。
- *トリガー* は、指定した表に対する削除、挿入、または更新操作によって実行される (引き起こされる) 一連のアクションを定義します。トリガーを使うことによって、他の表に書き込んだり、入力値を修正したり、アラート・メッセージを発行したりできます。

最初の部分では、キーに関する重要な概念について説明します。その後、参照保全、制約、およびトリガーについて例や図を使って説明します。

キー

キー は、特定の行を識別したりアクセスしたりするのに使うことのできる列の集まりです。

複数の列で構成されるキーのことを*複合キー* といいます。複合キーの指定されている表では、その複合キー内の列の順序は、その表内での列の順序に対応するとは限りません。

固有キー

*固有*キー は、値がすべて違っている列 (または列集合) です。固有キーの列の内容を NULL 値にすることはできません。この制約は、INSERT および UPDATE ステートメントの実行中にデータベース・マネージャーによって適用されます。 1 つの表に対して、複数の固有キーが可能です。固有キーはオプションであり、CREATE TABLE または ALTER TABLE ステートメントの中で定義できます。

基本キー

*基本*キー は、表の定義の一部を構成している固有キーです。 1 つの表に対して基本キーは 1 つしか定義できず、基本キーを構成する列の内容を NULL 値にすることはできません。基本キーはオプションであり、CREATE TABLE または ALTER TABLE ステートメントの中で定義できます。

外部キー

*外部*キー は、参照制約の定義の中で指定します。 1 つの表に対して、0 個以上の外部キーが可能です。複合外部キーの値のどの成分も NULL である場合、その外部キーの値は NULL です。外部キーはオプションであり、CREATE TABLE ステートメントまたは ALTER TABLE ステートメントの中で定義できます。

固有制約

固有制約は、1 つの表の中でキーの値が固有であることを保証します。固有制約はオプションであり、CREATE TABLE または ALTER TABLE ステートメントの中で PRIMARY KEY または UNIQUE 文節を指定することによって定義します。たとえば、ある表の従業員番号の列に固有制約を定義することによって、どの従業員の番号も固有の番号になるようにすることができます。

参照保全制約

固有制約と外部キーを定義することによって、表と表の間の関係を定義し、それによって特定の業務規則を適用させることができます。固有キーと外部キーの制約の組み合わせは、一般に「参照保全制約」と呼ばれています。外部キー

によって参照される固有制約は、*親キー* と呼ばれます。外部キーは、特定の親キーを参照します (または特定の親キーに関連しています)。たとえば、どの従業員 (EMPLOYEE 表) も既存の部署 (DEPARTMENT 表) に属していなければならない、という規則があるとします。この場合、EMPLOYEE 表の部署番号を外部キーとして定義し、DEPARTMENT 表の部署番号を基本キーとして定義します。参照保全制約について、下記の図に示します。

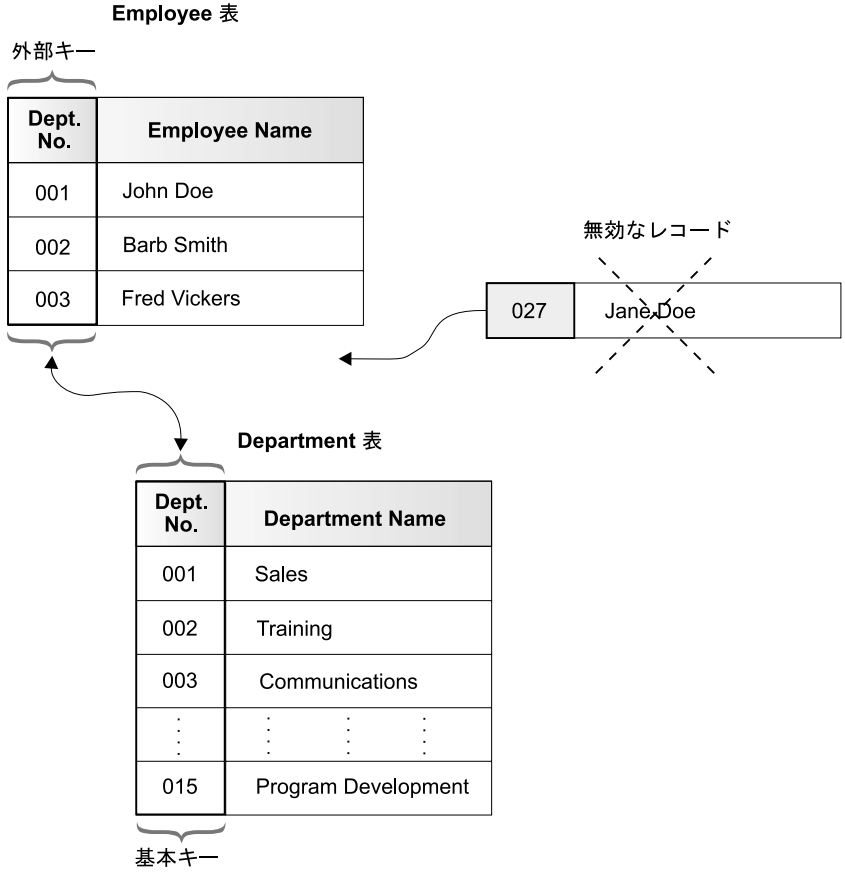


図 4. 外部制約と基本制約により関係を定義しデータを保護する

表検査制約

表検査制約 は、表の各行ごとに評価される条件を指定します。検査制約は、個々の列ごとに指定できます。検査制約は、CREATE または ALTER TABLE ステートメントを使って追加します。

この後に示すステートメントでは、次の制約を含む表が作成されます。

- 部署番号の値は 10〜100 の範囲内でなければならない。
- 従業員の肩書き (JOB) は、“Sales”、“Mgr”、または “Clerk” のいずれかでなければならない。
- 1986 年より前に採用されたどの従業員についても、給与は $40, 500 より多くなければならない。

```
CREATE TABLE EMP
      (ID           SMALLINT NOT NULL,
       NAME         VARCHAR(9),
       DEPT         SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
       JOB          CHAR(5)   CHECK (JOB IN ('Sales', 'Mgr', 'Clerk')),
       HIREDATE     DATE,
       SALARY       DECIMAL(7,2),
       COMM         DECIMAL(7,2),
       PRIMARY KEY (ID),
       CONSTRAINT YEARSAL CHECK
                          (YEAR(HIREDATE) >= 1986 OR SALARY > 40500) )
```

制約違反になるのは、条件の評価結果が偽になる場合だけです。たとえば、DEPT が NULL の行を挿入する場合、DEPT の値が 10〜100 の範囲でなければならないことが制約の中で定義されているとしても、その挿入操作はエラーなしで実行されます。

次のステートメントでは、COMP という名前の制約を EMPLOYEE 表に追加しています。この制約は、従業員の総所得は $15, 000 より大きい値でなければならない、というものです。

```
ALTER TABLE EMP
   ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
```

表に既存の行が新しい制約に違反していないかどうか検査されます。この検査は、下記のように SET CONSTRAINTS ステートメントを使うことによって据え置くことができます。

```
SET CONSTRAINTS FOR EMP OFF
ALTER TABLE EMP ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
SET CONSTRAINTS FOR EMP IMMEDIATE CHECKED
```

まず、SET CONSTRAINTS ステートメントを使って、表に対する制約検査を据え置きます。その後、実際に制約を検査することなく、1 つまたは複数の制約を表に追加できます。その後、もう一度 SET CONSTRAINTS ステートメントを使って、制約検査をオンにすることにより、据え置かれた制約検査を実行します。

トリガー

トリガー は、一連のアクションを定義します。トリガーを活動化するには、指定の基本表のデータを修正する操作を使います。

トリガーには、以下のような用途があります。

- 入力データの妥当性検査を実行すること
- 新しく挿入した行の値を自動的に生成すること
- 相互参照のために他の表から読み込むこと
- 監査証跡のために他の表に書き込むこと
- 電子メール・メッセージによるアラートをサポートすること

トリガーを使うことは、アプリケーション開発の効率を高めたり、業務規則を広い範囲に適用したり、アプリケーションとデータの早期メンテナンスを実現したりすることにつながります。

DB2 ユニバーサル・データベースは、いくつかのタイプのトリガーをサポートしています。トリガーは、DELETE、INSERT、または UPDATE 操作の前または後に活動化されるように定義できます。各トリガーには、*トリガー・アクション* と呼ばれる一連の SQL ステートメントが含まれます。それには、検索条件も含めることができます。

後トリガー の場合は、トリガー・アクションを各行ごとに 1 回実行するか、ステートメント全体で 1 回実行するかをさらに定義できます。*前トリガー* の場合は、必ず各行ごとにトリガー・アクションを実行します。

トリガー操作実行の前に特定の条件を検査する場合、または入力値を表に格納する前にその値を変更する場合は、 INSERT、UPDATE、または DELETE ステートメントの前にトリガーを使います。

必要に応じて値を波及させたり、トリガー操作の一部として必要となるその他の作業 (メッセージ送信など) を実行したりするには、後トリガーを使います。

次の例では、前トリガーと後トリガーの使用例を示します。株価の変動を記録追跡するアプリケーションのことを考えてみましょう。データベースには、CURRENTQUOTE と QUOTEHISTORY という 2 つの表が含まれており、それぞれ次のように定義されています。

```
CREATE TABLE CURRENTQUOTE
(SYMBOL VARCHAR(10),
 QUOTE DECIMAL(5,2),
 STATUS VARCHAR(9))
```

```
CREATE TABLE  QUOTEHISTORY
(SYMBOL VARCHAR(10),
 QUOTE DECIMAL(5,2),
 TIMESTAMP TIMESTAMP)
```

CURRENTQUOTE の QUOTE 列が次のようなステートメントによって更新されたなら、

```
UPDATE CURRENTQUOTE
   SET QUOTE = 68.5
   WHERE SYMBOL = 'IBM'
```

CURRENTQUOTE の STATUS 列を、下記のような株価の変動の状況を反映するように更新する必要があります。

- 値上がり (Rising)
- その年の新高値 (High)
- 値下がり (Dropping)
- その年の新安値 (Low)
- 変化なし (Steady)

これは、次の前トリガーによって実現できます。

1

```
CREATE TRIGGER STOCK_STATUS
   NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
   REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
   FOR EACH ROW MODE DB2SQL
```

2

```
 SET NEWQUOTE.STATUS =
```

3

```
    CASE
```

4

```
       WHEN NEWQUOTE.QUOTE >=
                 (SELECT MAX(QUOTE)
                     FROM QUOTEHISTORY
                     WHERE SYMBOL = NEWQUOTE.SYMBOL
                       AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) )
       THEN 'High'
```

5

```
  WHEN NEWQUOTE.QUOTE <=
              (SELECT MIN(QUOTE)
                  FROM QUOTEHISTORY
                  WHERE SYMBOL = NEWQUOTE.SYMBOL
                  AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) )
  THEN 'Low'
```

6

```
  WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
     THEN 'Rising'
  WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
     THEN 'Dropping'
  WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
     THEN 'Steady'
END
```

1 このコード・ブロックでは、 CURRENTQUOTE 表の QUOTE 列の更新の前に活動化するトリガーとして STOCK_STATUS を定義しています。第 2 行では、CURRENTQUOTE 表の実際の更新によって引き起こされる変更がデータベースに適用される前に、トリガー・アクションを適用するよう指定しています。 NO CASCADE 文節は、このトリガー・アクションが他のトリガーを活動化することはないということを示しています。第 3 行では、新しい値 (NEWQUOTE) と古い値 (OLDQUOTE) の列名の修飾子として使う名前を指定しています。それらの相関名 (NEWQUOTE および OLDQUOTE) で修飾した列名は、*遷移変数* と呼ばれます。第 4 行では、このトリガー・アクションを各行ごとに実行することを指定しています。

2 これは、このトリガーのトリガー・アクションの中の最初で唯一の SQL ステートメントの始まりです。トリガーの中でも、トリガーによって活動化されるステートメントによって更新される表の行の中の列に値を代入するという動作をするトリガーでは、 ″SET 遷移変数″ というステートメントが使われます。このステートメントでは、CURRENTQUOTE 表の STATUS 列に値を代入しています。

3 代入文の右辺に使う式は、1 つの CASE 式です。その CASE 式は、END キーワードのところまで続いています。

4 最初の WHEN では、新しい相場 (NEWQUOTE.QUOTE) が、その年 (カレンダー年) のその銘柄の最高値よりさらに高いかどうかを調べています。その副照会では、この後に後トリガーで更新する QUOTEHISTORY 表を使っています。

5 第 2 の WHEN では、新しい相場 (NEWQUOTE.QUOTE) が、その年 (カレンダー年) のその銘柄の最低値よりさらに低いかどうかを調べて

います。その副照会では、この後に後トリガーで更新する QUOTEHISTORY 表を使っています。

6 最後の 3 つの WHEN では、新しい相場 (NEWQUOTE.QUOTE) を表に保管されていた相場 (OLDQUOTE.QUOTE) と比較して、高いか安いか同じかを調べています。 ″SET 遷移変数″ ステートメントはここで終わっています。

CURRENTQUOTE 表の中の項目を更新するだけでなく、新しい相場とそのタイム・スタンプを QUOTEHISTORY 表にコピーすることによって監査記録を作成する必要があります。これは、次の後トリガーによって実現できます。

1

```
CREATE TRIGGER RECORD_HISTORY
AFTER UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
```

2

```
INSERT INTO QUOTEHISTORY
   VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
END
```

1 このコード・ブロックでは、 CURRENTQUOTE 表の QUOTE 列の更新の後に活動化するトリガーとして RECORD_HISTORY というトリガーを定義しています。第 3 行では、新しい値 (NEWQUOTE) の列名の修飾子として使う名前を指定しています。第 4 行では、このトリガー・アクションを各行ごとに実行することを指定しています。

2 このトリガーのトリガー・アクションに含まれるのは、更新された行のデータ (NEWQUOTE.SYMBOL と NEWQUOTE.QUOTE) と現在のタイム・スタンプを使って、行を QUOTEHISTORY 表に挿入するための単一の SQL ステートメントです。

CURRENT TIMESTAMP は、タイム・スタンプを内容とする特殊レジスターです。特殊レジスターのリストとその説明については、 75ページの『特殊レジスター』を参照してください。

結合

複数の表のデータを組み合わせることを、「表を結合する」といいます。データベース・マネージャーは、指定された表の行のすべての組み合わせを生成します。各組み合わせごとに、*結合条件* が検査されます。結合条件は、いくつかの制限を伴う検索条件です。それらの制限のリストについては、 *SQL 解説書* を参照してください。

結合条件に関係している列のデータ・タイプは、同じものである必要はありませんが、互換性のあるものでなければなりません。結合条件は、他の検索条件と同じようにして評価され、その比較操作には同じ規則が適用されます。

FROM 文節に指定されている表から取られた行が、互いに全く関係のないものであったとしても、結合条件を指定しない場合には、そのすべての組み合わせが戻されます。その結果は 2 つの表の*直積* と呼ばれます。

この後に示す例は、次の 2 つの表に基づいています。それらは、サンプル・データベースの表を簡略化したものですが、サンプル・データベース内には存在していません。それらは、結合一般に関係していることの概要を示すために使われています。 SAMP_STAFF の内容はパートとして雇われているわけではない従業員の名前とその肩書きのリストであり、 SAMP_PROJECT の内容は従業員 (パートとフルタイムの両方) の名前と担当するプロジェクトの名前のリストです。

それらの表は、以下のとおりです。

NAME	PROJ
Haas	AD3100
Thompson	PL2100
Walker	MA2112
Lutz	MA2111

図 5. SAMP_PROJECT TABLE

NAME	JOB
Haas	PRES
Thompson	MANAGER
Lucchessi	SALESREP
Nicholls	ANALYST

図 6. SAMP_STAFF TABLE

次の例では、この 2 つの表の直積を生成しています。結合条件を指定していないため、行のすべての組み合わせが出力されます。

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
   FROM SAMP_PROJECT, SAMP_STAFF
```

このステートメントの結果は、次のとおりです。

```
NAME       PROJ   NAME       JOB
---------- ------ ---------- --------
Haas       AD3100 Haas       PRES
Thompson   PL2100 Haas       PRES
Walker     MA2112 Haas       PRES
Lutz       MA2111 Haas       PRES
Haas       AD3100 Thompson   MANAGER
Thompson   PL2100 Thompson   MANAGER
Walker     MA2112 Thompson   MANAGER
Lutz       MA2111 Thompson   MANAGER
Haas       AD3100 Lucchessi  SALESREP
Thompson   PL2100 Lucchessi  SALESREP
Walker     MA2112 Lucchessi  SALESREP
Lutz       MA2111 Lucchessi  SALESREP
Haas       AD3100 Nicholls   ANALYST
Thompson   PL2100 Nicholls   ANALYST
Walker     MA2112 Nicholls   ANALYST
Lutz       MA2111 Nicholls   ANALYST
```

結合には、大きく分けて*内部結合* と*外部結合* の 2 種類があります。ここまでに上げた例では、内部結合しか使っていません。内部結合では、直積列のうち結合条件を満たすものだけが残されます。一方の表には存在するがもう一方の表には存在しない行の情報は、結果表に含められません。

次の例では、2 つの表の内部結合を生成しています。内部結合では、フルタイムの従業員のうちプロジェクトに割り当てられている従業員が出力されます。

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
   FROM SAMP_PROJECT, SAMP_STAFF
   WHERE SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

この内部結合は、次のようにも指定できます。

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT INNER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

結果は、次のとおりです。

```
NAME       PROJ   NAME       JOB
---------- ------ ---------- --------
Haas       AD3100 Haas       PRES
Thompson   PL2100 Thompson   MANAGER
```

内部結合の結果は、右側の表と左側の表の NAME 列の値が一致している行で構成されています。 'Haas' と 'Thompson' の 2 人は、フルタイム従業員の表 SAMP_STAFF に含まれており、さらにプロジェクトに割り当てられているフルタイムとパートの従業員の表 SAMP_PROJECT にも含まれています。

外部結合は、左側の表の行と右側の表の行 (あるいはその両方に含まれる行) のうち、内部結合には含まれない行と、内部結合とを連結したものです。 2 つの表の外部結合を実行する場合、そのどちらか一方を左側の表、他方を右側の表として割り当てることになります。外部結合には、次の 3 種類のものがあります。

1. ***左外部結合***。これには、内部結合に加えて、左側の表の行のうち内部結合に含まれないものが含まれます。
2. ***右外部結合***。これには、内部結合に加えて、右側の表の行のうち内部結合に含まれないものが含まれます。
3. ***全外部結合***。これには、内部結合に加えて、左側の表の行と右側の表の行のうち内部結合に含まれないものが含まれます。

表示する列を指定するには、SELECT ステートメントを使います。 FROM 文節の中で、第 1 の表の名前の後に、LEFT OUTER JOIN、RIGHT OUTER JOIN、または FULL OUTER JOIN のキーワードを指定します。その後、第 2 の表と ON キーワードを指定します。その ON キーワードの後に、結合する表と表の間の関係を示す結合条件を指定します。

次の例では、SAMP_STAFF が右側の表、SAMP_PROJECT が左側の表です。 LEFT OUTER JOIN を使うことによって、 SAMP_PROJECT の中のすべての従業員 (フルタイムおよびパート) の名前 (NAME) とプロジェクト番号 (PROJ)、およびフルタイムの従業員 (SAMP_STAFF に含まれる従業員) については肩書き (JOB) を出力します。

```
SELECT SAMP_PROJECT.NAME, SAMP_PROJECT.PROJ,
       SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT LEFT OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

このステートメントの結果は、次のとおりです。

```
NAME       PROJ                 NAME       JOB
---------- -------------------- ---------- --------------------
Haas       AD3100               Haas       PRES
Lutz       MA2111               -          -
Thompson   PL2100               Thompson   MANAGER
Walker     MA2112               -          -
```

すべての列に値が含まれている行は、内部結合の結果です。それらは結合条件を満たす行です。 'Haas' と 'Thompson' の 2 人は、SAMP_PROJECT (左側の表) と SAMP_STAFF (右側の表) の両方に含まれています。結合条件が満たされていない行については、右側の表の列が NULL 値になっています。 'Lutz' と 'Walker' は、SAMP_PROJECT 表に含まれているパートの従業員であり、SAMP_STAFF 表には含まれていません。結果セットには、左側の表のすべての行が含まれていることに注意してください。

次の例では、SAMP_STAFF が右側の表、SAMP_PROJECT が左側の表です。RIGHT OUTER JOIN を使うことによって、 SAMP_STAFF の中のすべてのフルタイム従業員の名前 (NAME) と肩書き (JOB)、およびプロジェクトに割り当てられている従業員 (SAMP_PROJECT に含まれる従業員) についてはそのプロジェクト番号 (PROJ) を出力します。

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT RIGHT OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

結果は、次のとおりです。

```
NAME       PROJ                 NAME       JOB
---------- -------------------- ---------- --------------------
Haas       AD3100               Haas       PRES
-          -                    Lucchessi  SALESREP
-          -                    Nicholls   ANALYST
Thompson   PL2100               Thompson   MANAGER
```

左外部結合の場合と同じように、すべての列に値が含まれている行は、内部結合の結果です。それらは結合条件を満たす行です。 'Haas' と 'Thompson' の 2 人は、SAMP_PROJECT (左側の表) と SAMP_STAFF (右側の表) の両方に含まれています。結合条件が満たされていない行については、左側の表の列が NULL 値になっています。 'Lucchessi' と 'Nicholls' は、プロジェクトに割り当てられていないフルタイムの従業員です。その 2 人は SAMP_STAFF には

含まれていますが、SAMP_PROJECT には含まれていません。結果セットには、右側の表のすべての行が含まれていることに注意してください。

次の例では、SAMP_PROJECT 表と SAMP_STAFF 表に対して、FULL OUTER JOIN を使っています。これにより、フルタイム (プロジェクトに割り当てられていない人も含む) の従業員とパートの従業員のすべての名前が出力されます。

```
SELECT SAMP_PROJECT.NAME, SAMP_PROJECT.PROJ,
       SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT FULL OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

結果は、次のとおりです。

```
NAME       PROJ                NAME       JOB
---------- ------------------- ---------- --------------------
Haas       AD3100              Haas       PRES
-          -                   Lucchessi  SALESREP
-          -                   Nicholls   ANALYST
Thompson   PL2100              Thompson   MANAGER
Lutz       MA2111              -          -
Walker     MA2112              -          -
```

この結果には、左外部結合、右外部結合、および内部結合が含まれています。フルタイムとパートのすべての従業員が含まれています。左外部結合および右外部結合の場合のように、結合条件が満たされていない値については、それぞれ該当する列が NULL 値になっています。結果セットには、SAMP_STAFF と SAMP_PROJECT のすべての行が含まれています。

複雑な照会

DB2 ユニバーサル・データベースには、ROLLUP および CUBE を使うことによって、単一の結果セットの中で複数の列をグループ化したり合併したり表示したりする機能が用意されています。この新しい強力な機能によって、SQL によるデータ分析が拡張され単純化されました。

データベースから有用な情報を抽出するための方式には、実にさまざまなものがあります。既存のデータ集合から結果表を生成するための再帰照会も実現可能です。

ROLLUP および CUBE 照会

ROLLUP 操作と CUBE 操作は、照会の GROUP BY 文節の中に指定します。

ROLLUP によるグループ化では、通常のグループ化行のほかに、*小計行* を含む結果セットが生成されます。 CUBE によるグループ化では、ROLLUP による行のほかに、クロス集計行が含まれる結果セットが生成されます。

それで、ROLLUP を使うと、月ごとの個人別売上情報に、月ごとの売上合計と総合計を含めたものが得られます。 CUBE を使うと、さらに個人別の売上合計のための行が含められます。

SQL 解説書 を参照してください。

再帰照会

再帰照会 とは、結果データを反復使用することによって、さらに次の結果を得る照会のことです。これは、ツリーまたはグラフの網羅と考えることができます。

実際の例としては、予約システム、ネットワーク計画、スケジュール設定などがあります。

再帰照会は、それ自身の名前を参照する共通表式を使うことによって構成します。

再帰照会の例については、 *SQL 解説書* を参照してください。

OLAP 関数

オンライン分析処理 (OnLine Analytical Processing: OLAP) 関数は、*ウィンドウ* で囲んだデータに対して列関数操作を実行します。このウィンドウでは、行の区分化や、区分内の行の配列や、*集約グループ* を指定できます。集約グループとは、現在行からの相対位置によって、計算に含める行集合を指定する機能です。そのようなウィンドウを使って、累積合計や移動平均などの操作を実行できます。

ウィンドウを指定して既存の列関数 (SUM や AVG など) を実行するほかに、OLAP 関数では、ランキング (RANK、DENSE_RANK) 操作を実行したり、行の区分化や配列に基づいて行の番号付け (ROW_NUMBER) を行ったりすることができます。

以下の照会例では、部署 15 と 38 について、給与の面から部署内における各従業員のランクを示し、部署内の給与の累積合計を計算します。

```
SELECT NAME, DEPT,
       RANK () OVER (PARTITION BY DEPT ORDER BY SALARY DESC) AS RANK,
       SUM (SALARY) OVER (PARTITION BY DEPT
```

```
                        ORDER BY SALARY DESC
                        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
          AS CUMULATIVE_SUM
   FROM STAFF
   WHERE DEPT IN (15,38)
   ORDER BY DEPT, RANK
```

このステートメントの結果は、次のとおりです。

```
NAME       DEPT      RANK       CUMULATIVE_SUM
---------- --------- ---------- --------------------
Hanes      15                 1             20659.80
Rothman    15                 2             37162.63
Ngan       15                 3             49670.83
Kermisch   15                 4             61929.33
O'Brien    38                 1             18006.00
Marenghi   38                 2             35512.75
Quigley    38                 3             52321.05
Naughton   38                 4             65275.80
Abrahams   38                 5             77285.55
```

第8章 データ操作のカスタマイズと拡張

この章では、DB2 ユニバーサル・データベースでの *オブジェクト指向拡張機能* について簡単に説明します。オブジェクト指向拡張機能を使うことには、たくさんの利点があります。*ユーザー定義タイプ (UDT)* を使えば、アプリケーションで使用できるデータ・タイプを増やすことができ、*ユーザー定義関数 (UDF)* を使えば、アプリケーション特有の関数を作成できます。 UDF は、データ・タイプの一貫した振る舞いとカプセル化を提供することによって、 UDT のメソッド となります。

さらに、*特殊レジスター* と *システム・カタログ* についても説明します。特殊レジスターとは、データベース・マネージャーによって定義される記憶域であり、 SQL ステートメントが参照する情報を保管するために使用されます。特殊レジスターは接続時に設定され、そのアプリケーションの処理だけに使用が限定されます。システム・カタログには、データベース・オブジェクトの論理構造と物理構造についての情報が含まれています。

この章の内容は、次のとおりです。

- ユーザー定義タイプ
- ユーザー定義関数
- ラージ・オブジェクト (LOB)
- 特殊レジスター
- カタログ視点の基礎知識

上記のことに関する詳しい説明は、このマニュアルの範囲を超えています。それについては、*SQL 解説書* と *管理の手引き* を参照してください。

ユーザー定義タイプ

固有タイプ は、内部表現の点では既存のタイプ (その「ソース」タイプ) と同じですが、別のものとみなされて、ほとんどの演算に関して互換性のないユーザー定義データ・タイプです。たとえば、年齢タイプ、体重タイプ、および身長タイプを定義する場合です。それらは意味はみな違いますが、その内部表現としては組み込みデータ・タイプ INTEGER を使います。

次の例は、PAY という固有タイプを作成する例です。

```
CREATE DISTINCT TYPE PAY AS DECIMAL(9,2) WITH COMPARISONS
```

PAY の表現は組み込みデータ・タイプ DECIMAL(9,2) と同じですが、別個のタイプとみなされ、DECIMAL(9,2) やその他のタイプとは互換性がありません。それと同じ固有タイプとしか比較できません。また、DECIMAL を操作するための演算子と関数は、ここでは適用されません。たとえば、PAY データ・タイプの値に INTEGER データ・タイプの値を乗算することはできません。したがって、PAY データ・タイプだけに適用される関数を作成する必要があります。

固有データ・タイプを使うと、うっかりミスをある程度防ぐことができます。たとえば、EMPLOYEE 表の SALARY 列が PAY データ・タイプとして定義されている場合、それと COMM とはソース・タイプが同じではありますが、それを COMM に加算することはできません。

固有データ・タイプではキャストがサポートされています。ソース・タイプを固有データ・タイプにキャストしたり、固有データ・タイプをソース・タイプにキャストしたりできます。たとえば、EMPLOYEE 表の SALARY 列が PAY データ・タイプとして定義されている場合、次の例は比較演算子のところでエラーにはなりません。

```
SELECT * FROM EMPLOYEE
   WHERE DECIMAL(SALARY) = 41250
```

DECIMAL(SALARY) は、10 進数のデータ・タイプを戻します。逆に、数値データ・タイプを PAY タイプにキャストすることもできます。たとえば、PAY(41250) を使うことによっても、数値 41250 をキャストできます。

ユーザー定義関数

28ページの『関数の使用』で触れたように、 DB2 ユニバーサル・データベースには組み込み関数とユーザー定義関数 (UDF) とが用意されています。しかし、それだけの関数ですべての要求を満たすことは到底できません。しばしば、特定の作業を行うためにカスタマイズした関数を作成する必要が生じます。ユーザー定義関数を使うなら、カスタマイズ済み関数を作成できます。

ユーザー定義関数には、ソース (またはテンプレート)、*外部スカラー*、*外部表*、 *OLE DB 外部表* という 4 つのタイプがあります。

ここでは、ソース関数と外部スカラー関数を取り上げます。外部表関数と OLE DB 外部表関数の詳細については、 *SQL 解説書* を参照してください。

ソース・ユーザー定義関数を使うと、ユーザー定義タイプにおいて、データベースで既知である別の組み込み関数またはユーザー定義関数を選択的に参照することが可能になります。スカラー関数と列関数の両方を使うことができます。

次の例では、DECIMAL データ・タイプを入力とする組み込み列関数 MAX に基づいて、 MAX というユーザー定義関数を作成しています。 MAX UDF は、PAY タイプを入力とし、PAY タイプを出力として戻します。

```
CREATE FUNCTION MAX(PAY) RETURNS PAY
   SOURCE MAX(DECIMAL)
```

外部ユーザー定義関数は、プログラミング言語によってユーザーが作成するものです。それには*外部スカラー関数* と*外部表関数* があります。いずれも *SQL 解説書* を参照してください。

別の例として、ストリング中の単語数を数えるための関数をすでに作成してあるなら、 CREATE FUNCTION ステートメントを使うことによって、WORDCOUNT という名前でデータベースにそれを登録できます。そのようにするなら、その関数を SQL ステートメントで使えるようになります。

次のステートメントは、従業員番号 (EMPNO) と、その履歴書の ASCII 形式での単語数を戻します。 WORDCOUNT はユーザーによってデータベースに登録されている外部スカラー関数であり、ここではそれをステートメントの中に使用しています。

```
SELECT EMPNO, WORDCOUNT(RESUME)
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

ユーザー定義関数の作成については、 *アプリケーション開発の手引き* を参照してください。

ラージ・オブジェクト (LOB)

ラージ・オブジェクト の語とその頭字語 *LOB* は、 BLOB、CLOB、または DBCLOB という 3 つのデータ・タイプのことを指して使われます。それらのタイプには、オーディオ、写真、およびワープロ文書などのオブジェクトの大量のデータを入れることができます。

バイナリー・ラージ・オブジェクト *(BLOB)* は、長さが 2GB 以下の可変長 (バイト単位) のストリングです。 BLOB は主として、画像、音声、およびマルチメディアなど、従来のタイプに当てはまらないデータを入れることを意図したものです。

文字ラージ・オブジェクト *(CLOB)* は、長さが 2GB 以下の可変長 (バイト単位) のストリングです。 CLOB は、ワープロ文書などの大規模な単一バイト文字セットのデータを入れるのに使われます。 CLOB は文字ストリングとみなすことができます。

2 バイト文字ラージ・オブジェクト *(DBCLOB)* は、長さが 2GB 以下 (2 バイト文字 1 073 741 823 文字) の可変長 2 バイト文字ストリングです。
DBCLOB は、ワープロ文書などの大規模な 2 バイト文字セットのデータを入れるのに使われます。 DBCLOB はグラフィック・ストリングとみなすことができます。

ラージ・オブジェクト (LOB) の操作

LOB 値は非常に大きいため、それらをデータベース・サーバーからクライアント・アプリケーション・プログラムに転送するには、かなり時間がかかることがあります。しかし、多くの場合、LOB 値は全体としてではなく一度に 1 つの部分が処理されます。 LOB 値の全体をアプリケーション・メモリーに保管することが必要ない、または望ましくないアプリケーションの場合は、その値をラージ・オブジェクト・ロケーター 変数によって参照できます。

それ以降のステートメントでは、そのロケーターを使うことによって、必ずしもラージ・オブジェクトの全体を取り出すことなく、データに対するいろいろな操作を実行することができます。ロケーター変数を使うことによって、クライアントとサーバーの間を流れるデータが少なくなり、その結果、アプリケーションに必要な記憶域を節約でき、パフォーマンスが向上します。

もう 1 つのメカニズムは*ファイル参照変数* です。ラージ・オブジェクトを直接ファイルに取り出したり、表内のラージ・オブジェクトをファイルから直接に更新したりするのに使います。ファイル参照変数を使うなら、ラージ・オブジェクトのデータを保管する必要がないため、アプリケーションに必要な記憶域を節約できます。 *アプリケーション開発の手引き* と *SQL 解説書* を参照してください。

特殊レジスター

特殊レジスター は、データベース・マネージャーによって 1 つの接続に対して定義されている記憶域であり、 SQL ステートメントの中で参照できる情報を格納するのに使われるものです。特殊レジスターのうち、特によく使われるものを以下に示します。すべての特殊レジスターのリストと、それらについての詳しい情報については、 *SQL 解説書* を参照してください。

- CURRENT DATE: SQL ステートメント実行時の時刻に対応する日付。
- CURRENT FUNCTION PATH: 関数とデータ・タイプの参照を変換するために使う関数パスを指定する値。
- CURRENT SERVER: 現行アプリケーション・サーバー。
- CURRENT TIME: SQL ステートメント実行時の時刻に対応する時刻。
- CURRENT TIMESTAMP: SQL ステートメント実行時の時刻に対応する時刻。
- CURRENT TIMEZONE: アプリケーション・サーバーにおける協定世界時とローカル時間の間の差。
- USER: 実行時許可 ID。

特殊レジスターの内容は、VALUES ステートメントを使うことによって表示できます。たとえば、

```
VALUES (CURRENT TIMESTAMP)
```

次のステートメントを使うこともできます。

```
SELECT CURRENT TIMESTAMP FROM ORG
```

この場合、表の中のすべての行項目の TIMESTAMP が戻されます。

カタログ視点の基礎知識

DB2 は、各データベースごとに広範囲の一連のシステム・カタログ表を作成し、維持しています。それらの表には、表、視点、パッケージ、参照保全関係、関数、固有タイプ、およびトリガーなどのデータベース・オブジェクトの論理構造と物理構造についての情報が含まれています。それらの表は、データベースが作成された時点で作成され、通常の操作手順の中で更新されます。明示的にそれらを作成したり除去したりすることはできませんが、その内容を照会したり表示したりすることはできます。

さらに詳しい情報については、 *SQL 解説書* を参照してください。

システム・カタログからの行選択

カタログ視点は、他のデータベース視点とよく似ています。そのデータを参照するには、システム内の他の視点に対してするのとまったく同じようにして SQL ステートメントを使います。

SYSCAT.TABLES カタログに含まれている表に関する情報は、非常に有用です。作成した既存の表の名前を調べるには、次のようなステートメントを使います。

```
SELECT TABNAME, TYPE, CREATE_TIME
   FROM SYSCAT.TABLES
   WHERE DEFINER = USER
```

このステートメントの結果は、次のとおりです。

```
TABNAME            TYPE CREATE_TIME
------------------ ---- --------------------------
ORG                T    1999-07-21-13.42.55.128005
STAFF              T    1999-07-21-13.42.55.609001
DEPARTMENT         T    1999-07-21-13.42.56.069001
EMPLOYEE           T    1999-07-21-13.42.56.310001
EMP_ACT            T    1999-07-21-13.42.56.710001
PROJECT            T    1999-07-21-13.42.57.051001
EMP_PHOTO          T    1999-07-21-13.42.57.361001
EMP_RESUME         T    1999-07-21-13.42.59.154001
SALES              T    1999-07-21-13.42.59.855001
CL_SCHED           T    1999-07-21-13.43.00.025002
IN_TRAY            T    1999-07-21-13.43.00.055001
```

次のリストは、さまざまなカタログ視点のうち、このマニュアルの内容に関係のあるもののリストです。このほかにもたくさんのカタログ視点があります。それらのリストについては、 *SQL 解説書* と*管理の手引き* を参照してください。

説明	カタログ視点
検査制約	SYSCAT.CHECKS
列	SYSCAT.COLUMNS
検査制約で参照されている列	SYSCAT.COLCHECKS
キーに使用されている列	SYSCAT.KEYCOLUSE
データ・タイプ	SYSCAT.DATATYPES
関数のパラメーターまたは関数の結果	SYSCAT.FUNCPARMS
参照制約	SYSCAT.REFERENCES
スキーマ	SYSCAT.SCHEMATA
表制約	SYSCAT.TABCONST
表	SYSCAT.TABLES

説明	カタログ視点
トリガー	SYSCAT.TRIGGERS
ユーザー定義関数	SYSCAT.FUNCTIONS
視点	SYSCAT.VIEWS

付録A. サンプル・データベース表

この付録では、サンプル・データベース SAMPLE のサンプル表に含まれている情報と、これらを作成および除去する方法を説明します。

DB2 ユニバーサル・データベースには追加のサンプル・データベースが提供されており、ビジネス・インテリジェンス機能を例示しています。これは、ビジネス・インテリジェンス・チュートリアルで使用されます。この付録では、サンプル・データベース SAMPLE の内容についてのみ説明します。ビジネス・インテリジェンス・サンプル・データベースについての詳細は、 データウェアハウスセンター *管理の手引き* を参照してください。

このサンプル表は、このマニュアルやこのライブラリーに属する他のマニュアルに含まれている例の中で使われているものです。また、サンプル・ファイルのうち、BLOB および CLOB データ・タイプのデータについても示します。

この付録の内容は、次のとおりです。

96ページの『Adamson の写真』
97ページの『Adamson の履歴書』
98ページの『Walker の写真』
98ページの『Walker の履歴書』

サンプル表のうち、ハイフン (-) は NULL 値を表しています。

サンプル・データベース

このマニュアルの例では、サンプル・データベース (SAMPLE) を使っています。それらの例を使うには、サンプル・データベースを作成する必要があります。それを使うには、データベース・マネージャーがインストールされていなければなりません。

サンプル・データベースを作成する

サンプル・データベースは、実行可能ファイルを使って作成します。[2] データベースを作成するには、SYSADM 権限がなければなりません。

- **UNIX 系のプラットフォームを使用している場合**

  オペレーティング・システムのコマンド・プロンプトを使っている場合は、次のように入力します。

  ```
  sqllib/bin/db2sampl <path>
  ```

  path は、サンプル・データベースの作成先のパスを指定するオプション・パラメーターです。 Enter を押してください。[3] DB2SAMPL のスキーマは、CURRENT SCHEMA 特殊レジスター値です。

- **OS/2 または Windows のプラットフォームを使用している場合**

  オペレーティング・システムのコマンド・プロンプトを使っている場合は、次のように入力します。

  ```
  db2sampl e
  ```

  e は、データベースの作成先のドライブを指定するオプション・パラメーターです。 Enter を押してください。[4]

2. このコマンドについては、*コマンド解説書* の中の DB2SAMPL コマンドの説明を参照してください。

3. path パラメーターを指定しないなら、サンプル・データベースはデータベース・マネージャー構成ファイルの中の DFTDBPATH パラメーターによって指定されるデフォルト・パスに作成されます。

4. ドライブ・パラメーターを指定しないなら、サンプル・データベースは DB2 と同じドライブにインストールされます。

ユーザー・プロファイル管理によってワークステーションにログオンしていないなら、そうするためのプロンプトが出されます。

サンプル・データベースの消去

サンプル・データベースにアクセスする必要がなくなったなら、 DROP DATABASE コマンドを使ってそれを消去できます。

```
db2 drop database sample
```

CL_SCHED 表

名前:	CLASS_CODE	DAY	STARTING	ENDING
タイプ:	char(7)	smallint	time	time
説明:	Class Code (room:teacher)	Day # of 4 day schedule	Class Start Time	Class End Time

DEPARTMENT 表

名前:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
タイプ:	char(3) not null	varchar(29) not null	char(6)	char(3) not null	char(16)
説明:	Department number	Name describing general activities of department	Employee number (EMPNO) of department manager	Department (DEPTNO) to which this department reports	Name of the remote location
値:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
	B01	PLANNING	000020	A00	-
	C01	INFORMATION CENTER	000030	A00	-
	D01	DEVELOPMENT CENTER	-	A00	-
	D11	MANUFACTURING SYSTEMS	000060	D01	-
	D21	ADMINISTRATION SYSTEMS	000070	D01	-
	E01	SUPPORT SERVICES	000050	A00	-
	E11	OPERATIONS	000090	E01	-
	E21	SOFTWARE SUPPORT	000100	E01	-

EMPLOYEE 表

名前:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
タイプ:	char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date
説明:	Employee number	First name	Middle initial	Last name	Department (DEPTNO) in which the employee works	Phone number	Date of hire

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
Job	Number of years of formal education	Sex (M male, F female)	Date of birth	Yearly salary	Yearly bonus	Yearly commission

EMPLOYEE 表の内容については、次のページを参照してください。

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date	char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

EMP_ACT 表

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
タイプ:	char(6) not null	char(6) not null	smallint not null	dec(5,2)	date	date
説明:	Employee number	Project number	Activity number	Proportion of employee's time spent on project	Date activity starts	Date activity ends
値:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01

名前:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

EMP_PHOTO 表

名前:	EMPNO	PHOTO_FORMAT	PICTURE
タイプ:	char(6) not null	varchar(10) not null	blob(100k)
説明:	Employee number	Photo format	Photo of employee
値:	000130	bitmap	db200130.bmp
	000130	gif	db200130.gif
	000130	xwd	db200130.xwd
	000140	bitmap	db200140.bmp
	000140	gif	db200140.gif
	000140	xwd	db200140.xwd
	000150	bitmap	db200150.bmp
	000150	gif	db200150.gif
	000150	xwd	db200150.xwd
	000190	bitmap	db200190.bmp
	000190	gif	db200190.gif
	000190	xwd	db200190.xwd

- 93ページの『Quintana の写真』に、従業員 Delores Quintana の写真を示します。
- 95ページの『Nicholls の写真』に、従業員 Heather Nicholls の写真を示します。
- 96ページの『Adamson の写真』に、従業員 Bruce Adamson の写真を示します。
- 98ページの『Walker の写真』に、従業員 James Walker の写真を示します。

EMP_RESUME 表

名前:	EMPNO	RESUME_FORMAT	RESUME
タイプ:	char(6) not null	varchar(10) not null	clob(5k)

名前:	EMPNO	RESUME_FORMAT	RESUME
説明:	Employee number	Resume Format	Resume of employee
値:	000130	ascii	db200130.asc
	000130	script	db200130.scr
	000140	ascii	db200140.asc
	000140	script	db200140.scr
	000150	ascii	db200150.asc
	000150	script	db200150.scr
	000190	ascii	db200190.asc
	000190	script	db200190.scr

- 94ページの『Quintana の履歴書』に、従業員 Delores Quintana の履歴書を示します。
- 95ページの『Nicholls の履歴書』に、従業員 Heather Nicholls の履歴書を示します。
- 97ページの『Adamson の履歴書』に、従業員 Bruce Adamson の履歴書を示します。
- 98ページの『Walker の履歴書』に、従業員 James Walker の履歴書を示します。

IN_TRAY 表

名前:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
タイプ:	timestamp	char(8)	char(64)	varchar(3000)
説明:	Date and Time received	User id of person sending note	Brief description	The note

ORG 表

名前:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
タイプ:	smallint not null	varchar(14)	smallint	varchar(10)	varchar(13)
説明:	Department number	Department name	Manager number	Division of corporation	City
値:	10	Head Office	160	Corporate	New York
	15	New England	50	Eastern	Boston
	20	Mid Atlantic	10	Eastern	Washington
	38	South Atlantic	30	Eastern	Atlanta
	42	Great Lakes	100	Midwest	Chicago
	51	Plains	140	Midwest	Dallas

名前:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
	66	Pacific	270	Western	San Francisco
	84	Mountain	290	Western	Denver

PROJECT 表

名前:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
タイプ:	char(6) not null	varchar(24) not null	char(3) not null	char(6) not null	dec(5,2)	date	date	char(6)
説明:	Project number	Project name	Department responsible	Employee responsible	Estimated mean staffing	Estimated start date	Estimated end date	Major project, for a subproject
値:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000

名前:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

SALES 表

名前:	SALES_DATE	SALES_PERSON	REGION	SALES
タイプ:	date	varchar(15)	varchar(15)	int
説明:	Date of sales	Employee's last name	Region of sales	Number of sales
値:	12/31/1995	LUCCHESSI	Ontario-South	1
	12/31/1995	LEE	Ontario-South	3
	12/31/1995	LEE	Quebec	1
	12/31/1995	LEE	Manitoba	2
	12/31/1995	GOUNOT	Quebec	1
	03/29/1996	LUCCHESSI	Ontario-South	3
	03/29/1996	LUCCHESSI	Quebec	1
	03/29/1996	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/1996	LEE	Quebec	3
	03/29/1996	LEE	Manitoba	5
	03/29/1996	GOUNOT	Ontario-South	3
	03/29/1996	GOUNOT	Quebec	1
	03/29/1996	GOUNOT	Manitoba	7
	03/30/1996	LUCCHESSI	Ontario-South	1
	03/30/1996	LUCCHESSI	Quebec	2
	03/30/1996	LUCCHESSI	Manitoba	1
	03/30/1996	LEE	Ontario-South	7
	03/30/1996	LEE	Ontario-North	3
	03/30/1996	LEE	Quebec	7
	03/30/1996	LEE	Manitoba	4
	03/30/1996	GOUNOT	Ontario-South	2
	03/30/1996	GOUNOT	Quebec	18
	03/30/1996	GOUNOT	Manitoba	1
	03/31/1996	LUCCHESSI	Manitoba	1
	03/31/1996	LEE	Ontario-South	14
	03/31/1996	LEE	Ontario-North	3

名前:	SALES_DATE	SALES_PERSON	REGION	SALES
	03/31/1996	LEE	Quebec	7
	03/31/1996	LEE	Manitoba	3
	03/31/1996	GOUNOT	Ontario-South	2
	03/31/1996	GOUNOT	Quebec	1
	04/01/1996	LUCCHESSI	Ontario-South	3
	04/01/1996	LUCCHESSI	Manitoba	1
	04/01/1996	LEE	Ontario-South	8
	04/01/1996	LEE	Ontario-North	-
	04/01/1996	LEE	Quebec	8
	04/01/1996	LEE	Manitoba	9
	04/01/1996	GOUNOT	Ontario-South	3
	04/01/1996	GOUNOT	Ontario-North	1
	04/01/1996	GOUNOT	Quebec	3
	04/01/1996	GOUNOT	Manitoba	7

STAFF 表

名前:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
タイプ:	smallint not null	varchar(9)	smallint	char(5)	smallint	dec(7,2)	dec(7,2)
説明:	Employee number	Employee name	Department number	Job type	Years of service	Current salary	Commission
値:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10

名前:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

STAFFG 表

注: STAFFG が作成されるのは、2 バイト・コード・ページの場合だけです。

名前:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
タイプ:	smallint not null	vargraphic(9)	smallint	graphic(5)	smallint	dec(9,0)	dec(9,0)
説明:	Employee number	Employee name	Department number	Job type	Years of service	Current salary	Commission
値:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-

名前:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

BLOB および CLOB データ・タイプのサンプル・ファイル

ここに示すのは、 EMP_PHOTO ファイル (従業員の写真) と EMP_RESUME ファイル (従業員の履歴書) に含まれているデータです。

Quintana の写真

図 7. Delores M. Quintana

Quintana の履歴書

db200130.asc および db200130.scr のファイルに含まれているテキストは、次のとおりです。

Resume: Delores M. Quintana

Personal Information	
Address:	1150 Eglinton Ave Mellonville, Idaho 83725
Phone:	(208) 555-9933
Birthdate:	September 15, 1925
Sex:	Female
Marital Status:	Married
Height:	5’2″
Weight:	120 lbs.
Department Information	
Employee Number:	000130
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-4578
Hire Date:	1971-07-28
Education	
1965	Math and English, B.A. Adelphi University

1960	Dental Technician Florida Institute of Technology

Work History

10/91 - present	Advisory Systems Analyst Producing documentation tools for engineering department.
12/85 - 9/91	Technical Writer Writer, text programmer, and planner.
1/79 - 11/85	COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

Interests

- Cooking
- Reading
- Sewing
- Remodeling

Nicholls の写真

図 8. Heather A. Nicholls

Nicholls の履歴書

db200140.asc および db200140.scr のファイルに含まれているテキストは、次のとおりです。

Resume: Heather A. Nicholls

Personal Information

Address:	844 Don Mills Ave Mellonville, Idaho 83734
Phone:	(208) 555-2310
Birthdate:	January 19, 1946
Sex:	Female
Marital Status:	Single
Height:	5’8″
Weight:	130 lbs.

Department Information

Employee Number:	000140
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-1793
Hire Date:	1976-12-15

Education

1972	Computer Engineering, Ph.D. University of Washington
1969	Music and Physics, M.A. Vassar College

Work History

2/83 - present	Architect, OCR Development Designing the architecture of OCR products.
12/76 - 1/83	Text Programmer Optical character recognition (OCR) programming in PL/I.
9/72 - 11/76	Punch Card Quality Analyst Checking punch cards met quality specifications.

Interests

- Model railroading
- Interior decorating
- Embroidery
- Knitting

Adamson の写真

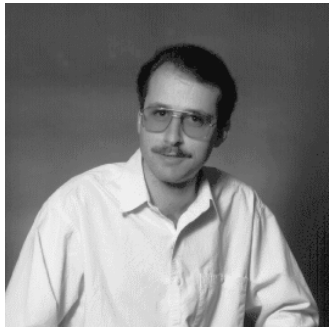


図 9. Bruce Adamson

Adamson の履歴書

db200150.asc および db200150.scr のファイルに含まれているテキストは、次のとおりです。

Resume: Bruce Adamson

Personal Information	
Address:	3600 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-4489
Birthdate:	May 17, 1947
Sex:	Male
Marital Status:	Married
Height:	6’0″
Weight:	175 lbs.
Department Information	
Employee Number:	000150
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-4510
Hire Date:	1972-02-12

Education

1971 Environmental Engineering, M.Sc. Johns Hopkins University

1968 American History, B.A. Northwestern University

Work History

8/79 - present Neural Network Design Developing neural networks for machine intelligence products.

2/72 - 7/79 Robot Vision Development Developing rule-based systems to emulate sight.

9/71 - 1/72 Numerical Integration Specialist Helping bank systems communicate with each other.

Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

Walker の写真

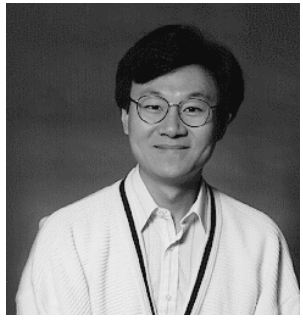


図 10. James H. Walker

Walker の履歴書

db200190.asc および db200190.scr のファイルに含まれているテキストは、次のとおりです。

Resume: James H. Walker

Personal Information

Address:	3500 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-7325
Birthdate:	June 25, 1952
Sex:	Male
Marital Status:	Single
Height:	5’11″
Weight:	166 lbs.

Department Information

Employee Number:	000190
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-2986
Hire Date:	1974-07-26

Education

1974	Computer Studies, B.Sc. University of Massachusetts
1972	Linguistic Anthropology, B.A. University of Toronto

Work History

6/87 - present	Microcode Design Optimizing algorithms for mathematical functions.
4/77 - 5/87	Printer Technical Support Installing and supporting laser printers.
9/74 - 3/77	Maintenance Programming Patching assembly language compiler for mainframes.

Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

付録B. DB2 ライブラリーの使用法

DB2 ユニバーサル・データベース ライブラリーは、オンライン・ヘルプ、ブック (PDF および HTML)、および HTML 形式のサンプル・プログラムから成っています。このセクションでは、ユーザーに提供される情報について紹介し、その入手方法を示します。

オンライン製品情報をご利用になるには、インフォメーション・センターを使用することができます。詳細については、117ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。ここではタスク情報、DB2 ブック、トラブルシューティング情報、サンプル・プログラム、および Web の DB2 情報を見ることができます。

DB2 PDF ファイルおよびハードコピー版資料

DB2 情報

以下に示す表では、DB2 ブックを 4 つのカテゴリーに分類しています。

DB2 の手引きおよび解説書
これらの資料は、すべてのプラットフォームに共通の DB2 情報を含んでいます。

DB2 のインストールおよび構成の情報
これらの資料は、特定のプラットフォーム上の DB2 ごとに用意されています。たとえば、 OS/2、Windows、および UNIX ベースのプラットフォームで稼働するそれぞれの DB2 用に、別個の*概説*および*インストール* 資料が用意されています。

プラットフォーム共通のサンプル・プログラム (HTML 形式)
これらのサンプルは、アプリケーション開発クライアントとともにインストールされるサンプル・プログラムの HTML 版です。これらのサンプルは参考用であり、実際のプログラムに代わるものではありません。

リリース情報
これらのファイルには、DB2 ブックには含められなかった最新の情報が記載されています。

インストール情報、リリース情報、およびチュートリアルは、製品 CD-ROM から HTML 形式で参照することができます。ほとんどの資料は、製品

CD-ROM から HTML 形式で表示できますし、 DB2 の資料 CD-ROM から Adobe Acrobat (PDF) 形式で表示し印刷することができます。 IBM にハードコピー版の資料を注文したい場合は、 113ページの『印刷資料の注文方法』を参照してください。注文可能な資料については、以下の表をご覧ください。

OS/2 および Windows プラットフォームの場合、 HTML ファイルは sqllib¥doc¥html ディレクトリーにインストールできます。 DB2 情報はいくつかの言語で提供されています。しかし、すべての言語に翻訳されているわけではありません。ある言語で情報が提供されていない場合は、英語版の情報が提供されます。

UNIX プラットフォームの場合、言語ごとに異なる複数の HTML ファイルを doc/%*L*/html ディレクトリーにインストールできます。ここで、*%L* は地域を表しています。詳細については、適切な「概説およびインストールの手引き」を参照してください。

DB2 ブックを入手して情報を利用するには、次のようなさまざまな方法があります。

- 116ページの『オンライン情報の表示』
- 121ページの『オンライン情報の検索』
- 113ページの『印刷資料の注文方法』
- 113ページの『PDF 資料の印刷』

表 1. DB2 情報

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
	DB2 の手引きおよび解説書情報		
管理の手引き	*管理の手引き: 計画* は、データベース概念について概説し、設計 (たとえば、論理および物理データベース設計) に関する情報を提供し、高い可用性について解説しています。 *管理の手引き: インプリメンテーション* は、設計、データベースへのアクセス、監査、バックアップ、および回復などのインプリメンテーションについて説明しています。 *管理の手引き: パフォーマンス* は、データベース環境について解説し、さらにアプリケーションのパフォーマンスの評価と調整の方法について説明しています。	第 1 巻 SC88-8513 db2d1x70 第 2 巻 SC88-8511 db2d2x70 第 3 巻 SC88-8512 db2d3x70	db2d0
管理 API 解説書	データベースの管理に使用できる DB2 アプリケーション・プログラミング・インターフェース (API) およびデータ構造について説明します。また、この資料は、アプリケーションから API を呼び出す方法も示します。	SC88-8514 db2b0x70	db2b0
アプリケーション構築の手引き	環境設定に関する情報を提供し、Windows、OS/2、および UNIX ベースのプラットフォームでの DB2 アプリケーションのコンパイル、リンク、実行の各ステップについて説明します。	SC88-8515 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 ユニバーサル・データベース製品をご使用中に発生する可能性のあるセンス・コード APPC、CPI-C、および SNA についての一般情報を提供します。 HTML 形式でのみご利用いただけます。	資料番号なし db2apx70	db2ap

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
アプリケーション開発の手引き	DB2 データベースにアクセスするアプリケーションを、組み込み SQL または Java (JDBC および SQLJ) を使用して開発する方法について説明します。さらに、ストアード・プロシージャーの作成方法、ユーザー定義関数の作成方法、ユーザー定義タイプの作成方法、トリガーの使用方法、区画化されている環境または統合されているシステムでのアプリケーションの開発方法などについて解説されています。	SC88-8516 db2a0x70	db2a0
コール・レベル・インターフェースの手引きおよび解説書	DB2 データベースにアクセスするアプリケーションを、 DB2 コール・レベル・インターフェース (Microsoft ODBC 仕様互換の呼び出し可能 SQL) を使用して開発する方法について説明します。	SC88-8517 db2l0x70	db2l0
コマンド解説書	コマンド行プロセッサーの使用法について説明し、データベースの管理に使用できる DB2 コマンドについて解説しています。	SC88-8518 db2n0x70	db2n0
コネクティビティー 補足	DB2 (AS/400 版)、DB2 (OS/390 版)、DB2 (MVS 版)、または DB2 (VM 版) を DRDA アプリケーション・リクエスターとして DB2 ユニバーサル・データベースとともに使用するためのセットアップ情報および参照情報を提供します。また、この資料は DRDA アプリケーション・サーバーを DB2 コネクト アプリケーション・リクエスターとともに使用する方法の詳細を示します。 HTML と PDF でのみ利用可能	資料番号なし db2h1x70	db2h1
データ移動ユーティリティー 手引きおよび解説書	データの移動を行う DB2 ユーティリティー (インポート、エクスポート、ロード、AutoLoader、および DPROP など) の使用法について説明しています。	SC88-8522 db2dmx70	db2dm

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
データウェアハウスセンター 管理の手引き	データウェアハウスセンターを使用してデータウェアハウスを構築および保守する方法を説明します。	SC88-8545 db2ddx70	db2dd
データウェアハウスセンター アプリケーション統合の手引き	プログラマーがアプリケーションをデータウェアハウスセンターおよび情報カタログ・マネージャーと統合するのに役立つ情報を提供します。	SC88-8546 db2adx70	db2ad
DB2 コネクト 使用者の手引き	DB2 コネクト製品の概念、プログラミング、および一般的な使用方法に関する情報を提供します。	SC88-8521 db2c0x70	db2c0
DB2 クエリー・パトローラー 管理の手引き	DB2 クエリー・パトローラー・システムの運用の概説を行い、運用および管理に関する詳細情報、および管理用グラフィカル・ユーザー・インターフェース・ユーティリティーについてのタスク情報を提供します。	SC88-8525 db2dwx70	db2dw
DB2 クエリー・パトローラー 使用者の手引き	DB2 クエリー・パトローラーのツールや関数の使用方法を説明します。	SC88-8527 db2wwx70	db2ww
用語集	DB2 およびその構成要素で使用される用語の定義を示します。 HTML 形式と SQL 解説書 で利用可能	資料番号なし db2t0x70	db2t0
イメージ、オーディオ、およびビデオ・エクステンダー 管理およびプログラミングの手引き	DB2 エクステンダーの一般情報について提供し、画像、音声、およびビデオ (IAV) エクステンダーの管理と構成について、および IAV エクステンダーを使用したプログラミングについて説明しています。さらに、参照情報、診断情報 (メッセージ解説)、およびサンプルも収録されています。	SC88-8609 dmbu7x70	dmbu7
情報カタログ・マネージャー 管理の手引き	情報カタログを管理するためのガイドです。	SC88-8547 db2dix70	db2di
情報カタログ・マネージャー プログラミングの手引きおよび解説書	情報カタログ・マネージャー用の体系化されたインターフェースの定義を示します。	SC88-8549 db2bix70	db2bi

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
情報カタログ・マネージャー 使用者の手引き	情報カタログ・マネージャー・ユーザー・インターフェースの使用に関する情報を提供します。	SC88-8548 db2aix70	db2ai
インストールおよび構成補足	プラットフォーム固有の DB2 クライアントの計画、インストール、およびセットアップのガイドです。この補足資料には、バインド、クライアント / サーバー通信の設定、 DB2 GUI ツール、DRDA AS、分散インストール、分散要求の構成、および異種データ・ソースへのアクセスについても説明されています。	GC88-8524 db2iyx70	db2iy
メッセージ解説書	DB2、情報カタログ・マネージャー、およびデータウェアハウスセンターから出されるメッセージとコードをリストし、取るべき処置を解説しています。	第 1 巻 GC88-8543 db2m1x70 第 2 巻 GC88-8544 db2m2x70	db2m0
OLAP Integration Server Administration Guide	OLAP Integration Server の Administration Manager 構成要素の使用方法を説明します。	SC27-0782 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	標準の OLAP Metaoutline インターフェースを使用して (Metaoutline Assistant を使用するのではなく) OLAP metaoutline を作成しデータを取り込む方法を説明しています。	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	(Model Assistant ではなく) 標準的な OLAP Model Interface を使用して OLAP モデルを作成する方法を説明します。	SC27-0783 db2lpx70	n/a
OLAP Setup and User's Guide	OLAP Starter Kit の構成およびセットアップに関する情報を提供します。	SC27-0702 db2ipx70	db2ip

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
OLAP Spreadsheet Add-in User's Guide for Excel	Excel 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC27-0786 db2epx70	db2ep
OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3	ロータス 1-2-3 作表計算プログラムを使用して OLAP データを分析する方法を説明します。	SC27-0785 db2tpx70	db2tp
レプリケーションの手引きおよび解説書	DB2 に付属の IBM レプリケーション・ツールの計画、構成、管理、および使用方法に関する情報を提供します。	SC88-8550 db2e0x70	db2e0
地理情報エクステンダー使用者の手引きおよび解説書	地理情報エクステンダーのインストール、構成、管理、プログラミング、およびトラブルシューティングに関する情報を提供します。また、地理情報データの概念についての重要事項を示し、地理情報エクステンダー固有の参照情報 (メッセージおよび SQL) を提供します。	SC88-8624 db2sbx70	db2sb
SQL 概説	SQL の概念を紹介し、構造体とタスクの例を多数提供しています。	SC88-8539 db2y0x70	db2y0
SQL 解説書	SQL の構文、セマンティクス、および言語規則について説明します。また、この資料には、各リリース間の互換性、製品の制限事項、およびカタログ・ビューも含まれます。	第 1 巻 SC88-8540 db2s1x70 第 2 巻 SC88-8657 db2s2x70	db2s0
システム・モニター 手引きおよび解説書	データベースおよびデータベース・マネージャーに関連したさまざまな情報を収集する方法を示します。この資料は、この情報を利用して、データベース活動の把握、パフォーマンス向上、および問題原因の判別を行う方法を説明しています。	SC88-8523 db2f0x70	db2f0

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
テキスト・エクステンダー *管理およびプログラミング*	DB2 エクステンダーの一般情報、テキスト・エクステンダーの管理および構成情報、およびテキスト・エクステンダーを使用したプログラミングの方法について解説します。この資料には、参照情報、診断情報 (メッセージ解説)、およびサンプルが含まれています。	SC88-8610 desu9x70	desu9
問題判別の手引き	エラーの原因の判別、問題からの回復、および DB2 カスタマー・サービスの支援の下での診断ツールの使用法を記載しています。	GD88-7271 db2p0x70	db2p0
新機能	DB2 ユニバーサル・データベース バージョン 7 の新しい機能および拡張機能について説明します。	SC88-8541 db2q0x70	db2q0
DB2 のインストールおよび構成の情報			
DB2 コネクト エンタープライズ・エディション (OS/2 および Windows 版) 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システム版の DB2 コネクト エンタープライズ・エディションで、計画、移行、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8520 db2c6x70	db2c6
DB2 コネクト エンタープライズ・エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 コネクト エンタープライズ・エディションの計画、移行、インストール、構成、およびタスクに関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8519 db2cyx70	db2cy

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 コネクト パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システムの DB2 コネクト パーソナル・エディションで、計画、移行、インストール、および構成を行う場合のタスク情報を提供します。また、この資料はサポートされているすべてのクライアントのインストールおよびセットアップについても説明します。	GC88-8533 db2c1x70	db2c1
DB2 コネクト パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 コネクト パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8528 db2c4x70	db2c4
DB2 データ・リンク・マネージャー (Windows 版) 概説およびインストール	AIX および Windows 32 ビット・オペレーティング・システムの DB2 データ・リンク・マネージャーで、計画、インストール、構成を行う場合の情報を提供します。	GC88-8532 db2z6x70	db2z6
DB2 エンタープライズ拡張エディション (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 エンタープライズ拡張エディションの計画、インストール、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8530 db2v3x70	db2v3
DB2 エンタープライズ拡張エディション (Windows 版) 概説およびインストール	Windows 32 ビット・オペレーティング・システムの DB2 エンタープライズ拡張エディションで、計画、インストール、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8529 db2v6x70	db2v6

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
DB2 ユニバーサル・データベース (OS/2 版) 概説およびインストール	OS/2 オペレーティング・システムでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8534 db2i2x70	db2i2
DB2 ユニバーサル・データベース (UNIX 版) 概説およびインストール	UNIX ベースのプラットフォームでの DB2 ユニバーサル・データベースの計画、インストール、移行、および構成に関する情報を提供します。 また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8536 db2ixx70	db2ix
DB2 ユニバーサル・データベース (Windows 版) 概説およびインストール	Windows 32 ビット オペレーティング・システムの DB2 ユニバーサル・データベースで、計画、インストール、移行、および構成を行う場合の情報を提供します。また、この資料はサポートされている多数のクライアントのインストールおよびセットアップについても説明します。	GC88-8537 db2i6x70	db2i6
DB2 パーソナル・エディション 概説およびインストール	OS/2 および Windows 32 ビット オペレーティング・システム 版の DB2 ユニバーサル・データベース パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8535 db2i1x70	db2i1
DB2 パーソナル・エディション (Linux 版) 概説およびインストール	サポートされる Linux 配布プログラムの DB2 ユニバーサル・データベース・パーソナル・エディションで、計画、インストール、移行、および構成を行う場合の情報を提供します。	GC88-8538 db2i4x70	db2i4
DB2 クエリー・パトローラー インストールの手引き	DB2 クエリー・パトローラーのインストール情報を提供します。	GC88-8526 db2iwx70	db2iw

表 1. DB2 情報 (続き)

資料名	説明	資料番号 PDF ファイル名	HTML ディレクトリー
ウェアハウス・マネージャー *インストールの手引き*	ウェアハウス・エージェント、ウェアハウス・トランスフォーマー、および情報カタログ・マネージャーのインストール情報を提供します。	GC88-8572 db2idx70	db2id
プラットフォーム共通のサンプル・プログラム (HTML 形式)			
サンプル・プログラム (HTML)	DB2 のサポートするすべてのプラットフォームでのプログラム言語用に、サンプル・プログラム (HTML 形式) を提供します。これらのサンプル・プログラムは、参照用としてのみ提供されています。サンプルは、すべてのプログラミング言語で利用できるわけではありません。HTML サンプルが利用できるのは、 DB2 アプリケーション開発クライアントがインストールされている場合だけです。 プログラムの詳細については、*アプリケーション構築の手引き* を参照してください。	資料番号なし	db2hs
リリース情報			
DB2 コネクト リリース情報	DB2 コネクトの資料には含められなかった最新の情報が収録されています。	注 #2 を参照してください。	db2cr
DB2 インストール情報	DB2 ブックには含められなかったインストールに関する最新の情報が収録されています。	製品 CD-ROM からのみ利用できます。	
DB2 リリース情報	DB2 ブックには含められなかった DB2 製品とその機能に関する最新の情報が収録されています。	注 #2 を参照してください。	db2ir

注:

1. ファイル名の 6 桁目の文字 *x* は、その資料の言語を表します。たとえば、ファイル名 `db2d0e70` は、*管理の手引き* の英語版であることを示し、ファイル名 `db2d0f70` は同じ資料のフランス語版を示します。資料の言語を表すためにファイル名の 6 桁目で使用されている文字は以下のとおりです。

言語	識別子
ブラジル・ポルトガル語	b
ブルガリア語	u
チェコ語	x
デンマーク語	d
オランダ語	q
英語	e
フィンランド語	y
フランス語	f
ドイツ語	g
ギリシャ語	a
ハンガリー語	h
イタリア語	i
日本語	j
韓国語	k
ノルウェー語	n
ポーランド語	p
ポルトガル語	v
ロシア語	r
簡体字中国語	c
スロベニア語	l
スペイン語	z
スウェーデン語	s
繁体字中国語	t
トルコ語	m

2. DB2 ブックには含められなかった最新の情報が、「リリース情報」で HTML 形式および ASCII ファイルとして利用できます。 HTML 版は、インフォメーション・センターおよび製品 CD-ROM からご利用になれます。 ASCII ファイルの参照方法:
   - UNIX ベースのプラットフォームでは、ファイル Release.Notes を参照してください。このファイルは DB2DIR/Readme/*%L* ディレクトリーにあります。ここで *%L* は地域名を、 DB2DIR は以下のものを表します。
     - /usr/lpp/db2_07_01 (AIX の場合)
     - /opt/IBMdb2/V7.1 (HP-UX、DYNIX/ptx、Solaris、および Silicon Graphics IRIX の場合)
     - /usr/IBMdb2/V7.1 (Linux の場合)
   - これ以外のプラットフォームでは、ファイル RELEASE.TXT を参照してください。このファイルは、製品がインストールされているディレクトリーにあります。 OS/2 プラットフォームでは、 **IBM DB2** フォルダーをダブルクリックし、 **Release Notes** アイコンをダブルクリックすることもできます。

PDF 資料の印刷

資料のハードコピー版が必要な場合、 DB2 の資料 CD-ROM にある PDF ファイルを印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷することができます。ライブラリー内の各資料のファイルについては、103ページの表1 を参照してください。

Adobe Acrobat Reader の最新版は、Adobe の Web サイト http://www.adobe.com から入手できます。

PDF ファイルは、DB2 の資料 CD-ROM に収録されており、ファイル拡張子 PDF が付いています。 PDF ファイルにアクセスするには以下のようにします。

1. DB2 の資料 CD-ROM を挿入します。 UNIX ベースのプラットフォームの場合は、 DB2 資料 CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。
2. Acrobat Reader を起動します。
3. 以下に示すいずれかの位置から必要な PDF ファイルを開きます。
   - OS/2 および Windows プラットフォームでは:

     x:¥doc¥*language* ディレクトリー。ここで、*x* は CD-ROM ドライブを、*language* は 2 桁の言語を表す国コード (たとえば、EN は英語) を示します。
   - UNIX ベースのプラットフォームでは:

     CD-ROM の */cdrom*/doc/*%L* ディレクトリー。ここで、*/cdrom* は CD-ROM のマウント・ポイントを、 *%L* は地域名を表します。

さらに、PDF ファイルを CD-ROM からローカル・ドライブまたはネットワーク・ドライブにコピーし、そこから参照することもできます。

印刷資料の注文方法

ハードコピー版の DB2 ブックは、個別に注文することができます。資料を注文するには、IBM 承認の販売業者または営業担当員に連絡してください。

DB2 オンライン文書

オンライン・ヘルプへのアクセス

すべての DB2 構成要素で、オンライン・ヘルプを利用できます。以下の表に、さまざまな種類のヘルプを示します。

ヘルプの種類	内容	利用方法
コマンド・ヘルプ	コマンド行プロセッサーのコマンド構文について説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `? command` ここで *command* はキーワードまたはコマンド全体を表します。 たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、`? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプが表示されます。
クライアント構成アシスタントのヘルプ **コマンド・センターのヘルプ** **コントロール・センターのヘルプ** **データウェアハウスセンターのヘルプ** **イベント・アナライザーのヘルプ** **情報カタログ・マネージャーのヘルプ** **サテライト管理センターのヘルプ** **スクリプト・センターのヘルプ**	そのウィンドウまたはノートブックで実行できるタスクについて説明します。このヘルプは、知っておく必要のある概説および前提条件に関する情報を含みます。また、ウィンドウやノートブックの制御の使用方法を示します。	ウィンドウまたはノートブックから、**「ヘルプ (Help)」**押しボタンをクリックするか、または **F1** キーを押します。

ヘルプの種類	内容	利用方法

メッセージ・ヘルプ	メッセージの原因、および取るべき処置を説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。
		`? XXXnnnnn`
		ここで、*XXXnnnnn* は有効なメッセージ識別子を表します。
		たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。
		一度に 1 画面分のメッセージ・ヘルプを表示させるには、次のように入力します。
		`? XXXnnnnn | more`
		メッセージ・ヘルプをファイルに保管するには、次のように入力します。
		`? XXXnnnnn > filename.ext`
		ここで、*filename.ext* はメッセージ・ヘルプを保管するファイルを表します。

SQL ヘルプ	SQL ステートメントの構文について説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `help statement` ここで、*statement* は SQL ステートメントを表します。 たとえば、help SELECT と入力すると、 SELECT ステートメントのヘルプが表示されます。 **注:** UNIX ベースのプラットフォームでは、SQL ヘルプを利用できません。
SQLSTATE ヘルプ	SQL 状態およびクラス・コードについて説明します。	コマンド行プロセッサーの対話モードから、次のように入力します。 `? sqlstate` or `? class code` ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。 たとえば、`? 08003` によって SQL 状態 08003 のヘルプが表示され、 `? 08` によってクラス・コード 08 のヘルプが表示されます。

オンライン情報の表示

この製品に付属のブックは、ハイパーテキスト・マークアップ言語 (HTML) ソフトコピー形式です。ソフトコピー形式では情報を検索または表示したり、ハイパーテキスト・リンクを利用して関連情報に移動したりすることができます。また、1 つの端末を超えてライブラリーを容易に共用することができます。

オンライン・ブックやサンプル・プログラムは、 HTML バージョン 3.2 仕様に準拠するすべてのブラウザーを使って表示できます。

オンライン・ブックまたはサンプル・プログラムは、次のようにして表示します。

- DB2 管理ツールを実行している場合、インフォメーション・センターを使用します。
- ブラウザーで、**ファイル (File) —>ページを開く (Open Page)** をクリックします。次のようなページを開いて、DB2 情報に関する説明とリンクを表示してください。
  - UNIX ベースのプラットフォームでは、以下のページを開きます。

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    ここで *%L* はロケール名です。
  - その他のプラットフォームでは、以下のページを開きます。

    sqllib¥doc¥html¥index.htm

    パスは DB2 がインストールされているドライブです。

    インフォメーション・センターをインストールしていない場合、 **DB2 Information** アイコンをダブルクリックしてページを開くことができます。このアイコンは、ご使用のシステムに応じて、製品のメイン・フォルダー内または Windows 「スタート」メニューにあります。

Netscape ブラウザーのインストール

システムに Web ブラウザーがインストールされていない場合、製品の箱の中にある Netscape CD-ROM から Netscape をインストールすることができます。インストールに関する詳細な説明については、以下を参照してください。

1. Netscape CD-ROM を挿入します。
2. UNIX ベースのプラットフォームでは、CD-ROM をマウントします。マウントの手順については、*概説およびインストール* を参照してください。

3. インストールの手順については、 CDNAV*nn*.txt ファイルを参照します。ここで、*nn* は 2 桁の言語識別子を表します。ファイルは CD-ROM のルート・ディレクトリーにあります。

インフォメーション・センターを使用した情報へのアクセス

インフォメーション・センターを使用すると、DB2 製品情報にすばやくアクセスすることができます。インフォメーション・センターは、DB2 管理ツールを使用できるすべてのプラットフォームで利用できます。

インフォメーション・センターは「インフォメーション・センター (Information Center)」アイコンをダブルクリックすることによってオープンできます。このアイコンのある場所はシステムによって異なります。メイン・プロダクト・フォルダーか Windows の**「スタート」**メニューのどちらかです。

Windows プラットフォームの DB2 では、ツールバーおよび**ヘルプ**・メニューを使用して、インフォメーション・センターにアクセスすることもできます。

インフォメーション・センターは 6 種類の情報を提供します。適切なタブをクリックすると、種類ごとに提供されているトピックが表示されます。

タスク (Tasks)
DB2 を使用して実行できる主要なタスク。

参照 (Reference)
DB2 参照情報 (キーワード、コマンド、API など)。

ブック (Books)
DB2 ブック。

トラブルシューティング (Troubleshooting)
エラー・メッセージのカテゴリーと、メッセージに対する回復処置。

サンプル・プログラム (Sample Programs)
DB2 アプリケーション開発クライアントに付属のサンプル・プログラム。 DB2 アプリケーション開発クライアントをインストールしていない場合、このタブは表示されません。

Web
WWW 上にある DB2 情報。この情報にアクセスするには、ご使用のシステムから Web への接続が必要です。

リストから項目を 1 つ選択すると、インフォメーション・センターはビューアーを立ち上げて情報を表示します。選択した情報の種類に応じて、ビューアーはシステム・ヘルプ・ビューアー、エディター、または Web ブラウザーです。

インフォメーション・センターには検索機能が備わっており、リストを参照せずに特定のトピックを探すことができます。

テキストの全検索を行うには、インフォメーション・センター内のハイパーテキスト・リンク「**DB2 オンライン情報の検索 (Search DB2 Online Information)**」検索フォームに従います。

通常、HTML 検索サーバーは自動的に始動します。 HTML 情報の検索がうまくいかない場合は、以下の方法の 1 つを使用して、検索サーバーを始動しなければならない場合もあります。

Windows では

「スタート」をクリックし、**「プログラム」—>「IBM DB2」—>「Information」—>「Start HTML Search Server」**を選択します。

OS/2 では

「DB2 (OS/2 版)」フォルダーをダブルクリックして、**「Start HTML Search Server」**アイコンをダブルクリックします。

HTML 情報の検索でこの他の問題が発生した場合は、リリース情報を参照してください。

注: 検索機能は、Linux、DYNIX/ptx、および Silicon Graphics IRIX 環境では利用できません。

DB2 ウィザードの使用

ウィザードを使用すると、各タスクをステップごとに進めることによって、さまざまな管理タスクを遂行することができます。ウィザードは、コントロール・センターおよびクライアント構成アシスタントを通して使用できます。以下の表では、ウィザードとその目的をリストしています。

注: データベース作成、索引作成、複数サイト更新の構成、およびパフォーマンス構成ウィザードは、区分データベース環境で使用できます。

ウィザード	内容	利用方法
データベース追加 (Add Database)	クライアント・ワークステーション上にデータベースのカタログを作成します。	クライアント構成アシスタントから、「**追加 (Add)**」をクリックします。

ウィザード	内容	利用方法
データベース・バックアップ (Back up Database)	バックアップ計画を決定、作成、およびスケジュールします。	「コントロール・センター (Control Center)」からバックアップするデータベースを右クリックし、**「バックアップ (Backup)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
複数サイト更新の構成 (Configure Multisite Update)	複数サイト更新、分散トランザクション、または 2 フェーズ・コミットを構成します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「複数サイト更新 (Multisite Update)」**を選択します。
データベース作成 (Create Database)	データベースを作成し、いくつかの基本的な構成タスクを実行します。	「コントロール・センター (Control Center)」から、**「データベース (Databases)」**フォルダーを右クリックして、**「作成 (Create)」—> 「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。
表作成 (Create Table)	基本的なデータ・タイプを選択して、表の基本キーを作成します。	「コントロール・センター (Control Center)」から、**「表 (Tables)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する表 (Table Using Wizard)」**を選択します。
表スペース作成 (Create Table Space)	新しい表スペースを作成します。	「コントロール・センター (Control Center)」から、**「表スペース (Table Spaces)」**アイコンを右クリックして、**「作成 (Create)」—> 「ウィザードを使用する表スペース (Table Space Using Wizard)」**を選択します。
索引作成 (Create Index)	すべての照会について、作成すべき索引および除去すべき索引を提案します。	「コントロール・センター (Control Center)」から、**「索引 (Index)」**アイコンを右クリックして、**「作成 (Create)」—>「ウィザードを使用する索引 (Index Using Wizard)」**を選択します。

ウィザード	内容	利用方法
パフォーマンス構成 (Performance Configuration)	ビジネス要件に適合するように構成パラメーターを更新して、データベースのパフォーマンスを調整します。	「コントロール・センター (Control Center)」から、調整したいデータベースを右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。 区分データベース環境では、「Database Partitions」視点から、調整したい最初のデータベース区画を右クリックして、**「ウィザードを使用するパフォーマンスの構成 (Configure Performance Using Wizard)」**を選択します。
データベース復元 (Restore Database)	障害の後、データベースを回復します。どのバックアップを使用し、どのログを再生するかの判別を支援します。	「コントロール・センター (Control Center)」から復元するデータベースを右クリックし、**「復元 (Restore)」—>「ウィザードを使用するデータベース (Database Using Wizard)」**を選択します。

文書サーバーのセットアップ

デフォルトでは、DB2 情報はローカル・システムにインストールされます。つまり、DB2 情報にアクセスする必要のある各担当者が同じファイルをインストールする必要があります。 DB2 情報を 1 か所に格納するには、次のようにします。

1. ¥sqllib¥doc¥html のすべてのファイルとサブディレクトリーを、ローカル・システムから Web サーバーにコピーします。各ブックには独自のサブディレクトリーがあり、そのブックを構成する必要な HTML および GIF ファイルが入っています。ディレクトリー構造は常に同じ状態に保つ必要があります。
2. Web サーバーを構成して、ファイルを新しい場所で検索するようにします。さらに詳しい情報については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。
3. インフォメーション・センターの Java バージョンをご使用の場合は、すべての HTML ファイルのベース URL を指定できます。この URL はブックのリストに使用してください。

4. 資料ファイルが表示されるようになったなら、よく使うトピックにはブックマークを付けておいてください。ブックマークを付けるページは、たとえば以下のものがあります。
   - ブックのリスト
   - 頻繁に使用されるブックの目次
   - 頻繁に参照する情報 (たとえば、ALTER TABLE トピックなど)
   - 検索フォーム

中央のマシンから DB2 ユニバーサル・データベース オンライン文書ファイルを提供する方法については、 *インストールおよび構成 補足* の NetQuestion 付録を参照してください。

オンライン情報の検索

HTML ファイルの情報を検索するには、以下の方法のどれか 1 つを使用してください。

- 最上部にある**「検索 (Search)」**をクリックします。検索フォームを使用して特定のトピックを見つけます。この機能は、 Linux、DYNIX/ptx、または Silicon Graphics IRIX 環境ではご利用になれません。
- 最上部にある**「索引 (Index)」**をクリックします。索引を使用して、ブック内の特定のトピックを見つけます。
- HTML 資料またはヘルプの目次あるいは索引を表示してから、 Web ブラウザーの検索機能を利用して資料内の特定のトピックを見つけます。
- Web ブラウザーのブックマーク機能を使用して、特定のトピックにすばやく戻ります。
- インフォメーション・センターの検索機能を使用して、特定のトピックを検索します。詳しくは、117ページの『インフォメーション・センターを使用した情報へのアクセス』を参照してください。

付録C. 特記事項

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミングまたはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミングまたはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書で、 IBM ライセンス・プログラムまたは他の IBM 製品に言及している部分があっても、このことは当該プログラムまたは製品のみが使用可能であることを意味するものではありません。これらのプログラムまたは製品に代えて、 IBM の知的所有権を侵害することのない機能的に同等な他社のプログラム、製品またはサービスを使用することができます。ただし、IBM によって明示的に指定されたものを除き、これらのプログラムまたは製品に関連する稼働の評価および検証はお客様の責任で行っていただきます。

IBM および他社は、本書で説明する主題に関する特許権 (特許出願を含む)、商標権、または著作権を所有している場合があります。本書は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。実施権、使用権等の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
AP 事業所
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

本書に含まれる情報には、技術的に不正確なもの、または誤植が含まれる場合があります。これらに対する変更は、定期的に行われます。これらの変更は、資料の改訂版に含まれます。 IBM は、本書で説明している製品、プログラムに対して、予告なく改良、変更を加える場合があります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するもので

はありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様になんら義務も負わせない適切な方法で、使用もしくは配布することがあります。

本プログラムのライセンス保持者で、 (i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

本プログラムに関する上記の情報は、適切な条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、 IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

本書に含まれるパフォーマンス・データは、制御された環境下で決定されています。したがいまして、その他の稼働環境で得られる結果とは、かなり異なる可能性もあります。一部の測定値は、開発中のシステムを使用している場合があり、これらの測定値が一般的に提供可能なシステムで同様の数値になることを保証するものではありません。さらに、一部の測定値が推定されたものもあります。実測値と異なる場合があります。本書のユーザーは、使用される特定の環境での該当データを確認してください。

IBM 以外の製品については、当該製品の提供者から直接、出版されている資料または一般公開されている情報から入手しました。 IBM は、これらの製品についてはテストを行っておらず、これらの IBM 以外の製品に関する性能、互換性またはその他の主張について確認することはできません。 IBM 以外の製品の機能に対する質問は、それぞれの製品提供者にお問い合わせください。

IBM の将来の方向性または意図については、予告なしに変更または中止する場合があり、 IBM の目的および目標のみを示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれていますが、これは説明に具体性を与えるために記載されたものであり、それらの例には、個人、企業、ブランドの、あるいは製品などの名前が含まれている場合があります。それらの名前はすべて架空のものであり、また名称や住所が類似する企業が実在しても、それは偶然に過ぎません。

著作権：

本書に含まれる情報には、サンプル・アプリケーション・プログラムがソース言語の形式で含まれており、様々な、オペレーティング・プラットフォームでのプログラミング技法を示しています。お客様は、これらのサンプル・プログラムが書かれているオペレーティング・プラットフォームでアプリケーション・プログラミング・インターフェースが実行可能となるためのアプリケーション・プログラムを開発、使用、販売または配布もしくは転送する目的のためにのみ、サンプル・プログラムを、IBM に対する別途料金を支払うことなく、複製、変更、配布または転送することができます。これらのサンプルは、すべての条件下で十分にテストを行っていません。したがいまして、IBM は、これらのプログラムの信頼性、実用性または機能について、いかなる保証も負いません。

サンプル・プログラムまたはその改変版の複製物には、全部複製か部分複製かを問わず、次の著作権表示を必ず行うものとします。

商標

次のものは、 IBM Corporation の米国およびその他の国における商標です。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

次のものは、他社の商標または登録商標です。

Tivoli およびその NetView は、米国およびその他の国における Tivoli Systems Inc. の商標です。

Microsoft、Windows、Windows NT、および Windows ロゴは Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group がライセンスしている米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標または登録商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

［ア行］

［カ行］

［サ行］

［タ行］

［ナ行］

［ハ行］

［マ行］

［ヤ行］

［ラ行］

［数字］

A

B

C

D

E

F

G

H

I

L

N

O

P

R

S

T

U

V

W

IBM と連絡をとる

技術上の問題がある場合は、時間をとって「問題判別の手引き」に定義されている処置を検討し、それらの提案を実行した後で、DB2 顧客サービスに連絡をとってください。この資料には、DB2 顧客サービスがお客さまを支援するために必要とする情報が説明されています。

製品情報

以下の情報は英語で提供されます。内容は英語版製品に関する情報です。

http://www.ibm.com/software/data/

DB2 World Wide Web ページには、ニュース、製品説明、研修スケジュールなどの DB2 に関する最新情報が提供されています。ただし、提供されている情報は英語です。

http://www.ibm.com/software/data/db2/library/

「DB2 Product and Service Technical Library」では、よくされる質問 (FAQ)、修正内容、資料、および最新の DB2 技術情報などの情報へのアクセスが提供されています。

注: この情報のご提供は英語のみとなりますのでご注意ください。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

「International Publications」注文用 Web サイトでは、マニュアルの注文方法についての情報を提供しています。ただし、提供されている情報は英語です。

http://www.ibm.com/education/certify/

IBM の「Professional Certification Program」Web サイトでは、 DB2 を含むさまざまな IBM 製品の認証テストの情報を提供しています。ただし、提供されている情報は英語です。

ftp.software.ibm.com

匿名でログオンしてください。ディレクトリー `/ps/products/db2` には、 DB2 および多数の他製品に関連したデモ、修正プログラム、情報、およびツールがあります。ただし、提供されている情報は英語です。

comp.databases.ibm-db2, bit.listserv.db2-l

これらのインターネット・ニュースグループは、ユーザーが DB2 製品に関する自分の経験について話し合うために利用できます。ただし、提供されている情報は英語です。

Compuserve: GO IBMDB2

このコマンドを入力すると、IBM DB2 Family forum にアクセスできます。すべての DB2 製品が、このフォーラムでサポートされています。ただし、提供されている情報は英語です。

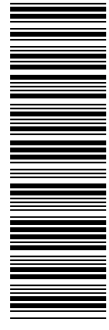
米国以外の国で IBM に連絡する方法については、 **IBM Software Support Handbook** の Appendix A を参照してください。この資料にアクセスするには、 Web ページ: http://www.ibm.com/support/ にアクセスし、ページの最下部にある「IBM Software Support Handbook」リンク・ボタンを選択します。

注: 国によっては、 IBM が承認している販売業者が、 IBM サポート・センターの代わりにそれら販売業者のサポート・センターに連絡する場合があります。

部品番号: CT7YHJA

Printed in Japan

CT7YHJA



SC88-8539-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12