

DB2 Universal Database V8 対応版

DB2 アドバイザー試験対策セミナー



セミナーテキスト

DB2 Universal Database V8 対応版 DB2 アドバイザー 試験対策セミナー

2003 年 6 月 第 1 版発行

編集・発行 日本アイ・ビー・エム株式会社
ソフトウェア・エデュケーション・サービスセンター
東京都品川区西五反田 8-1-5

(C)Copyright IBM Japan, Ltd. 2003
All Rights Reserved. Printed in Japan.

日本アイ・ビー・エム株式会社 の文書による同意なく本製品およびテキストの一部または全部の無断転載、無断複写することを、禁止いたします。

本トレーニングテキストの内容は、DB2 Universal Database Version 8.1 で検証しています。

本トレーニングテキストの内容は、予告なく変更されることがあります。

Lotus、Lotus Notes、Domino は IBM Corporation および Lotus Development Corporation の商標または登録商標です。

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

次のものは、他社の商標または登録商標です。

Microsoft、Windows、Windows NT、Internet Explorer、Word、PowerPoint は Microsoft Corporation の米国及びその他の国における登録商標です。

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。

このコースウェアについて

対象

「DB2 アドバイザー試験対応セミナー」は、RDB の操作経験のない方で、DB2 アドバイザー認定試験に合格するスキルを身につけたい方を対象としています。

前提条件

- (必須) Windows によるコンピューターの基本操作に慣れている方。

概要

「DB2 アドバイザー試験対応セミナー」は、RDB の操作経験のない方が RDB の基本や DB2 UDB V8 の概要について理解し、DB2 アドバイザー認定試験に合格するためのスキルを学習します。

形式と期間

「DB2 アドバイザー試験対応セミナー」は、1日間の研修 (実習なし) として実施されます。

コースウェアで扱うトピック

このコースでは、次のトピックを取り扱っています。

- リレーショナル・データベースの概要
- DB2 UDB の製品コンポーネント
- DB2 UDB のデータベース・オブジェクト
- 基本的な SQL の利用
- インスタンスとセキュリティ
- データベースの同時実行制御

目次

このコースウェアについて
目次

レッスン 1 リレーショナル・データベースの概要

1-1. データベースとは	2
1-2. リレーショナル・データベースの特徴	5
練習問題:	9

レッスン 2 DB2 UDB の製品コンポーネント

2-1. DB2 UDB の機能と製品コンポーネント	12
2-2. DB2 UDB の GUI ツール	16
練習問題:	20

レッスン 3 基本的な SQL の利用

3-1. SQL の基本	22
3-2. データベースへの接続	25
3-3. 基本的な表の照会	26
練習問題:	31
3-4. 検索条件の指定	32
3-5. 演算と関数の利用	37
練習問題:	43
3-6. データのグループ化	44
3-7. 操作の順序	46
3-8. 表の結合	47
3-9. 集合演算	50
3-10. 副照会	54
練習問題:	55
3-11. データの挿入・変更・削除	56
練習問題:	60

レッスン 4 データベース・オブジェクト

4-1. 表、視点(ビュー)、索引	62
4-2. 表やビューの作成 削除	66
練習問題:	69
4-3. データ記憶管理	70
4-4. DB2 のデータ型	73
4-5. システム・カタログ・ビュー	75
練習問題:	77

レッスン 5 インスタンスとセキュリティ

5-1. インスタンスと管理サーバー	80
5-2. オブジェクトの権限と特権	81
5-4. データの参照と保全性	85
練習問題:	89

レッスン 6 データベースの同時実行制御

6-1. トランザクション 並行性	92
6-2. ロック	93
練習問題:	97

付録

SAMPLE データベースのデータ	付録-2
練習問題の解答	付録-4



レッスン 1

リレーショナル・データベースの概要

このレッスンでは、次のトピックを学習します。

- 1-1. データベースとは
- 1-2. リレーショナル・データベースの特徴

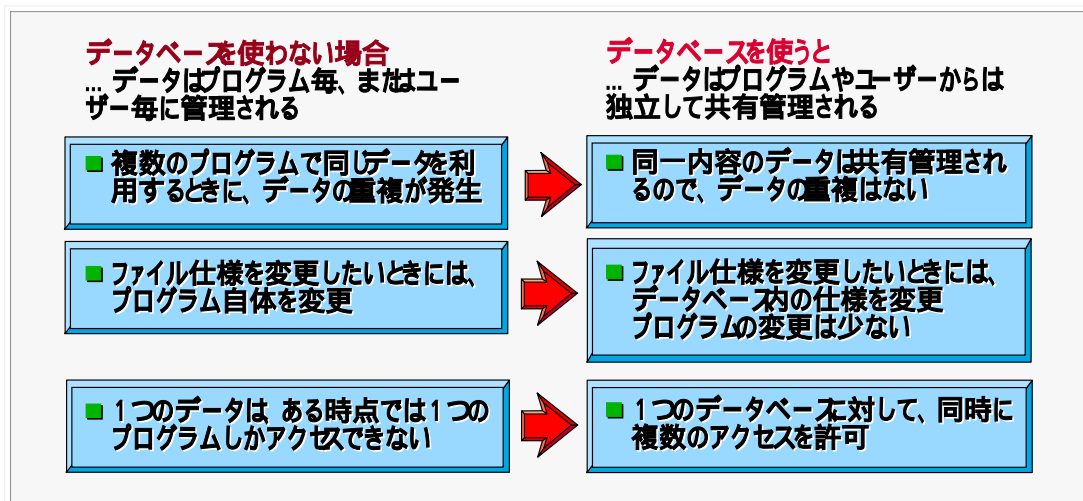
レッスン 1 リレーショナル・データベースの概要

1-1. データベースとは

1-1-1. データベースとは

データベースの基本的な考え方は、データやデータ仕様など共有資源の一元管理の手法から発しています。

次の図は、データベースの定義について説明しています。



データベースを操作、監視、運用するしくみは、データベース管理システム (DBMS: DataBase Management System) によって提供されます。

メモ: DB2 UDB や Oracle など一般的なデータベース系ソフトウェア製品は、データベースではなくデータベース管理システム (DBMS) です。

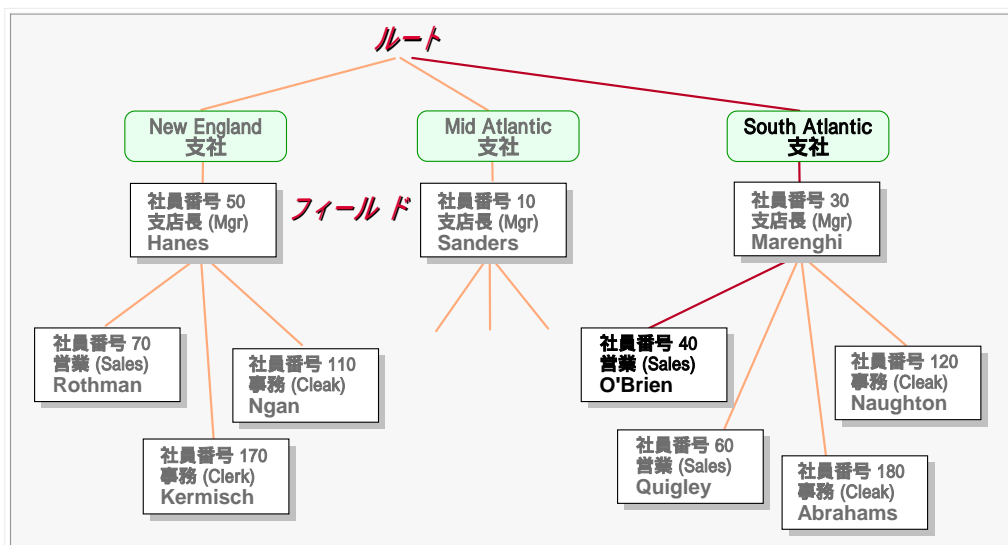
1-1-2. データベースのデータ構造モデル

データベースの構造を示すデータ・モデルには、大きく分けて次の 3 種類があります。

- 階層型モデル
- ネットワーク型 (網型) モデル
- リレーショナル型 (関係型) モデル

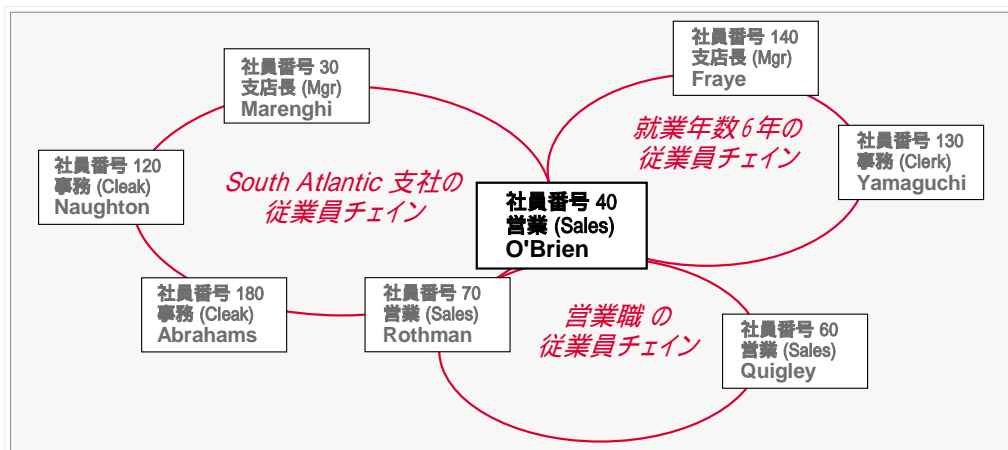
階層型モデル

- データは 木構造 (ツリー構造) で表します。
- 1 つの情報が複数の情報と親子関係を持っていますが、1 つの情報について親となる情報は 1 つだけ (情報へのアクセス経路は 1 つだけ) です。
- データの配置など、物理的な構造が検索時の処理速度に影響します。



ネットワーク型モデル

- データを ネットワーク構造 で表します。
- 1 つの情報について親となる情報は 1 つとは限りません。情報へのアクセス経路は複数存在し、ネットワークをたどって検索します。
- データの配置など、物理的な構造が検索時の処理速度に影響します。



リレーショナル型モデル

- データを列と行からなる 2次元の表 で表します。
- 情報へのアクセス経路は複数存在します。
- データ構造は論理的に定義され、アクセス時の処理速度は物理的な構造に直接影響されません。
- データの重複排除などデータ相互の関係を示すルールが明確に定義され、演算機能など高度なデータ操作を行えます。

ID	NAME	DEPT	JOB	YEARS
10	Sanders	20	Mgr	7
20	Pernal	20	Sales	8
30	Marenghi	38	Mgr	5
40	O'Brien	38	Sales	6
50	Hanes	15	Mgr	10
60	Quigley	38	Sales	10
70	Rothman	15	S	DEPT
:	:	:	:	:

表 (STAFF 表)

DEPT	DEPTNAME	MANAGER	LOCATION
10	Head Office	160	New York
15	New England	50	Boston
20	Mid Atlantic	10	Washington
38	South Atlantic	30	Atlanta
42	Great Lakes	100	Chicago
:	:	:	:

表 (ORG 表)

ORG表を参照

メモ: リレーショナルデータベースの詳細については、次のトピックで説明します。

レッスン 1 リレーショナル・データベースの概要

1-2. リレーショナル・データベースの特徴

1-2-1. リレーショナル・データベースのデータ構造

リレーショナル・データベースの理論は、1970年、IBM 研究所の E.F. Codd 博士により発表された原理です。複数の表の関係が、論理的に表現されるため管理しやすいのが特徴です。

リレーショナル型データベースは、表 (テーブル)、列、行 によって表現されます。列と行で表される値は、原則として複数の値を含みません。

表 (STAFF 表)

ID	NAME	DEPT	JOB	YEARS
10	Sanders	20	Mgr	7
20	Pernal	20	Sales	8
30	Marenghi	38	Sales	5
40	O'Brien	38	Sales	6
50	Hanes	15	Mgr	10
60	Quigley	38	Sales	-
70	Rothman	15	Sales	7
:	:	:	:	:

列 (Column) の注釈: 列のラベル (ID, NAME, DEPT, JOB, YEARS) は属性 (アトリビュート) と呼ばれます。

行 (Row) の注釈: 行のラベル (O'Brien, Sales, 38, 6) は値 (Value) と呼ばれます。

リレーショナル・データベースのデータ型

リレーショナル・データベースの列は、同じデータ型の値の集合です。データ型には、日付型、文字列型、数値型などの種類があります。

リレーショナル・データベースのデータ型については、標準的な仕様の規格がありますが、実際には、各データベース製品により大きく異なっています。データ設計を行う際には、使用するデータベース製品の仕様を予め確認しておく必要があります。

メモ: DB2 UDB のデータ型については、レッスン3で説明します。

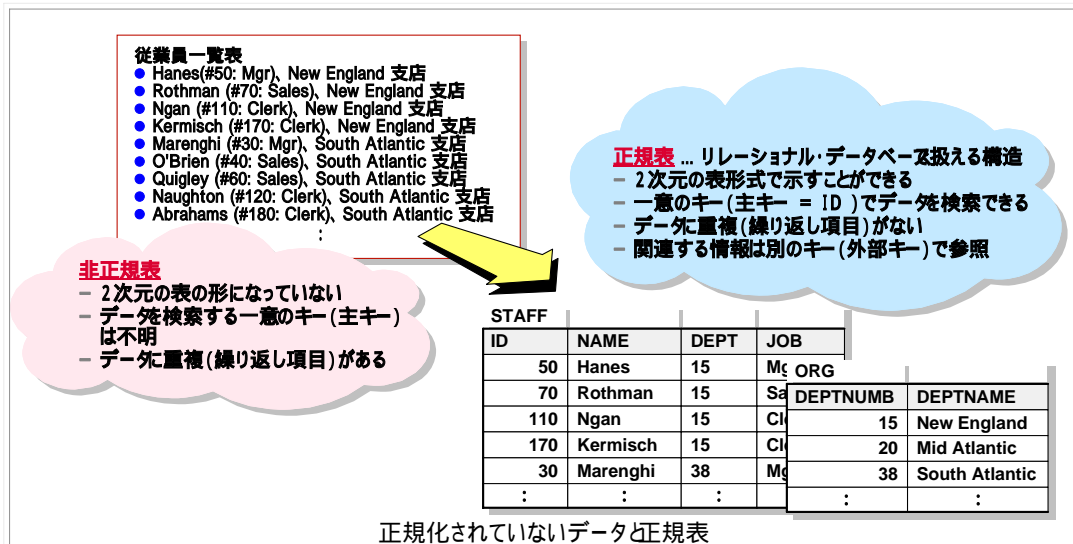
1-2-2. 表の正規化

現実世界の情報をデータベースで扱うデータの構造にするためには、情報の整理が必要です。

たとえば、ある会社には、支社ごとに次のような従業員が所属していたとします。

New England 支社	Hanes(#50: Mgr), Rothman(#70: Sales), Ngan(#110: Clerk), Kermisch(#170: Clerk)
Mid Atlantic 支社	Sanders(#10: Mgr), Pernal(#20: Sales), James(#80: Clerk), Sneider(#190: Clerk)
South Atlantic 支社	Marenghi(#30: Mgr), O'Brien(#40: Sales), Quigley(#60: Sales), Naughton(#120: Clerk), Abrahams(#180: Clerk)
Great Lakes 支社	Plotz(#100: Mgr), Koonitz(#90: Sales), Yamaguchi(#130: Clerk), Scoutten(#200: Clerk)

このような情報は、そのままの形ではリレーショナル型データベースでは取り扱うことができません。リレーショナル型データベースで情報を扱うためには、前項で述べたような特徴に基づき、データの関係を整理しておきます。このことを、表の **正規化** と呼びます。



1-2-3. 主キーと外部キー

リレーショナルデータベースでは、1つの表内に、行を特定するための一意の値を持つ列 (値の重複しない列) が必要です。表内で行を特定する列の値のことを、**主キー**と呼びます。

また、表内で行を識別するために、重複しない値の組み合わせを持つ列を指定しなければなりません。こうした列の値のことを、表の **主キー** と呼びます。

表 (STAFF 表)

ID	NAME	DEPT	JOB	YEARS
10	Sanders	20	Mgr	7
20	Pernal	20	Sales	8
30	Marenghi	38	Mgr	5
40	O'Brien	38	Sales	6
50	Hanes	15	Mgr	10
60	Quigley	38	Sales	-
70	Rothman	15	Sales	7
:	:	:	:	:

注: 主キーとなる列は 1 つ、または複数列の組み合わせで指定することができます。

表の正規化を行う過程では、重複したデータを持つ行が少なくなるように、表を分割して別の表を作成します。分割された新しい表は、分割の要になる列を主キーとし、元表からはこの列の値で参照することができます。このような外部表参照用の値を、**外部キー** と呼びます。

たとえば、下の例では、STAFF 表の主キーは「ID」、ORG 表を参照するための外部キーは「DEPT」です。また、ORG 表では、「DEPTNUMB」が主キーとなっています。

表 (STAFF 表)

ID	NAME	DEPT	JOB	YEARS
10	Sanders	20	Mgr	7
20	Pernal	20	Sales	8
30	Marenghi	38	Mgr	5
40	O'Brien	38	Sales	6
50	Hanes	15	Mgr	10
60	Quigley	38	Sales	-
70	Rothman	15	Sales	7
:	:	:	:	:

表 (ORG 表)

DEPTNUMB	DEPTNAME	MANAGER	LOCATION
10	Head Office	160	New York
15	New England	50	Boston
20	Mid Atlantic	10	Washington
38	South Atlantic	30	Atlanta
42	Great Lakes	100	Chicago
:	:	:	:

注: 外部キーは、STAFF 表の DEPT 列と ORG 表の DEPTNUMB 列の値を指します。

1-2-4. リレーショナル・データベースのデータ操作とルール

データベース問い合わせ言語

リレーショナル・データベースを操作するためには、一般的に、SQL 言語を使用します。SQL には、次のような特徴があります。

- ISO、ANSI、JIS により標準規格化 例: SQL-92/ SQL-99
- 会話型の実行環境 (インタプリタ) で、DBMS からは独立
- SQL を介してプログラミングすることにより、プログラムの変更部分は少なく、データの独立性が保たれる

メモ: SQL 言語の詳細については、レッスン4で説明します。

データの一貫性と制約

リレーショナル・データベースでは、外部表の参照などデータの論理関係に矛盾や不都合が生じないことが必要とされます。

従って、データの重複を許さない (例: UNIQUE 指定)、不確定なデータを許さない (例: NOT NULL 指定)、不正な値を許さない (制約) など、データの一貫性を守るためのさまざまなしくみが用意されています。

これらのしくみは、通常、テーブルの作成時にデータベース管理者によって設定されます。

メモ: UNIQUE、NOT NULL、制約の詳細については、レッスン5で説明します。

1-2-5. リレーショナル・データベースのその他の特徴

リレーショナル・データベースには、次のような機能が必要とされています。

- **同時実行制御機能**
複数の端末から同時に更新を要求することにより問題が発生しないようにする機能
- **トランザクション管理機能**
データの作業をしている途中になんらかの原因で更新が行えない場合に、意図的に作業内容を破棄するなど、作業単位を管理する機能
- **障害回復機能**
データベースがなんらかのトラブルにみまわれた時にデータが消失、またはデータの整合性が損なわれることを防ぐために、バックアップを行う機能
- **機密保護機能**
アクセスするユーザーごとに参照できるデータを制御する機能

これらの機能は、通常、データベース管理システム (DBMS: DataBase Management System) によって実現されます。

メモ: DB2 UDB や Oracle などのデータベース関連のソフトウェア製品は、データベース管理システム (DBMS:DataBase Management System)と呼ばれます。

レッスン 1 リレーショナル・データベースの概要**練習問題:**

リレーショナル型データベースについて説明している次の文章のうち、誤っているものはどれですか？

- A. 行と列からなる2次元の表であらわす
- B. 複数の同時アクセスに対応するしきみを持つ
- C. 複数の経路でネットワークをたどって検索できる
- D. データの論理関係に矛盾や不都合が生じないためのしきみを持つ

SQL に関する次の説明文のうち、正しいものはどれですか？

- A. ANSI により標準規格化されている
- B. System Query Language の略である
- C. 階層型のデータベースで利用されている
- D. IBM DB2 UDB 独自の高度なプログラミング言語である

- メモ -



レッスン 2

DB2 UDB の製品コンポーネント

このレッスンでは、次のトピックを学習します。

- 2-1. DB2 UDB の機能と製品コンポーネント
- 2-2. DB2 UDB の GUI ツール

レッスン 2 DB2 UDB の製品コンポーネント

2-1. DB2 UDB の機能と製品コンポーネント

2-1-1. DB2 ユニバーサル・データベース

DB2 ユニバーサル・データベース (以下、DB2 UDB と記載) は、クライアント/ サーバー型の RDBMS (リレーショナル型データベース管理システム) です。

DB2 UDB には、次のような特長があります。

- **高いスケーラビリティ**
 - モバイル端末から超並列マシンまで、あらゆる規模のユーザーに対応します。
 - 導入初期の小規模運営から業務拡大に伴うシステム拡張まで、同一アプリケーションを継続的に運用できます。
 - ソース・コードのベースを統一することにより、プラットフォームの差異に左右されない共通の機能や一貫した環境を提供します。
 - システム拡張に伴うデータ量やユーザーの増加の際に、ノードやクラスターを追加するだけで容易に対応可能です。
- **ビジネス・インテリジェンス機能**
 - データ・ウェアハウス機能 ... 手軽にデータ・ウェアハウス構築を始めることのできる機能 (データ・ウェアハウス・センター) を提供しています。
 - 透過アクセス機能 ... 他社製データベースにアクセスするためのテーブル関数や、オプション製品である DB2 リレーショナル・コネクタの利用により、Oracle や Microsoft SQL Server、Sybase への透過アクセス機能を提供します。
- **強力な e-business 対応**
 - IBM WebSphere Studio Site Developer Advanced版を標準装備し、データベース・アプリケーションのインターネット/ イン트라ネット化を容易に実現します。
 - リレーショナル・データベースとして業界で初めて Web サービスとの連携機能をサポートします。
- **豊富なデータ対応**
 - XML エクステンダーを使用して、XML (eXtended Markup Language) 文書を新しいデータ型として定義し格納することができます。
 - 文章、イメージといった従来のデータはもちろん、動画、音声、地理情報といった豊富なマルチメディア・データに対応します。
- **GUI ツール群**
 - 快適にデータベースを利用するために、データベース管理者やユーザーの双方に対してさまざまな機能を提供する統合ビジュアル・ツールが用意されています。
- **使いやすい開発環境**
 - 業界標準の SQL-99 をサポートします。
 - ディベロップメント・センターのサポートにより、効果的なアプリケーション開発を行うことができます。
 - IBM WebSphere StudioやMicrosoft Visual Studio との統合により、慣れ親しんだアプリケーション開発環境を使用して DB2 のアプリケーション開発が可能です。

2-1-2. DB2 UDB 製品構成

DB2 製品群は、**DB2 ファミリー** とも呼ばれています。DB2 製品ファミリーは、小さい携帯用の装置から最大の IBM メインフレームに至るまで、広範囲にわたるリレーショナル・データベース・ソリューションを提供します。

DB2 ファミリーには、次の主要製品があります。

DB2 ユニバーサル・データベース (DB2 UDB)

DB2 UDB には、次のデータベース・サーバー製品群が用意されています。

- パーソナル・エディション (PE)
- ワークグループ・サーバー・エディション (WSE)
- ワークグループ・サーバー・アンリミテッド・エディション (WSUE)
- エンタープライズ・サーバー・エディション (ESE)

DB2 コネクト

DRDA プロトコルを使用して、ホスト・データベースにアクセスする機能を提供します。DB2 UDB では、次の DB2 コネクト製品群が用意されています。

- DB2 コネクト パーソナル・エディション (CPE)
- DB2 コネクト エンタープライズ・エディション (CEE)
- DB2 コネクト アンリミテッド・エディション (CUE)
- DB2 コネクト アプリケーション・サーバー・エディション (CASE)

DB2 UDB 開発者エディション

データベース・アプリケーションを開発し、テストする機能を提供します。DB2 UDB では、次の開発者版製品群が用意されています。

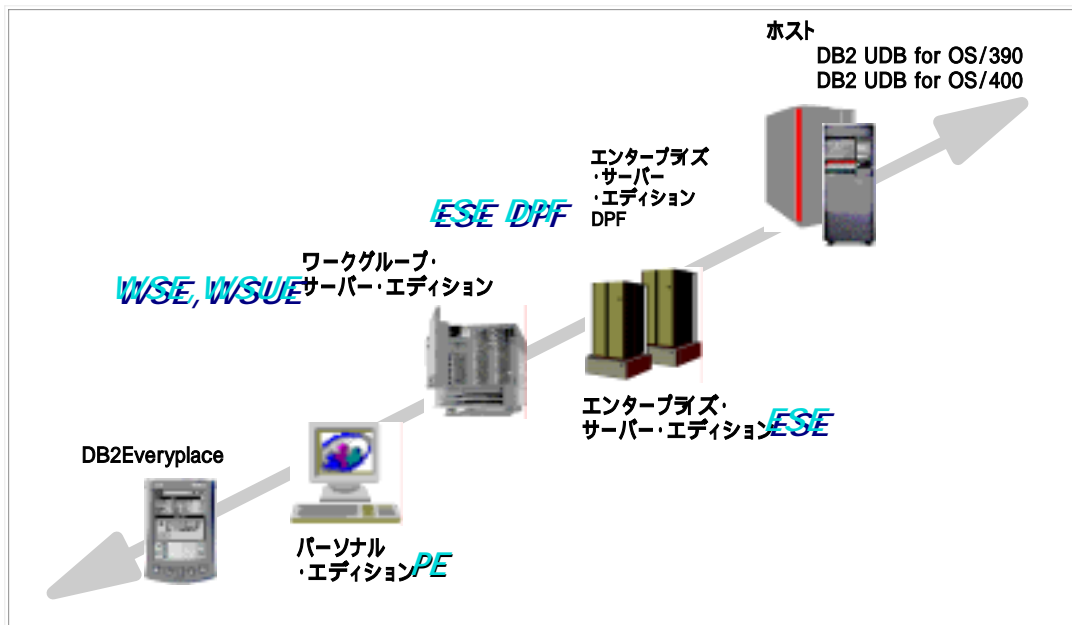
- DB2 パーソナル開発者版 (PDE)
- DB2 ユニバーサル開発者版 (UDE)

この他、モバイル端末から双方向にデータベース間のデータ同期を行うことのできる DB2 Everyplace があります。

2-1-3. DB2 UDB サーバー製品群

DB2 ユニバーサル・データベースには、次の製品パッケージがあります。

DB2 UDB エディション	説明
パーソナルエディション (PE)	<ul style="list-style-type: none"> リモート・クライアントからのアクセスをサポートしません。
ワークグループ・サーバー・エディション (WSE, WSUE)	<ul style="list-style-type: none"> リモート・クライアントからのアクセスをサポートします。 DB2 コネクト製品を利用すると、DRDA ホスト接続を行えます。 アンリミテッド・エディション (WSUE) では、無制限ユーザーをサポートします。
エンタープライズ・サーバー・エディション (ESE)	<ul style="list-style-type: none"> リモート・クライアントからのアクセスをサポートします。 無制限ユーザーをサポートします。 DRDA ホスト接続の機能をサポートします。
エンタープライズ・サーバー・エディション データベース・パーティショニング・フィーチャー (ESE DPF)	<ul style="list-style-type: none"> ESE と同様の機能に加え、区分化データベースによる超並列処理をサポートします。



以下の表は、DB2 ユニバーサル・データベースの製品パッケージごとにサポートするプラットフォームの一覧を示しています。

エディション	Windows 95/98, Me	Windows NT/Windows 2000	Linux	AIX	HP-UX	Solaris
パーソナル						
ワークグループ						
エンタープライズ						
エンタープライズ DPF						

[注意]: 対応する OS のバージョン等、詳細については製品カタログ等を確認してください。

2-1-4. DB2 製品のコンポーネント

DB2 UDB 製品のコンポーネントは、次のとおりです。

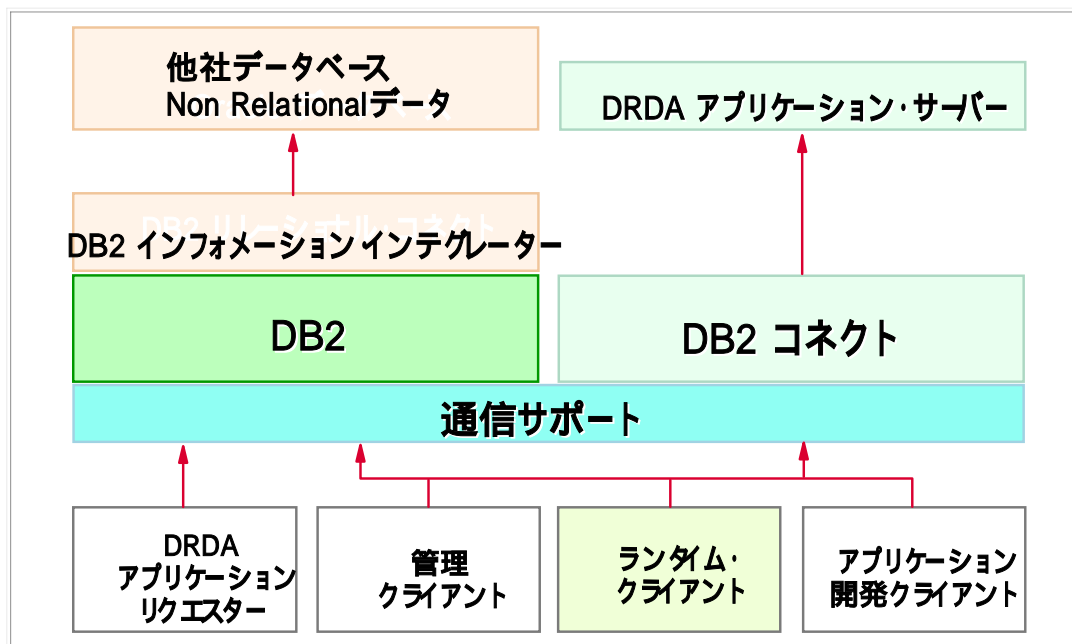
クライアントのコンポーネント

- **ランタイム・クライアント** (実行クライアント) ... リモートでのアプリケーションの実行や、DB2 データベース・サーバーへのアクセスを実現
- **アプリケーション開発クライアント** ... 各プラットフォーム用の開発環境を提供
- **管理クライアント** ... DB2 UDB の管理権限をサポート

通信サポート ... DB2 のサーバー / クライアント環境を実現します。

サーバー側のコンポーネント

- **DB2 コネクト** ... DRDA アプリケーション・リクエスター (AR) 機能を含み、OS/390・OS/400 用など DRDA 準拠のデータベース管理システムへのデータアクセスを実現
- **DB2 インフォメーション・インテグレーター** ... 他社データベース、Non-Relationalなデータへのアクセスを実現



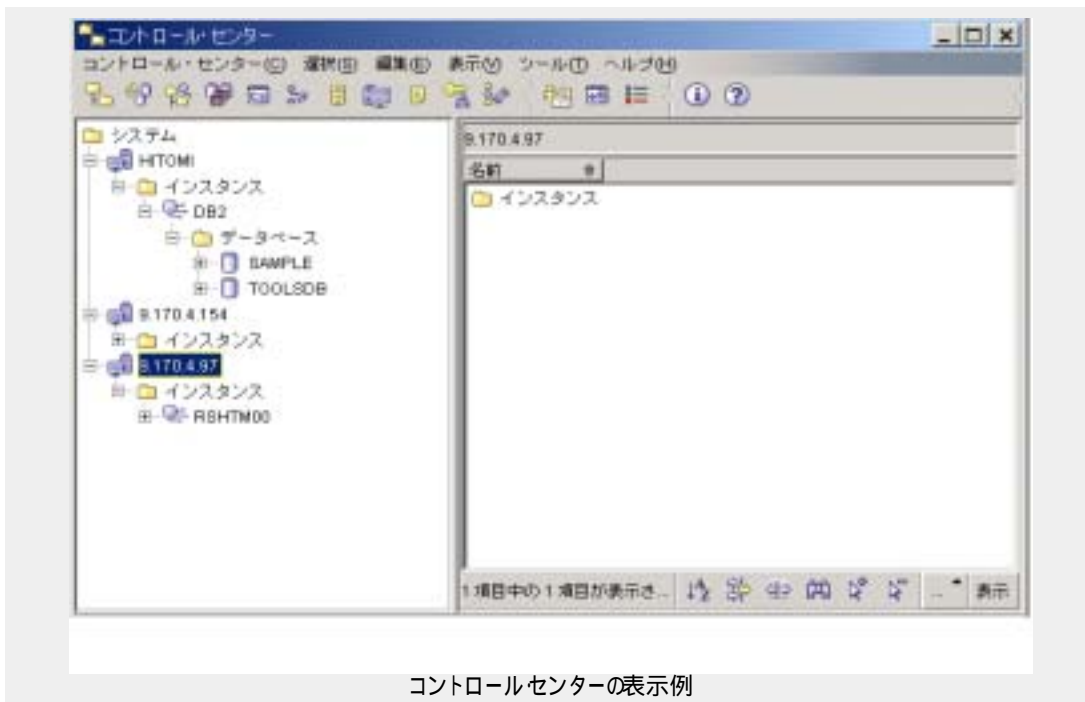
レッスン 2 DB2 UDB の製品コンポーネント

2-2. DB2 UDB の GUI ツール

DB2 UDB 製品には、GUI によってデータベースを快適に管理するための多彩なビジュアル・ツールが用意されています。

2-2-1. コントロール・センターを使用したデータベースの管理

コントロール・センター を使用すると、データベース・サーバーのインスタンスや、データベース・オブジェクト (表、ビュー、表スペースなど) の管理を、統合された 1 つのグラフィカル・インターフェースから行うことができます。



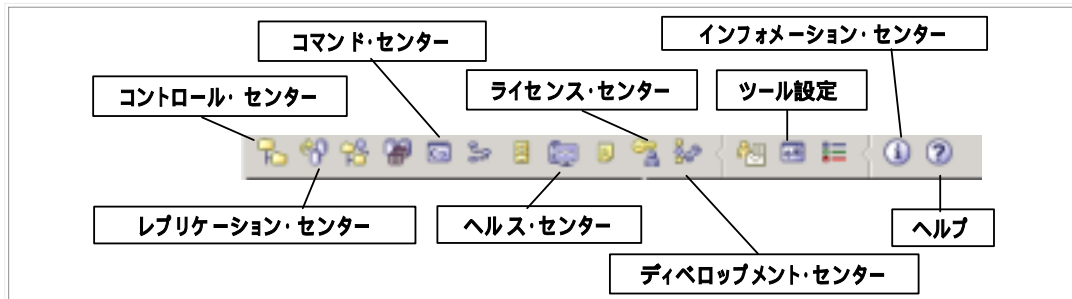
コントロールセンターの表示例

コントロール・センターを利用すると、データベース・オブジェクトについて次のような操作を行えます。







- データベースの作成および除去
- 表スペースまたは表の作成、更新、および除去
- 索引の作成、更新、および除去
- データベースまたは表スペースのバックアップと回復
- サーバー上のリソースおよびイベントの監視

2-2-2. その他の DB2 管理ツール

コントロール・センターのインターフェースには、サーバーを管理するために役立つ他のツールを起動するためのアイコンが用意されています。



次の表は、コントロール・センターのアイコンから起動できるおもな DB2 管理ツールを示しています。

	ツールの名前	説明
	コントロール・センター	コントロール・センターの別のセッションを開始します
	コマンド・センター	DB2 コマンドや SQL ステートメントを対話式ウィンドウに入力し、その実行結果を結果ウィンドウに表示します。出力結果をファイルに保管することもできます。
	ヘルス・センター	潜在的な問題をアラートし、問題の解決に役立つ情報を提供することによって、データベース管理者 (DBA) を支援します。データベース管理者は、インスタンスの稼働状況をモニターすることができます。
	ライセンス・センター	ライセンスを管理し、ライセンス状況、およびシステム上にインストールされている DB2 プロダクトの使用状況を表示します。
	ツール設定	DB2 管理ツールの設定値を変更します。
	インフォメーション・センター	データベース・タスク、参照資料、DB2 文書、ウェアハウス管理情報、トラブルシューティング補助機能、アプリケーション開発用のサンプル・プログラム、DB2 Web 関連の URL 等の DB2 製品情報に、すばやくアクセスすることができます。

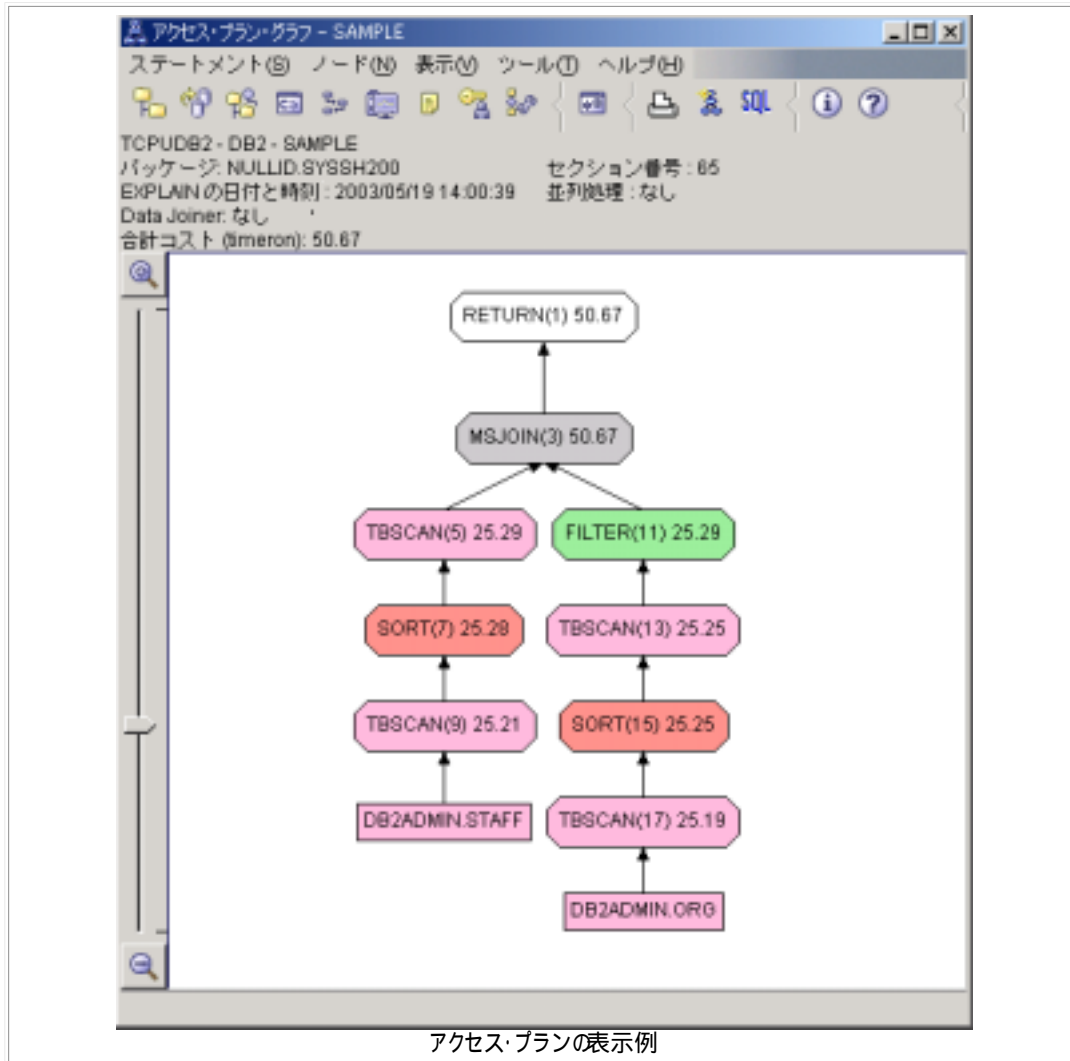
メモ: 上記の他にも、コントロール・センターから起動できる多くの便利なツールがあります。詳しくは、ヘルプ文書『管理の手引き』等を参照してください。

この他にも、DB2 UDB には、次のような便利なツールが用意されています。

- Visual Explain
- クライアント構成アシスタント

Visual Explain

Visual Explain は、データベース・マネージャの最適化プログラムが選んだ SQL ステートメントのアクセス・プランをグラフとして表示します。グラフから読み取る情報を使って、SQL 照会を調整したり、表に索引を作成してパフォーマンスを改善することができます。



アクセス・プランの表示例

クライアント構成アシスタント (CCA)

クライアント構成アシスタントは、DB2 接続の設定を容易に行うためのウィザードです。データベースの自動検出機能により、エンド・ユーザーは、複雑な設定をすることなく使用したいデータベースに接続することができます。



2-2-3. SQL 作成ツール

DB2 UDB では、DB2 コマンドや SQL での作業を行うために、次のツールが用意されています。用途に応じて使い分けてください。

ツール	説明
コマンド・センター	<ul style="list-style-type: none"> 対話型の GUI ツールです。 コントロール・センターから起動して、DB2 のコマンドや SQL 文を実行できます。
コマンド行プロセッサ	<ul style="list-style-type: none"> 入力行の先頭に db2 => と表示され、DB2 のコマンドや SQL 文を入力できます。 quit と入力すると、コマンド・ウィンドウの画面に戻ります。
コマンド・ウィンドウ	<ul style="list-style-type: none"> OS のコマンド等を実行できるプロンプト画面です。 DB2 のコマンドや SQL 文を入力する場合は、先頭に「DB2」を入力します。 db2 と入力すると、コマンド行プロセッサが起動します。

レッスン 2 DB2 UDB の製品コンポーネント**練習問題:**

DB2 UDB の製品群について説明した次の文章のうち、正しいものはどれですか？（正解は複数あります）

- A. DB2 Everyplace は、携帯端末から双方向にデータベース間のデータ同期を行うことのできる製品である
- B. DB2 UDB パーソナル・エディションは、リモート・クライアントからのアクセスをサポートしている
- C. DB2 UDB ワークグループ・サーバー・エディションのデータベース・サーバーは、Windows98 にインストールできる
- D. DB2 UDB データベース・パーティショニング・フィーチャーは、超並列処理をサポートしている

DB2 UDB の GUI ツールについて説明した次の文章のうち、正しいものはどれですか？

- A. インフォメーション・センターは、サーバーを管理するセッションを開始する
- B. スクリプト・センターは、DB2 コマンドや SQL ステートメントなどを対話式に入力し、結果ウィンドウに実行結果を表示する
- C. コントロール・センターでは、DB2 UDB のデータベース・オブジェクトの操作や管理を行える
- D. コマンド・センターでは、よく使う DB2 コマンドを格納しておき、スケジュールによって実行することができる

次のツールのうち、SQL ステートメントを入力できるものはどれですか？（正解は複数あります）

- A. サテライト管理センター
- B. コマンド行プロセッサ
- C. ライセンス・センター
- D. コマンド・センター

DB2 サーバーへの接続構成を容易に定義するウィザードは、次のうちどれですか？

- A. コントロール・センター
- B. クライアント構成アシスタント
- C. インフォメーション・センター
- D. コマンド・センター



レッスン 3

基本的な SQL の利用

このレッスンでは、次のトピックを学習します。

- 3-1. SQL の基本
- 3-2. データベースへの接続とエラーの調査
- 3-3. 基本的な表の照会
- 3-4. 検索条件の指定
- 3-5. 演算と関数の利用
- 3-6. データのグループ化
- 3-7. 操作の順序
- 3-8. 表の結合
- 3-9. 集合演算
- 3-10. 副照会
- 3-11. データの挿入・更新・削除

レッスン 3 基本的な SQL の利用

3-1. SQL の基本

3-1-1. SQL とは

SQL は、リレーショナル・データベースに表などのオブジェクトを定義したり、データを操作したりするために使用する照会 (問い合わせ) 言語です。SQL は ISO、ANSI、JIS などの標準化団体により標準化されていますので、操作する RDBMS が異なるベンダーの製品であっても、同じような操作で扱うことができます。

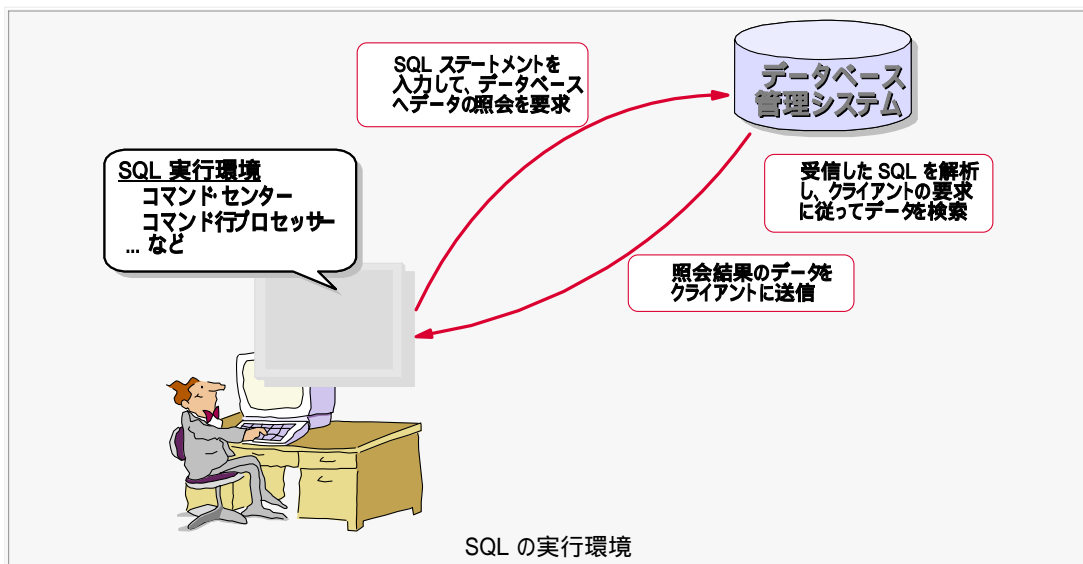
現在では、SQL-92 または SQL-99 と呼ばれる規格が、もっとも一般的に使用されています。IBM DB2 UDB V7 は、ANSI SQL-99 をサポートしています。

ヒント: SQL 言語の歴史的背景

リレーショナル・データベース・モデルの理論にともない、データベースの問い合わせを行うために開発された言語の 1 つが、「構造化英文照会言語 (SEQUEL)」です。この SEQUEL が、商標上の問題から名を改め SQL と呼ばれるようになりました。さらに、SQL は 1986年に ANSI 規格として承認され、1987年に国際標準化機構 (ISO) によって国際標準として出版されました。

3-1-2. SQL の実行環境

SQL の実行環境は、データベース管理システムからは独立したミドルウェアとして存在します。通常は、エンド・ユーザーが対話的に利用できるような UI ツールが提供されています。



DB2 UDB では、データベースにアクセスしたり、データの操作を行うために、コマンド行プロセッサ (CLP) や、コマンド・センターといった GUI ツールを利用して対話的に SQL を使うことができます。

3-1-3. SQL とは

SQL は、リレーショナル・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用する、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。

SQL は、次の3種類に分類することができます。

- **DML (Data Manipulate Language) ... データ操作言語**
データの参照、登録、更新などを行う SQL です。一般のユーザーが最も頻繁に使用する言語で、おもな命令は次の 4 つです。
 - SELECT ... 表のデータを検索する
 - INSERT ... 表にデータを追加 (挿入) する
 - UPDATE ... 表に存在するデータを更新する
 - DELETE ... データ行を削除する

- **DDL (Data Definition Language) ... データ定義言語**
データベースやデータベース・オブジェクトの定義を行う SQL です。通常は、データベース管理者などが使用します。おもな命令には次のものがあります。
 - CREATE TABLE ... 表の作成
 - ALTER TABLE ... 表定義の変更
 - CREATE VIEW ... ビューの作成
 - CREATE INDEX ... 索引の作成
 - GRANT/ REVOKE ... 権限の設定

- **DCL (Data Control Language) ... データ制御言語**
トランザクションを制御する SQL です。おもな命令には次のものがあります。
 - COMMIT
 - ROLLBACK

メモ: トランザクションについては、「レッスン6」を参照してください。

3-1-4. DB2 UDB の SQL で使用できる文字

DB2 UDB で使用する SQL 言語のキーワードや演算子は、基本的に 1 バイトの文字 (英字)、数字、特殊文字からなっています。

文字 は、26 個の大文字 (A ~ Z) および 26 個の小文字 (a ~ z)、3 個の文字 (\$, #, @) のいずれかです。**数字** は、0 ~ 9 のいずれかの文字です。

特殊文字 は、以下のいずれかの文字です。

ブランク	+ 正符号	; セミコロン
" 引用符または二重引用符	, コンマ	< 不等号 (より小さい)
% パーセント	縦線	= 等号
& アンパーサンド	! 感嘆符	> 不等号 (より大きい)
' アポストロフィ (') または一重引用符	- 負符号	? 疑問符
(左括弧	. ピリオド	_ 下線
) 右括弧	/ スラッシュ	^ 脱字記号
* アスタリスク	: コロン	

マルチバイト文字

- マルチバイト文字は、すべて文字として扱われます。ただし、特殊文字である 2 バイトのブランクは例外です。
- a ~ z の 1 バイトの小文字は大文字に変換されますが、マルチバイトの英小文字は大文字には変換されません。

スペースの扱い

SQL ステートメントを構成するキーワードや文字はスペース (空白) で区切られますが、文中の余分な空白は無視されます。

スペース は、1 つまたは複数のブランク文字です。文字列定数と区切り識別子以外で、トークンにスペースを含めることはできません。

大文字と小文字

ステートメントを構成する文字のうち、区切りトークンを除くすべてのトークンで、小文字と大文字は区別されません。

たとえば、次の 2 つのステートメントは全く同じです。

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

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

【注意】 検索条件に指定する文字列などは、小文字と大文字を正しく指定するようにします。たとえば、上の例で LASTNAME='SMITH' と指定すると、'Smith' という値は一致したものととして検索されません。

行末文字

ANSI SQL では、行末文字として ";" (セミコロン) を指定しますが、DB2 UDB のツール (例: コマンド行プロセッサなど) では、行末文字のセミコロンは入力しません。

レッスン 3 基本的な SQL の利用

3-2. データベースへの接続

3-2-1. 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、まず最初に、操作するデータベースに **接続** する必要があります。データベースに接続するには、CONNECT ステートメントを使用します。

```
書式 CONNECT TO <データベース名> USER <ユーザー名>  
      USING <パスワード>
```

- ユーザー名を省略した場合、現在 OS にログオンしているユーザー名でデータベースへの接続を試みます。

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、TYAMADA (ユーザー名: TYAMADA、パスワード: PASSWORD) というユーザー ID で 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で次のように入力します。

```
CONNECT TO SAMPLE USER TYAMADA USING PASSWORD
```

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      = TYAMADA  
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。

レッスン 3 基本的な SQL の利用

3-3. 基本的な表の照会

3-3-1. 列の照会

表のデータを照会するには、SELECT ステートメントを使います。

- SELECT ステートメントは、表から指定した列のデータを照会し、結果を返します。
- デフォルトでは、返される行の順序は保証されません。

書式 SELECT <列名> FROM <表名>

複数列の照会

表から複数の列を照会するには、列の名前をコンマで区切ったリストを指定します。SELECT に続いて指定する列名のことを **選択リスト** と呼びます。

書式 SELECT <列名1>, <列名2>, ...<列名n> FROM <表名>

列1	列2	列3	列4	列5

例: SAMPLE データベースの ORG 表には次のようなデータが入っています。

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

次のステートメントでは、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

全列の照会

アスタリスク (*) を使うと、表のすべての列を選択できます。

```
書式 SELECT * FROM <表名>
```

例：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

3-3-2. 行のソート

SELECT 文で 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。

- 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ます。
- ORDER BY 文節には列名または列の式を指定しますが、この名前は選択リストに指定したものである必要はありません。
- 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。

書式 SELECT <列名> FROM <表名> ORDER BY <ソートする列名> (DESC)

例：次のステートメントでは、STAFF 表から従業員の NAME、JOB、YEARS が就業年数 (YEARS) でソートして表示されます。

```
SELECT NAME, JOB, YEARS
FROM STAFF
ORDER BY YEARS
```

このステートメントの結果の表示例は、次のとおりです。

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

ソート列 (デフォルトで昇順)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
70	Davis	84	Sales	5	17230.00	150.00
120	Gafney	84	Clerk	5	16790.50	100.00
150	Edwards	84	Sales	7	12800.75	150.00
290	Quill	84	Mgr	10	19818.00	

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

また、ORDER BY 文節に ASC または DESC を指定すると、明示的に昇順または降順にソートすることができます。どちらも指定しない場合はデフォルトのソート順序となります。DB2 の場合は、デフォルトで昇順 (ASC) となります。

例：次のステートメントでは、STAFF 表から従業員の NAME、JOB、YEARS が就業年数 (YEARS) の降順でソートして表示されます。

```
SELECT NAME, JOB, YEARS
FROM STAFF
ORDER BY YEARS DESC
```

このステートメントの結果の表示例は、次のとおりです。

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

例：次のステートメントでは、STAFF 表から従業員の NAME、JOB、YEARS が名前 (NAME) のアルファベット順序で表示されます。

```
SELECT NAME, JOB, YEARS
FROM STAFF
ORDER BY NAME
```

このステートメントの結果の表示例は、次のとおりです。

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

ソート列 (昇順)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
70	Davis	84	Sales	5	17230.00	150.00
120	Gafney	84	Clerk	5	16790.50	100.00
150	Edwards	84	Sales	7	12800.75	150.00
290	Quill	84	Mgr	10	19818.00	

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

Diagram description: The diagram shows a table with columns ID, NAME, DEPT, JOB, YEARS, SALARY, and COMM. The NAME and YEARS columns are highlighted with a red hatched pattern. A red arrow points to the NAME column with the label 'ソート列 (昇順)'. A yellow arrow points from the NAME and YEARS columns of the main table to a separate table below that shows the result of sorting by NAME in ascending order.

ヒント:

- ORDER BY に指定する列名が、SELECT で照会する列名に含まれる場合、番号で指定することができます。
例: `SELECT NAME, JOB, YEARS FROM STAFF ORDER BY 1`
は、上記の SELECT 文と同じ結果を返します。
- ORDER BY 句に複数の列を指定して降順にソートしたい場合は、それぞれの列に DESC 指定が必要です。
例: `SELECT NAME, JOB, YEARS FROM STAFF ORDER BY NAME DESC, YEARS
DESC`

レッスン 3 基本的な SQL の利用

練習問題:

次の表があります。

STAFFTABLE

STID	LNAME	FNAME
01	Izumi	Ichiro
02	Yamada	Taro
03	Fujimoto	Mako

次の SQL 文を発行してこの表を照会したときに取得する結果表の行は、どのような順序になりますか？

```
SELECT * FROM STAFFTABLE
```

- A. 保証された順序はない
- B. 主キーの順序
- C. STAFFTABLE 表に行が挿入された順序
- D. テーブルの最後の REORG に使われた索引に基づく順序

次の表があります。

ORG

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

次の SELECT 文を実行した場合、結果表の行数はいくつですか？

```
SELECT * FROM ORG
```

- A. 0
- B. 1
- C. 5
- D. 8

レッスン 3 基本的な SQL の利用

3-4. 検索条件の指定

3-4-1. 行の絞り込み

表から特定の行を照会するには、SELECT ステートメントに WHERE 文節を伴って、行を選択する条件を指定します。表から特定の行を選択する基準のことを、**検索条件** とよびます。

```
書式 SELECT <列名> FROM <表名> WHERE <検索条件>
```

列1	列2	列3	列4	列5

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、ある行に関して結果が真か偽 (または不明) となる条件をしています。

3-4-2. WHERE 文節による検索条件の指定

WHERE 文節の中で条件を指定するために使用できる基本的な演算子には、次のものがあります。

- WHERE 句の基本的な演算子 -

演算子	説明	演算子	説明
=	等しい	!<	小さくない
<>	等しくない	>	大なり
!=	等しくない	>=	大または等しい
<	小なり	!>	大きくない
<=	小または等しい		

検索条件指定時の注意:

- 検索条件を作成する場合は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します。たとえば、文字列と数値の比較は行えません。
- 数値データ・タイプ以外では算術演算を実行しないようにします。
- 文字列の演算で値を指定する場合、単一引用符 (') で囲む必要があります。また、文字列の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しなければなりません。たとえば、データベース内の文字列値で 1 文字目だけ大文字の値を選択したい場合、同じように 1 文字目だけ大文字にして検索条件を指定します。

例: WHERE JOB = 'Clerk'

- 検索条件で特定の数値を指定する場合、引用符は使用しません。

例: WHERE DEPT = 20

例: 次の例では、「=」演算子を使って 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の表示例は、次のとおりです。

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

例: 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
```

その他の演算子

WHERE 句に指定できる演算子には、比較演算子の他にも、次のものがあります。

- WHERE の基本的な演算子・その2 -

演算子	説明
AND	条件の積 (A かつ B)
OR	条件の和 (A または B)
NOT	条件の否定
LIKE	文字列のワイルドカード比較
BETWEEN	指定した 2 つの値の間 <-> NOT BETWEEN
IS NULL	NULL 値である <-> IS NOT NULL
IN	かっこで囲まれ、カンマで区切られた値の一覧の中の 1 つと一致する <-> NOT IN

AND の使用

例：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 (DEPT=20) かつ事務員 (JOB='Clerk') であるレコードを選択しています。

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

このステートメントの結果の表示例は、次のとおりです。

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

ヒント: 演算子の評価順序

AND 演算子と OR 演算子を組み合わせて使用する場合、AND 演算子の方が先に評価されます。たとえば、部署が 15 または 20 で、かつ Clerk であるデータを検出しようとする場合、次の検索条件を指定したとします。

```
WHERE DEPT=15 OR DEPT=20 AND JOB='Clerk'
```

この条件では、(DEPT=20 AND JOB='Clerk') が先に評価されてしまいますので、検出されるデータは「DEPT=15」または「DEPT=20 AND JOB='Clerk」となり、期待したものと異なります。このような問題を防ぐためには、かっこをつけて条件の評価順位を調整することができます。

```
WHERE (DEPT=15 OR DEPT=20) AND JOB='Clerk'
```

WHERE 句の中で AND と OR を共用する場合は、明示的にかっこを使用するとよいでしょう。

NULL の使用

また、値が特に NULL として設定されている場合もあります。列の値が NULL 値かどうかを調べるには、IS NULL 述部および IS NOT NULL 演算子を使います。

例: 次のステートメントは、歩合が該当しない (COMM 列が NULL 値である) 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の表示例は、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。

IN の使用

IN 演算子は、ある値を他のいくつかの値と比較するときに使います。たとえば、次の 2 つのステートメントは同じです。

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

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

BETWEEN の使用

BETWEEN 演算子は、1 つの値と一定範囲の値 (BETWEEN で指定) を比較します。

例: 次の 2 つの例では、いずれも給与が \$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
```

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

例: 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 の使用

LIKE 演算子は、特定のパターンにマッチするSTRING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。LIKE 演算子を使った検索を、**ワイルドカード 検索**と呼ぶこともあります。

- アンダースコア (_) 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STRINGを表します。
- その他の文字は、その文字自体を表します。

例: 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 _ _ _ _ _ _ _'
```

例: 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%'
```

レッスン 3 基本的な SQL の利用

3-5. 演算と関数の利用

3-5-1. 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。

例：次のステートメントでは、STAFF 表で給与 (SALARY) と歩合 (COMM) の合計が \$13,000 より小さい従業員の名前 (NAME)、職種 (JOB)、給与と歩合の合計 (SALARY + COMM) が表示されます。

```
SELECT NAME, JOB, SALARY + COMM
FROM STAFF
WHERE (SALARY + COMM) < 13000
ORDER BY 3
```

結果の表示例は、次のとおりです。

NAME	JOB	3
Yamaguchi	Clerk	10581.50
Burke	Clerk	11043.50
Scoutten	Clerk	11592.80
Abrahams	Clerk	12246.25
Kermisch	Clerk	12368.60
Ngan	Clerk	12714.80

結果表第 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、SELECT 文で指定した (SALARY + COMM) が列名そのものを指定していないため、システムが生成した番号です。この列番号は、ORDER BY で使用することができます。

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。演算子のオペランド (被演算値) としては、次のものがあります。

- 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)

3-5-2. 式に別名をつける

オプションの AS 文節を使うと、式に意味のある名前 (別名) 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。

書式 <式> AS <別名>

例: 次のステートメントは前の SQL と同様の条件で給与と歩合の合計額が \$13,000 より少ない従業員を表示しますが、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の表示例は、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、結果表の列名には番号ではなく指定した列名が表示されます。また、この別名は ORDER BY 文節の中で使用することもできます。

しかし、WHERE 文節は SELECT 文節よりも先に評価されるため、SELECT 文節に記述した別名を WHERE 句の中で使用することはできません。表記の順序と実行の順序が異なっていますので、注意が必要です。

3-5-3. 複数の表からのデータ検索

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 文節に表名を指定します。WHERE 文節はオプションです。

書式 SELECT <列名> FROM <表名1>,<表名2>

例：次の例では、ORG 表と STAFF 表という 2 つの異なる表の列を比較して、検索条件を満たすデータを照会しています。

WHERE 文節による検索条件で ORG 表の MANAGER 列と STAFF 表の ID 列を比較し、一致する行のみを 2 つの表から検出します。

で検索条件に一致したレコードのうち、結果表には ORG 表から部署名 (DEPTNAME) を、STAFF 表からマネージャー (MANAGER) の名前 (NAME) を表示しています。

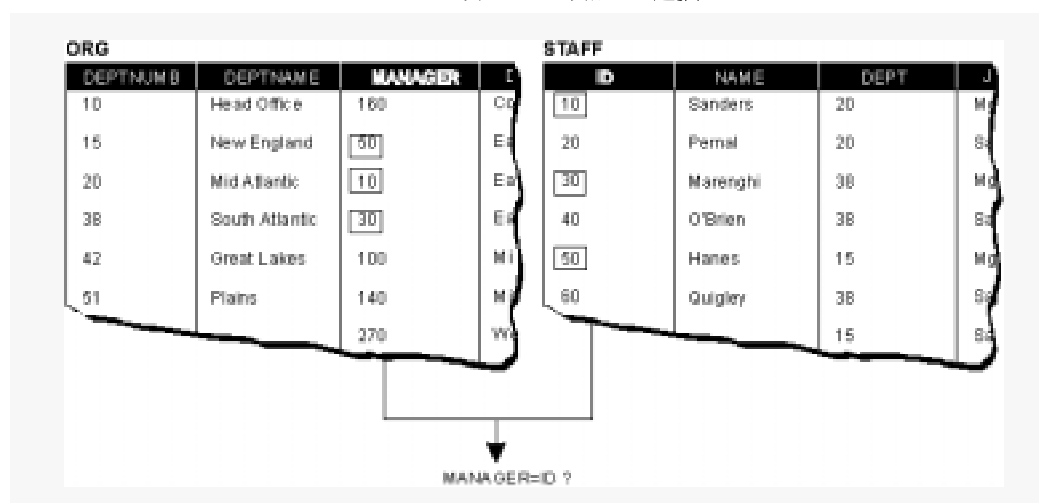
```
SELECT DEPTNAME, NAME
FROM ORG, STAFF
WHERE MANAGER = ID
```

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
```

結果表には、マネージャーの名前と、その部署とが示されています。

STAFF 表と ORG 表からの選択



3-5-4. 関数の使用

関数 とは、データベース内の入力値と結果値との間の関係付けです。

組み込み関数とユーザー定義関数

DB2 UDB の関数は、**組み込み関数** と **ユーザー定義関数** に分類することができます。

組み込み関数 は、データベース・マネージャーに用意されている関数で、予約スキーマである SYSIBM スキーマの一部に含まれています。組み込み関数には、AVG などの列関数、"+" などの演算関数、DECIMAL などのキャスト関数、SUBSTR など文字列関数などが用意されています。

ユーザー定義関数 では、組み込み関数に含まれない機能を関数として定義することができます。たとえば、カスタマイズされた業務用の機能や他社製品との互換機能など、広い範囲の拡張性を実現することができます。ユーザー定義関数は、組み込み関数の派生関数として定義することもできます。

ユーザー定義関数は、CREATE FUNCTION ステートメントを使って SYSCAT.FUNCTIONS データベースに登録されます。

メモ: DB2 UDB では、事前インストール済みの関数も用意されています。事前インストール済みのユーザー定義関数は、SYSFUN という名前のスキーマに含まれています。

その他の分類

関数の分類として、機能別に次のように分類することもできます。

- **スカラー関数** ... 呼び出されるたびに単一の値を戻します。たとえば、組み込み関数 SUBSTR() などがあります。
- **列関数** ... 1 つの列の集合から、単一の値を戻します。
- **行関数** ... 値を一行で戻します。
- **表関数** ... 値を表の形で戻します。

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。次の表は、DB2 UDB で使用できるスカラー関数の例です。

関数	説明
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。

メモ: スカラー関数の完全な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。

例: 次のステートメントでは、LENGTH() を利用して ORG 表から部署名 (DEPTNAME) と、その文字列の長さを戻します。

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

このステートメントの結果の表示例は、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: SELECT文節で演算や関数を使用し、かつ、AS文節で、該当する列に列名をつけていない場合には、列の順番にあたる番号を列名として表示します。この例では、LENGTH(DEPTNAME) に対して名前を付ける 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。

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。次の表は、DB2 UDB で使用できる列関数の例です。

関数	説明
SUM	ある集合内の値の和を戻します。
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。

メモ: 列関数の完全な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。

例: 次のステートメントでは、MAX() を利用して、STAFF 表から給与 (SALARY) の最大値を求め、MAX_SALARY 列として表示しています。

```
SELECT MAX(SALARY) AS 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。

```
MAX_SALARY
-----
22959.20
```

例: 次のステートメントでは、AVG() を利用して、STAFF 表から給与 (SALARY) の平均値を求め、AVG_SALARY 列として表示しています。

```
SELECT AVG(SALARY) AS AVG_SALARY FROM STAFF
```

このステートメントの結果の表示例は、次のとおりです。

```
AVG_SALARY
-----
16675.64228571428571428571428571
```

3-5-5. 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 表から部署番号 30 未満の部署名を照会しようとしたとします。次のような SQL ステートメントを実行すると、結果表には同じ値の行が何度も出現します。

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

結果表

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

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、前のステートメントに 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。

レッスン 3 基本的な SQL の利用

練習問題:

次の表があります。

ORG

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

以下の SELECT 文を実行した場合、結果表の行数はいくつですか？

```
SELECT DEPTNUMB, DEPTNAME FROM ORG
WHERE DIVISION='Eastern'
```

- A. 0
- B. 1
- C. 2
- D. 3

次の表があります。

COUNTRY

ID	NAME	PERSON	CITIES
1	Argentina	1	10
2	Canada	2	20
3	Cuba	2	10
4	Germany	1	0
5	France	7	5

1 行も結果セットを返さないのは、次のうちどのステートメントですか？

- A. SELECT PERSON FROM COUNTRY WHERE NAME LIKE 'a%'
- B. SELECT COUNT(*) FROM COUNTRY
- C. SELECT COUNT(1) FROM COUNTRY
- D. SELECT ID,NAME FROM COUNTRY WHERE PERSON BETWEEN 7 AND 7

レッスン 3 基本的な SQL の利用

3-6. データのグループ化

3-6-1. データのグループ化

DB2 UDB では、表の中の特定の列に基づいてデータをグループ化して分析する機能があります。表内の特定の列をグループ化するには、GROUP BY 文節を利用します。

```
書式 SELECT <列名> FROM <表名>
      GROUP BY <列名> (HAVING <フィルター条件> )
```

SELECT 文節に指定する列名には、グループ化する列か列関数のいずれかを指定しなければなりません。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。

例：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 WHERE 検索条件を含めると、グループ化の計算よりも先に WHERE 検索条件が評価されます。

例：次のステートメントは、上の SQL 文に WHERE 検索条件を追加することにより、部署番号 50 未満の部署のみ照会しています。

```
SELECT DEPT, MAX(SALARY) AS MAXIMUM
FROM STAFF
WHERE DEPT < 50
GROUP BY DEPT
```

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

GROUP BY 文節の後に HAVING 文節を使う

グループ化を行う際に、特定の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に、HAVING 文節を利用してフィルター条件を指定することができます。

HAVING 文節の条件設定では、WHERE 文節に使用できる演算子はほとんど利用できます。

HAVING 文節によるフィルター条件は、GROUP BY 文節の後ろに追加します。

例：たとえば、次のステートメントでは、部署 (DEPT) ごとに所属する従業員の人数 (レコード数) を表示しています。

```
SELECT DEPT, COUNT(*)
FROM STAFF
GROUP BY DEPT
```

< 結果表 >

DEPT	2
10	4
15	4
20	4
38	5
42	4
51	5
66	5
84	4

グループ化を行う際に、「所属スタッフが5人以上」というフィルター条件を指定する場合は、次のようなステートメントを実行します。

```
SELECT DEPT, COUNT(*)
FROM STAFF
GROUP BY DEPT
HAVING COUNT(*) > 4
```

< 結果表 >

DEPT	2
38	5
51	5
66	5

ヒント: グループ化とソート

GROUP BY によって、暗黙的にデータはソートされますが、照会されるデータの順序を保証する唯一の指定は ORDER BY 句です。検索結果に順序を指定する必要がある場合は、明示的に ORDER BY を追加するようにします。

レッスン 3 基本的な SQL の利用

3-7. 操作の順序

SQL 文の操作の実行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。

操作の実行順序は、次のとおりです。

- FROM 文節
- WHERE 文節
- GROUP BY 文節
- HAVING 文節
- SELECT 文節 (副照会)
- ORDER BY 文節

注意: 上記の操作記述順序は、実際に処理が実行される順序であるとは限りません。

メモ: AS 文節で別名を指定する場合などは、注意が必要です。

メモ: 副照会については、あとのトピックで説明しています。

レッスン 3 基本的な SQL の利用

3-8. 表の結合

3-8-1. 表の結合

複数の表のデータを組み合わせることを、表の **結合** と呼びます。複数の表を結合して照会する場合には、どのような条件で表のデータを組み合わせるかという **結合条件** を指定します。

例: たとえば、次の 2 つの表 SAMP_PROJET と SAMP_STAFF があります。

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

2 つの表から、一致する名前 (NAME) を持つ行のみを表示する場合は、次のようなステートメントを指定することができます。

```
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
```

< 結果表 >

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

このステートメントでは、WHERE 文節に指定した内容が 2 つの表の結合条件です。

ヒント:

結合条件を指定しない場合、複数の表から抽出する行のすべての組み合わせが結果表に出力されます。たとえば、上記のサンプルでは 2 つの表に 4 行ずつデータがありますので、 $4 * 4 = 16$ 行のレコードが結果表に出力されます。この場合の結果表は、2 つの表の **直積** と呼ばれます。

3-8-2. 内部結合と外部結合

複数の表を結合する際に、すべての表に含まれるデータのみを結合することを **内部結合** と呼びます。逆に、いずれかの表に含まれないデータをも含む結合を **外部結合** と呼びます。

内部結合

前のトピックで紹介したステートメントは、2つの表に共通するデータを結合しますので、内部結合を生成します。内部結合を指定するには、明示的に **INNER JOIN** 文節を使用して次のように記述することもできます。

```
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
```

この照会の結果は、4-8-1 のステートメントと同じです。

内部結合では、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、結果として出力されません。

外部結合

外部結合では、結合するいずれかの表に含まれない行のデータも結果表に出力することができます。外部結合には、次の3種類があります。

- **左外部結合** ... 内部結合に加え、左側の表にのみ含まれる行も結果表に含まれます。
- **右外部結合** ... 内部結合に加え、右側の表にのみ含まれる行も結果表に含まれます。
- **全外部結合** ... 内部結合に加え、片方の表にのみ含まれる行もすべて結果表に含まれます。

例: **左外部結合**では、**LEFT OUTER JOIN** 文節を使用します。次のステートメントは、**LEFT OUTER JOIN** 文節以外は前の内部結合のステートメントと全く同じですが、結果表では、左側の表 (**SAMP_PROJECT**) にのみ含まれる行も出力されています。結合条件の満たされない行 (右側の表にないデータ) は、**NULL** 値として表示されます。

```
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
```

< 結果表 >

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

右外部結合では **RIGHT OUTER JOIN** 文節を使用します。左外部結合の例とは逆に、結果表には、右の表 (SAMP_STAFF) にのみ含まれる行も出力されます。

```
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
```

< 結果表 >

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

全外部結合には **FULL OUTER JOIN** 文節を使用します。全外部結合では、内部結合に含まれないすべての行が結果に出力されます。つまり、全外部結合には、内部結合、左外部結合、右外部結合のすべての結果表が含まれています。

```
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
```

< 結果表 >

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

レッスン 3 基本的な SQL の利用

3-9. 集合演算

3-9-1. 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せられます。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 つの照会を組み合わせ、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。

例: たとえば、次の 2 つのステートメント 「給与 (SALARY) が \$21,000 以上のスタッフを照会」、「職種がマネージャー (JOB='Mgr') のスタッフを照会」とそれぞれの結果表があります。

```
SELECT ID,NAME,JOB,SALARY
FROM STAFF
WHERE SALARY >=21000
```

< 結果表 >

ID	NAME	JOB	SALARY
140	Fraye	Mgr	21150.00
160	Molinare	Mgr	22959.20
260	Jones	Mgr	21234.00
310	Graham	Sales	21000.00

```
SELECT ID,NAME,JOB,SALARY
FROM STAFF
WHERE JOB='Mgr'
```

< 結果表 >

ID	NAME	JOB	SALARY
10	Sanders	Mgr	18357.50
30	Marenghi	Mgr	17506.75
50	Hanes	Mgr	20659.80
100	Plotz	Mgr	18352.80
140	Fraye	Mgr	21150.00
160	Molinare	Mgr	22959.20
210	Lu	Mgr	20010.00
240	Daniels	Mgr	19260.25
260	Jones	Mgr	21234.00
270	Lea	Mgr	18555.50
290	Quill	Mgr	19818.00

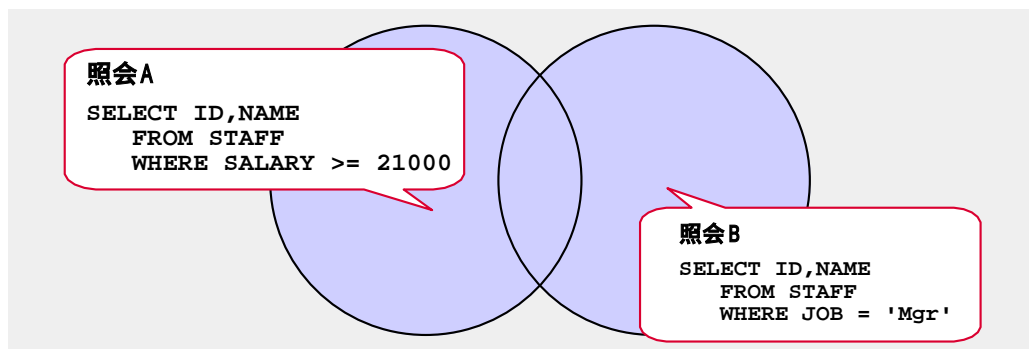
2 つの照会を UNION 演算子で結合で組み合わせると、2 つの結果表のすべての行から重複の除去された結果が表示されます。

```
SELECT ID,NAME,JOB,SALARY FROM STAFF WHERE SALARY >= 21000
UNION
SELECT ID,NAME,JOB,SALARY FROM STAFF WHERE JOB = 'Mgr'
```

< 結果表 >

ID	NAME	JOB	SALARY
30	Marenghi	Mgr	17506.75
100	Plotz	Mgr	18352.80
10	Sanders	Mgr	18357.50
270	Lea	Mgr	18555.50
240	Daniels	Mgr	19260.25
290	Quill	Mgr	19818.00
210	Lu	Mgr	20010.00
50	Hanes	Mgr	20659.80
310	Graham	Sales	21000.00
140	Fraye	Mgr	21150.00
260	Jones	Mgr	21234.00
160	Molinare	Mgr	22959.20

このステートメントでは、 の照会結果と の照会結果の和集合、つまり「給与が \$21,000 以上のスタッフ」または「職務がマネージャー」または「マネージャーでかつ給与が \$21,000 以上」のスタッフが照会結果として返されます。個々の照会で重複する行は削除されています。



ヒント:

上の照会では、ID の順序が指定されていません。結果表のソートを指定するには、ORDER BY 文節を利用します。集合演算で ORDER BY 文節を利用する場合、必ず最後の照会の後に指定します。

例:

```
SELECT ID,NAME,JOB,SALARY FROM STAFF WHERE SALARY >= 21000
UNION
SELECT ID,NAME,JOB,SALARY FROM STAFF WHERE JOB = 'Mgr'
ORDER BY ID
```

EXCEPT 演算子

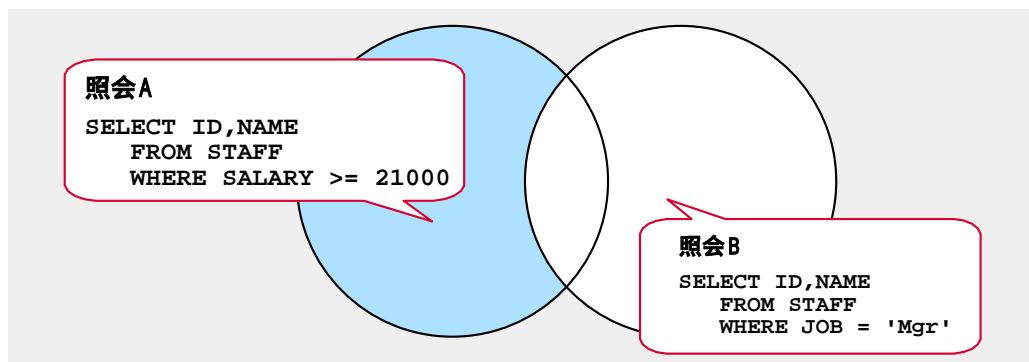
EXCEPT 演算子は、照会 A には含まれているが照会 B には含まれていない行を取り出し、行の重複をすべて除去することによって結果表を生成します。EXCEPT と共に ALL を使った場合 (EXCEPT ALL)、行の重複は除去されません。

例：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 (JOB='Mgr') ではないスタッフを戻す照会の例です。

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

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	JOB	SALARY
310	Graham	Sales	21000.00



INTERSECT 演算子

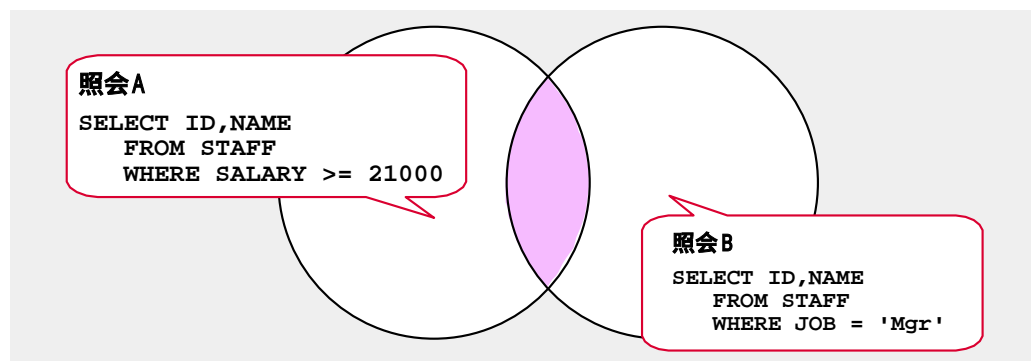
INTERSECT 演算子は、照会 にも照会 にも含まれている行を取り出し、行の重複をすべて除去することによって結果表を生成します。INTERSECT と共に ALL を使った場合 (INTERSECT ALL)、重複した行は削除されません。

例：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 (JOB='Mgr') のスタッフを戻す照会の例です。

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

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	JOB	SALARY
140	Fraye	Mgr	21150.00
260	Jones	Mgr	21234.00
160	Molinare	Mgr	22959.20



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。
- 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。
- 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。

レッスン 3 基本的な SQL の利用

3-10. 副照会

3-10-1. 副照会

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。

例：次のステートメントでは、STAFF 表の中で、所属年数 (YEARS) が平均年数 (SELECT AVG(YEARS) FROM STAFF の副照会ステートメント) を上回る従業員の ID、名前 (NAME)、所属年数 (YEARS) を検索しています。

```
SELECT ID, NAME, YEARS
FROM STAFF
WHERE YEARS > (SELECT AVG(YEARS)
                FROM STAFF)
```

このステートメントでは、最初に副照会のステートメントが評価され、STAFF 表に登録されたデータの所属年数平均を算出します。次に、メイン照会の WHERE 検索条件が副照会で算出された値に置換され、メイン照会が評価されます。

ヒント:

AVG() は列関数ですので、WHERE YEARS > AVG(YEARS) などのように記述することはできません。

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することもできます。

レッスン 3 基本的な SQL の利用

練習問題:

SQL 文には、複数の表を論理的に操作するための演算機能があります。
 複数の異なる照会の結果表の和を取る集合演算で使用されるキーワードは、次のうちどれですか？

- A. JOIN
- B. UNION
- C. PROJECTION
- D. INTERSECTION

次の 2 つの表 EMP, DEPT があります。

EMP

NAME	DEPTNO
Abe	100
Katoh	110
Suzuki	110
Yoshida	120
Sakai	130

DEPT

DEPTNO	NAME
100	Marketing
110	Sales
120	Support
150	Technical

次の結果表を得るためには、どのような SQL 文を使用しますか？

NAME	DEPTNO	NAME
Abe	100	Marketing
Katoh	110	Sales
Suzuki	110	Sales
Yohida	120	Support
Sakai	130	-
-	150	Technical

- A. SELECT EMP.NAME,EMP.DEPTNO,DEPT.NAME
FROM EMP FULL OUTER JOIN DEPT ON EMP.DEPTNO=DEPT.DEPTNO
- B. SELECT EMP.NAME,EMP.DEPTNO,DEPT.NAME
FROM EMP RIGHT OUTER JOIN DEPT ON EMP.DEPTNO=DEPT.DEPTNO
- C. SELECT EMP.NAME,EMP.DEPTNO,DEPT.NAME
FROM EMP LEFT OUTER JOIN DEPT ON EMP.DEPTNO=DEPT.DEPTNO
- D. SELECT EMP.NAME,EMP.DEPTNO,DEPT.NAME
FROM EMP,DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO

レッスン 3 基本的な SQL の利用

3-11. データの挿入・変更・削除

3-11-1. 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。

```
書式  INSERT INTO <表名> [( <列名>, ... )] VALUES ( <値>, ... )
```

INSERT ステートメントには、次の 2 種類の形式があります。

- **VALUES 文節を使った挿入** ... VALUES 文節を使って、1 行または複数の行に挿入する値を直接指定します。
- **副照会を使った挿入** ... VALUES 文節を使わずに、他の表やビューに含まれている行から選択した列の値を指定します。

いずれの形式についても、行のすべての列に値を指定する方法と、挿入列を指定して省略可能な列への値の挿入を省略する方法のいずれかを選択することができます。

VALUES 文節を使った挿入

VALUES 文節を使って、挿入する値を直接指定します。

例: 全列の値を指定

次の例では、VALUES 文節を使って、PRES 表に 1 行分のデータを挿入しています。ここでは、VALUES 文節として PRES 表の定義に従い、すべての列のデータを指定しています。

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

例: 列名を指定して入力不要な列への入力を省略

次のステートメントでは、同じ PRES 表に 3 行分のデータを挿入しています。ここでは、挿入する列を NAME、JOB、ID に限り、対応する列のデータのみ指定しています。

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

上の例で、指定されなかった列には NULL 値が挿入されます。

データを挿入する表において、主キーなど NOT NULL として定義されている列で省略時の値が設定されていない場合は、必ず値を指定しなければなりません (省略できません)。

メモ: CREATE TABLE ステートメントで表を作成する際、列定義の中の NOT NULL 文節に WITH DEFAULT 句でデフォルト値を設定することができます。このような指定が予め行われている列については、NOT NULL 列であっても入力を省略することができます。

前出の 2 つの挿入レコードを照会した結果表の表示例です。レコード中の NULL 値はハイフン (-) で表示されています。

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	-	-	-	-

副照会を使った挿入

新規行を作成する際に、他の表やビューに含まれる行の値を指定することができます。この場合には、値の指定に VALUES 句ではなく副照会を使用します。

例: 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 (副照会)、抽出した結果表のデータを PRES 表に挿入しています。

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

INSERT ステートメントの注意事項

- NOT NULL が指定されている列でデフォルト値の設定がない場合は、入力を省略できません。
- 列名を省略して INSERT を行う場合、VALUES で指定するデータのリストは、必ず挿入先の表定義の列の順序と同じで、同じ数の値を指定します。
- 挿入データとして指定する値は、挿入先の列のデータタイプと互換性のある値を指定しなければなりません。

3-11-2. データの変更

表の中のデータ行を更新するには、UPDATE ステートメントを使います。更新する行の条件は、WHERE 文節で検索条件を指定します。検索条件の指定方法は、SELECT ステートメントでの指定と同じです。

```
書式 UPDATE <表名> SET <列名> = <値>, ... [WHERE <検索条件>]
```

UPDATE ステートメントを使うと、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。

例：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。

このステートメントを実行すると、ID = 410 の従業員レコード行で、肩書き (JOB) を 'Prgmr' に変更し、給与 (SALARY) に \$300 加算した値をセットしています。

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

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。

例：次の例では、すべての営業マン (JOB = 'Sales') の給与を 15% アップしています。

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

UPDATE ステートメントの注意事項

- COMMIT ステートメントを発行するまで、データの変更トランザクションはシステムに反映されません。
- WHERE 文節で指定する検索条件はオプションですが、WHERE 文節を省略した場合、表またはビューの中のすべてのレコード行を SET 文節以下の内容で更新します。
- WHERE 文節で指定した検索条件に一致した行が複数存在する場合は、複数の行を一度に更新します。

3-11-3. データの削除

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。削除する行の条件は、WHERE 文節で検索条件を指定します。検索条件の指定方法は、SELECT ステートメントでの指定と同じです。

```
書式  DELETE FROM <表名> [WHERE <検索条件>]
```

例：次の例では、従業員 ID が 120 の行を削除しています。

```
DELETE FROM PRES
WHERE ID = 120
```

DELETE ステートメントの注意事項

- COMMIT ステートメントを発行するまで、データの変更トランザクションはシステムに反映されません。
- WHERE 文節で指定する検索条件はオプションですが、WHERE 文節を省略した場合、表またはビューの中のすべてのレコード行を削除します。
注意：表の内容レコードだけでなく表定義そのものを削除する場合は、DROP TABLE ステートメント (DCL) を使います。
- WHERE 文節で指定した検索条件に一致した行が複数存在する場合は、複数の行を一度に削除します。
- DELETE ステートメントでは行全体を削除しますので、特定の列の値を削除することはできません。

レッスン 3 基本的な SQL の利用**練習問題：**

SQL に関する次の説明文のうち、正しいものを選択しなさい。

- A. UPDATE 文は、DDL (Data Definition Language) に分類される
- B. 1 つの SQL 文で、複数のデータを同時に変更することができる
- C. UPDATE 文で変更したデータは元の状態には戻らないため、慎重に入力する必要がある
- D. UPDATE 文を使用する場合は、主キーなど一意のキーを指定して変更行を指定しなければならない

SQL に関する次の説明文のうち、正しいものを選択しなさい。

- A. DELETE 文を使用する場合は、主キーなど一意のキーを指定して変更行を指定しなければならない
- B. DELETE 文で削除したデータは元の状態には戻らないため、慎重に入力する必要がある
- C. DELETE 文で列名にアスタリスク (*) を指定すると、表内のすべての行が削除される
- D. DELETE 文は、DML (Data Manipulate Language) に分類される



レッスン4

データベース・オブジェクト

このレッスンでは、次のトピックを学習します。

- 4-1. 表、視点 (ビュー)、索引
- 4-2. 表やビューの作成、削除
- 4-3. データと記憶管理
- 4-4. DB2 のデータ型
- 4-5. システムカタログビュー

レッスン 4 データベース・オブジェクト

4-1. 表、視点(ビュー)、索引

4-1-1. DB2 UDB の構造とデータベース・オブジェクト

DB2 UDB はリレーショナル・データベース管理システム (RDBMS) です。リレーショナル・データベースは、データを **表** (テーブル) の集合という形で保管します。表は、定義された数の **列** と任意の数の **行** によって構成されています。

表や列を指定したり、それらの間のさまざまな関係を指定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、**SQL 言語** を使用します。

メモ: **SQL 言語** ... SQL 言語の詳細については、レッスン4で説明します。

ヒント: DB2 UDB のデータベースには、データを格納する表の他に、次のものが含まれます。

- **システム・カタログ表** ... データの論理構造および物理構造の定義情報などを含む
- **データベース構成ファイル** ... データベースの属性を指定するパラメーターを含む
- **ログ** ... 進行中のトランザクションおよびアーカイブ可能トランザクションの回復ログ

ここでは、DB2 UDB で使用されるおもなデータベース・オブジェクトについて説明しています。

4-1-2. 表 (テーブル)

表 は、決まった数の **列** と可変数の **行** から構成される論理的な構造です。列は、同じデータ型の値の集合です。行は、表の中の 1 つのレコードを構成する値の集合です。任意の列を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。

たとえば、下の表の例では、'山田 太郎' は値の一例です。

		列				
		社員番号	氏名	部門コード	部門名	給与
行 [0010	山田 太郎	20	営業部	305000
		0022	大泉 一郎	20	営業部	410000
		0130	山口 淳子	38	経理部	287000
		0048	田坂 真衣子	38	経理部	340000
		0452	鈴木 義彦	15	総務部	262000
		⋮	⋮	⋮	⋮	⋮

例: 従業員表

- すべてのデータベースおよび表データは、表スペースに割り当てられます。
- 1 つの表の中で、行は必ずしも一定の順序に並んでいるわけではありません。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。

メモ: **表スペース** については、次のトピック「データ記憶管理」を参照してください。

4-1-3. 視点 (ビュー)

視点 (ビュー) 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。ビューは実際の表ではなく、また永続的な記憶域を必要としない動的な表、つまり「仮想表」です。ビューには、元となっている表の列や行のすべてまたは一部を含めることができます。

ビューを使うと、同じデータでも、ユーザーごとに違う方法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。

従業員表 ... 人事部のみ参照可能; 従業員の全データ物理的に格納

社員番号	氏名	部門コード	部門名	給与	歩合
0010	山田 太郎	20	営業部	305000	0.3
0022	大泉 一郎	20	営業部	410000	0.25
0130	山口 淳子	38	経理部	287000	-
0048	田坂 真衣子	38	経理部	340000	-
...

従業員一覧ビュー
(全社員参照可能)
...従業員表に格納されているデータのうち、全社員に見せてもよい情報のみを定義したビュー

社員番号	氏名	部門コード	部門名
0010	山田 太郎	20	営業部
0022	大泉 一郎	20	営業部
0130	山口 淳子	38	経理部
0048	田坂 真衣子	38	経理部
...

例: 従業員表と従業員一覧ビュー

それらのユーザーは、それぞれ実際の表から派生したビューを使って作業をします。どのビューも、表によく似ており、独自の名前が付けられています。

ビューを使うと、次のようなメリットがあります。

- ユーザーに見せたくないデータ構造を隠蔽することができる
(ユーザーごとにアクセスできる列や行を変えたビューを用意することにより、機密データへのアクセスを制御することもできる)
- ユーザーにわかりやすい形でシステムを説明できる
- アプリケーション開発者にとって使いやすい形に整理して表データを提供できる

4-1-4. 索引 (インデックス)

索引 は一式のキーであり、それぞれのキーは表の行を示しています。

たとえば、下の従業員表では、従業員番号に基づいた索引が作成されています。索引のキー値は、表の行を示すポインターを提供します。

索引を使用すると、ポインターを介して直接データへのパスを作成できるので、さらに効率よく行のアクセスを行えます。

従業員表

社員番号	氏名	部門コード	部門名	給与	歩合
0010	山田 太郎	20	営業部	305000	0.3
0022	大泉 一郎	20	営業部	410000	0.25
0130	山口 淳子	38	経理部	287000	-
0048	田坂 真衣子	38	経理部	340000	-
⋮	⋮	⋮	⋮	⋮	⋮

従業員表の索引 索引キー

(主キー)	(ポインタ)
0010	0010 レコードの格納先
0022	0022 レコードの格納先
0048	0048 レコードの格納先
0130	0130 レコードの格納先
⋮	⋮

- 表のデータはソートされていないので検索効率が悪い
- 索引の主キーはソートされているので検索効率がよい
- 索引キーは主キーとポインタの情報からなる 表のようなもの

索引の例

メモ: SQL 最適化プログラム は表のデータにアクセスする効率的な方法を自動的に選択します。最適化プログラムは、データへの最も速いアクセス・パスを判別する際に、索引を考慮に入れます。「レッスン2」の Visual Explain に関する説明を参照してください。

4-1-5. スキーマ

スキーマは、名前のつけられたオブジェクト(表やビューなど)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、表やビューなど名前のつ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。

メモ: CREATE SCHEMA ステートメントについては、ヘルプ『SQL 解説書』を参照してください。

名前のつ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スキーマの名前で修飾することにより、オブジェクトの名前とスキーマの名前を関連づけることができます。

書式	<スキーマ名>.<オブジェクト名>
----	-------------------

オブジェクト名は 2 つの部分からなっており、第一の部分は、そのオブジェクトの割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。

メモ: 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。

レッスン 4 データベース・オブジェクト

4-2. 表やビューの作成・削除

表やビューなどのデータベース・オブジェクトを作成したり、削除したりするには、DDL (データ定義言語) を使用します。

ヒント:

DDL については、標準の種別により定義内容が異なります。また、製品ベンダーによってもサポートしている機能が異なります。

4-2-1. 表の作成

表を作成するには、CREATE TABLE ステートメントを使います。表の作成では、列の名前やタイプ、制約などの定義を行います。

書式 CREATE TABLE <表名> (<列定義>, <列定義>, ...<列定義>)

メモ: 制約 については、後のレッスンで説明します。

例: CREATE TABLE ステートメントの実行例

次のステートメントは、CREATE TABLE ステートメントで PRES という名前の表を作成する例です。表で定義する列の名前やデータ・タイプ、オプション (NOT NULL、WITH DEFAULT) が指定されています。

```
CREATE TABLE PRES
( 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)
```

メモ: ここで使用されている列定義のオプションは、表の作成時に指定できるオプションのごく一部です。すべてのオプションについては、『SQL 解説書』の CREATE TABLE ステートメントの説明を参照してください。

4-2-2. ビューの作成

ビューを作成するには、CREATE VIEW ステートメントを使用します。

```
書式 CREATE VIEW <ビュー名> AS <基本表の照会>
```

例: 機密データを隠蔽するためのビューの作成

次のステートメントでは、STAFF 表から ID、氏名 (NAME)、部署 (DEPT)、職種 (JOB)、就業年数 (YEARS) の列のみを取り出した STAFF_ONLY ビューを作成しています。

```
CREATE VIEW STAFF_ONLY
AS SELECT ID, NAME, DEPT, JOB, YEARS
FROM STAFF
```

たとえば、ここで参照している STAFF 表には給与 (SALARY)、歩合 (COMM) といったデータが入っていますので、人事部門など特定の部門の担当者以外には見せたくないデータであったとします。しかし、給与と歩合を除いたデータについては、逆に、すべてのユーザーが自由に参照できるようにしたいような場合には、STAFF 表を基本表とした STAFF_ONLY ビューを作成することにより実現できます。

ビューは実データをとまなわれない仮想表ですので、別の表を作成してデータの重複管理を行う必要はありません。

作成したビューの内容は、表の照会と同様に、SELECT ステートメントを利用して照会することができます。

```
SELECT *
FROM STAFF_ONLY
```

例: 利用者の必要な情報だけを表示するビューの作成

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ビューを作成できます。そのビュー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
```

ヒント: ビューの操作と基本表

ビューは仮想表ですのでビュー自体には実データを持ちませんが、ビュー・オブジェクトに対して挿入や更新の操作を行うことによって、基本表に含まれる実際のデータを更新することができます。

ビューを使って基本表のデータを更新する際に、WITH CHECK OPTION 文節を指定して、定義に違反した操作を行わないようにすることができます。詳しくは、『SQL 解説書』を参照してください。

4-2-3. 表やビューの削除

SQL を使って表やビューなどのデータベース・オブジェクトを削除するには、**DROP** を使用します。

書式	DROP TABLE <表名>
	DROP VIEW <ビュー名>

- ビューの削除:
この SQL によって指定したビューは削除されますが、元になる表のデータが削除されることはありません。

4-2-4. DDL のまとめ

データベース・オブジェクトの作成や変更、削除を行うおもな DDL には、次のものがあります。

データベース・オブジェクトの作成

CREATE ステートメントで作成できるデータベース・オブジェクトは、次のとおりです。

- 表
- ビュー
- 表スペース
- 索引
- スキーマ
- ユーザー定義関数 (UDF)、ユーザー定義データ・タイプ (UDT)
- バッファ・プール
- トリガー

データベース・オブジェクトの変更

ALTER ステートメントを使用すると、次のデータベース・オブジェクトの特性のいくつかを変更することができます。

- 表
- 表スペース
- ビュー
- バッファ・プール

メモ: 索引の定義は変更することができません。索引の定義を変更したい場合は、一旦削除してから新規作成するしかありません。

データベース・オブジェクトの削除

DROP 文では、CREATE 文で作成したオブジェクトをすべて削除することができます。

メモ: DDL など DB2 UDB で使用できる SQL 言語の詳細については、「DB2 情報」の『SQL 解説書』を参照してください。

レッスン 4 データベース・オブジェクト

練習問題:

次のデータベース・オブジェクトのうち、ALTER 文で定義を変更できないものはどれですか？

- A. 表
- B. ビュー
- C. バッファ・プール
- D. 索引

次のデータベース・オブジェクトのうち、DROP 文で削除できないものはどれですか？

- A. 表
- B. 列
- C. ビュー
- D. 索引

レッスン 4 データベース・オブジェクト

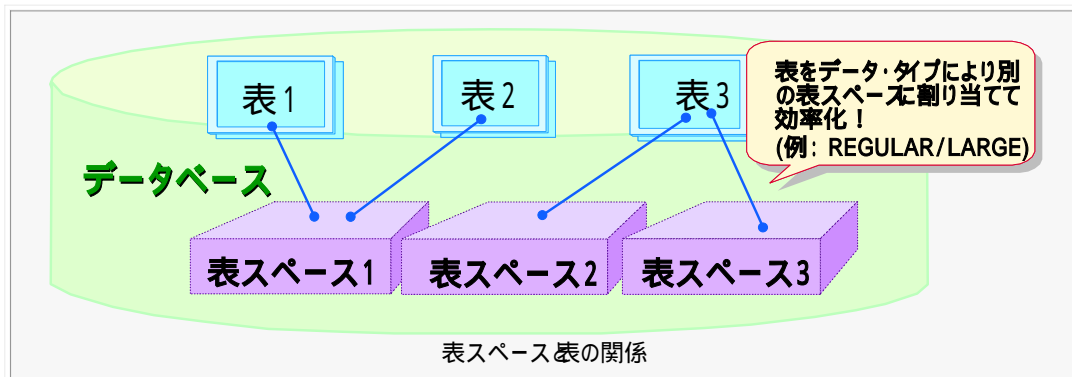
4-3. データ記憶管理

4-3-1. 表スペースとは

DB2 UDB では、データベースを格納する物理的な領域を **表スペース** や **コンテナ** という単位で管理しています。

表スペース とは、DB2 UDB データベースで表のデータを格納する領域です。表スペースは、データベース・オブジェクトの 1 つとして管理されます。

- 1 つの表スペースに、複数の表を割り当てることができます。表スペース単位でデータのバックアップ/復旧 (リストア) を行うことができます。
- 1 つの表を、複数の表スペースに割り当てることができます。(例: 使用するデータ・タイプによって表スペースを分けることにより、パフォーマンスを向上させる場合など)



ヒント: データベースを作成すると、デフォルトで次の3つの表スペースが作成されます。ユーザーが表を作成するときには、これらの表スペースを使用することもできますし、新しい表スペースを追加して使用することもできます。

- SYSCATSPACE ... すべてのシステム表を保持
- TEMPSPACE1 ... 表のソートや再編成を行うための SQL 動作中に使用されるクラッシュ空間
- USERSPACE1 ... ユーザーの表や索引などのオブジェクトすべてを格納

4-3-2. 表スペースの分類

表スペースは、構成する物理領域やデータ・タイプにより、次のように分類できます。

分類1: 表スペースを構成する物理領域による 分類

- SMS (システム管理ストレージ) 表スペース ... SMS 記憶域
- DMS (データベース管理ストレージ) 表スペース ... DMS 記憶域

分類2: 格納するデータ・タイプによる 分類

- Regular 表スペース ... 通常のデータを格納し、最も頻繁に使用される
- Large 表スペース ... LOB/LONGデータを効率よく格納する
索引の格納も可能 (DMS 記憶域のみ指定可能)
- Temporary 表スペース ... システムがスクラッチ領域として使用する

表スペースの分類のまとめ

表スペース	SMS	DMS
Regular	Y	Y
Large		Y
Temporary	Y	Y

4-3-3. 表スペースとコンテナ

表スペースに物理的な領域を割り当てるには、**コンテナ** という単位で定義を行います。

1 つの表スペースには、複数のコンテナを割り当てることができます。複数のコンテナを利用することにより、次のメリットがあります。

- 物理領域を複数のデバイスに分散することにより、データの検索速度が向上し、アクセスの付加も分散できる
- 表サイズが増加したときに、表スペースにコンテナを追加するだけで物理領域を増加させることができる (DMS 記憶域の場合)

コンテナには、次の 2 つの記憶域があります。

SMS (システム管理ストレージ) 記憶域

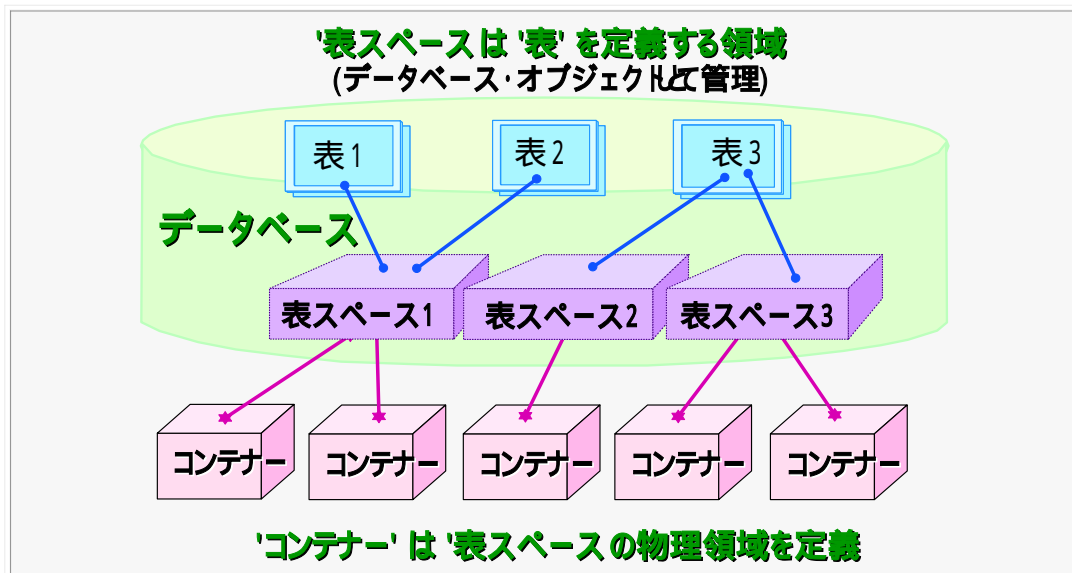
SMS 記憶域には、次のような特徴があります。

- OS がファイルを管理するので管理は容易
- 動的にサイズが割り当てられるので、サイズ見積もりは不要
- コンテナの追加はできない
- サブディレクトリーを指定

DMS (データベース管理ストレージ) 記憶域

DMS 記憶域には、次のような特徴があります。

- データベース管理システムが管理する
- SMS に比べてアクセスは速い
- 固定サイズなので、作成時に表サイズの見積もりが必要
- コンテナの追加やコンテナ・サイズの変更により、容量の増減が可能
- ファイル、未フォーマットディスクの領域 (RAW デバイス) を指定



表スペースとコンテナのまとめ

注意事項:

- LOB/LONGデータ型は Large 表スペースに格納する方がパフォーマンスが得られますが、Large表スペースの作成には、DMS を指定する必要があります。
- 1 つの表スペースに複数のコンテナを割り当てるときには、SMS か DMS のいずれかのタイプで統一する必要があります。
- 表と索引を別の表スペースに配置する場合は、DMS を使用します。

レッスン 4 データベース・オブジェクト

4-4. DB2 のデータ型

4-4-1. DB2 UDB のデータ・タイプ (型)

次の表は、DB2 UDB で使用できる主なデータ・タイプ (型) と、それぞれのデータ・タイプの特性をまとめています。

データ・タイプ	種類	特性	例または範囲
CHAR (X)	固定長文字列	最大長 254	'Sunny day'
VARCHAR (X)	可変長文字列	最大長 32672	'Sunny day'
SMALLINT	数値 (短整数)	長さ 2 バイト、精度 5 桁	範囲は -32768 ~ 32767
INTEGER	数値 (長整数)	長さ 4 バイト、精度 10 桁	範囲は -2147483648 ~ 2147483647
BIGINT	数値 (大整数)	長さ 8 バイト、精度 19 桁	範囲は -9223372036854775808 ~ 9223372036854775807
REAL	数値 (単精度浮動小数点数)	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 (倍精度浮動小数点数)	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 (P, S)	数値 (10 進数)	精度 5、位取り 2	範囲は $-10^{**31+1} \sim 10^{**31-1}$
DATE	日付 / 時刻 (日付)	3 つの部分で構成される値	1991-10-27
TIME	日付 / 時刻 (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

メモ: 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。

4-4-2. DB2 UDB のその他のデータ・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の値として受入可能なものを定義します。

ここでは、もっとも基本的なデータ・タイプの一部について説明します。

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
- 可変長グラフィック・ストリング

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。

NULL 値

NULL 値 は、その行のその列に、その他の値がないことを意味する特殊な値です。NULL 値は、どのデータ・タイプについても存在します。

ラージ・オブジェクト

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)** ... 長さが 2GB 以下の可変長 (バイト単位) の文字列です。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)** ... 長さが 2GB 以下の可変長 (バイト単位) の文字列です。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 は文字文字列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)** ... 長さが 2GB 以下 (2 バイト文字 1 073 741 823 文字) の可変長 2 バイト文字文字列です。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・文字列とみなすことができます。

4-4-3. ユーザー定義タイプ

ユーザー定義タイプ (UDT) は、ユーザーの用途に応じて作成できるデータ・タイプです。既存の基本データ型や、他のユーザー定義タイプ (UDT) を継承して、新しいデータ・タイプを定義することができます。

ヒント: ユーザー定義関数

DB2 UDB では、ユーザー定義タイプ (UDT) の他に、ユーザー独自のロジックを関数として定義しておくユーザー定義関数 (UDF) を利用することができます。

レッスン 4 データベース・オブジェクト

4-5. システム・カタログ・ビュー

4-5-1. システム・カタログ表

DB2 UDB は、各データベースごとに **システム・カタログ表** を作成し、維持しています。システム・カタログ表には、表、視点 (ビュー)、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。

システム・カタログ表は、データベースが作成された時点で作成され、データベースの操作中に自動的に更新されます。システム・カタログ表内の情報は明示的作成したり除去したりすることはできませんが、SQL ステートメントなどを利用して **システム・カタログ・ビュー** (カタログ視点) を照会し、内容を確認することができます。

4-5-2. システム・カタログ・ビューの参照

システム・カタログ・ビューは、データベース内の一般のビューとよく似ています。システム・カタログ・ビューデータを参照するには、他のビューに対してするのと同様にして SQL ステートメントを使います。

例: 次のステートメントは、SYSCAT.TABLES カタログに含まれている表の名前などの情報を表示する例です。

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

このステートメントの結果の表示例は、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

その他のシステム・カタログビュー

次の表は、おもなカタログ視点 (カタログ・ビュー) の一覧です。

一覧で紹介されている、スキーマ名 SYSCAT を持つカタログ視点の他に、スキーマ名 SYSSTAT のカタログ視点もあります。SYSCAT のカタログ視点を読み取りのみであるのに対し、SYSSTAT のカタログ視点は更新が可能であり、統計情報の一部を変更することができます。

説明	カタログビュー
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.ROUTINEPARMS
参照制約	SYSCAT.REFERENCES
スキーマ	SYSCAT.SCHEMATA
表制約	SYSCAT.TABCONST
表	SYSCAT.TABLES
トリガー	SYSCAT.TRIGGERS
ユーザー定義関数	SYSCAT.ROUTINES
視点 (ビュー)	SYSCAT.VIEWS

メモ：このほかにも多くのカタログビューが用意されています。詳しくは、『SQL 解説書』や『管理の手引き』を参照してください。

レッスン 4 データベース・オブジェクト**練習問題:**

DB2 UDB について説明した次の文章のうち、正しいものを選択しなさい。

- A. データベース・オブジェクトの定義や権限などの情報は、データ・ディクショナリに格納されている
- B. データベース・オブジェクトの定義や権限などの情報は、システム・カタログ表に格納されている
- C. データベース・オブジェクトの定義や権限などの情報は、システム・カタログビューに格納されている
- D. データベース・オブジェクトの定義や権限などの情報は、データベース構成ファイルに格納されている

DB2 UDB のコンテナや表スペースについて説明した次の文章のうち誤っているものを選択しなさい。

- A. DB2 UDB では、1 つの表スペースに複数のコンテナを割り当てることができる。
- B. DB2 UDB では、1 つの表スペースに複数の表を割り当てることができる。
- C. DB2 UDB では、1 つのコンテナを複数の表スペースに割り当てることができる。
- D. DB2 UDB では、1 つの表を複数の表スペースに割り当てることができる。

DB2 UDB のコンテナや表スペースについて説明した次の文章のうち正しいものを選択しなさい。

- A. SMS は通常固定サイズなので、作成時にサイズの見積りが必要である
- B. SMS は OS の管理する表スペースなので、アクセスの速度が速い
- C. DMS は DBMS の管理する表スペースなので、データ・サイズの変化に伴って自動的にサイズが拡大・縮小する
- D. Large 表スペースは DMS コンテナを使用しなくてはならない

- メモ -



レッスン5

インスタンスとセキュリティ

このレッスンでは、次のトピックを学習します。

- 5-1. インスタンスと管理サーバー
- 5-2. オブジェクトの権限と特権
- 5-3. データの参照と保全性

レッスン 5 インスタンスとセキュリティ

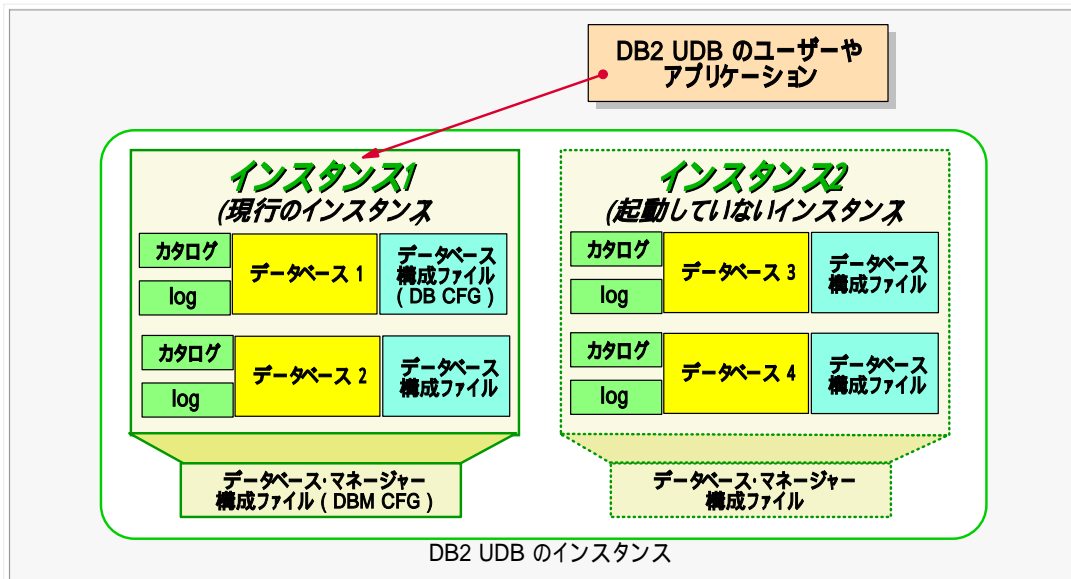
5-1. インスタンスと管理サーバー

5-1-1. インスタンスとは

インスタンスとは、DB2 UDB のシステム・プロセス起動の単位で、データベース・マネージャーとも呼ばれます。

- 1 台のマシンで複数のインスタンスを作成可能
ただし、アプリケーションやユーザーが参照するインスタンスは一度に 1 インスタンスのみ (環境変数で指定)
- データベース・マネージャー構成ファイル (DBM CFG) のパラメーターで属性を指定
- 1 つのインスタンス上に複数のデータベースを作成可能
- OS コマンドや GUI ツール (コントロール・センター) から管理

ユーザーやアプリケーションは、環境変数 (または、レジストリー変数) の指定により起動している 1 つのインスタンスを参照します。



5-1-2. 管理サーバーについて

DB2 UDB 管理サーバー (DAS) は、サーバー管理専用の特別なコンポーネントです。管理サーバーは、自動的に作成および開始されます。

DB2 管理ツールや構成アシスタントは、DB2 UDB サーバーに常駐する管理サーバーを使用してデータベースをカタログ化 (登録すること) します。

レッスン 5 インスタンスとセキュリティ

5-2. オブジェクトの権限と特権

DB2 UDB のデータベース資源にアクセスするプロセスは、次のとおりです。

- DB2 UDB サーバーにアクセスするための認証
- DB2 UDB で管理するデータベースへのアクセス制御の検査

認証に成功すると、DB2 UDB で管理されるデータベースへのアクセス制御に対して、正しいデータベース権限とオブジェクト特権を持っているかどうかを検査されます。DB2 UDB では、**権限**と**特権**によりこの検査を行います。

5-2-1. 権限とは

DB2 UDB には、次の 5 種類の **権限** があります。

- **SYSADM** システム管理者権限
- **DBADM** データベース管理者権限
- **SYSCTRL** システム制御権限
- **SYSMAINT** システム保守権限
- **LOAD** ロード権限

SYSADM システム管理者権限

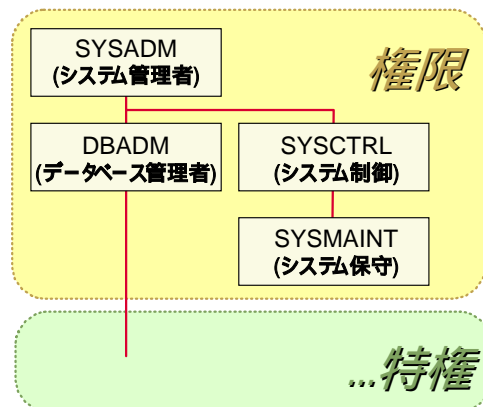
- 最高位の権限 ... データベース・マネージャーによって作成/管理されるすべての資源を制御
- データベース・マネージャー構成ファイル (DBM CFG) の変更
- SYSADM、SYSCTRL、SYSMAINT 権限の付与/取り消し (DBM CFG で設定)
- DBADM 権限の付与/取り消し (GRANT/REVOKE で設定)

SYSCTRL システム制御権限

- システム資源に影響を与える操作を行える権限
- インスタンスやデータベースを停止できる
- データベースや表スペースの作成/削除、バックアップ/リストアを行える
- データベース内のデータへの直接アクセスはできない

SYSMAINT システム保守権限

- 第2レベルのシステム制御権限
- 特定のインスタンスに関連したすべてのデータベースに対する保守権限
- データベースや表スペースの作成/削除は行えないが、バックアップ/リストアを行える
- データベース内のデータへの直接アクセスはできない



DBADM データベース管理者権限

- 単一のデータベースに固有の管理権限
- データベース内のオブジェクトの作成/ 除去
- データベース内のデータへのアクセス特権 (CONTROL 特権など) の付与/ 取り消し
- データベースの作成/ 削除変更は行えない

LOAD ロード権限

- LOADユーティリティ、AutoLoaderユーティリティによるロードを実行できる権限
- LOADのオプションに応じて、INSERT権限、DELETE権限も必要となる

権限のまとめ

機能	SYSADM	SYSCTRL	SYSMAINT	DBADM	LOAD
データベースの移行					
DBM CFG の更新					
DBADM の許可/ 取消し					
db/node/dcs ディレクトリの更新					
システムからのユーザーの強制ログオフ					
データベースの作成/ 削除					
表スペースの作成/ 削除/ 変更					
新しいデータベースへの復元					
DB CFG の更新					
データベースまたは表スペースのバックアップ					
既存のデータベースへの復元					
ロールフォワード回復の実行					
データベース・インスタンスの開始/ 停止					
表スペースの復元					
トレースの実行					
DBM または DB のスナップショットの記録					
表スペースの状態の照会					
ログ履歴ファイルの更新					
表スペースの静止化					
表の再編成					
RUNSTATS ユーティリティの実行					
ログ・ファイルの読み取り					
イベント・モニタの作成/ 活動化/ 削除					
ロードの実行					

ヒント:

- SYSADM、SYSCTRL、SYSMAINT の 3 つの権限は、データベース・マネージャー構成ファイル内のパラメータに、各権限が付与されるグループ名を指定します。
- DBADM 権限は、SYSADM 権限を持つユーザーから GRANT 文で付与されます。
- LOAD 権限は、SYSADMまたはDBADM 権限を持つユーザーから GRANT 文で付与されます。

5-2-2. 特権とは

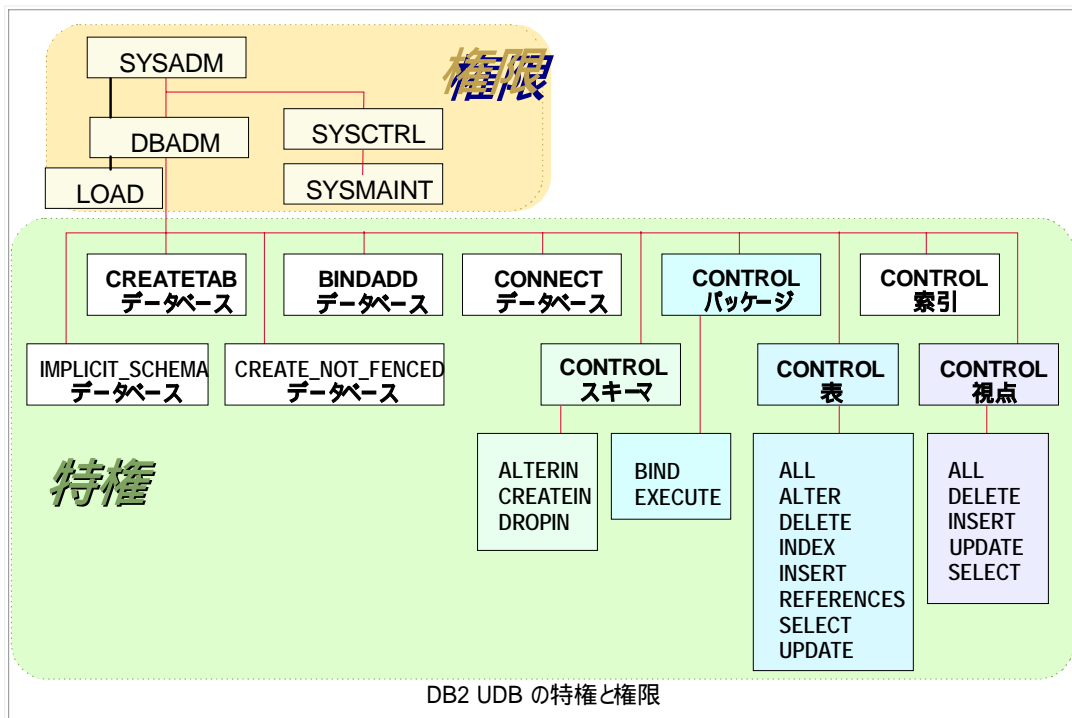
SYSADM または DBADM 権限をもつユーザーは、データベース内のオブジェクトでさまざまな操作を行うことができます。

メモ: SYSCTRL 権限と SYSM maint 権限では、データベース内のデータの操作を行うことはできません。

それらの権限に対して、**特権**とは、あるユーザーまたはグループに対し、特定のデータベース・オブジェクトに指定した方法でアクセスする権利を与えるものです。

特権の与えられたユーザーは、SYSADM ユーザーや DBADM ユーザーとは異なり、データベース・オブジェクトの制限された操作のみを行えます。

次の図は、主なDB2 UDB の権限と特権を示しています。



5-2-3. 権限と特権のまとめ

DB2 UDB の権限と特権のしくみは、次のとおりです。

- 権限/ 特権は、ユーザー、グループ、PUBLIC に対して付与します。
ここでいうユーザーとグループはオペレーティング・システムの管理するユーザー、グループです。
- DBADM 権限と特権の付与には、DDL の GRANT/ REVOKE を使用します。
DBADM 権限や特権の付与の状況は、システム・カタログの照会で確認できます。
- データベースが作成されると、自動的に、PUBLIC に対して CREATETAB、BINDADD、CONNECT 特権が付与されます。
- CONTROL 特権を与えられたユーザーは、付与された権限の内容により、特定のデータベース・オブジェクトにアクセスしたり、そのオブジェクトに関する特権を他のユーザーに付与 (GRANT) したり、取り消し (REVOKE) したりすることができます。

レッスン 5 インスタンスとセキュリティ

5-3. データの参照と保全性

5-3-1. 制約

制約とは、データベース・マネージャーによって課せられる規則のことです。

制約には次の3つのタイプがあります。

- **固有限制** (ユニーク制約) ... 表の1つまたは複数の列に、重複する値を指定することを禁止する規則
- **参照制約** ... 1つまたは複数の表の列の値に関する論理的な規則
- **表検査制約** ... 特定の表に追加されるデータに対して制限を設定する規則

ヒント:

参照制約および表検査制約はオン / オフの切り換えが可能です。一般に、大量のデータをデータベースにロードする場合には、制約の実施検査をオフにします。

5-3-2. 固有限制 (ユニーク制約)

固有限制 (ユニーク制約) は、表内でその値が一意である場合にのみキーの値を有効とする規則です。

固有限制を定義するには、CREATE TABLE または ALTER TABLE ステートメントで、次の定義を行います。

- PRIMARY KEY 文節の使用 ... 指定した列で構成される主キーを定義します。
- UNIQUE 文節の使用 ... 指定した列で構成される固有キー (ユニークキー) を定義します。

固有限制を指定すると、定義された列に対して **固有索引** (ユニーク索引) が自動的に作成されます。固有索引は、固有限制がつけられた列の変更時に、キーの固有性を維持するために、データベース・マネージャーによって使用されます。固有限制に指定される列は、必ずしもNOT NULLとして定義する必要はありませんが、NULL列が許されるのは1列のみになります。主キーを定義する列は、NOT NULLとして定義する必要があります。

例: 次の例は、EMP 表を作成する CREATE ステートメント内で、ID を主キーに定義している部分を示しています。PRIMARY KEY 指定を行う ID 列には、あらかじめ NOT NULL オプションを設定しておく必要があります。

```
CREATE TABLE EMP
  (ID          SMALLINT NOT NULL,
   NAME       VARCHAR(9),
   :
   :
   PRIMARY KEY (ID),
   :)
```

5-3-3. 参照制約の概念と用語

参照保全とは、すべての外部キーのすべての値が有効であるデータベースの状態のことです。**参照制約**は、外部キーの値を以下の場合にのみ有効とする規則です。

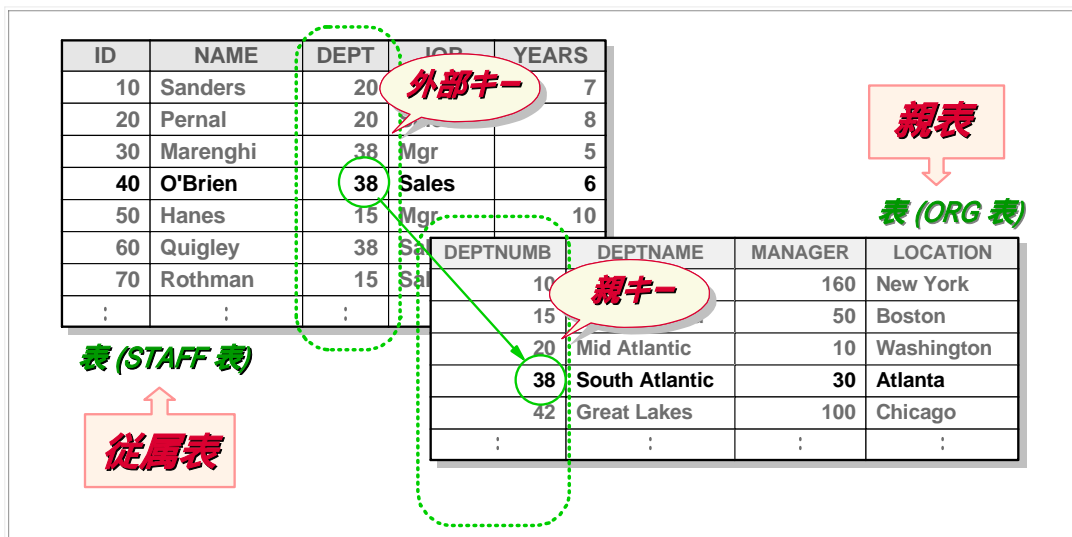
- 外部キーの値が親キーの値として現れる場合、または
- 外部キーの構成要素の一部がヌル値の場合

参照制約は、CREATE TABLE または ALTER TABLE ステートメントで定義することができます。参照制約は、INSERT、UPDATE、DELETE などのステートメント実行の過程で評価されます。

メモ：固有制約、参照制約を定義する SQL ステートメントの構文の詳細については、『SQL 解説書』などを参照してください。

参照保全の規則に関係したおもな用語には、次のものがあります。

用語	説明
親キー	参照制約の基本キーまたは固有キー（ユニークキー）。
親表	参照制約の親キーを含む表。表は、任意の数の参照制約に関して親として定義することができます。ある参照制約で親となっている表が、他の参照制約の従属となる場合もあります。
親行	その親キーの値が従属表の中の少なくとも 1 つの外部キーと一致する、親表の 1 つの行のことです。
外部キー	同じ表または別の表の固有キー（ユニークキー）または基本キーを参照する、表内の 1 つの列または 1 組の列のことです。
従属表	最低限 1 つの参照制約が定義に含まれている表。表は、任意の数の参照制約に関して従属として定義することができます。ある参照制約で従属となっている表が、他の参照制約の親となる場合もあります。
従属行	親キーの値と一致する非ヌルの外部キー値を持つ従属表の 1 つの行のことです。



5-3-4. 参照制約の規則

参照制約には、次のような規則があります。

- 挿入規則
- 更新規則
- 削除規則

挿入規則 (INSERT 規則)

参照制約の挿入規則は、外部キーが指定されている場合に暗黙的に指定されます。この規則が指定されている場合、外部キーのヌル値以外の挿入値が、親表の親キーの何らかの値と一致していなければなりません。

ID	NAME	DEPT	JOB	YEARS
10	Sanders	20	Mgr	7
20	Pernal	20	Sales	8
30	Marenghi	38	Mgr	5
40	O'Brien	38	Sale	
50	Hanes	15	Mgr	
60	Quigley	38	Sale	
70	Rothman	15	Sale	
:	:	:	:	:

表 (STAFF 表)

DEPTNUMB	DEPTNAME	MANAGER	LOCATION
10	Head Office	160	New York
15	New England	50	Boston
20	Mid Atlantic	10	Washington
38	South Atlantic	30	Atlanta
42	Great Lakes	100	Chicago
:	:	:	:

表 (ORG 表)

親キー

未登録

160	Suzuki	98	Clerk	1
-----	--------	----	-------	---

親キーに登録されていない値は外部キー列に指定できない
STAFF 表に行を挿入できない

更新規則 (UPDATE 規則)

参照制約の更新規則は、参照制約の定義時に指定します。指定項目として、ON UPDATE 文節で NO ACTION (デフォルト値) または RESTRICT を指定します。

更新規則のオプション

RESTRICT	● 従属行がある場合は、親行の更新を禁止します。
NO ACTION	● (デフォルト値) 親キーの更新を禁止します。

削除規則 (DELETE 規則)

参照制約の削除規則は、参照制約の定義時に指定されます。指定項目として、ON DELETE 文節で NO ACTION (デフォルト値)、RESTRICT、CASCADE、SET NULL を指定します。SET NULL を指定できるのは、外部キーの列の中にヌル値が可能なものがある場合だけです。

削除規則のオプション

RESTRICT	<ul style="list-style-type: none"> 従属行がある場合は、親行の削除を禁止します。
NO ACTION	<ul style="list-style-type: none"> (デフォルト値) 親行の削除を禁止します。
SET NULL	<ul style="list-style-type: none"> 親行を削除します。 従属行がある場合は、外部キーの値を NULL に設定します。従属表の外部キー列がヌル値を許可していない場合には、このオプションを設定できません。
CASCADE	<ul style="list-style-type: none"> 親行を削除します。 従属行がある場合、削除作業が伝播し、従属するすべての行を削除します。

メモ: ON UPDATE 文節や ON DELETE 文節で指定する RESTRICT と NO ACTION の違いなど、詳しくは、『SQL 解説書』などを参照してください。

5-3-5. 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。

例: 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 (DEPT) の値は 10 ~ 100 の範囲内でないといけない。

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 AND SALARY > 40500) )
```

レッスン 5 インスタンスとセキュリティ**練習問題：**

次のうち、データベースを作成できるのはどの権限ですか？

- A. SYSADM
- B. SYSMAINT
- C. DBADM
- D. DBCTRL

行に対応するキーの値が子テーブルに存在する場合、親テーブルからその行を削除できないようにするための削除規則は、次のうちのどれですか？

- A. CASCADE
- B. RESTRICT
- C. NOT DELETE ON
- D. SET NULL

従業員表に給与データを入力する列を作成し、その列には一定の金額までしか入力できないように制限するためには、次のうちどの方法を定義しますか？

- A. 列にプライマリー・キー制約を定義する
- B. ユーザー定義のスカラー関数を定義する
- C. 表の検査制約を定義する
- D. 参照制約を定義する

次のような定義の JINJI 表があります。

JID	INTEGER
JNAME	CHAR(30)
SALARY	DECIMAL(10,2)
DEPT	INTEGER

この表のデータは、原則として DB2 のユーザー全員に参照させたいのですが、一部の列だけは特定の人だけにしか見せないようにしたいと考えています。どのような方法をとればよいでしょうか？

- A. GRANT 文で PUBLIC に SELECT 特権を与え、その後、SALARY 列のみ SELECT 特権を除く
- B. GRANT SELECT ON JINJI TO PUBLIC
- C. JINJI 表にビューを作成する
- D. GRANT 文で CONTROL 特権を与え、その後、REVOKE 文で SELECT 特権を取り消す

- メモ -



レッスン 6

データベースの同時実行制御

このレッスンでは、次のトピックを学習します。

- 6-1. トランザクションと並行性
- 6-2. ロック

レッスン 6 データベースの同時実行制御

6-1. トランザクションと並行性

6-1-1. トランザクションとは

トランザクションは、作業単位とも呼ばれます。トランザクションには、少なくとも 1 つの SQL 文が含まれます。言いかえると、個々の SQL は、トランザクションの一部であるといえます。SQL 文によって実行された処理は DBMS によって追跡され、トランザクション・ログに出力されます。

DB2 のトランザクションは、最初の SQL 文の処理時に暗黙的に開始され、COMMIT 文または ROLLBACK 文によって明示的に、あるいは暗黙的に終了します。

- **COMMIT 文**
トランザクション内で処理されたステートメントを永続的に反映させます。
- **ROLLBACK 文**
トランザクションを元の状態に戻します。データ破損時の回復処理などに使用します。

例: トランザクションの例 (1 行目 ~ 4 行目までが 1 つのトランザクション)

```

1: >INSERT INTO STAFF (ID, NAME, DEPT) VALUES (190, 'Suzuki', 84)
2: >SELECT ID,NAME,DEPT FROM STAFF WHERE ID=190

  ID   NAME      DEPT
  ----  -
  190 Suzuki      84

3: >SELECT ID,NAME,DEPT FROM STAFF WHERE ID=150

  ID   NAME      DEPT
  ----  -
  150 Edwards    84

4: >UPDATE STAFF SET DEPT='38' WHERE ID=150
5: >COMMIT
6: >SELECT ID,NAME DEPT FROM STAFF WHERE ID=150

  ID   NAME      DEPT
  ----  -
  150 Edwards    38

```

メモ: COMMIT 文や ROLLBACK 文は、DCL (データ制御言語) と呼ばれます。

レッスン 6 データベースの同時実行制御

6-2. ロック

6-2-1. DB2 UDB のロックのしくみ

DB2 では、**ロック制御** により、トランザクション・レベルまたは文レベルで、常に整合性のある最新状態のデータを提供します。

DB2 では、読み込み / 書込みに関わらず、緻密なロック制御が行われます。

ロック・モード

DB2 UDB には、12 種類のロック・モードがあります。

- | | | | | |
|-----------|---------|----------|-----------|---------|
| 1. 意図なし | 2. 意図共用 | 3. 次キー共用 | 4. 共用 | 5. 意図排他 |
| 6. 意図排他共用 | 7. 更新 | 8. 次キー排他 | 9. 次キー弱排他 | 10. 排他 |
| 11. 弱排他 | 12. 超排他 | | | |

ロックの単位

DB2 では、次の単位でロックが取得されます。

- 行 (デフォルト)
- 表
- 表スペース

ロックの期間やモードを調整するためには、分離レベルを変更します。

6-2-2. 分離レベル

DB2 では、ANSI に準拠した 4 種類の分離レベルを実装しています。

- 未コミット読み取り (UR: Uncommitted Read)
- カーソル固定 (CS: Cursor Stability) (デフォルト)
- 読み取り固定 (RS: Read Stability)
- 反復可能読み取り (RR: Repeatable Read)

未コミット読み取り (UR)

- DB2 UDB がサポートする、最も低い分離レベルです。
- コミットされていないデータを含むすべてのレコードにアクセスすることができます。
- 更新が確定していない未確定なデータにもアクセスが可能なため、ダーティー読み取りとも呼ばれます。

カーソル固定 (CS)

- カーソル固定分離レベルは、デフォルトの分離レベルです。
- トランザクションの実行時に、カーソルが置かれている行をロックします。固定したカーソル行のデータは保証され他のアプリケーションから変更されることはありませんが、それ以外の行が保証されません。
- 行のロックは、カーソルが移動するか、COMMIT か ROLLBACK によりトランザクションが終了するまで保持されます。

読み取り固定 (RS)

- 読み取り固定は、検索結果に含まれている行をすべてロックします。
- COMMIT か ROLLBACK によりトランザクションが終了するまでロックが保持されます。

反復可能読み取り (RR)

- DB2 UDB でサポートする、最も高い分離レベルです。
- 検索結果に含まれているすべての行に加え、アプリケーションが作業単位内で参照するすべての行をロックします。
ロックされる要素が大きい場合は、表ロックが獲得される場合もあります。
- COMMIT か ROLLBACK によりトランザクションが終了するまでロックが保持されます。

6-2-3. ロック待機とデッドロック

あるトランザクションで、すでに排他 (ロック) されている行の更新を要求した場合、排他したアプリケーションがロックを解放することを待ちます。このことを **ロック待機** と呼びます。

DB2 では、ロック待機が無限に続くことを防ぐために、ロック待ちの制限時間 (**ロック・タイムアウト**) を設定することができます。

メモ: ロック・タイムアウトは、デフォルトでは使用不可に設定されています。

デッドロック

並行するトランザクションの処理内容により、ロックの競合が起きることがあります。このことを **デッドロック** と呼びます。

デッドロックの例:

鈴木さんと山田さんは、同じ表に対してトランザクションを実行しています。

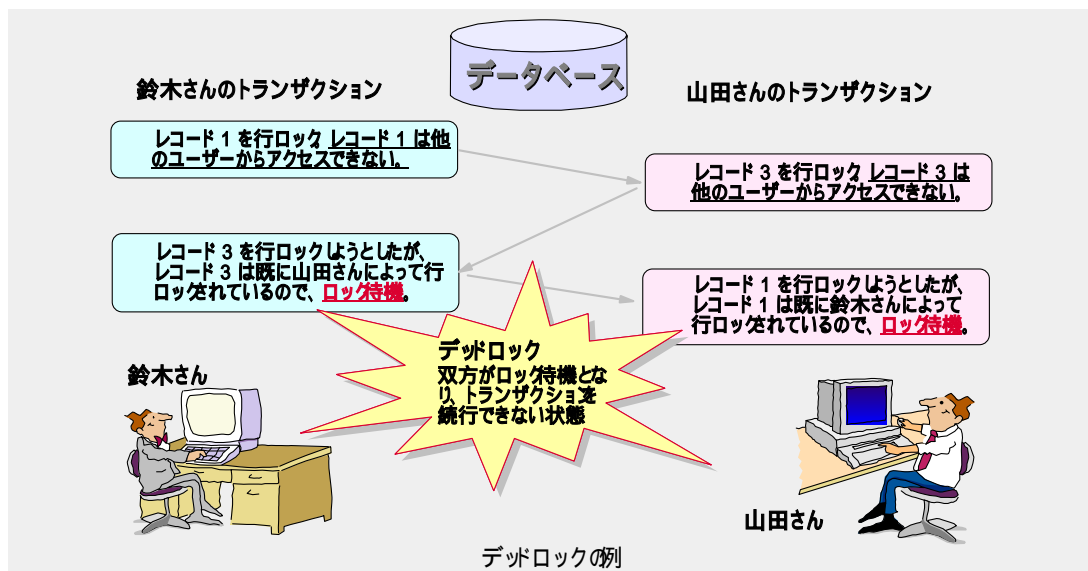
鈴木さんは、レコード 1 を行ロックした。

山田さんは、レコード 3 を行ロックした。

鈴木さんは、レコード 3 を行ロックしようとしたが、レコード 3 は既に山田さんによってロックされているので、鈴木さんの処理はロック待機となる。

山田さんは、レコード 1 を行ロックしようとしたが、レコード 1 は既に鈴木さんによってロックされているので、山田さんの処理はロック待機となる。

このような状態の場合、鈴木さんと山田さんは、お互いのロック解放を無限に待機することになります。



DB2 UDB では、バックグラウンド・プロセスにより、デッドロックを検出することができます。この機能によりデッドロック状態が検出されると、片方のトランザクションはエラー・コードを受け取り、自動的にロールバックされます。片方のトランザクションがロールバックされるとロックが解放されるので、もう一方の待機しているトランザクションは、処理を実行することができます。

DB2 UDB では、デフォルト設定で、デッドロックを起こしている片方のトランザクション全体をロールバックしますが、設定を変更することにより、文単位のロールバックを指定することもできます。

レッスン 6 データベースの同時実行制御**練習問題:**

ロックの取得される単位として誤っているのは、次のうちどれですか？

- A. 行
- B. 表スペース
- C. 列
- D. 表

RR (Repeatable Read) 分離レベルを使用して、アクティビティによって更新ロックが開放されるのは、次のうちどの場合ですか？

- A. トランザクションがコミットされた
- B. トランザクションが、UPDATE 文で変更された
- C. 行をアクセスするカーソルが閉じられた
- D. 行をアクセスするカーソルが次の行に移動した

次のうち、結果セットを作成するために読み込まれる行のすべてをロックする分離レベルはどれですか？

- A. カーソル固定 (CS: Cursor Stability)
- B. 未コミット読み取り (UR: Uncommitted Read)
- C. 反復可能読み取り (RR: Repeatable Read)
- D. コミットなし (NC: No Commit)

DB2 のデッドロックについて述べている次の文章のうち、正しいものはどれですか？

- A. DB2 でデッドロック状態が発生すると、自動的に両方のトランザクションがエラーとなり、双方の処理内容が破棄される
- B. DB2 でデッドロック状態が発生すると、両方のトランザクションに警告メッセージが表示され、いずれか一方がロールバックを行うまでロック待機が持続する
- C. DB2 でデッドロック状態が発生すると、自動的に両方のトランザクションがエラーとなり、双方の処理内容がロールバックされる
- D. DB2 でデッドロック状態が発生すると、自動的に片方のトランザクションがエラーとなり、処理内容がロールバックされる

- メモ -



付録
参考資料

付録

SAMPLE データベースのデータ

次の表は、SAMPLE データベースで作成されるおもな表のうち、このテキストで頻繁に参照している表に入力されているデータの一覧です。

ORG 表の定義

列名	データ型	制約
DEPTNUMB	SMALLINT	NOT NULL
DEPTNAME	VARCHAR(14)	
MANAGER	SMALLINT	
DIVISION	VARCHAR(10)	
LOCATION	VARCHAR(13)	

ORG 表に登録済みのサンプルデータ

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

STAFF 表の定義

列名	データ型	制約
ID	SMALLINT	NOT NULL
NAME	VARCHAR(9)	
DEPT	SMALLINT	
JOB	CHAR(5)	
YEARS	SMALLINT	
SALARY	DECIMAL(7,2)	
COMM	DECIMAL(7,2)	

STAFF 表に登録済みのサンプルデータ

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	18357.50	-
20	Pernal	20	Sales	8	18171.25	612.45
30	Marenghi	38	Mgr	5	17506.75	-
40	O'Brien	38	Sales	6	18006.00	846.55
50	Hanes	15	Mgr	10	20659.80	-
60	Quigley	38	Sales	-	16808.30	650.25
70	Rothman	15	Sales	7	16502.83	1152.00
80	James	20	Clerk	-	13504.60	128.20
90	Koonitz	42	Sales	6	18001.75	1386.70
100	Plotz	42	Mgr	7	18352.80	-
110	Ngan	15	Clerk	5	12508.20	206.60
120	Naughton	38	Clerk	-	12954.75	180.00
130	Yamaguchi	42	Clerk	6	10505.90	75.60
140	Fraye	51	Mgr	6	21150.00	-
150	Williams	51	Sales	6	19456.50	637.65
160	Molinare	10	Mgr	7	22959.20	-
170	Kermisch	15	Clerk	4	12258.50	110.10
180	Abrahams	38	Clerk	3	12009.75	236.50
190	Sneider	20	Clerk	8	14252.75	126.50
200	Scoutten	42	Clerk	-	11508.60	84.20
210	Lu	10	Mgr	10	20010.00	-
220	Smith	51	Sales	7	17654.50	992.80
230	Lundquist	51	Clerk	3	13369.80	189.65
240	Daniels	10	Mgr	5	19260.25	-
250	Wheeler	51	Clerk	6	14460.00	513.30
260	Jones	10	Mgr	12	21234.00	-
270	Lea	66	Mgr	9	18555.50	-
280	Wilson	66	Sales	9	18674.50	811.50
290	Quill	84	Mgr	10	19818.00	-
300	Davis	84	Sales	5	15454.50	806.10
310	Graham	66	Sales	13	21000.00	200.30
320	Gonzales	66	Sales	4	16858.20	844.00
330	Burke	66	Clerk	1	10988.00	55.50
340	Edwards	84	Sales	7	17844.00	1285.00
350	Gafney	84	Clerk	5	13030.50	188.00

付録

練習問題の解答

レッスン 1 練習問題の解答

C A

レッスン 2 練習問題の解答

A,D C B,D B

レッスン 3 練習問題の解答

A D
D A
B A
B D**レッスン 4 練習問題の解答**

D B
B C D**レッスン 5 練習問題の解答**

A B C C

レッスン 6 練習問題の解答

C A C D

