01

データベースとDBMS

データベース

データベースは、たくさんの情報をコンピューターの中に整理して保管する大きなファイルキャビネットのようなものです。キャビネットには、たくさんの引き出しがありデータが整理されています。

☑ データベースはデータの入れ物であり“しくみ”

データベースとは、情報を整理して保存する特別な方法や場所のことです。たとえば、学校のデータベースでは、生徒や先生の名前、生年月日、成績、出席記録などが情報として保存されています。この情報はコンピューターの中にデジタルで保管されており、簡単に情報を探したり、更新したり、整理したりすることができます。データベースは大事な情報を整理し、必要なときにすぐに使えるようにするための便利なしくみや場所なのです。

☑ データベースとDBMS

また、データベースは、**情報やデータの集合体を意味**します。この集合体は、テキスト、数値、画像、音声など、さまざまな種類のデータを含みます。通常、**データは表と呼ばれる形式で整理され、これらの表はフィールド（列）とレコード（行）から構成**されます。

データベースと密接な関係を持つものが**DBMS**（ディービーエムエス）（Database Management System（データベース マネジメント システム））です。DBMSは、データベースを効果的に管理するソフトウェアシステムで、データベースを作成、更新、操作、維持する役割を果たします。DBMSは、データベースにアクセスするための標準的なインターフェイスを提供し、ユーザーやアプリケーションがデータベースに対してSQL（エスキューエル）を実行（P.20参照）したり、データを操作したりするのに役立ちます。

つまり、データベースは情報を格納し、DBMSはそのデータベースを**管理・操作するためのシステムやソフトウェア**といえます。データベースは情報の宝庫であり、DBMSはその宝庫を効果的に管理し、データを利用可能にするためのシステムなのです。

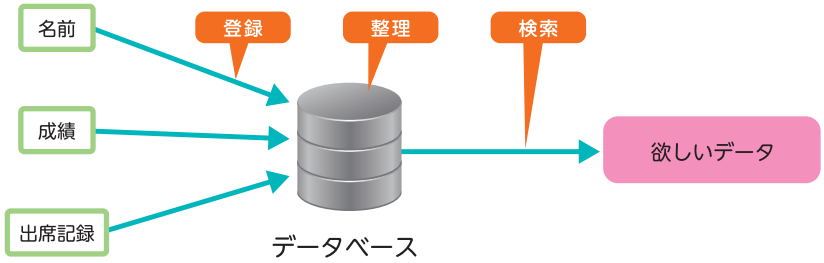
データベース登場の背景

データベースは、ミサイルの弾道情報（弾頭の質量、初速、風速、目標の距離など）を一元的に収集し、管理するためのツールとして、1940年アメリカのベル研究所の試作機で登場しました。

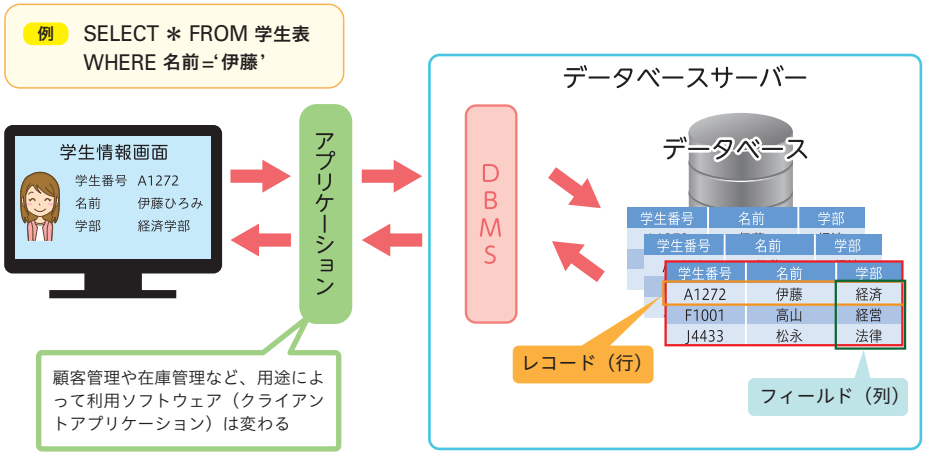
表

表の作成は、データの種類や関係性などを考慮する必要があります。表の設計が適切でないと、データの検索や更新などの操作が効率的に行えなかったり、データの整合性が保てなくなったりする可能性があります。なお、データベースは、すべて表で保存されるわけではありません。画像や動画などは、表とは別の場所に保存されるケースもあります。

データベースの役割



データベース活用におけるDBMSとアプリケーションの位置付け



ONE POINT

アプリケーションとは

アプリケーションは、組織やビジネスにおいて日常業務を効率的に支援し、管理するためのソフトウェアであり、特定の業界や業務プロセスに合わせて設計されます。データベースを利用したアプリケーションには、顧客管理ソフトウェア、在庫管理ソフトウェア、給与計算ソフトウェアなどさまざまなものがあり、業務プロセスの自動化やデータの集計と分析、情報共有を目的とし、組織内で重要な役割を果たしています。

関連用語 SQL（P.20）、データ制御言語（P.22）、データ操作言語（P.22）、データ定義言語（P.22）、トランザクション制御言語（P.22）

01

SELECT文からの理解

検索の基本

SQLは、データベースから必要な情報を取得するための強力なツールです。データベースから情報を取り出す基本的なSQL文がSELECT文です。ここでは、SQLを使った検索の基本的な方法について説明します。

☑ データベースの検索でできること

SELECT文は、データベース内の表からデータを取り出す際に使用します。たとえば顧客情報を保存した顧客表から顧客名とメールアドレスを取得することができます。多くのデータの中から特定の条件に合うデータだけを取り出すことができ、たとえば年齢が30以上の顧客を検索する場合などはWHERE句を使います。

検索結果を特定の順序で並べることもできます。顧客の名前順（昇順）に並べたいようなケースでは、ORDER BY句を使います。

さらには、特定のパターンに一致するデータを検索することもできます。たとえば名前に「田」が含まれる顧客を検索する場合はLIKEを使います。

またSQLには便利な集計関数があります。たとえば、顧客の総数を数えるなどのデータを集計したい場合、COUNT関数を使います。ほかにも、特定の列の合計を計算するSUM関数、平均を計算するAVG関数、最大値や最小値を取得するMAX関数、MIN関数などがあります。

☑ 検索の基本となるSELECT文

以上のように、SQL文ではSELECT文を基本に、WHERE句で条件を付けたり、ORDER BY句で並べ替えたり、LIKEやIN、BETWEENで柔軟な検索を行ったりできます。また、集計関数やGROUP BY句を使ってデータを集計することも可能です。**SELECT文を理解することで、SQLの基本が身に付き**、データベースの操作がより身近で便利なものになります。

文と句

SQLクエリでいうところの「文」と「句」の違いは、前者がデータベースに対して実行できる単位の命令であるのに対し、後者は文を構成する要素であるということです。たとえば、SELECT文やあとで解説するINSERT文、UPDATE文、DELETE文などは前者に該当します。これらの文は、それぞれ異なるデータベース操作を行うことができます。一方、WHERE句やあとで解説するFROM句などは、言葉のとおり後者の句に分類されます。これらの句は、文に付加することで、データの抽出条件や対象となる表などを指定することができます。

SELECT文の記載例

顧客表

顧客番号	顧客名	メールアドレス	年齢
1000	高橋	taxxxx@xx.com	22
2000	小林	koxxxx@xx.com	31
3000	清田	kixxxx@xx.com	25
4000	黒木	kuxxxx@xx.com	40
5000	中田	naxxxx@xx.com	18

●SELECT文による検索

例
```
SELECT 顧客名,メールアドレス FROM 顧客表;
```

顧客名	メールアドレス
高橋	taxxxx@xx.com
小林	koxxxx@xx.com
清田	kixxxx@xx.com
黒木	kuxxxx@xx.com
中田	naxxxx@xx.com

●条件指定による検索

例
```
SELECT 顧客名,年齢 FROM 顧客表 WHERE 年齢 >= 30;
```

顧客名	年齢
小林	31
黒木	40

●検索結果を並べ替える

例
```
SELECT 顧客名,メールアドレス FROM 顧客表 ORDER BY 顧客名 ASC;
```

顧客名	メールアドレス
清田	kixxxx@xx.com
黒木	kuxxxx@xx.com
小林	koxxxx@xx.com
高橋	taxxxx@xx.com
中田	naxxxx@xx.com

●名前の一部に‘田’の文字を含む顧客データを指定する

例
```
SELECT 顧客名,年齢 FROM 顧客表 WHERE 顧客名 LIKE'%田%';
```

顧客名	年齢
清田	25
中田	18

●顧客の総数を数える

例
```
SELECT COUNT(*) FROM 顧客表;
```

COUNT(*)
5

関連用語 LIKE（P.50）、ORDER BY句（P.54）、WHERE句（P.40）、集計関数（P.116）

02

データの取得／フィルタリング／整形

SELECT文

SELECT文はSQLクエリの基本であり、データベースから情報を取得する際に頻繁に使用されます。表、列、条件を正確に指定することで、必要なデータを取得し、データベース操作を行う基盤となります。

本書で紹介する書式と例

本書ではOracleデータベースを想定してSQL文の書式および例を構成しています。多くのDBMSで共通していますが、各DBMSは特有の拡張や機能を持っており、そのため一部の書式および例で違いが生じる場合もあります。

☑ SELECT文の役割

SELECT文の主な役割は、データベースからデータを抽出することです。これにより、以下の操作が可能になります

①**データの取得**：表内のデータを取得し、その実行結果をセットに格納します（結果セット）。これにより、**データを閲覧したり、後続の操作を行ったりする**ことができます。

②**データのフィルタリング**：WHERE句を使用して、**特定の条件を満たす行だけを選択**できます。たとえば、特定の日付範囲内の注文、特定のカテゴリの商品などを取得できます。

③**データの整形**：SELECT文は、計算や文字列の結合など、**データの変換や整形を行う**際にも使用できます。これにより、必要な形式でデータを取得できます。

☑ SELECT文の基本構造

SELECT文の基本構造は、SELECT文、FROM句、WHERE句（オプション）の3つで構成され、それぞれ以下のような役割を持っています。

①**SELECT文**：**取得する列を指定**します。列名はカンマで区切って列挙します。または、「＊」（アスタリスク）を使用してすべての列を選択できます。

②**FROM句**：**データを取得する表を指定**します。1つ以上の表を指定することができます。

③**WHERE句（オプション）**：**取得条件を指定**します。条件式が真と評価される行だけが結果に含まれます。条件を指定しない場合、表内のすべての行が返されます。

» SELECT文の役割

●データベースからデータを取り出す

書式

```
SELECT 列名, 列名, ……
FROM 表名
WHERE 条件;
```

文を見やすくするために改行を入れることができる

文の最後には；（セミコロン）を入れる

●主に以下の3つの演算を組み合わせた操作を行う

選択……行の抽出

射影……指定した列の抽出

結合……複数の表を結合して1つの表にする

» SELECT文の例

以下は、社員表から従業員の名前と給与を取得するシンプルなSELECT文の例です。

例

```
SELECT 名前,給与
FROM 社員表;
```

さらに、条件を追加して、給与が500,000円以上の従業員のみを取得するクエリは、以下のようになります。

例

```
SELECT 名前,給与
FROM 社員表
WHERE 給与 >= 500000;
```

関連用語 SQLの各句が実行される順番（P.60）

01

データの挿入

INSERT文

SQLのINSERT文は、新しいデータをデータベースに追加するためのクエリです。このクエリを使用することで指定した表にデータを挿入することができます。データは列ごとに指定し、新しい行が作成されます。

☑ データベースにおけるデータの追加

新しい行を表に追加することで、**データベースを更新**することができます。指定した表内の特定の列にデータを挿入することができ、これにより、データベース内の情報を正しく制御したり、カスタマイズしたりできます。

データベース表に新しい行を挿入するには、INSERT文を利用します。SQLの基本的な操作の1つであり、INSERT文であるクエリのことをINSERT命令と呼ぶこともあります。

INSERT文は単一の行だけでなく、**複数の行を一度に挿入することも可能**で、これは効率的なデータの追加に役立ちます。

☑ 値の挿入方法と挿入できる値

INSERT文は、**VALUES句**を使用して**挿入するデータの値を指定**します。データはVALUES句内に列ごとに対応する値を指定し、その順序に従って表内の列に挿入されます。また、**INSERT INTO文内でSELECT文を使用**すれば、別の表からデータを抽出（コピー）し、そのデータを挿入先の表に挿入（ペースト）できます（副問い合わせ）。この方法は、**データの複製や結合**などの操作に非常に便利です。

また、**シーケンス**や**デフォルト値**といった概念を使用して、**自動的に生成された値や既定の値を挿入**できます。これは、CREATE文（第6章参照）を使った表内の**値の生成**や**初期化**に使用されます。

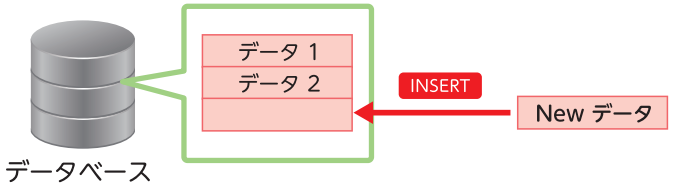
INSERT命令は非常に重要で、データベース内の情報を管理し、アプリケーションやシステムを正確かつ効果的に機能させるために欠かせないものです。

シーケンス
イベントやアクションは時系列でつながり、前後の関係があります。シーケンスとは、そうした物事が順番に起こるしくみの概念のことで、たとえば、料理の手順や物語の出来事もシーケンスといえます。

デフォルト
デフォルトは、標準設定や初期状態を指し、ユーザーが特別な設定をしない場合の自動的な選択や状態を意味します。右ページの図では、表作成時の価格のところで「DEFUALT 100」を指定しています。

INSERT文の役割

●データベースにデータを追加する



書式
INSERT INTO 表名(列名1, 列名2, 列名3, …) VALUES(値1, 値2, 値3, …);
例
INSERT INTO 食品表(ID, 商品, 価格) VALUES(0001, 豚肉, 500),(0002, 玉ねぎ, 150);

食品表

ID	商品	価格	
0001	豚肉	500	← データ1
0002	玉ねぎ	150	← データ2
			← 列名（必ず行の先頭）
↑ 値1	↑ 値2	↑ 値3	

CREATE TABLE文を使ったデータの自動生成

書式
CREATE TABLE 表名(列名 データ型 DEFAULT デフォルト値, …);
例
CREATE TABLE 食品表(ID Int, 商品 Text, 価格 Int DEFAULT 100);
INSERT INTO 食品表(ID, 商品) VALUES(0003, サーモン);

デフォルト値を100に設定

※IntやTextはデータベースにおけるデータ型の1つ。Intは小数点を含まない整数値（integer type）を扱い、Textは文字列を扱う。

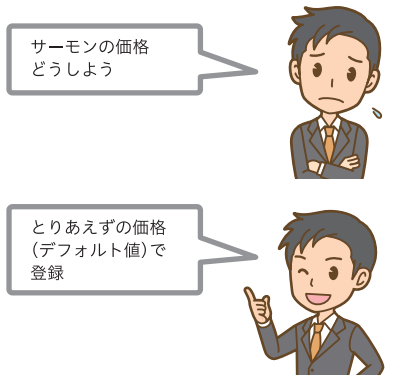
追加データ

0003	サーモン	?

食品表

ID	商品	価格
0001	豚肉	500
0002	玉ねぎ	150
0003	サーモン	100

デフォルト値



関連用語 副問い合わせ（P.108）、CREATE TABLE文（P.146）

Chapter4

07

サブクエリ

副問い合わせ

SQLの**副問い合わせ**とは、**メインクエリの処理中に、別のクエリ（サブクエリ）を実行する機能**です。SQLクエリの中に別のクエリを埋め込むことで、データをより柔軟かつ効果的に抽出することができます。

副問い合わせとは
副問い合わせとは、入れ子SELECT文（サブクエリ）のことで、外側のSELECT文の条件や結果に基づいて内側のSELECT文が実行され、結果が外側のSELECT文に組み込まれます。

☑ 条件指定がシンプルになり可読性が向上する

副問い合わせは、通常、メインクエリに対する単一の条件として利用されます。結果、**結合処理と同じデータセットを得る**ことができます。クエリを１文で組み立てる場合と異なり、副問い合わせではクエリを**メイン**と**サブ（副問い合わせ）**とに**分割して組み立てる**ため、メインクエリの条件が単純になる傾向が高くなります。この単純な条件指定は理解しやすく、SQLの可読性を向上させます。

☑ メインクエリの処理を効率化

たとえば「顧客情報」表と「注文情報」表を結合して、ある日に注文した顧客の情報を取得するとします。結合処理では両方の表を結合する処理と、指定された日付の注文日である注文情報を検索する処理が同時に同じタイミングで実行されます。

一方、副問い合わせを使用すると、まず注文日が指定された日付の注文情報を取得する副問い合わせが実行され、次にその結果を基に、「顧客情報」表から顧客情報を取得します。このとき、副問い合わせの結果はメモリ内にキャッシュされ再利用されるため、**メインクエリの処理時間を短縮**することができます。

このように副問い合わせを使用すると、サブクエリの結果を基にメインクエリの条件を動的に設定できます。これにより、**特定の条件に基づいたデータセットを取得する柔軟性が向上**します。そのため副問い合わせは、データベースクエリの柔軟性と効率性を向上させ、特定の条件を持つデータを的確に取得する手段として非常に有用です。

副問い合わせの例

書式
```
SELECT 列名 FROM 表名 WHERE 列名=(サブクエリ);
```
例
```
SELECT * FROM 従業員表
WHERE 部署ID= (SELECT ID FROM 部署表 WHERE 部署名 = '経理部');
```

従業員表

ID	氏名	部署ID
0001	山本	1111
0002	田中	2222
0003	中村	3333

部署表

ID	部署名
1111	総務部
2222	営業部
3333	経理部

副問合せで抜き出したデータ

ID	氏名	部署ID
0003	中村	3333

メインクエリの処理を効率化

例
```
SELECT * FROM 顧客情報表
WHERE ID= (SELECT 顧客ID FROM 注文情報表 WHERE 注文日 = '2024/1/1');
```

顧客情報表

ID	顧客名	パスワード
0001	山本	1111
0002	田中	2222
0003	中村	3333

注文情報表

注文ID	顧客ID	注文日
10001	0001	2024/1/1
10002	0002	2024/1/2
10003	0003	2024/1/3

サブクエリがまず実行される
(SELECT 顧客ID FROM 注文情報表 WHERE 注文日= '2024/1/1')

顧客情報表

ID	顧客名	パスワード
0001	山本	1111
0002	田中	2222
0003	中村	3333

注文情報表

顧客ID
0001

サブクエリの結果を基にメインクエリが実行される
（SELECT * FROM 顧客情報表 WHERE ID= 0001）

ID	顧客名	パスワード
0001	山本	1111

関連用語 相関副問い合わせ（P.104）、非結合副問い合わせ／結合副問い合わせ（P.102）、副問い合わせを含むDELTE文（P.112）、副問い合わせを含むUPDATE文（P.110）

Chapter5

01

単一行関数と集計関数

関数の種類

SQLには単一行関数と集計関数の2つの主要な関数カテゴリがあります。単一行関数は個々のデータの操作に、集計関数はデータの全体的な概要を得るために利用されます。

関数
関数は、プログラムで使われる「便利な箱」です。この箱には特定の処理が詰め込まれており、必要なときに呼び出して使います。たとえば、料理のレシピ本のレシピが関数だと考えると理解しやすいでしょう。レシピを活用すれば、同じ処理（料理の作り方）を繰り返し簡単に行うことができます。

☑ 単一行関数とは

単一行関数は**各行に対して個別に適用**され、一般的に**データの変換**や**操作**に使用されます。たとえば、**大文字や小文字の区別なく検索を行いたい場合**、文字列を一律に変換する単一行関数を使用します。また、データベース内の日付や時刻の表現が異なる場合、単一行関数を使用して、**異なる形式の日付を統一**し、データの一貫性を維持することができます。

さらに、**文字列の長さを知りたい場合**や、**複数の文字列を組み合わせて新しい情報を作成したい場合**も単一行関数が不可欠です。単一行関数はデータベース内の情報をより使いやすく整え、必要な情報を素早く取得するために不可欠なツールといえます。

☑ 集計関数とは

集計関数は、**複数の行から集計データを計算**するために使用されます。主にグループ化や合計の算出に役立ちます。データの合計や平均、最大値や最小値を計算することで、全体の傾向や特徴を理解しやすくします。集計関数を利用することで、**膨大な売上データの売上合計を算出**したり、**製品価格の平均値**を求めたりすることができます。また特定の条件を満たす顧客数や注文数を数えることで、ビジネスの成長傾向や需要の変化を迅速に把握できます。

これらの集計関数は、大量のデータから重要な統計情報を抽出するため、ビジネス上の意思決定や効果的なデータ分析に欠かせないものです。データを全体的に理解し、パターンやトレンドを発見する際には、これらの集計関数が強力なツールとなります。

単一行関数の利用例

例
```
SELECT LOWER (ソフトウェア) FROM インストール表;
```

インストール表

ID	ソフトウェア	インストール日
0001	Microsoft	2023/12/1
0002	Python	2024/1/2
0003	Google	2024/2/10

文字列を小文字に変更

ID	ソフトウェア	インストール日
0001	microsoft	2023/12/1
0002	python	2024/1/2
0003	google	2024/2/10

集計関数の利用例

例
```
SELECT SUM (売上),AVG(売上) FROM 注文表;
SELECT MAX (売上),MIN(売上) FROM 注文表;
```

2/15	500万
4/10	110万
5/28	500万
7/31	1000万
8/30	800万

❶売上データの合計値、平均値を算出

❷データの最大値、最小値の抽出

例
```
SELECT COUNT (＊) FROM 顧客表 WHERE 性別 = '男性';
```

氏名	性別
田中博美	女性
吉田ひろみ	女性
野口宏美	女性
牧浩美	男性

男性の数を数える

関連用語 数値関数（P.120）、日付関数（P.122）、変換関数（P.124）、文字関数（P.118）

01

代表的なデータベースオブジェクト

データベースオブジェクトの種類

データベースオブジェクトとは、データベースに格納されるデータの単位です。RDBにはいくつかのオブジェクトがあり、このオブジェクトの適切な設計と運用がデータベースの信頼性と効率性を確保します。

☑ 代表的なデータベースオブジェクト

ここではいくつかの代表的なデータベースオブジェクトを紹介します。

①**表（Table）**：表はデータの物理的な格納場所であり、情報を構造的に保持します。各表は**特定のテーマやエンティティ（例：顧客、注文、商品）に対応**し、行はそれぞれの**データレコード**を表し、列は**データの属性**を定義します。

②**インデックス（Index）**：インデックスはデータベースの**検索性能を向上**させるために利用されます。特定の列に対して作成され、データの位置を素早く特定することができます。

③**ビュー（View）**：ビューは**仮想的な表**であり、特定のクエリによって動的に生成された結果を表に見たてたものです。複雑なデータ構造を単純に表示するために利用され、**データの隠蔽**や**簡素化**に役立ちます。

④**トリガー（Trigger）**：トリガーはデータベース内で特定のイベントが発生した場合に**自動的に実行されるプログラム**です。これにより、データの変更に対して自動的な制御や処理が行われ、データベースの一貫性が維持されます。

⑤**ストアドプロシージャ（Stored Procedure）**：ストアドプロシージャはデータベース内に保存された**一連のSQL文の集まり**であり、アプリケーションから呼び出されることがあります。SQL文を再利用可能にし、処理の効率性を向上させます。

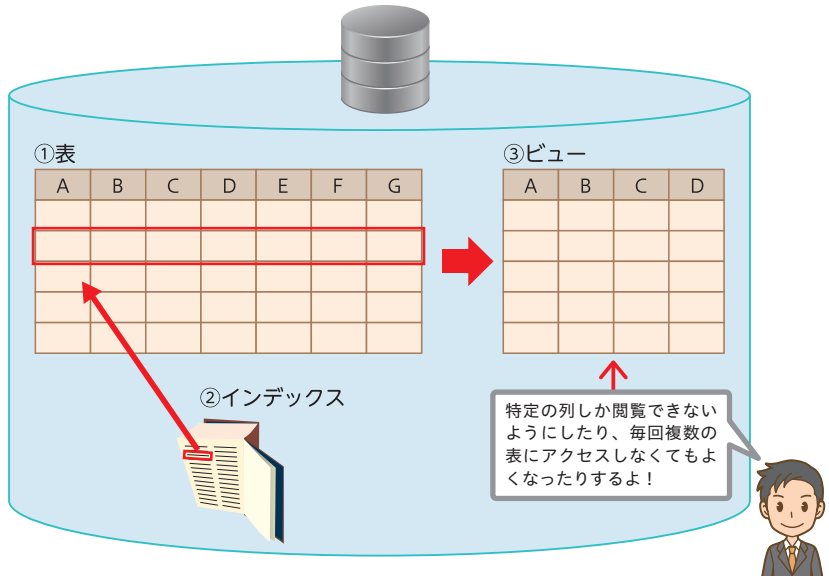
オブジェクト
データベースのオブジェクトは、データを整理して保管するための道具や部品です。表やクエリなどが該当します。これらを使うことで、情報を簡単に見つけたり、整理したりすることができます。

インデックス
インデックスは、データベース内の情報を高速に見つけるための手がかりです。辞書の目次のように、重要なデータの位置を示し、検索を迅速化します。

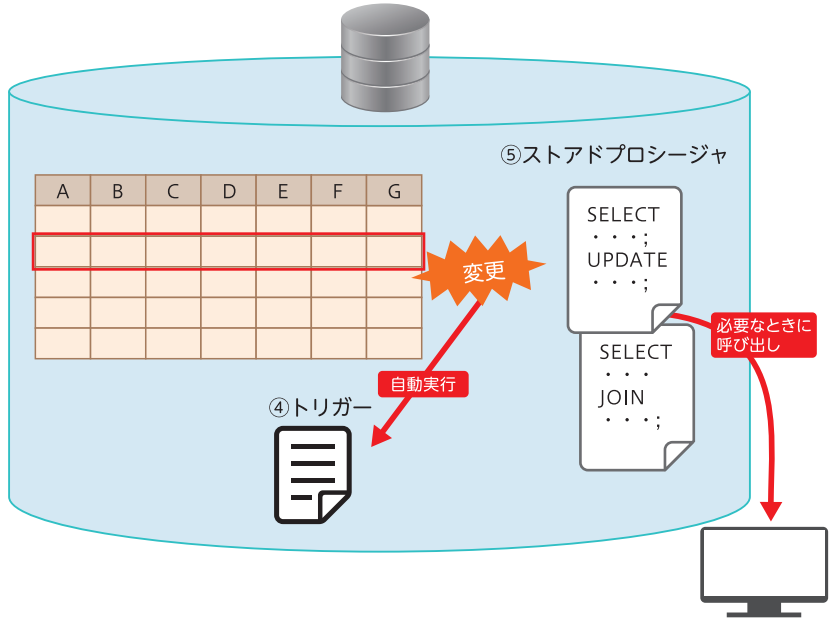
トリガー
トリガーとは引き金という意味で、何かしらの動作や状況が特定の行動や反応を引き起こすものです。特定の条件が満たされるとある行動が自動的に起こるしくみです。

ストアドプロシージャ
ストアドプロシージャは、データベースに保存されており、必要なときに呼び出してまとまった特定の処理を行います。

表・インデックス・ビューのイメージ



トリガー・ストアドプロシージャのイメージ



関連用語 インデックス（P.156、158）、ビュー（P.106、160、162）、表（P.18）

Chapter7

01

ユーザーアカウントの作成と削除

CREATE USER文／DROP USER文

ユーザーアカウントは、セキュリティの観点から極めて重要です。適切な管理と対策を講じることで、データへの不正アクセスや悪意ある操作から守られ、データベース全体の信頼性とセキュリティが確保されます。

☑ ユーザーアカウントの必要性

ユーザーアカウント
ユーザーアカウントには、ユーザー名、パスワード、環境設定、使用権限などが含まれます。 コンピューターやネットワーク上のサービスにおいて、個々の利用者がどのような機能を利用できるかといった使用権限の設定内容のことです。

ユーザーアカウントは、データベースにアクセスするための**特定の識別子と権限のセット**です。これにより、データベースの利用者を識別し、利用者に対して異なる操作権限を設定できます。

セキュリティの観点から、すべてのアクセスを匿名でなく特定のユーザーに関連付けることが重要です。ユーザーアカウントには、**操作権限（読み取り、書き込み、更新など）**を適切に設定することがポイントとなります。必要な操作にのみ権限を与え、不必要な操作を制限することで、誤ったデータ変更や機密情報の漏洩を防ぎます。

☑ ユーザーアカウントの作成と管理

ユーザーアカウントは、**管理者**によって作成され、各ユーザーに対して必要な権限が与えられます。アカウント作成時には、**強力なパスワード**の使用や**二要素認証**の導入など、アカウントのセキュリティを向上させる対策が取られます。

強力なパスワードの使用は基本であり、定期的な**パスワード変更**やパスワードの複雑さを要求する**ポリシーの設定**により、不正アクセスからの保護が強化されます。ユーザーアカウントの削除や権限変更も定期的に行うことで、不要な権限を持つアカウントを排除します。

ユーザーアカウントの操作は**監査ログ**に記録され、不審なアクティビティを検知することができます。定期的にこれらのログを確認することで、セキュリティインシデントの早期発見と対応が可能となります。

ユーザーアカウント作成／削除の書式

書式
```
CREATE USER ユーザー名;
DROP USER ユーザー名;
```

銀行口座から見たアカウント管理

Aさん口座表

取引日	入出金	残高
2024/01/01	￥5,000	￥105,000
2024/01/03	￥-3,000	￥102,000
2024/01/05	￥50,000	￥152,000

アカウント管理の鉄則

パスワードポリシー設定の基本

以下の項目を適切に設定することで、パスワードの推測や漏洩による不正アクセスリスクを大幅に低減することができる。

項目	内容
パスワードの長さ	推測困難性を意識して設定することが重要。一般的には、8文字以上、できれば12文字以上が推奨されている
パスワードの複雑性	英字（大文字／小文字）、数字、記号の組み合わせを必須にする。連続する文字やキーボード配列順の文字列の使用を禁止することも有効
パスワード変更禁止期間	短期間でのパスワード変更を制限する。これはカウントの乗っ取りを防ぐためで、一般的には、30日以上が推奨されている
パスワード有効期限	定期的なパスワード変更を強制し、パスワードの漏洩リスクを低減する。一般的には、90日～180日程度が推奨されている
パスワード履歴の記録	パスワード履歴を記録することで、過去に使用したパスワードの再利用を防ぐ。一般的には、過去12回分のパスワード履歴を記録することが推奨されている
パスワード強度チェック	パスワード強度チェックツールを導入し、設定されたパスワードの強度を評価する。強度が弱い場合は、より強固なパスワードを設定するようにユーザーを指導する
二要素認証	パスワードに加えて別の認証要素を追加することで、セキュリティを強化する。その強化方法としては、SMS認証やワンタイムパスワードなどが一般的に使われている

関連用語 権限（P.168）、権限付与（P.174）、所有者（P.170）、ロール（P.176）

Chapter8

01

データの重複排除と整合性確保

正規化とは

正規化は、「正規形」と呼ばれるルールに基づいて行われ、正規化を行うことで、表が整理され、データの冗長性や重複が減り、データベースのパフォーマンスや保守性が向上します。ここでは、正規化の重要性を解説します。

正規化
一般的に、正規化はデータベース設計において使用され、データの構造を効率的かつ一貫した形に整理し、冗長性を排除してデータの柔軟性と効率性を向上させるプロセスです。

☑ 正規形が保たれていないと起こり得る問題

正規形が保たれていないと次のような問題を引き起こす可能性があります。

①**データの冗長性**：正規形が保たれていない場合、同じ情報が複数の場所に重複して格納されることがあります。たとえば、クラスごとに生徒情報を持つ表で、クラスの情報が各生徒の行に繰り返し保存されているとします。これでは**データが冗長**になり、スペースの無駄使いとなります。

②**更新の複雑さ**：正規形が保たれていないと、**データの変更や更新が複雑**になります。たとえば、教師の連絡先情報がクラス表と生徒表の両方に存在する場合、教師の電話番号が変わると、それをすべての関連するクラスと生徒に更新する必要があります。

③**データの整合性の喪失**：正規形が保たれていないと、**データの整合性が損なわれる**可能性があります。たとえば、クラス情報と生徒情報が同じ表に混在している場合、一部の生徒のクラス情報がほかの生徒と異なることがあり、これがデータの不整合を引き起こす可能性があります。

④**検索の非効率性**：正規形が保たれていないと、必要な情報を取得するために複雑なクエリが必要になり、**データベースの検索が非効率**になることがあります。

正規化には正規形の種類に対応して3つの段階（P.184参照）があり、初めの段階である第一正規形を達成することにより、より高度な正規形への移行が可能になります。

正規化が保たれていない表の例 ①

クラスID	クラス名	教師名	教師連絡先
C01	Aクラス	齋藤	XXX
C01	Aクラス	齋藤	XXX
C02	Bクラス	山田	YYY
C02	Bクラス	山田	YYY

教師の連絡先を更新したいが、連絡先が複数の場所に保存されているため両方とも変更する必要があり、かつスペースの無駄遣いでもある

教師名	教師連絡先
齋藤	AAA

教師名	教師連絡先	生徒ID	生徒名	生徒住所
齋藤	XXX	183012	田中太郎	品川区
齋藤	XXX	183609	小林次郎	渋谷区
山田	YYY	190383	山本花子	新宿区
山田	YYY	194789	鈴木桃子	中央区

正規化が保たれていない表の例 ②

クラスID	クラス名	教師名	教師連絡先	生徒ID	生徒名	生徒住所
C01	Aクラス	齋藤	XXX	183012	田中太郎	品川区
C01	Aクラス	齋藤	XXX	183609	小林次郎	渋谷区
C02	Bクラス	山田	YYY	190383	山本花子	新宿区
C02	Cクラス	山田	YYY	194789	鈴木桃子	中央区

同じことが繰り返し書かれているため、ミスの状況次第でデータに矛盾が発生する場合がある

あれ？
CO2はBクラスでは……？
入力ミスかな？

正規化が保たれていない表の例 ③

生徒ID	生徒名	クラブ名
183012	田中太郎	サッカー部
183609	小林次郎	サッカー部
~~190383~~	~~山本花子~~	~~写真部~~

山本さんが転校したから削除するよ

あれ？
写真部の情報がなくなってる？

関連用語 第一正規化（P.186）、第二正規化（P.188）、第三正規化（P.190）、非正規化の表（P.202）

02

表の種類

1NF／2NF／3NF

RDBの表には「正規形（NF：Normal Form）」と呼ばれる3段階の種類があり、データを整理しやすくするためのルールを持っています。段階が上がると重複や冗長性排除のレベルも上がります。

正規化の目標
正規化には、正規形の種類に応じた数の正規化のステップが存在します。正規化は、1NF（第一正規形）から3NF（第三正規形）へ進化しています。1NFでは重複データの排除、2NFでは部分関数従属の排除、3NFでは推移関数従属の排除（P.190参照）が目指されました。

☑ 第一正規形（1NF）

第一正規形（1NF）は、RDBにおいて基本的な表の形式であり、データの整理と整合性を確保するための最初のステップです。**1つの列に複数の値が含まれないこと**と**各列は同じデータ型を持っていること**といった特徴が満たされている必要があります。このルールにより、データが整理されて混乱することがなくなります。

☑ 第二正規形（2NF）

第二正規形（2NF）は、表の各列が、**主キー（一意でほかの行と区別できる列）に完全に依存している状態**です。たとえば、学校のクラスごとに生徒を管理する表であれば、クラスの情報は主キー（例：クラスID）に完全に関連していることが求められます。

このルールにより、データの整理がより効果的になり、データの重複が減ります。

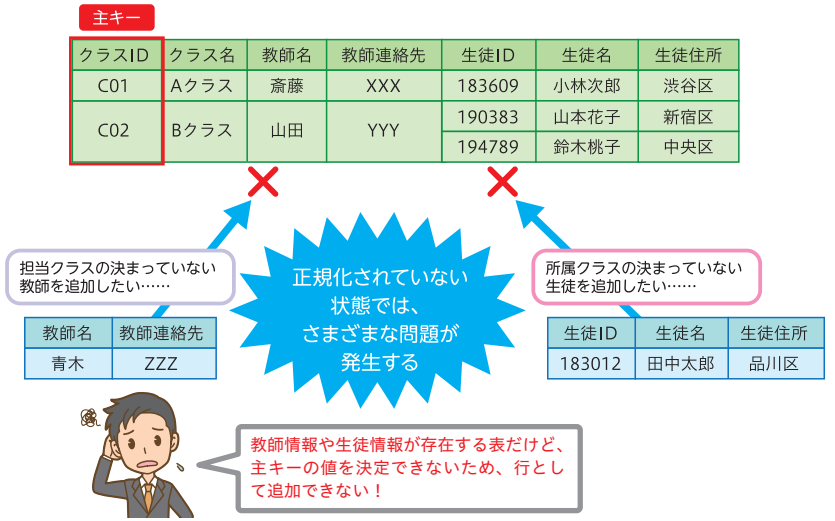
☑ 第三正規形（3NF）

第三正規形（3NF）は、**表の各列が主キーに直接関連しており、ほかの非キー列には関連していない状態**です。たとえば、生徒の住所がクラス全体に依存せず、生徒自体にのみ関連していることが求められます。

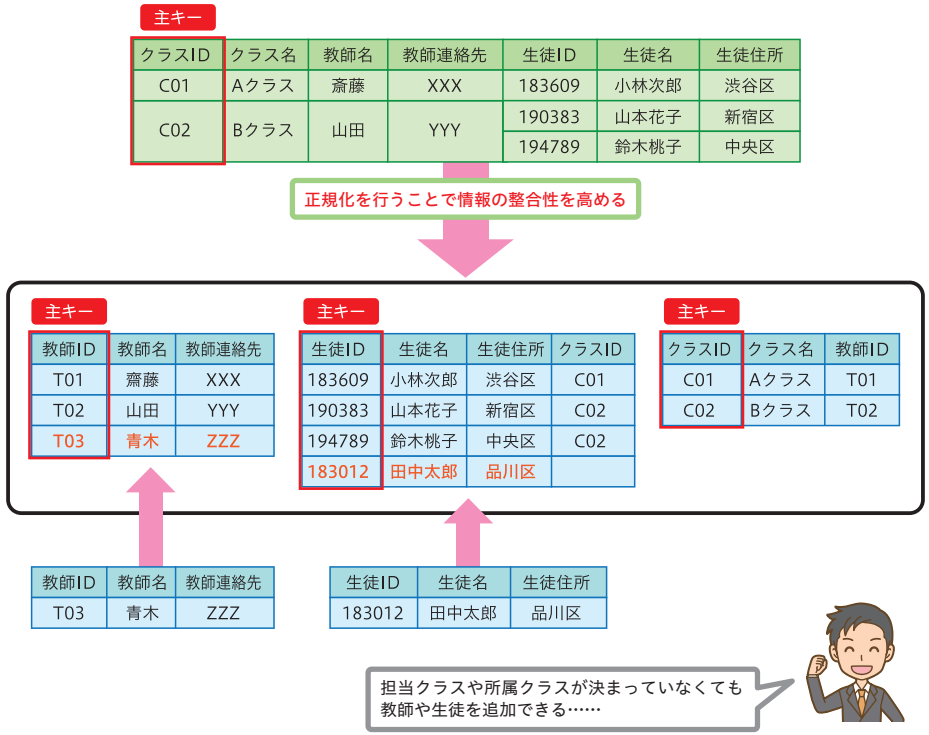
このルールにより、データが冗長になりにくくなり、データの整合性が向上します。

正規形はデータベースの設計や管理において非常に重要です。正規形に適合していると、情報の整合性が高まります。

正規化されていない場合の問題点



正規化を行うことの目的（上の図の問題を解消）



関連用語 主キー（P.18、192、194）、正規化（P.182）