

# ICT基礎 データベース

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.



■第1章 データベースの概要	4
1-1 データベースとは	5
【演習1】ディスカッション	7
1-2 データベースの用語	12
1-3 データベースの機能	15
■第2章 データベース設計	24
2-1 データベース設計とは	25
2-2 データベース設計のプロセス	28
2-3 ER図	33
【演習2】ER図の作成	41
2-4 正規化	42
【演習3】正規化	54
■第3章 SQL	55
3-1 SQLとは	56
3-2 PostgreSQLとは	61
3-3 基本的な検索	65
【演習4】基本的な検索	71
【演習5】WHERE句の利用	78
3-4 グループ関数	79
3-5 データのグループ化	81
【演習6】グループ化とグループ関数の利用	85
3-6 表の結合	86
【演習7】表の結合	92
3-7 追加・更新・削除	93
【演習8】追加・更新・削除	98
3-8 表の作成と削除	99
3-9 ビューの作成と削除	103



■ 付録	108
付-1 演習解答	109
付-2 PostgreSQLのインストール	118
付-3 サンプルのデータベース	128
付-4 参考書籍	135

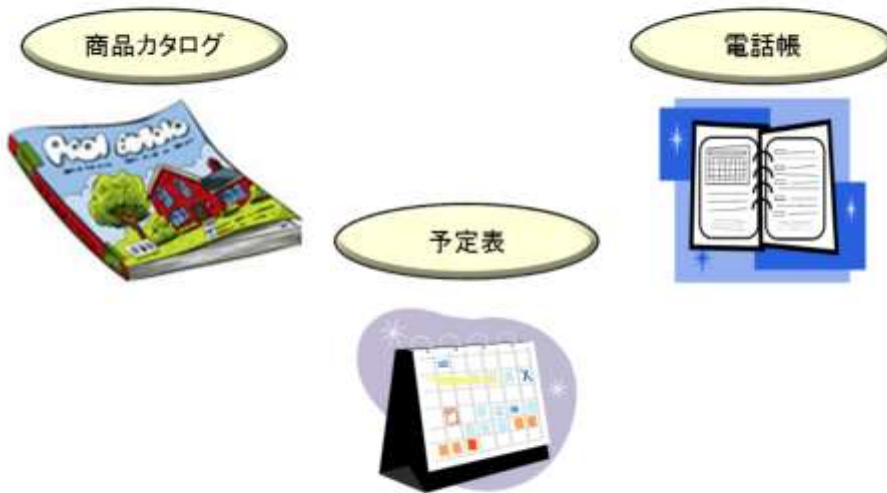
# 第1章 データベースの概要

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 1-1 データベースとは

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# データベースとは？



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データベースとは、「データ（情報）」の「ベース（基地）」のことです。すなわち、データを1カ所に集め、様々な用途に活用できるようにしたものを「データベース」と呼びます。

データが色々な場所に分散していると、いざ利用したいと思った時に探すことが困難になります。また、同じデータが複数箇所にあった場合、どのデータを信頼すればよいのか分からなくなります。

そこで、データベースでデータを一元管理することで、データの利活用を容易に行うことができるのです。



普段の生活の中で、直接もしくは間接的に利用している「データベース」には何があるか、挙げてみましょう。

# データベース管理システムとは？



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データの管理は、コンピューターが最も得意とするところです。

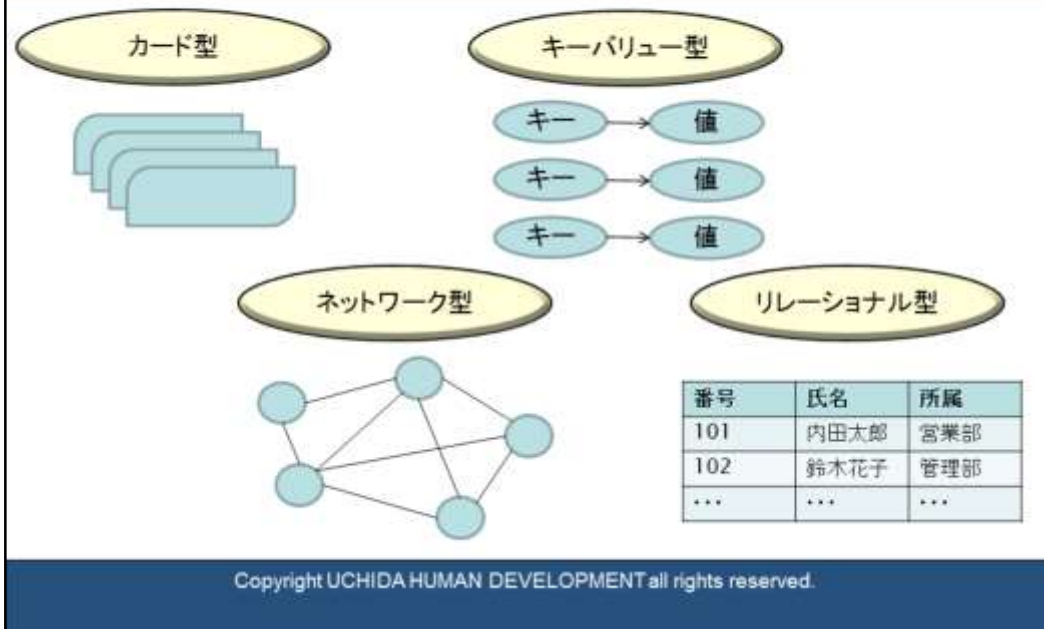
データベース管理システム（DataBase Management System : DBMS）とはその名の通り、データベースを管理するためのシステムです。

データベース管理システムは、データを一元管理し、データの検索・追加・更新・削除などを行います。また、データを加工して、合計や平均などを求めることも可能です。

データベース管理システムの特徴は、人間の手では扱いきれない大量の情報を、高速に処理できるということです。Excelのような表計算ソフトでも扱いきれないような何億件という情報を、高速に処理することができます。



# データベース管理システムの種類



## ■カード型

名刺のようなカード単位でデータを保存するデータベース管理システムです。

## ■ネットワーク型

各データがネットワーク状につながっている形式のデータベース管理システムです。

## ■キーバリュー型 (Key Value Store)

「キー」と、それに紐付く「値」をセットで保存するデータベース管理システムです。

NoSQLデータベースとも呼ばれます。

## ■リレーショナル型

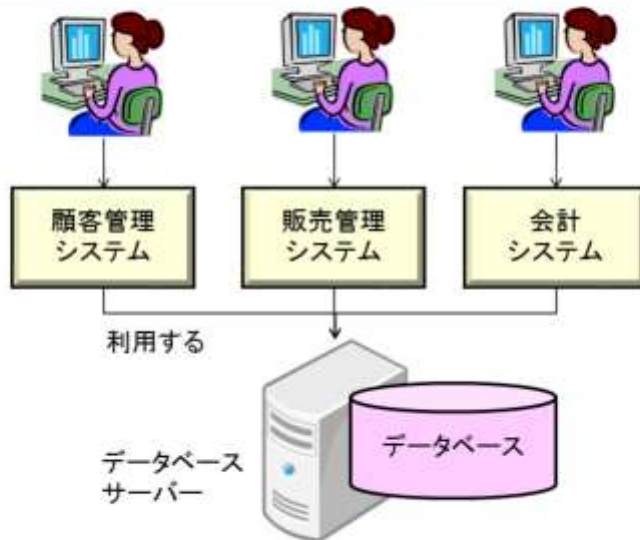
データを「テーブル」という表の形式で保存するデータベース管理システムです。

これらの中で、業務システムなどで現在最もよく利用されているのはリレーショナル型です。

非常に大量なデータを高速に処理したい場合には、キーバリュー型も使われ始めています。FacebookやTwitterでは、キーバリュー型が使われています。

以降では、単に「データベース」と記述した場合、リレーショナル型のデータベース管理システムを指すこととします。

## データベースの利用形態



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

通常、データベースは、専用のデータベースサーバー内で動作します。一般的に、大規模なデータを管理するデータベースは処理負荷が大きいため、専用のサーバーを作ります。

データベースは、顧客管理システム・販売管理システム・会計システムなど、様々なシステムで利用されるデータを一元管理し、データの加工・提供などを行います。

## 主なデータベース製品

### 有償

製品名	ベンダー
Oracle Database	Oracle
SQL Server	Microsoft
DB2	IBM
Access	Microsoft

### 無償

製品名	ベンダー
MySQL	Oracle
PostgreSQL	PostgreSQL.org

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

### ■Oracle Database

Oracle社製のデータベースで、世界シェアNo.1です。

### ■SQL Server

Microsoft社製のデータベースで、Windows Serverなど他のMicrosoft製品と組み合わせて使用することが多いです。

### ■DB2

IBM社製のデータベースで、世界シェアNo.2です。

### ■Access

Microsoft社製のデータベースで、データ量が少ない一般業務向けの簡易的な製品です。

### ■MySQL

以前はSun Microsystems社が開発していましたが、買収に伴ってOracle社に開発が移管されました。オープンソースなので、無償で利用できます。

### ■PostgreSQL

MySQLと同様に、オープンソースのデータベースです。

## 1-2 データベースの用語

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 行と列

番号	氏名	所属
101	内田太郎	営業部
102	鈴木花子	管理部
...	...	...

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

リレーショナル型データベースでは、「テーブル」という表の形式でデータを保存します。

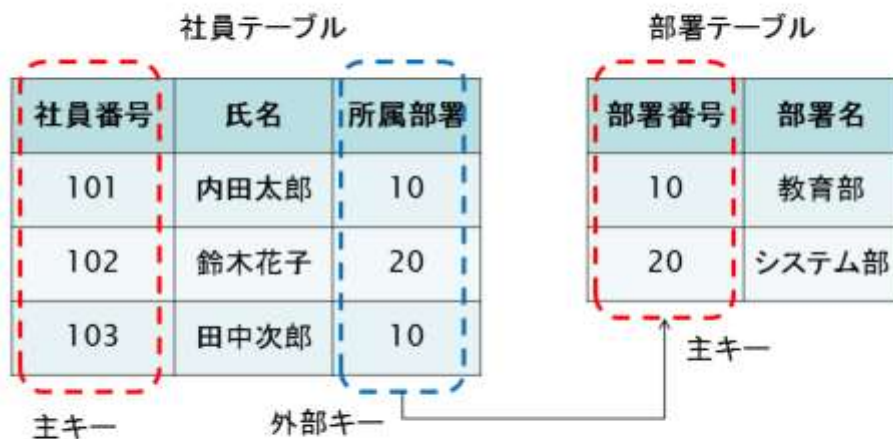
表のうち、横のことを「行」、縦のことを「列」と呼びます。

行は、「レコード」「ロウ」「タプル」とも呼ばれます。

列は、「カラム」「属性」とも呼ばれます。

行と列が交わった1つ1つのセルのことを「フィールド」と呼びます。フィールドには、1つの値しか格納することは出来ません。

# キー



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## ■主キー (Primary Key : PK)

テーブルの列のうち、行を一意に特定できる（行が1つに決まる）ものを「主キー」と言います。

図の例では、社員テーブルは社員番号、部署テーブルは部署番号が主キーです。

主キーは、1つの列でも、複数列の組み合わせでも構いません。

## ■外部キー (Foreign Key : FK)

テーブルの列のうち、他のテーブルとの関連を表す列のことを「外部キー」と言います。

図の例では、社員テーブルの所属部署が、部署テーブルとの関連を表す外部キーです。

## 1-3 データベースの機能

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 迅速なデータ操作(インデックス)



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データベースの特徴は、大量のデータを迅速に操作できることです。

そのために、データベースでは「インデックス」と呼ばれる、本の索引のような機能を持っています。

インデックスを付加すると、一般的には検索速度が向上しますが、むやみに付けばよいという訳ではありません。

検索速度が向上するように、インデックスを最適化することを「チューニング」と言います。チューニングは、データベース管理者の重要な仕事の1つです。



## データの誤入力防止(制約)

※参照整合性制約の例

社員番号	氏名	所属部署
101	内田太郎	10
102	鈴木花子	20
103	田中次郎	10
<del>104</del>	<del>佐藤恵子</del>	<del>30</del>



部署番号	部署名
10	教育部
20	システム部

部署番号が「30」の  
部署は存在しないので、  
入力できない

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

制約とは、データベースに対する誤ったデータ入力を防ぐ機能です。制約は、テーブルの各列に対して設定します。

### ■一意性制約

その列は一意でなければならないという制約です。

### ■NOT NULL制約

その列はNULL（空）であってはならないという制約です。

### ■主キー制約

主キーである列は、一意でなければならない、かつNULLであってはならないという制約です。

「主キー制約＝一意性制約＋NOT NULL制約」です。

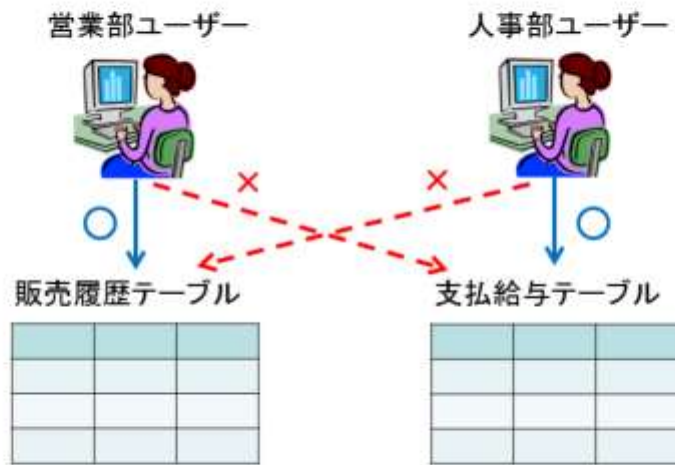
### ■検査制約

指定した条件を満たさない値は入力できないという制約です。例えば、「年齢に負の値は入力できない」などです。

### ■参照整合性制約（外部キー制約）

他のテーブルへの外部キーになっている列には、参照先に無い値は指定できないという制約です。また、外部キーに存在する値は、主キーの存在するテーブルから削除できません。

# セキュリティ管理



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

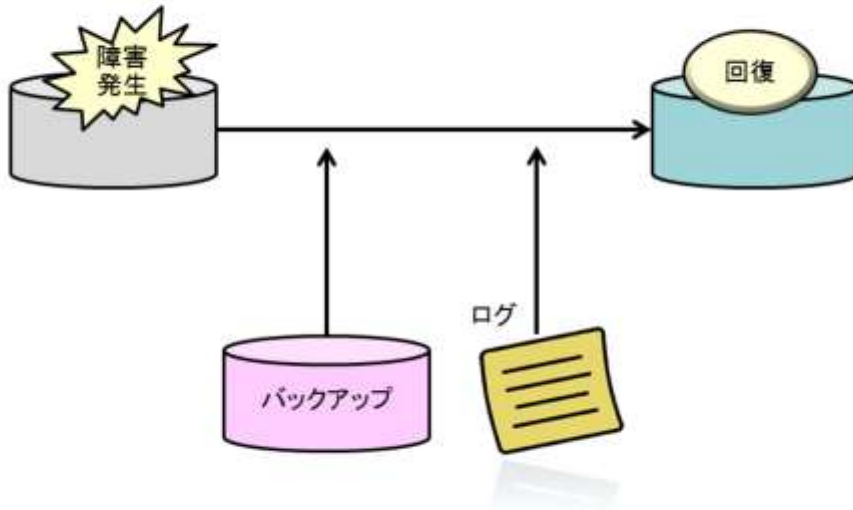
データベースには重要なデータが一元管理されているため、全ての人が全てのデータを利用できてしまうと、不都合が起こる可能性があります。

また、機密上、特定の人以外には見せたくないデータなども存在します。

そのため、データベースには「ユーザー」を管理する機能があります。

そして、ユーザーごとに権限を設定し、各ユーザーが利用できるテーブルを制限したりすることができます。

## 障害回復



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データは貴重なものです。オペレーションミスや故意の破壊、機械の故障や自然災害にも対応する必要があります。

そのためデータベースは、バックアップ（コピー）から復元できることはもちろん、バックアップした時点から障害発生時点までの履歴情報（ログ）を利用して、最新の状態まで回復することもできます。

# トランザクション

トランザクションを定義していない場合



トランザクションを定義した場合



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

トランザクションとは、実社会から見て「ひとまとまり」になる処理や操作の集まりを指します。

「ひとまとまり」の処理をトランザクションとして定義すると、トランザクション内のすべての処理は、「全てが処理される」か「全てが破棄される」のどちらかとなることをデータベースが保障します。

## ■例：口座Aから口座Bに10万円送金する

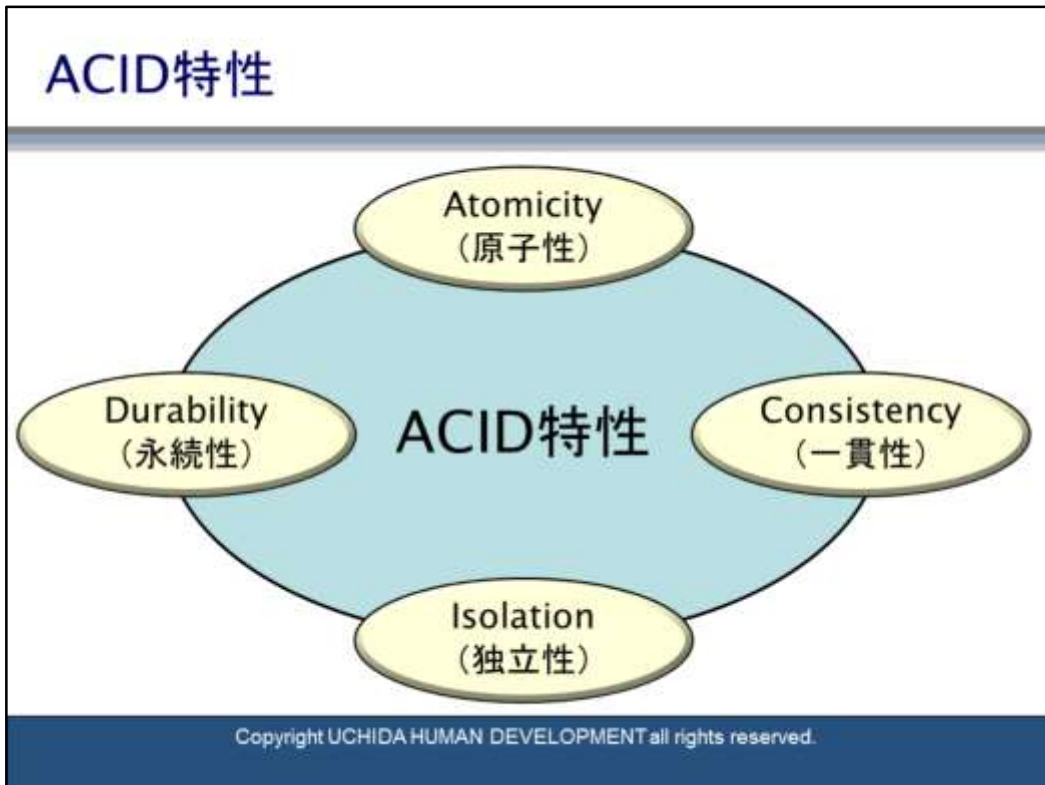
この処理は、

- ①「Aから10万円引く」
- ②「Bに10万円加える」

という2つの処理からなります。

一方の処理だけが実行されることがあってはなりません。①だけが行われ、障害により②が実行されないと、矛盾が生じます。

2つの処理を1つのトランザクションとすれば、片方だけが実行されてしまうことを防ぎます。



トランザクションは、「ACID特性」という4つの特性を持っています。

■Atomicity（原子性）

トランザクションは、全てのタスクが実行されるか、全く実行されないかのどちらかでなければなりません。

■Consistency（一貫性）

トランザクションの実行によって、データの整合性が失われてはなりません。

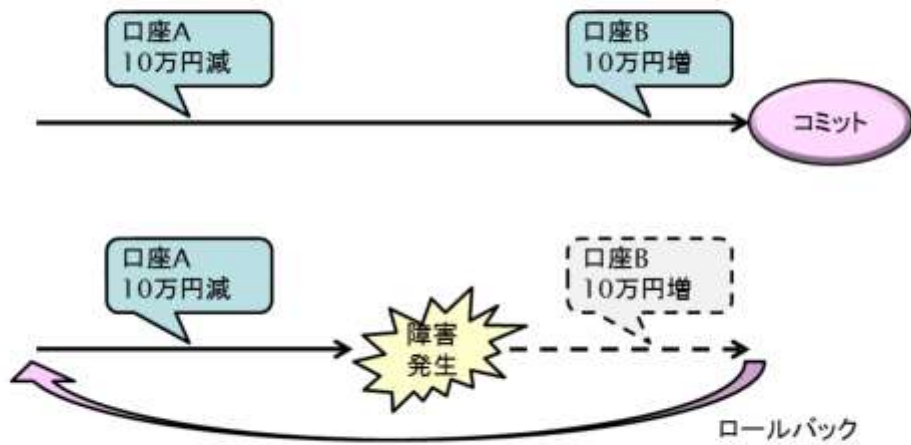
■Isolation（独立性）

トランザクション実行中の様子は、他のタスクからは見えてはいけません。

■Durability（永続性）

トランザクションの結果は、永遠に失われてはなりません。

## コミットとロールバック



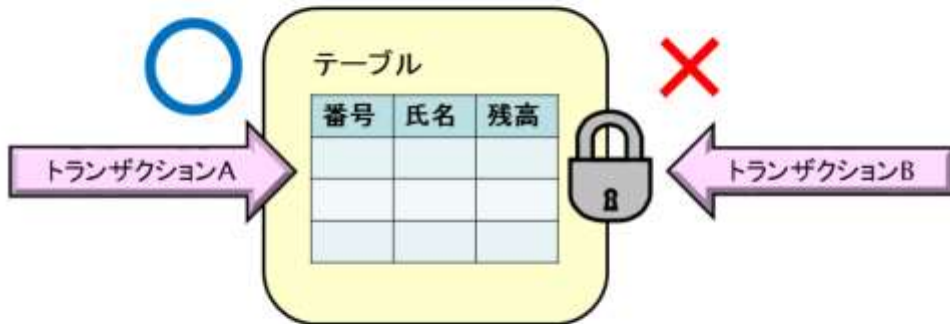
Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

トランザクションの**Atomicity**（原子性）を保つために、データベースにはコミットとロールバックという機能があります。

コミットは、トランザクションの完了を確定する操作です。コミットが無ければ、トランザクションは完了したと見なされません。

ロールバックは、トランザクションが途中で失敗した場合、実行前の状態に戻す操作です。

## ロック



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

例えば、Cさんの銀行口座へ1万円を振り込む処理は、以下の2つから成り立ちます。

- ①Cさんの現在の口座残高を確認する
- ②Cさんの口座残高を（①で確認した額+1万円）に更新する

AさんとBさんが、同時にCさんの口座に1万円を振り込んだ場合、以下のような処理の順番になると、矛盾が生じます。（最初、Cさんの口座残高は10万円とします）

- ①Aが残高を確認（10万円）
- ②Bが残高を確認（10万円）
- ③Aが残高を更新（11万円）
- ④Bが残高を更新（11万円） ←矛盾

このような現象を防ぐために、データベースには「ロック」という機能があります。

トランザクションAが処理を実行している最中は、対象のテーブルをロックし、トランザクションBからテーブルを使えなくします。

ロックには、検索はできるが更新はできない「共有ロック」と、検索すらできない「排他ロック」があります。

## 第2章 データベース設計

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.



## 2-1 データベース設計とは

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# データベース設計とは



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データベース設計とは、システムを開発する際に「どのようなテーブルが必要か」「各テーブルにはどのような属性が必要か」を考え、図にまとめることです。

データベース設計は非常に手間のかかる作業ですが、非常に重要です。

しっかりとした設計を行わなかった場合、以下のような事態に陥る可能性があります。

- ・ 検索処理の応答時間が異常にかかる
- ・ 検索処理が突然実行できなくなる
- ・ 入力したはずのデータが消失してしまう
- ・ あるデータを削除したら、それに伴って他のデータも削除されてしまう
- ・ 更新するつもりのないデータを更新してしまう
- ・ テーブルのメンテナンスをしたら他のプログラムが動作しなくなってしまう
- ・ 帳票や画面を新たに作成できない

# データベース設計の目標

- ニーズを満たす
- 信頼性の高い正確なデータを格納する
- データの独立性が高く拡張性に優れている

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## ■ニーズを満たす

蓄積したデータも、ビジネス活動の経営判断の材料として用いなければ意味がありません。また、正しくデータベースの設計を行う過程で、業務上の問題点が浮き彫りになることもあります。これによって、ユーザーのニーズをより正確に掘り下げることができるようになります。

## ■信頼性の高い正確なデータを格納する

データは正確かつ信頼性が保障されていなければなりません。たとえば、人事台帳で退社年月日が入社年月日より早いことは絶対にあってはなりません。このような実世界での「常識」をデータベースにルールとして登録しておくことで、論理的に正しくないデータが格納されることを防げます。

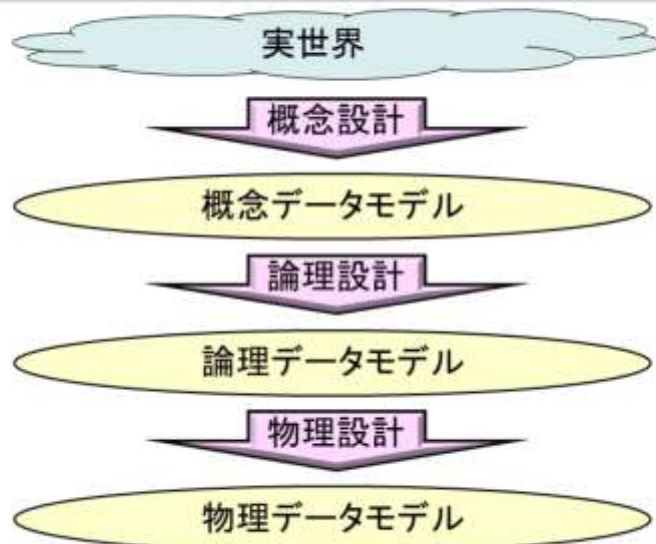
## ■データの独立性が高く拡張性に優れている

設計段階で、システム化の対象範囲全体を俯瞰しており、一貫した方針のもとで構築すると、後から修正が必要になったときにも適切な対応をとることができ、メンテナンスを容易にすることができます。1つの変更が発生したときに、それに該当する部分すべてを修正するようなシステムでは困ります。上手く設計を行うと、データ構造が変化してもほとんどプログラムを変更せずに済みます。

## 2-2 データベース設計のプロセス

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# 設計のプロセス



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

実世界をデータベース化していく設計過程は、「概念設計」「論理設計」「物理設計」と呼ばれる設計段階からなっています。

大まかに言えば、それぞれのデータモデルは

概念データモデル：どんなテーブルがあるか

論理データモデル：どんな属性があるか

物理データモデル：属性などにどのような設定をするかを表しています。

# 概念設計



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データベースの概念設計は、データベースシステムの設計過程において最初に行う作業となります。実世界を概念データモデルとしてモデル化していくことが、主な作業になります。

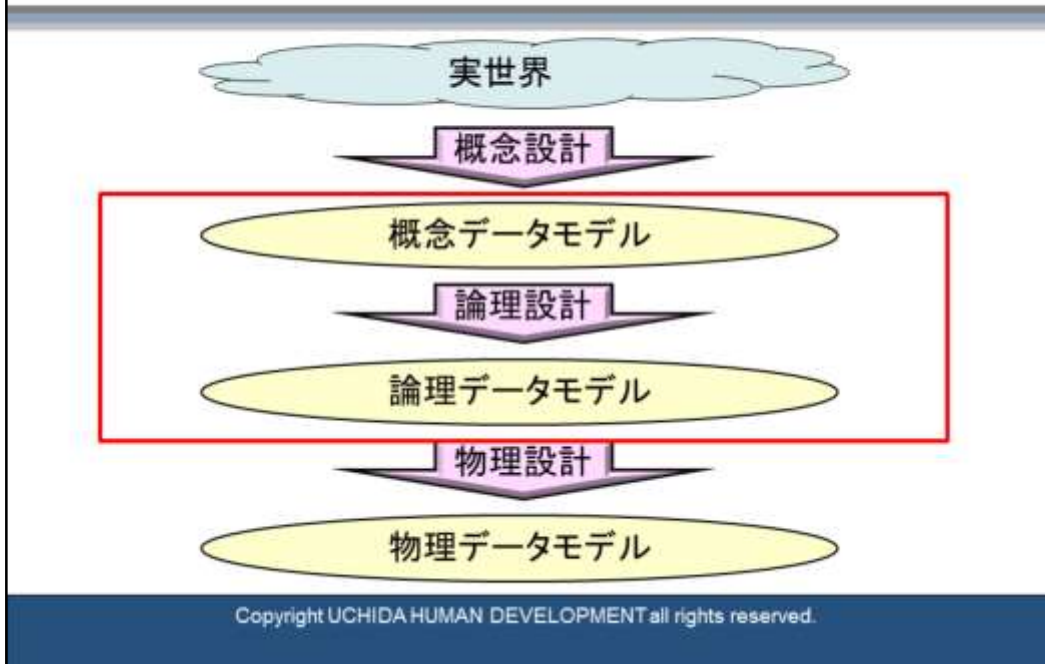
概念データモデルで表現されるのは、企業をとりまく実世界がモデル化されたものです。しかし、すべてのデータを構築の対象とするのではなく、企業戦略の観点から、必要なデータのみをデータベース化の対象範囲とします。

対象範囲や、業務運営の過程で作成・参照されるデータを把握し、業務の流れと関係を定義し明確にすることを目的とします。

概念設計のプロセスは以下の通りです。

- ①対象となる業務及び機能を分析し、システム化の対象領域を明確に定義する
- ②業務の管理対象となるエンティティ（実体）を抽出する
- ③管理対象と管理対象間のリレーション（関係）を把握する
- ④主なデータ項目やキーを定義する
- ⑤概念データモデルをER図により表現する（後述）

# 論理設計



論理設計とは、よりデータベースを意識して詳細化し、論理データモデルとしてモデル化することです。概念設計で作成したER図も詳細化され、実装を意識した情報を揃えます。つまり、各エンティティのすべての属性（データ項目）を定義します。

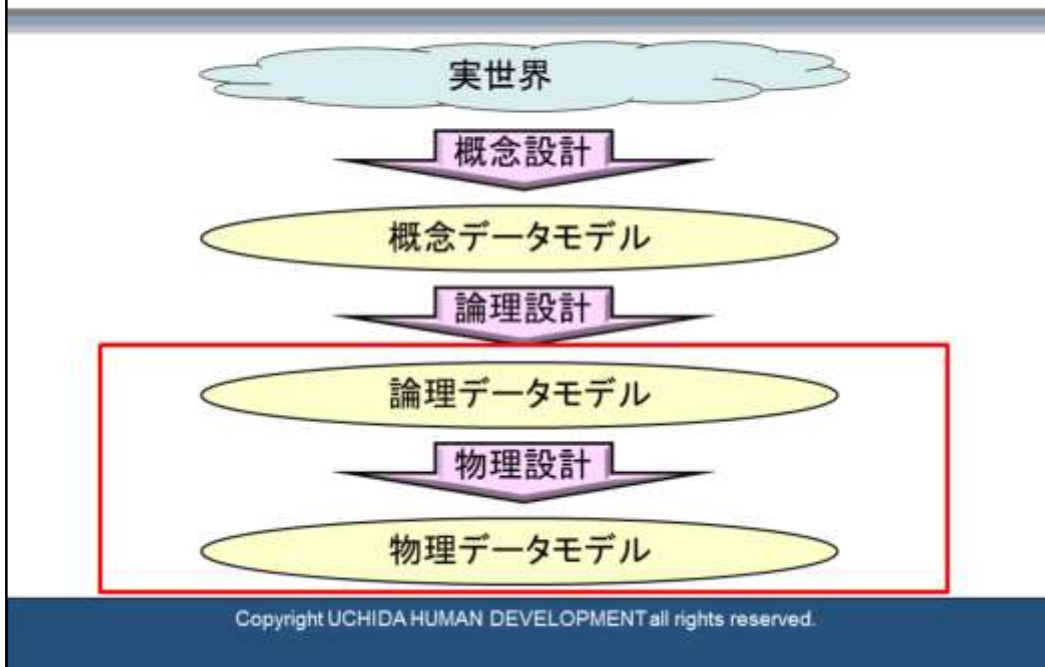
また、アプリケーションから見て必要な情報を取り出す際に用いる「ビュー」も論理設計で検討します。

設計プロセスのアプローチには、概念設計を基に行う「トップダウン」と、現状で利用されている帳票や画面を基に行う「ボトムアップ」の2種類があります。

論理設計のプロセスは以下の通りです。

- ①属性を洗い出す
- ②正規化を行う（後述）
- ③論理データモデルを作成する

# 物理設計



物理設計は、データベースへ実装するための設計です。

従って、利用するデータベース製品に応じた設計になります。実際のコンピュータ資源上にそのデータ構造を実現していく作業であり、データベース設計の最終段階です。必要に応じて、実際のデータベースの使用条件を考えて、データ構造の見直しを図ります。また、物理設計はデータベース製品に依存するため、物理設計を行う前にデータベース製品を決定する必要があります。

物理設計のプロセスは以下の通りです。

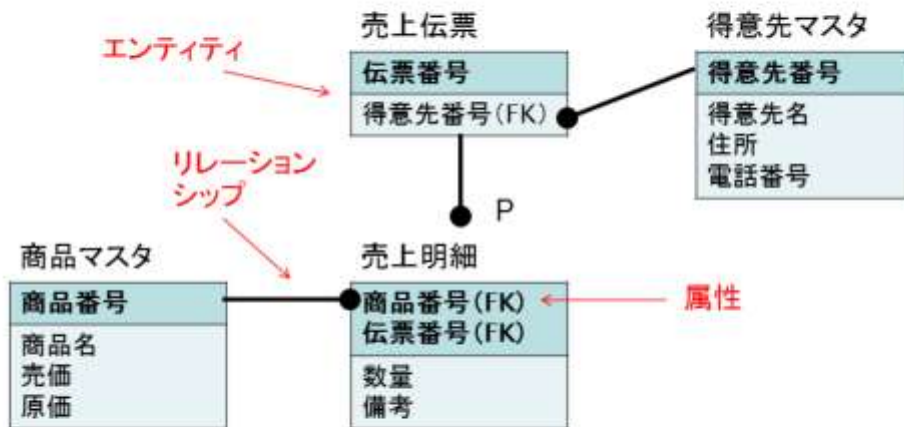
- ①テーブル名・列名を決定する
- ②データ型・データ長を決定する
- ③エンティティを統合する（非正規化）
- ④データ容量を見積る
- ⑤インデックスを検討する
- ⑥ディスク装置へ配置する



## 2-3 ER図

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## ER図とは



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ER図 (Entity Relationship Diagram) とは、対象業務の分析を行い、モデル化した結果を表現するものです。

ER図は、以下の要素から構成されています。

### ■エンティティ (Entity)

システム化対象範囲で、管理対象としたいものを示します。「実体」とも表記します。データベースのテーブルに相当します。

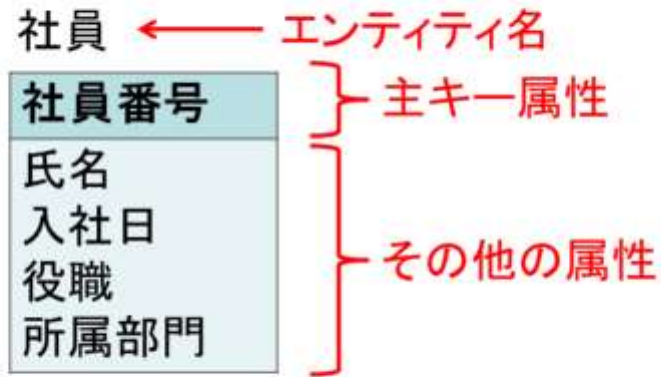
### ■リレーションシップ (Relationship)

エンティティ同士の関連を示します。「関連」「関係」とも表記します。主キーと外部キーの関連に相当します。

### ■属性 (Attribute)

エンティティを特徴づける要素を示します。各テーブルの列に相当します。

# エンティティ



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## ■エンティティ

エンティティは「実体」という日本語で表現されます。有形・無形を問わずシステムで「管理対象」としてとらえたいものを指します。最終的には、テーブルとほぼ同じ単位になることが多いです。

## ■インスタンス

エンティティを集合としてとらえた場合、集合を構成するひとつひとつを「インスタンス」と呼びます。例えば、「社員」エンティティにおいて、山田さん・佐藤さん・・・はインスタンスとなります（行／レコードに相当します）。

## ■属性

エンティティは、属性によって説明されます。「社員」エンティティは社員番号、氏名、入社日、役職、所属部門などの属性から構成されます（列に相当します）。

## エンティティの選定



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

### ■エンティティの選定

一般的にエンティティは以下のように決定します。

- ・ユーザーヒアリング等により分析した対象業務からの「名詞」や「イベント」を抽出します。
- ・その中から、システム化の要件の観点からデータとして蓄積すべきであり、さらに、「管理対象」となりうるものを抽出します。

### ■主キーの選定

エンティティを選定すると同時に、エンティティを構成する属性も自ずとはっきりしてきます。属性の中から主キーとなる属性を選定します。

主キーを選び出すガイドラインは次の通りです。

- ・インスタンス（行）を一意に識別できる
- ・値が変更されることが少ない
- ・長さが短い
- ・必ず値が設定される

# リレーションシップ



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

リレーションシップとは、エンティティとエンティティの関係や結びつきを表現したものです。

リレーションシップを決定するにあたって、個々のリレーションシップのカーディナリティ（**Cardinality**：多重性）とコンディション（**Condition**：条件性）を明確にしておく必要があります。

## ■カーディナリティ

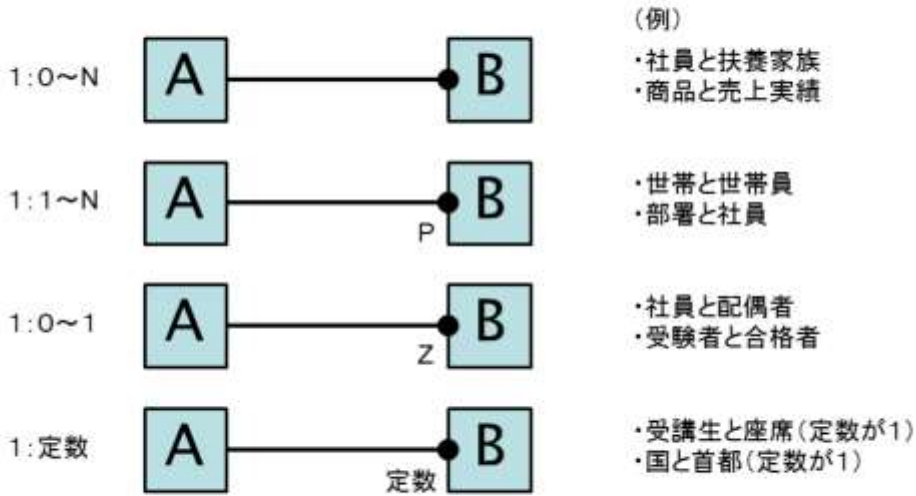
いわゆる「何：何の関係」をあらわします。例えば、「社員」と「配偶者」は1：1、「世帯」と「世帯員」は1：Nです。

## ■コンディション

リレーションシップで結ばれたエンティティの一方に対して、もう一方が任意であるか必須であるかを示します。例えば、「社員」に対して「配偶者」は任意、「世帯」に対して「世帯員」は必須です。

カーディナリティとコンディションは、システム化範囲のビジネスルールが表現されたものになります。正確に反映することが必要です。

## リレーションシップの表記



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

リレーションシップは、上記のように記述します。

### ■ 1 : 0 ~ N

例えば、「社員と扶養家族」の関係の場合、1人の社員に対して、扶養家族は0人の場合、1人の場合、2人の場合、・・・と何人でも考えられます。この状態のことを「1 : 0 ~ N」と表現します。この場合、リレーションシップの線の「0 ~ N」の側に「●」を付けます。

### ■ 1 : 1 ~ N

例えば、「世帯と世帯員」の関係の場合、1つの世帯に対して、必ず1人以上の世帯員が必要です。この状態のことを「1 : 1 ~ N」と表現します。この場合、リレーションシップの線の「1 ~ N」の側に「●」と「P」を付けます。

### ■ 1 : 0 ~ 1

例えば、「社員と配偶者」の関係の場合、1人の社員に対して、配偶者は0人か1人かのどちらかです。この状態のことを「1 : 0 ~ 1」と表現します。この場合、リレーションシップの線の「0 ~ 1」の側に「●」と「Z」を付けます。

### ■ 1 : 定数

例えば、「受講生と座席」の関係の場合、1人の受講者に対して、必ず座席は1つです。この状態のことを「1 : 1 (定数)」と表現します。定数は、場合によって値が変化します。この場合、リレーションシップの線の「1 (定数)」の側に「●」と「1 (定数)」を付けます。

## 外部キーの原則

部署

部署番号

部署名

社員

社員番号

氏名

部署番号(FK)

P

(テーブルの例)

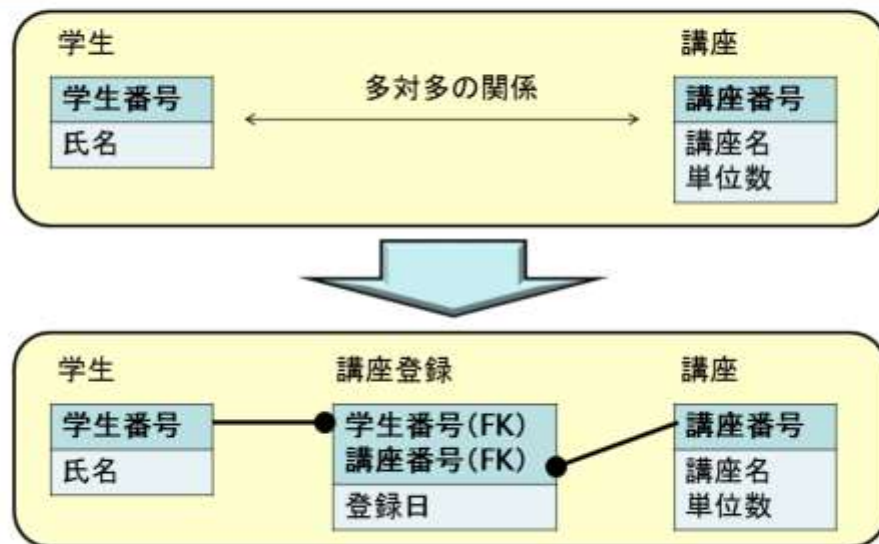
部署番号	部署名
10	教育部
20	システム部

社員番号	氏名	部署番号
101	内田太郎	10
102	鈴木花子	20
103	田中次郎	10

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

リレーションシップが「1 : 多」の場合、必ず「多」の側に外部キーを設定します。  
図のような「部署と社員」の関係の場合、社員が「多」になりますので、社員エンティティに外部キーとして「部署番号」を設定します。  
もし仮に、「1」である部署エンティティに社員番号（外部キー）があった場合、1つの部署に1人の社員しか所属できないということになってしまいます。

## 多対多のリレーションシップ



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

例えば、図のような「学生」と「講座」の関係は、「多対多」になります。

すなわち、1人の学生は複数の講座を受講しますし、1つの講座には複数の受講者がいます。

このような関係は、リレーショナル型のデータベースでは表現できません。

このような場合、2つのエンティティの間に、図の「講座登録」のようなエンティティを追加します。

これによって、「1対多」のリレーションシップが2つできることになります。





次の要件に従って、概念設計レベルのER図を作成してください。

主キー、外部キー以外にも、エンティティを説明するのに必要な属性も挙げてください。  
解答にあたって条件が不足する場合は、必要に応じて適切な前提条件を設定してください。

#### ■要件

ある企業では、社員が保有している資格を管理するデータベースを構築しようとしています。

このデータベースを利用して、特定の資格を有する社員の一覧を作ったり、部門別に有資格者の人数を数えるのに使おうと考えています。

この企業は、いくつかの部門が存在し、社員は必ず部門に所属しています。

管理したい資格は、あらかじめ登録しておきたいと考えています。社員が資格を取得したら、取得した日付も登録します。

社員が資格を取得するたびに、資格手当を社員に支給する制度があります。資格手当は、資格の種類によって決まっています。

## 2-4 正規化

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 正規化とは

社員番号	氏名	部署番号	部署名
101	内田太郎	10	教育部
102	鈴木花子	20	システム部
103	田中次郎	10	教育部



社員番号	氏名	部署番号
101	内田太郎	10
102	鈴木花子	20
103	田中次郎	10

部署番号	部署名
10	教育部
20	システム部

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ひとことで表現すれば「1事実1箇所」（「1つのデータは1つの場所に」）を実現するための手法です。

正規化の作業をすすめていくと、テーブルが複数のテーブルに分割され、データ構造がわかりやすく単純になっていきます。

## 正規化していない表の問題点

社員取得資格

社員番号	社員名	部門番号	部門名	部門住所	資格番号	資格名	認定団体	取得日
101	西田正	12	研修部	〇〇ビル3F	11263	MCSE	MS	2010/1/5
					11256	MCT	MS	2011/8/10
102	野平友之	12	研修部	〇〇ビル3F	11563	MCSD	MS	2011/12/8
					11256	MCT	MS	2012/4/3
103	木山悟	10	開発部	〇〇ビル5F	11563	MCSD	MS	2012/4/3
104	大川剛夫	11	企画部	〇〇ビル2F	11263	MCSE	MS	2010/6/18
105	梶山恵子	10	開発部	〇〇ビル5F	11256	MCT	MS	2010/6/18
106	香坂恵一	11	企画部	〇〇ビル2F	12101	Silver	Ora	2012/9/10
					12102	Gold	Ora	2012/9/30
107	西川智信	12	研修部	〇〇ビル3F	11563	MCSD	MS	2012/11/5
					12102	Gold	Ora	2013/3/10

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

図のような、社員の氏名や部門、所持している資格を管理する表を例とします。

このように、全てのデータを1つの表で管理した場合、以下のような問題があります。

### ■データベースへの実装時の問題

図のような「セルの結合」がされた表は、リレーショナル型データベースではそもそも扱うことができません。

### ■追加時の問題

例えば、新たな部門として「営業部」が出来た場合、営業部に配属される社員が決定するまで、この表にレコードを追加することができません。

### ■更新時の問題

例えば、研修部の部門住所が「〇〇ビル1F」に更新された場合、表中の複数レコードを更新しなければなりません。そのため、更新忘れによる矛盾が生じる可能性があります。

### ■削除時の問題

例えば、社員番号104の大川さんと106の香坂さんが退職し、表からレコードが削除されると、企画部が表から無くなってしまいます。

## 正規化のメリット

- データの冗長性を排除できる
- データ構造が分かりやすくなる

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

### ■ データの冗長性を排除できる

同一種類のデータが分散格納されません。従って、処理対象データを特定しやすくなり更新時異常を抑制できます。また、必要な格納メディア容量も小さくなります。

### ■ データ構造が分かりやすくなる

正規化をすすめていくと、格納されるデータは2次元の表であらわすことができるようになり、構造を把握しやすいテーブルになります。

## 正規形の種類



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

正規化の段階は、図のように7つに分けられます。

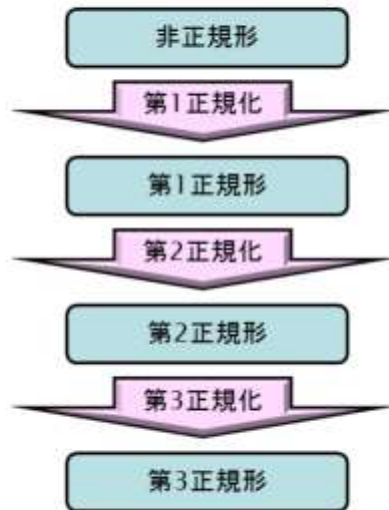
一番上段の非正規形は、全く正規化されていない状態です。

そこから段階的に、第1正規形→第2正規形→第3正規形→ボイス・コード正規形→第4正規形→第5正規形と進んでいきます。

ただし、ボイス・コード正規形以降は、かなり特殊な場合にしか使われませんので、一般的には、第3正規形まで正規化されていれば十分とされています。

このテキストでも、第3正規形までを解説します。

## 第1～第3正規化



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

### ■第1正規化：繰り返しを無くす

Excelで言うところの「セルの結合」のような状態のことを「繰り返し」と言います。この繰り返しを無くすことが第1正規化です。

### ■第2正規化：部分関数従属性を無くす

主キーが複数の列から構成されている場合に、主キーの一部が決まれば他の列の値が決まることを「部分関数従属性」と言います。これを無くすことが第2正規化です。

### ■第3正規化：推移関数従属性を無くす

「列Aが決まれば列Bが決まり、列Bが決まれば列Cが決まる」という「A→B→C」の関係のことを、「推移関数従属性」と言います。これを無くすことが第3正規化です。

## 非正規形

社員取得資格

社員番号	社員名	部門番号	部門名	部門住所	資格番号	資格名	認定団体	取得日	
101	西田正	12	研修部	〇〇ビル3F	11263	MCSE	MS	2010/1/5	} 繰り返し
					11256	MCT	MS	2011/8/10	
102	野平友之	12	研修部	〇〇ビル3F	11563	MCSD	MS	2011/12/8	}
					11256	MCT	MS	2012/4/3	
103	木山悟	10	開発部	〇〇ビル5F	11563	MCSD	MS	2012/4/3	
104	大川邦夫	11	企画部	〇〇ビル2F	11263	MCSE	MS	2010/6/18	
105	槐山恵子	10	開発部	〇〇ビル5F	11256	MCT	MS	2010/6/18	
106	香坂恵一	11	企画部	〇〇ビル2F	12101	Silver	Ora	2012/9/10	}
					12102	Gold	Ora	2012/9/30	
107	西川智信	12	研修部	〇〇ビル3F	11563	MCSD	MS	2012/11/5	}
					12102	Gold	Ora	2013/3/10	

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

先ほどの社員取得資格表を例に、正規化を段階に沿って行います。

図のように、非正規形のこの表には繰り返しが存在しており、このままではデータベースに実装することができません。

この表の属性定義は、以下のように表記します。

下線がついている属性は主キー、{ } で囲われた属性は繰り返しを表しています。

社員取得資格 = 社員番号 + 社員名 + 部門番号 + 部門名 + 部門住所 + { 資格番号 + 資格名 + 認定団体 + 取得日 }



## 第1正規形

社員

社員番号	社員名	部門番号	部門名	部門住所
101	西田正	12	研修部	〇〇ビル3F
102	野平友之	12	研修部	〇〇ビル3F
103	木山悟	10	開発部	〇〇ビル5F
104	大川邦夫	11	企画部	〇〇ビル2F
105	槐山恵子	10	開発部	〇〇ビル5F
106	香坂恵一	11	企画部	〇〇ビル2F
107	西川智信	12	研修部	〇〇ビル3F

資格取得

社員番号	資格番号	資格名	認定団体	取得日
101	11263	MCSE	MS	2010/1/5
101	11256	MCT	MS	2011/8/10
102	11563	MCSD	MS	2011/12/8
102	11256	MCT	MS	2012/4/3
103	11563	MCSD	MS	2012/4/3
104	11263	MCSE	MS	2010/6/18
105	11256	MCT	MS	2010/6/18
106	12101	Silver	Ora	2012/9/10
106	12102	Gold	Ora	2012/9/30
107	11563	MCSD	MS	2012/11/5
107	12102	Gold	Ora	2013/3/10

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

非正規形の表から、繰り返し部分を分離したものが第1正規形です。

社員＝社員番号＋社員名＋部門番号＋部門名＋部門住所

資格取得＝社員番号（FK）＋資格番号＋資格名＋認定団体＋取得日

## 第2正規形

### 社員

社員番号	社員名	部門番号	部門名	部門住所
101	西田正	12	研修部	〇〇ビル3F
102	野平友之	12	研修部	〇〇ビル3F
103	木山悟	10	開発部	〇〇ビル5F
104	大川邦夫	11	企画部	〇〇ビル2F
105	横山恵子	10	開発部	〇〇ビル5F
106	香坂恵一	11	企画部	〇〇ビル2F
107	西川智信	12	研修部	〇〇ビル3F

### 資格

資格番号	資格名	認定団体
11263	MCSE	MS
11256	MCT	MS
11563	MCSD	MS
12101	Silver	Ora
12102	Gold	Ora

### 資格取得

社員番号	資格番号	取得日
101	11263	2010/1/5
101	11256	2011/8/10
102	11563	2011/12/8
102	11256	2012/4/3
103	11563	2012/4/3
104	11263	2010/6/18
105	11256	2010/6/18
106	12101	2012/9/10
106	12102	2012/9/30
107	11563	2012/11/5
107	12102	2013/3/10

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

第1正規形の表から、主キーの一部のみに従属する属性を分離したものが、第2正規形です。

第1正規形の資格取得表では、社員番号と資格番号を合わせて主キーとなっていますが、主キーの一部である資格番号が決まれば、資格名と認定団体は決まります。

この部分を別の表に分離します。

社員＝社員番号＋社員名＋部門番号＋部門名＋部門住所

資格＝資格番号＋資格名＋認定団体

資格取得＝社員番号 (FK)＋資格番号 (FK)＋取得日

## 第3正規形

### 社員

社員番号	社員名	部門番号
101	西田正	12
102	野平友之	12
103	木山悟	10
104	大川邦夫	11
105	横山恵子	10
106	香坂恵一	11
107	西川智信	12

### 部門

部門番号	部門名	部門住所
10	開発部	〇〇ビル5F
11	企画部	〇〇ビル2F
12	研修部	〇〇ビル3F

### 資格取得

社員番号	資格番号	取得日
101	11263	2010/1/5
101	11256	2011/8/10
102	11563	2011/12/8
102	11256	2012/4/3
103	11563	2012/4/3
104	11263	2010/6/18
105	11256	2010/6/18
106	12101	2012/9/10
106	12102	2012/9/30
107	11563	2012/11/5
107	12102	2013/3/10

### 資格

資格番号	資格名	認定団体
11263	MCSE	MS
11256	MCT	MS
11563	MCSD	MS
12101	Silver	Ora
12102	Gold	Ora

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

第2正規形の表から、「A→B→C」というような推移関数従属性を無くしたものが、第3正規形です。

第2正規形の社員表では、「社員番号→部門番号→部門名、部門住所」という関係があるので、この部分を別の表に分離します。

社員＝社員番号＋社員名＋部門番号

部門＝部門番号＋部門名＋部門住所

資格＝資格番号＋資格名＋認定団体

資格取得＝社員番号 (FK)＋資格番号 (FK)＋取得日

これで、正規化は完了です。

# 非正規化

社員番号	氏名	部署番号
101	内田太郎	10
102	鈴木花子	20
103	田中次郎	10

部署番号	部署名
10	教育部
20	システム部



社員番号	氏名	部署番号	部署名
101	内田太郎	10	教育部
102	鈴木花子	20	システム部
103	田中次郎	10	教育部

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ここまで説明した通り、データベースの正規化は第3正規化まで行うことが基本です。しかし、正規化で表を分離すると、データを表示する際には、表と表を合体させる「結合」という処理が必要になります。結合の処理は非常に負荷が高い（時間がかかる）ため、実際のシステムでは、あえて正規化を崩す「非正規化」を行うことがあります。

## 属性の注意点

- 明確な名前を付ける
- 導出項目は含めない

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

### ■ 明確な名前を付ける

属性には明確な名前をつけなければなりません。それらが持っている意味を正確に表現する名前をつけるべきです。

また、用語は統一すべきです。データベース内に「社員番号」と「社員ID」の2つの属性が共存していたとき、両者が同一のものを指しているのか、異なるのかを判断することが困難になります。

### ■ 導出項目は含めない

導出項目とは、計算すれば求めることができる項目です（合計金額など）。そのため、一般的にはデータベースには含めません。

ただし、計算処理の付加が高い場合などは、データベースに導出項目を含めることもあります。



以下は、あるレストランのレシートです。

これを元に、第3正規化まで行ったER図を作成してください。

解答にあたって条件が不足する場合は、必要に応じて適切な前提条件を設定してください。

レストランマロニエ		
2014年1月24日(金) 17:07		2名
伝票NO:008343		
メニュー名称	数量	金額
オニオングラタンスープ	1	¥400
コーンスープ	1	¥350
オムライス	1	¥580
ビーフシチュー	1	¥800
オレンジジュース	1	¥220
コーラ	1	¥220
合計金額	¥2,570	
お預かり金額	¥10,070	
お釣り	¥7,500	

## 第3章 SQL

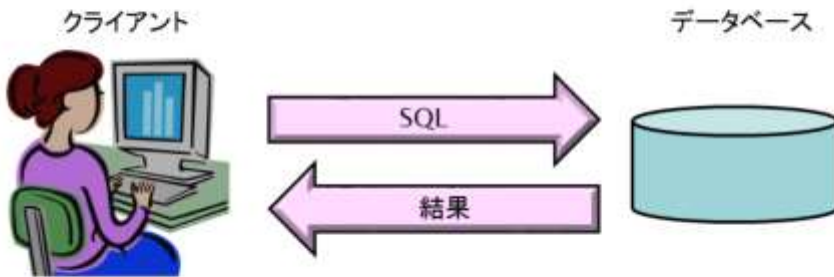
Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 3-1 SQLとは

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.



# SQLとは



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

SQLとは、データベースを操作するための言語です。

SQLで出来ることには、以下のようなことがあげられます。

- ①オブジェクト(表など)の定義をつくる
- ②表の操作(データの読み書き)をする
- ③データベースの管理をする

SQLには、ANSI・ISO・JISで制定された標準SQLと、各データベースメーカーが作ったSQLとがあります。

それぞれのSQLには、方言のように多少の違いはありますが、標準SQLもメーカーSQLも基本は同じです。

どれかひとつを理解できれば、他のSQLを使う場合でも心配ないでしょう。

## SQLの種類

- DDL
- DML
- DCL

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

SQLには、以下の3種類があります。

### ■DDL (Data Definition Language)

表の作成・変更・削除など、データを定義するためのSQLです。

### ■DML (Data Manipulation Language)

表に対する検索・追加・更新・削除など、データを操作するためのSQLです。

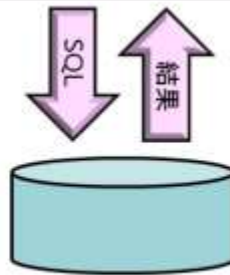
### ■DCL (Data Control Language)

トランザクション管理や権限設定のためのSQLです。

# プログラムとSQL

※Javaプログラムの例

```
...  
Connection con = DriverManager.getConnection(url, user, password);  
Statement st = con.createStatement();  
String sql = "SELECT empno,ename,job FROM emp";  
ResultSet rs = st.executeQuery(sql);  
...
```



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

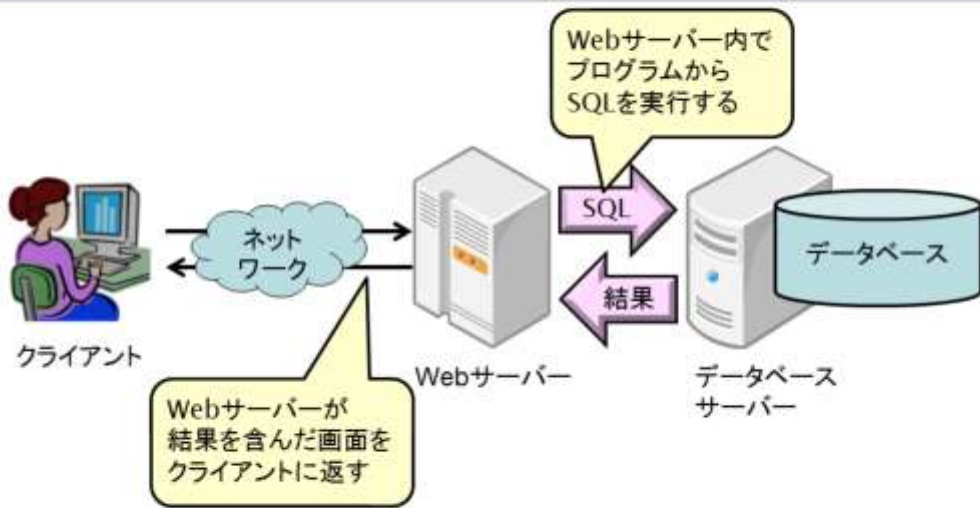
一般的な情報システムの場合、直接SQLでデータベースを操作することは少なく、図のようにプログラム中にSQLを埋め込むことが多くなります。

プログラムの中からSQLを実行し、検索結果などをプログラムが受け取り、表示を行います。

何故なら、直接SQLを触ることは、ITに詳しいとは限らないエンドユーザーにとっては難しいことだからです。

プログラムの役割は、複雑なSQLを隠蔽（ユーザーから見えなくする）し、使いやすいようにすることと言っても過言ではありません。

## WebシステムとSQL



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

近年の企業内のシステムは、**Web**を使ったイントラネットのシステムが多くを占めます。

**Webシステム**では、図のようにクライアント⇄Webサーバー⇄データベースサーバーという**3層構造**のシステム構成になっていることがほとんどです。

クライアントが**Webシステム**を利用する際は、まず**Webサーバー**に対してリクエスト（要求）を送ります。

そのリクエストを受け取った**Webサーバー**は、プログラム内から**SQL**を実行して、データベースから結果を受け取ります。

そして**Webサーバー**は、その結果を含んだ画面をクライアントにレスポンス（返す）します。

## 3-2 PostgreSQLとは

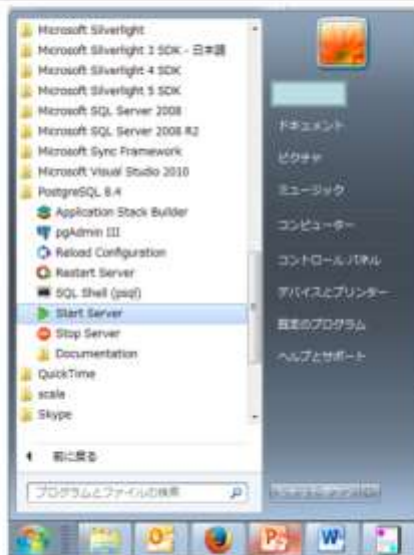
Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# PostgreSQLとは



今回の研修では、「PostgreSQL」というオープンソースのデータベースを使用します。オープンソースなので、Webからダウンロードして無償で利用できます。

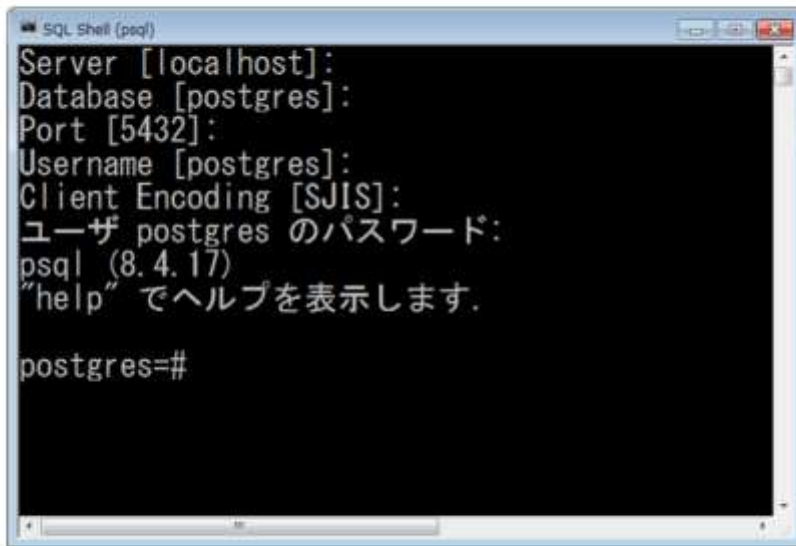
## PostgreSQLの起動・停止



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

PostgreSQLデータベースは、  
[スタート]メニュー-[すべてのプログラム]-[PostgreSQL 8.4]-[Start Server]  
で起動します。  
停止する際は[Stop Server]から行います。

## psqlの起動



```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Client Encoding [SJIS]:
ユーザ postgres のパスワード:
psql (8.4.17)
"help" でヘルプを表示します。

postgres=#
```

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

psqlとは、PostgreSQLデータベースを操作するためのツールです。

SQLを入力し、実行することができます。

psqlは、

[スタート]メニュー-[すべてのプログラム]-[PostgreSQL 8.4]-[SQL Shell(psql)]

で起動します。

図のようなコマンドプロンプト画面が立ち上がったら、**Enter**キーを5回押します。

その後、パスワード「**postgres**」を入力し、**Enter**を押します（画面には入力した文字は表示されません）。

「**postgres=#**」というプロンプトが表示されれば、起動成功です。



### 3-3 基本的な検索

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# 基本的なSELECT文

## 文法

```
SELECT 列 [, 列...] FROM 表;
```

## 例

```
SELECT empno,ename FROM emp;
```

## 結果

empno	ename
101	Nishida
102	Nohira
...	...

※大文字・小文字は区別されない  
※psql上で実行する際は、  
最後に必ず「;」が必要

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データを検索する際は、「SELECT文」を使用します。

「SELECT」というキーワードの後に列名を指定し（複数列の場合は「,」で区切る）、  
「FROM」の後に表名を指定します。

## 全列検索

### 文法

```
SELECT * FROM 表;
```

### 例

```
SELECT * FROM emp;
```

### 結果

empno	ename	job	mgr	hiredate	sal	comm	deptno
101	Nishida	Director		1981-11-17	500,000		10
102	Nohira	Manager	101	1981-5-1	285,000		30
...	...	...	...	...	...	...	...

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

「SELECT」の後に「\*」を指定すると、全列を検索することができます。

## 列の別名

### 文法

```
SELECT 列 AS 別名 [, 列 AS 別名...] FROM 表;
```

### 例

```
SELECT empno AS id, ename AS name FROM emp;
```

### 結果

id	name
101	Nishida
102	Nohira
...	...

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

SELECT文で列名を指定する際、列名の後に「AS 別名」と指定すると、検索結果の列が別名で表示されます。

# データの並び替え

## 文法

```
SELECT 列 [, 列...] FROM 表 ORDER BY 列 [, 列...];
```

## 例

```
SELECT * FROM emp ORDER BY empno;
```

## 結果

empno	ename	job	mgr	hiredate	sal	comm	deptno
101	Nishida	Director		1981-11-17	500,000		10
102	Nohira	Manager	101	1981-5-1	285,000		30
...	...	...	...	...	...	...	...

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ORDER BY句を使用すると、検索結果の並び替えを行うことができます。

デフォルトの並び替えは昇順（値が小さいものから大きいものへ）です。以下のようにも記述できます。

```
SELECT * FROM emp ORDER BY empno ASC;
```

降順（大きいものから小さいものへ）で並び替えたい場合は、以下のように指定します。

```
SELECT * FROM emp ORDER BY empno DESC;
```

ORDER BY句で指定する列名は、AS句で付けた別名でも可能です。

## 重複行の削除

### 文法

```
SELECT DISTINCT 列 [, 列...] FROM 表;
```

### 例

```
SELECT DISTINCT job FROM emp;
```

### 結果

job
Clerk
Engineer
Manager
Salesman
Director

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

DISTINCT句を使用すると、検索結果から重複行を削除することができます。  
例えば、**emp**表から**job**（職種）を一覧したい場合などに便利です。



以下のSQLを作成してください

- (1)DEPT表の、所在地、部門名を（この順番で）表示する
- (2)LIC表の、資格名、認定団体を、認定団体の降順で表示する
- (3)LIC表を用いて、認定団体の一覧を表示する。ひとつの団体は1行のみの表示とすること

## 特定行の検索

### 文法

```
SELECT 列 [, 列...] FROM 表 WHERE 条件式;
```

### 例

```
SELECT empno,ename,job FROM emp WHERE job = 'Salesman';
```

### 結果

empno	ename	job
105	Kajiyama	Salesman
106	Kohsaka	Salesman
107	Nishikawa	Salesman
109	Ichikawa	Salesman

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

WHERE句を使用すると、指定した条件に一致する行を検索することができます。



## 比較演算子

演算子	意味・例
=	等しい SELECT * FROM emp WHERE hiredate = '1981-04-02';
!= または <>	等しくない SELECT * FROM emp WHERE job != 'Salesman';
>	より大きい SELECT * FROM emp WHERE sal > 150000;
>=	以上 SELECT * FROM emp WHERE sal >= 150000;
<	より小さい SELECT * FROM emp WHERE sal < 150000;
<=	以下 SELECT * FROM emp WHERE sal <= 150000;

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## その他の比較演算子

演算子	意味・例
BETWEEN x AND y	x以上y以下 SELECT * FROM emp WHERE sal BETWEEN 100001 AND 199999;
IS NULL	NULL(空)である SELECT * FROM emp WHERE comm IS NULL;
LIKE	文字パターンと一致する行を検索する %...任意の複数文字(0文字も可) _...任意の1文字 SELECT * FROM emp WHERE ename LIKE '____i'; SELECT * FROM emp WHERE ename LIKE '%i%';

## 論理演算子

演算子	意味・例
AND	かつ SELECT * FROM emp WHERE job = 'Salesman' AND sal > 150000;
OR	または SELECT * FROM emp WHERE job = 'Salesman' OR job = 'Clerk';
NOT	否定 SELECT * FROM emp WHERE ename NOT LIKE 'S%';

## 論理演算子の優先順位

例

```
SELECT empno,ename,job,sal FROM emp  
WHERE job = 'Clerk' OR job = 'Salesman' AND sal > 150000;
```

結果

empno	ename	job	sal
106	Kohsaka	Salesman	160,000
108	Sasaki	Clerk	95,000
111	Komatsu	Clerk	80,000
113	Saitoh	Clerk	110,000
114	Inoue	Clerk	130,000

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

論理演算子の優先順位は以下のとおりです。

- 1.NOT
- 2.AND
- 3.OR

## 論理演算子の優先順位

例

```
SELECT empno,ename,job,sal FROM emp  
WHERE (job = 'Clerk' OR job = 'Salesman') AND sal > 150000;
```

結果

empno	ename	job	sal
106	Kohsaka	Salesman	160,000

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

( )を使用して、優先順位を変更することができます。



以下のSQLを作成してください

(1)LICCTF表を用いて、資格番号12102を取得した社員の社員番号と、取得日を表示する

(2)EMP表を用いて、所属部門が無い（NULL）社員の社員番号と名前を表示する

(3)LIC表を用い、資格手当が40,000以上で、さらに認定団体が「MS」でない資格の、資格名・認定団体・資格手当を表示する

## 3-4 グループ関数

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## グループ関数

関数	意味・例
SUM()	NULLを除いた合計値を返す SELECT SUM(sal) FROM emp;
AVG()	NULLを除いた平均値を返す SELECT AVG(sal) FROM emp;
MAX()	NULLを除いた最大値を返す SELECT MAX(sal) FROM emp;
MIN()	NULLを除いた最小値を返す SELECT MIN(sal) FROM emp;
COUNT()	列を指定した場合は、NULL値を除いた行数を、*を指定した場合はNULL値を含むすべての行数を返す SELECT COUNT(*) FROM emp; SELECT COUNT(mgr) FROM emp;

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

関数とは、データベースがあらかじめ用意したデータ操作のためのプログラムです。  
SQL文の中で、データの計算や、データ型の変換を行うために使用します。  
その中でもグループ関数とは、グループ単位で一つの結果を求める関数です。  
グループ分けしていない場合は、表全体が一つのグループとなります。



## 3-5 データのグループ化

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# データのグループ化

## 文法

```
SELECT 列 [, 列...] FROM 表 [GROUP BY 列 [, 列...]];
```

## 例

```
SELECT job,COUNT(*) AS COUNT FROM emp GROUP BY job;
```

## 結果

job	count
Clerk	4
Director	1
Engineer	2
Manager	3
Salesman	4

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

GROUP BY句を使用すると、表のデータをグループ化することができます。

例えば、**emp**表で**job**ごとにグループ化し、職種ごとの人数を求めることができます。

GROUP BY句を使った場合、SELECT句に指定できるものは、GROUP BYで指定した列、グループ関数、定数のみとなります。

## グループに対する条件指定

### 文法

```
SELECT 列 [, 列...] FROM 表 [WHERE 条件]  
[GROUP BY 列 [, 列...]] [HAVING 条件];
```

### 例

```
SELECT job,AVG(sal) FROM emp WHERE empno > 105  
GROUP BY job HAVING AVG(sal) > 120000;
```

### 結果

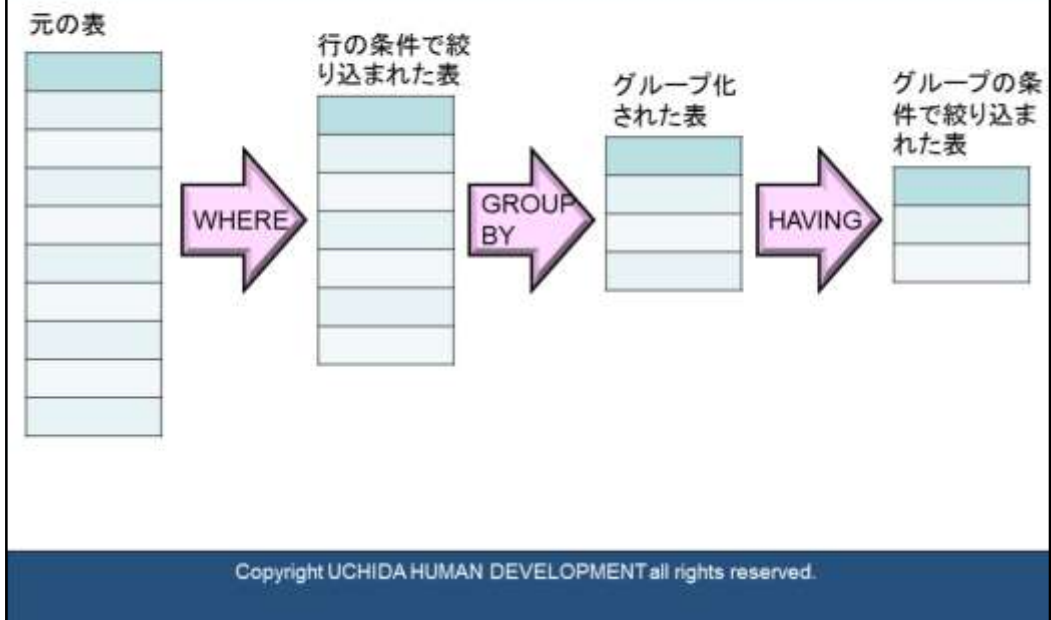
job	avg
Engineer	300000.00
Salesman	145000.00

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

HAVING句を使用すると、GROUP BYで作成したグループに対する条件を指定することができます。

例えば、emp表でjobごとにグループ化し、平均給与が120000を超えるグループのみ抽出することができます。

## 句の評価順序



**WHERE**は行に対する条件、**HAVING**はグループに対する条件を指定します。

**WHERE**句、**GROUP BY**句、**HAVING**句があった場合、評価の順序は以下のようになります。

- ①**WHERE**句で行を絞り込む
- ②**GROUP BY**句で残った行をグループ化する
- ③**HAVING**句で各グループから更に絞り込む



以下のSQLを作成してください

(1)全社員の平均給与を表示する

(2)(1)の表示結果を、職種ごとの平均給与を表示するように変更する

(3)LIC表を用いて、認定団体ごとの資格数を表示する。ただし、資格数が2つ以上の団体のみ表示すること

(4)LICCTF表を用いて、社員番号と、社員それぞれが取得した資格数を表示する。ただし、資格数の多い順に表示すること

## 3-6 表の結合

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 結合とは

emp

empno	ename	...	deptno
101	Nishida	...	10
102	Nohira	...	30
103	Kiyama	...	10
...	...	...	...

dept

deptno	dname	...
10	Admin	...
20	Planning	...
30	Sales	...
40	Operations	...

結合

ename	deptno	dname
Nishida	10	Admin
Nohira	30	Sales
Kiyama	10	Admin
...	...	...

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

リレーショナル型データベースでは、表は正規化（＝分割）されています。

そのため、例えば「社員名と部署名を表示したい」となった場合、**emp**表と**dept**表を合体させる必要があります。

この表の合体のことを「結合」と言います。

## 等価結合

### 文法

```
SELECT [表1.]列, [表2.]列, ... FROM 表1 JOIN 表2  
ON [表1.] 列1 = [表2.] 列1;
```

### 例

```
SELECT emp.empno, emp.ename, dept.deptno, dept.dname  
FROM emp JOIN dept ON emp.deptno = dept.deptno;
```

### 結果

empno	ename	deptno	dname
101	Nishida	10	Admin
102	Nohira	30	Sales
...	...	...	...

※「113 Saitoh」のレコードは、deptnoがNULLのため結果には含まれない

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

複数の表を結合するには、**FROM**句と**JOIN**句に結合したい表を指定し、**ON**句に結合条件を指定します。

結合条件とは、どの表のどの列と、どの表のどの列が結合の基準となるのか、結合の手がかりを指定するものです。

等価結合は、結合条件に「=」を使用し、列の値が等しいもの同士を結合します。

**ON**句や**SELECT**句で指定した列が、2つの表に共通して存在する場合、どちらの表の列か明確にする必要があります。そのためには、列名の前に表名を付けます。

2つの表に共通する列名ではない場合は、必ずしも表名を付ける必要はありません。



# 左外部結合

## 文法

```
SELECT [表1.]列, [表2.]列, ... FROM 表1 LEFT OUTER JOIN 表2
ON [表1.] 列1 = [表2.] 列1;
```

## 例

```
SELECT emp.empno, emp.ename, dept.deptno, dept.dname
FROM emp LEFT OUTER JOIN dept ON emp.deptno = dept.deptno;
```

## 結果

empno	ename	deptno	dname
101	Nishida	10	Admin
102	Nohira	30	Sales
...	...	...	...
113	Saitoh		
114	Inoue	10	Admin

←deptnoがNULLのレコードも  
結果に含まれる

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

左外部結合では、結合する「左側」の表（例の場合はemp表）を基準として、結合を行います。

左外部結合の場合、部署に所属していない（結合条件deptnoがNULLである）社員も、結果に含まれます。

## 右外部結合

### 文法

```
SELECT [表1.]列, [表2.]列, ... FROM 表1 RIGHT OUTER JOIN 表2  
ON [表1.] 列1 = [表2.] 列1;
```

### 例

```
SELECT emp.empno, emp.ename, dept.deptno, dept.dname  
FROM emp RIGHT OUTER JOIN dept ON emp.deptno = dept.deptno;
```

### 結果

empno	ename	deptno	dname
101	Nishida	10	Admin
102	Nohira	30	Sales
...	...	...	...
114	Inoue	10	Admin
		40	Operations

←社員が所属しない部署の  
レコードも結果に含まれる

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

右外部結合では、結合する「右側」の表（例の場合はdept表）を基準として、結合を行います。

右外部結合の場合、社員が所属しない部署のレコードも、結果に含まれます。

## 全外部結合

### 文法

```
SELECT [表1.]列, [表2.]列, ... FROM 表1 FULL OUTER JOIN 表2  
ON [表1.] 列1 = [表2.] 列1;
```

### 例

```
SELECT emp.empno, emp.ename, dept.deptno, dept.dname  
FROM emp FULL OUTER JOIN dept ON emp.deptno = dept.deptno;
```

### 結果

empno	ename	deptno	dname
101	Nishida	10	Admin
...	...	...	...
113	Saitoh		
		40	Operations

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

全外部結合では、左外部結合と右外部結合を合わせたような結合です。

全外部結合の場合、部署に所属しない社員、社員が所属しない部署、両方のレコードが結果に含まれます。



以下のSQLを作成してください

- (1)EMP表とLICCTF表を用いて、社員名・取得した資格番号を表示する
- (2)EMP表とDEPT表を用いて、社員名・部門番号・所在地を表示する。ただし、所在地が「3F」の社員のみ対象とすること
- (3)EMP表、LICCTF表を用いて、社員名と、その社員が取得した資格数を表示する

## 3-7 追加・更新・削除

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## データの追加

### 文法

```
INSERT INTO 表 [ (列1 [,列2・・・] ) ]VALUES (値1 [,値2・・・]);
```

### 例

```
INSERT INTO emp(empno,ename,deptno) VALUES(115,'Miyabe',20);  
INSERT INTO emp VALUES(116,'Ezawa',NULL,104,'2000-03-23',180000,NULL,20);  
INSERT INTO emp VALUES(117,'Furuta','Salesman',102,'2000-04-01',150000,0,30);
```

### 結果

empno	ename	job	mgr	hiredate	sal	comm	deptno
...	...	...	...	...	...	...	...
115	Miyabe						20
116	Ezawa		104	2000-03-23	180,000		20
117	Furuta	Salesman	102	2000-04-01	150,000	0	30

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

テーブルにレコードを追加する場合は、INSERT文を使います。

INSERT句の列指定を省略すると、VALUES句で指定した値を、表の列定義の順序通りに格納します。

# データの更新

## 文法

```
UPDATE 表 SET 列1 = 値1 [,列2 = 値2... ] WHERE 条件;
```

## 例

```
UPDATE emp SET ename = 'Hamada', job = 'Clerk' WHERE empno = 115;
```

## 結果

empno	ename	job	mgr	hiredate	sal	comm	deptno
...	...	...	...	...	...	...	...
115	Hamada	Clerk					20
116	Ezawa		104	2000-03-23	180,000		20
117	Furuta	Salesman	102	2000-04-01	150,000	0	30

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

レコードを更新する場合は、**UPDATE**文を使います。

**SET**句では、更新したい列のみ指定します。

**WHERE**句を指定すると、条件を満たすレコードのみ更新されます。

**WHERE**句を省略すると、表内の全てのレコードが更新されます。

# データの削除

## 文法

```
DELETE FROM 表 [WHERE 条件];
```

## 例

```
DELETE FROM emp WHERE empno = 115;
```

## 結果

empno	ename	job	mgr	hiredate	sal	comm	deptno
...	...	...	...	...	...	...	...
115	Hamada	Clerk					20
116	Ezawa		104	2000-03-23	180,000		20
117	Furuta	Salesman	102	2000-04-01	150,000	0	30

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

レコードを削除する場合は、**DELETE**文を使います。

**WHERE**句を指定すると、条件を満たす行のみ削除されます。

**WHERE**句を省略すると、表内の全ての行が削除されます。



# CRUD

- Create
- Refer
- Update
- Delete

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データベースを利用する際に、よく「**CRUD**」という用語を使います。

これは、以下の言葉の頭文字を取ったものです。

**Create**・・・データの追加 (**INSERT**)

**Refer**・・・データの参照 (**SELECT**)

**Update**・・・データの更新 (**UPDATE**)

**Delete**・・・データの削除 (**DELETE**)



以下のSQLを作成してください

(1)LIC表に、資格番号17101、資格名SJCP、認定団体Sun、資格手当20,000であるレコードを追加する

(2)(1)で追加したレコードの、資格番号を12301、資格名をOCJP、認定団体をOraに更新する

(3)(2)で更新したレコードを削除する

## 3-8 表の作成と削除

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

# 表の作成

## 文法

```
CREATE TABLE 表名 (  
  列名 データ型 [制約]  
  [ ,列名 データ型 [制約]... ]  
);
```

## 例

```
CREATE TABLE emp2 (  
  id integer PRIMARY KEY  
  ,name varchar(20)  
  ,deptno integer REFERENCES dept(deptno)  
);
```

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

表を作成するときは、**CREATE TABLE**文を使います。

例では、**emp2**という表を作成しています。列は**id**、**name**、**deptno**の3つです。

**id**には主キー制約、**deptno**には**dept**表の**deptno**への外部キー制約を付けています。

# データ型

## 文字列型

データ型	説明
char(N)	固定長文字列。Nは文字数を指定する
varchar(N)	可変長文字列。Nは最大文字数を指定する

## 数値型

データ型	説明
integer	4バイト符号付整数(-2147483648~2147483647)

## 日付型

データ型	説明
date	日付(年月日)

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データ型とは、その列がどのような種類のデータかを表すものです。

前頁の例では、**id**は**integer**というデータ型になっています。**integer**は整数という意味です。

このように設定すると、**id**には整数の値しか入れることができません。

例えば、**id**に「AAA」などといった文字列を入れることは不可能になります。

## 表の削除

文法

```
DROP TABLE 表名;
```

例

```
DROP TABLE emp2;
```

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

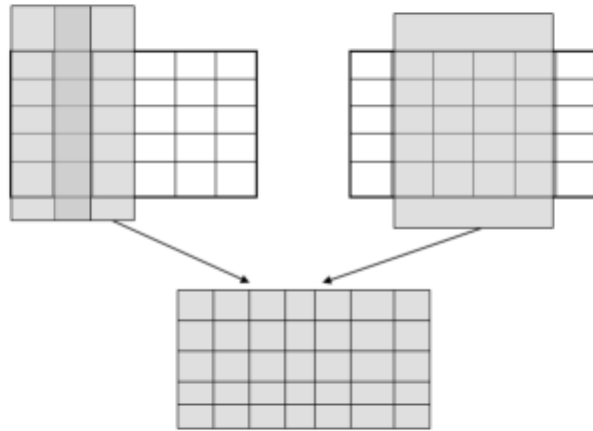
表を削除するときは、**DROP TABLE**文を使います。

「**DROP TABLE**」の後に、削除する表の名前を指定します。

## 3-9 ビューの作成と削除

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## ビューとは



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ビューとは、実際の表（複数の場合もあり）を元に、ある条件で抽出した「仮想表」です。

正規化された表は、データベース管理者、開発者にとっては、メンテナンス効率に優れたデータモデルですが、一般ユーザーにとっては使いやすいとは限りません。

ビューをすることによって、ユーザーにとって使いやすい「表」を作ることができます。また、セキュリティの面から、見せたくないデータを隠すこともできます。



# ビューの作成

## 文法

```
CREATE VIEW ビュー名 AS SELECT...
```

## 例

```
CREATE VIEW emp3 AS SELECT empno,ename FROM emp;
```

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ビューを作成するときは、**CREATE VIEW**文を使います。

例では、**emp**表から**empno**と**ename**のみを切り出した、**emp3**というビューを作成しています。

## ビューからの検索

例

```
SELECT * FROM emp3 WHERE empno = 101;
```

結果

empno	ename
101	Nishida

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ビューから検索する際の文法は、通常の**SELECT**文と全く同じです。  
**FROM**句の後に、検索したいビューの名前を指定します。

## ビューの削除

文法

```
DROP VIEW ビュー名;
```

例

```
DROP VIEW emp3;
```

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ビューを削除するときは、**DROP VIEW**文を使います。

「**DROP VIEW**」の後に、削除するビューの名前を指定します。

# 付録

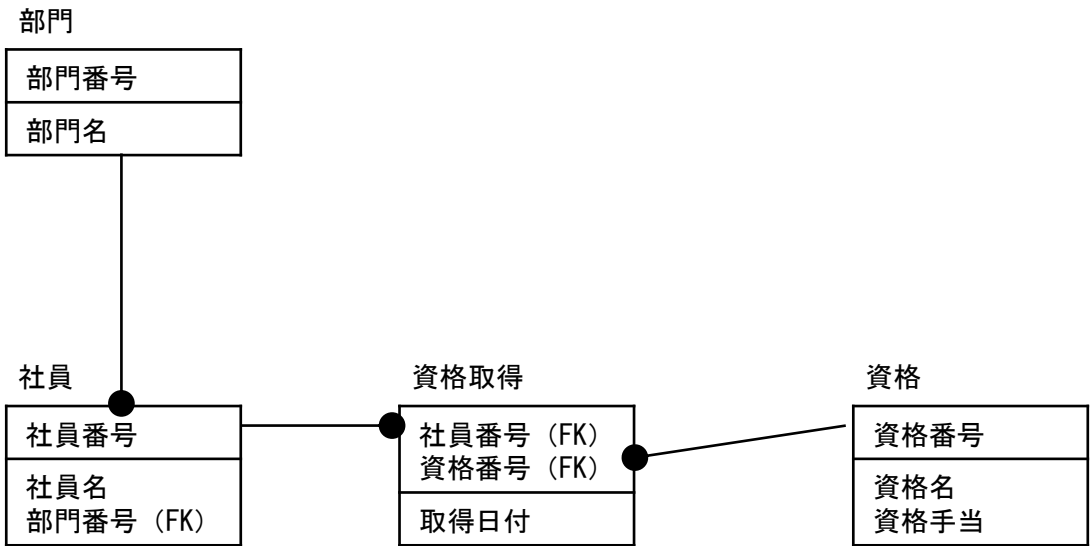
Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 付-1 演習解答

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.



解答はありません。自由に挙げてください。





●非正規形

レシート＝伝票NO＋会計日付＋会計時刻＋人数  
 ＋ {メニューNO＋メニュー名称＋注文数量＋金額}  
 ＋合計金額＋お預かり金額＋お釣り

●第1正規形

レシート＝伝票NO＋会計日付＋会計時刻＋人数＋合計金額＋お預かり金額＋お釣り  
 レシート明細＝伝票NO(FK)＋メニューNo＋メニュー名称＋注文数量＋金額

●第2正規形

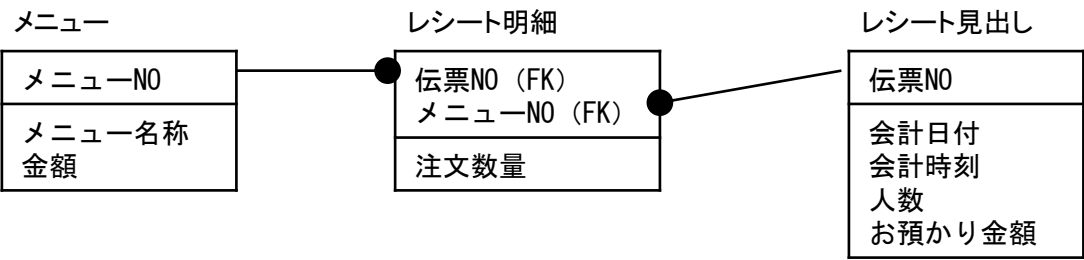
レシート見出し＝伝票NO＋会計日付＋会計時刻＋人数＋合計金額  
 ＋お預かり金額＋お釣り

レシート明細＝伝票NO(FK)＋メニューNo(FK)＋注文数量  
 メニュー＝メニューNO＋メニュー名称＋金額

●第3正規形

レシート見出し＝伝票NO＋会計日付＋会計時刻＋人数＋お預かり金額  
 レシート明細＝伝票NO(FK)＋メニューNo(FK)＋注文数量  
 メニュー＝メニューNO＋メニュー名称＋金額

※この演習では、第2正規形と第3正規形は同じである。  
 ただし、合計金額とお釣りは導出項目であるため省いた。







- (1) `SELECT loc,dname FROM dept;`
- (2) `SELECT lname,lvdr FROM lic ORDER BY lvdr DESC;`
- (3) `SELECT DISTINCT lvdr FROM lic;`



- (1) `SELECT empno,ctfdate FROM licctf WHERE licno = 12102;`
- (2) `SELECT empno,ename FROM emp WHERE deptno IS NULL;`
- (3) `SELECT lname,lvdr,licinc FROM lic WHERE licinc >= 40000 AND lvdr != 'MS';`



- (1) `SELECT AVG(sal) FROM emp;`
- (2) `SELECT job,AVG(sal) FROM emp GROUP BY job;`
- (3) `SELECT lvdr,COUNT(*) FROM lic GROUP BY lvdr HAVING COUNT(*) >= 2;`
- (4) `SELECT empno,COUNT(*) FROM licctf GROUP BY empno ORDER BY COUNT(*)  
DESC;`



- (1) `SELECT emp.ename,licctf.licno FROM emp JOIN licctf ON emp.empno =  
licctf.empno;`
- (2) `SELECT emp.ename,dept.deptno,dept.loc FROM emp JOIN dept ON  
emp.deptno = dept.deptno WHERE loc = '3F';`
- (3) `SELECT emp.ename,COUNT(*) FROM emp JOIN licctf ON emp.empno =  
licctf.empno GROUP BY emp.ename;`



- (1) `INSERT INTO lic VALUES(17101,'SJCP','Sun',20000);`
- (2) `UPDATE lic SET licno = 12301,lname = 'OCJP',lvdr = 'Ora' WHERE licno = 17101;`
- (3) `DELETE FROM lic WHERE licno = 12301;`

## 付-2 PostgreSQLのインストール

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## ①インストーラーの入手



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

下記のWebサイトからインストーラーを入手します。

<http://www.enterprisedb.com/products-services-training/pgdownload#windows>

今回は、PostgreSQL 8.4.17のWindows用インストーラー（postgresql-8.4.17-1-windows.exe）をダウンロードします。

## ②Setup

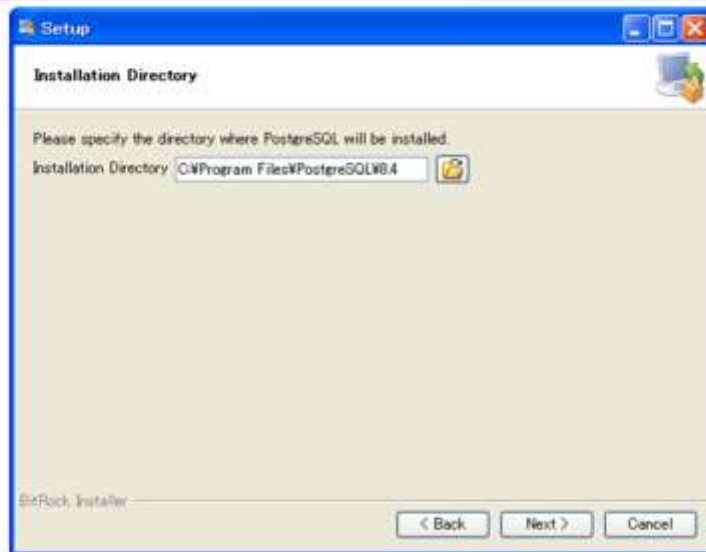


Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ダウンロードしたインストーラーをダブルクリックして実行すると、図のようなセットアップ画面が表示されるので、**[Next]**をクリックします。



### ③ インストールフォルダの実行

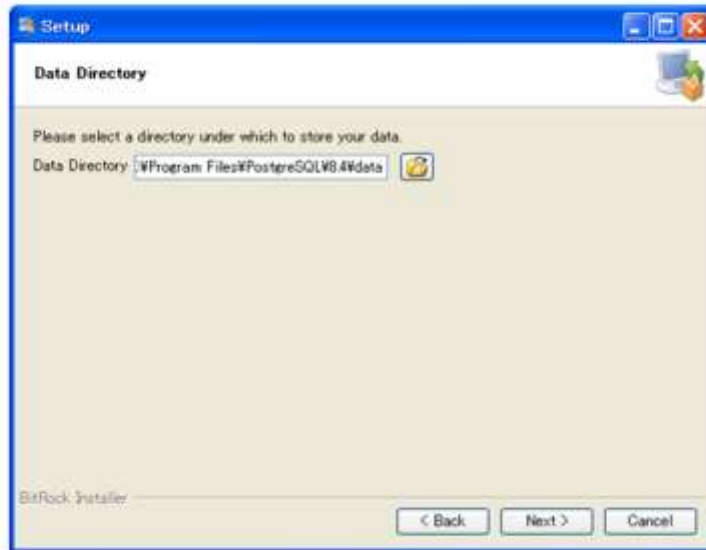


Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

PostgreSQLをインストールするフォルダを選択します。

デフォルトでは[Installation Directory]が「C:¥Program Files¥PostgreSQL¥8.4」となっているため、そのまま[Next]を選択します。

## ④データ格納フォルダの選択

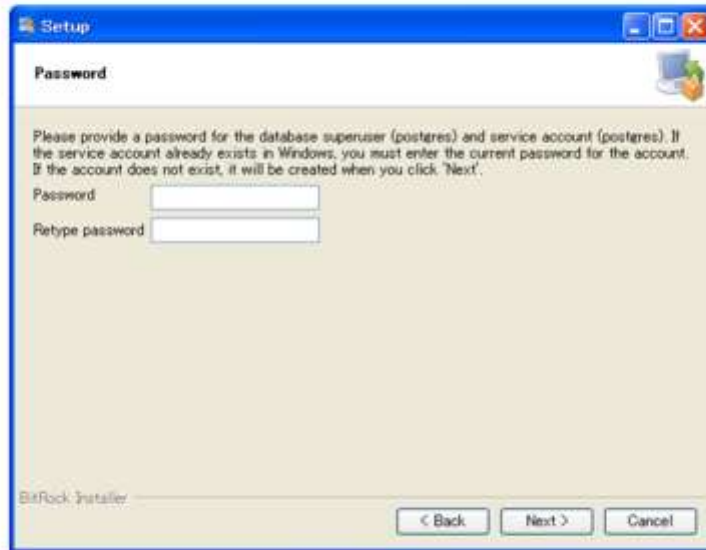


Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

データベースのデータを格納するフォルダを選択します。

[Data Directory]は、デフォルトでは「C:¥Program Files¥PostgreSQL¥8.4¥data」となっていますが、パスに半角スペースが含まれているとインストール出来ないことがあるため、「C:¥PostgreData」と入力して[Next]を選択します。

## ⑤パスワードの設定



The screenshot shows a Windows-style installer window titled "Setup". The main heading is "Password". Below the heading, there is a paragraph of instructions: "Please provide a password for the database superuser (postgres) and service account (postgres). If the service account already exists in Windows, you must enter the current password for the account. If the account does not exist, it will be created when you click 'Next'." There are two text input fields: "Password" and "Retype password". At the bottom of the window, there are three buttons: "< Back", "Next >", and "Cancel". The text "BitRock Installer" is visible in the bottom left corner of the window.

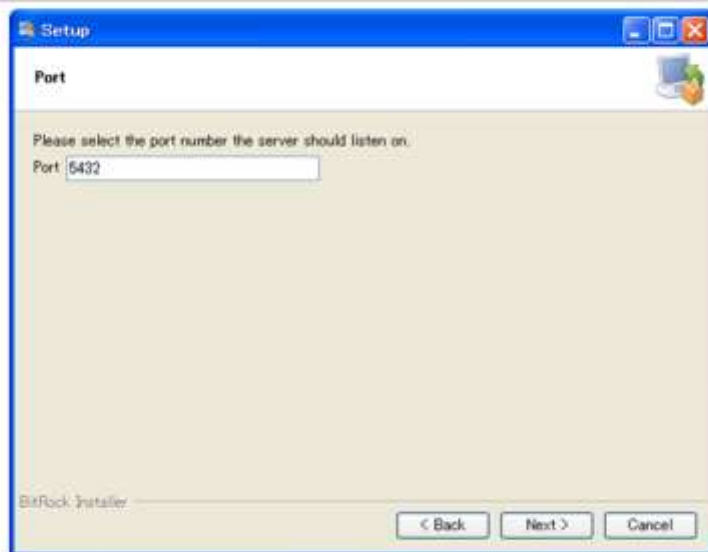
Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

パスワードを設定します。

[Password]、[Retype password]ともに「postgres」と入力して、[Next]を選択します。

※Windowsセキュリティポリシーの設定によっては、単純なパスワードでは受け付けられないこともあります。その場合は、ポリシーに従って複雑なパスワードを設定してください。

## ⑥ポート番号の設定

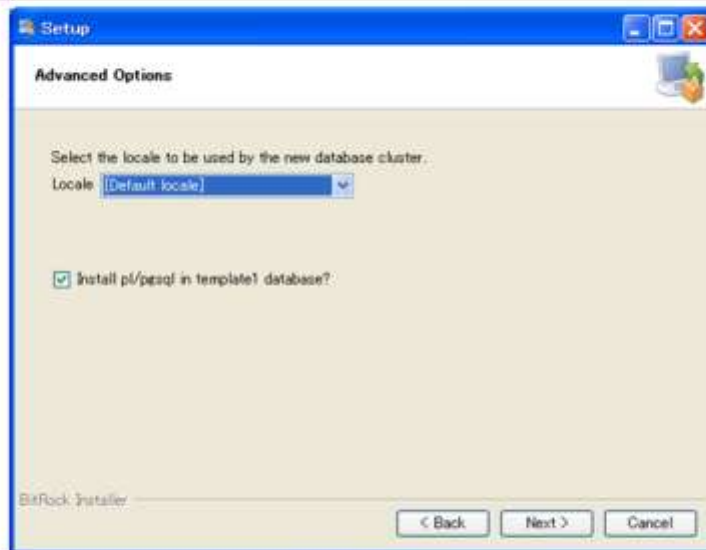


Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

PostgreSQLが利用するポート番号を設定します。

デフォルトでは[Port]が「5432」となっているため、そのまま[Next]を選択します。

## ⑦ロケールの選択

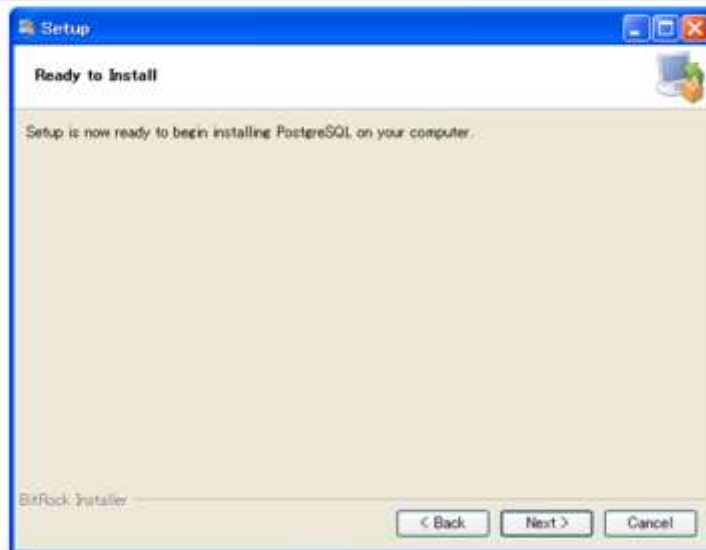


Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

ロケールを選択します。

[Locale]で「C」を選択し、[Next]を選択します。

## ⑧インストールの開始



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

[Next]を選択すると、インストールが開始します。  
この処理には数分かかる場合があります。

## ⑨インストールの完了



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

インストールが完了すると、図のような画面が表示されます。

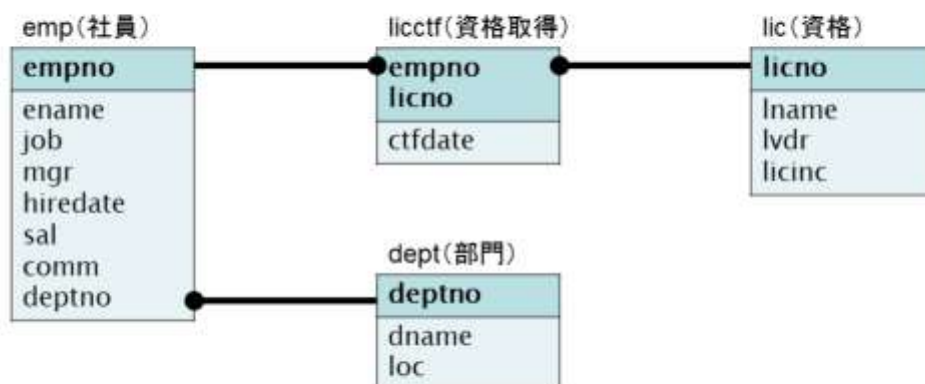
[Launch Stack Builder at exit? ... ]のチェックを外して[Finish]を選択します。

## 付-3 サンプルのデータベース

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.



## ER図



Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 各表の説明

emp表(社員表) EMPLOYEE

カラム(列)	型(データ型)	説明
empno	smallint	社員番号
ename	character varying	社員名
job	character varying	職種
mgr	smallint	上司の社員番号
hiredate	date	入社日
sal	integer	給与
comm	integer	歩合給
deptno	smallint	部門番号

dept表(部門表) DEPARTMENT

カラム(列)	型(データ型)	説明
deptno	smallint	部門番号
dname	character varying(14)	部門名
loc	character varying(13)	所在地

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 各表の説明(つづき)

lic表(資格表)

LiCense

カラム(列)	型(データ型)	説明
licno	smallint	資格番号
lname	character varying	資格名
lvdr	character varying	認定団体
licinc	integer	資格手当

licctf表(資格取得表) LiCense CerTiFied

カラム(列)	型(データ型)	説明
empno	smallint	社員番号
licno	smallint	資格番号
ctfdate	date	資格取得日

## データ内容

emp表

empno	ename	job	mgr	hiredate	sal	comm	deptno
101	Nishida	Director		1981-11-17	500,000		10
102	Nohira	Manager	101	1981-5-1	285,000		30
103	Kiyama	Manager	101	1981-6-9	245,000		10
104	Ohkawa	Manager	101	1981-4-2	297,500		20
105	Kajiyama	Salesman	102	1981-9-28	125,000	140,000	30
106	Kohsaka	Salesman	102	1981-2-20	160,000	30,000	30
107	Nishikawa	Salesman	102	1981-9-8	150,000	0	30
108	Sasaki	Clerk	102	1981-12-3	95,000		30
109	Ichikawa	Salesman	102	1981-2-22	125,000	50,000	30
110	Yamamoto	Engineer	104	1981-12-3	300,000		20
111	Komatsu	Clerk	110	1980-12-17	80,000		20
112	Aizawa	Engineer	104	1982-12-9	300,000		20
113	Saitoh	Clerk	112	1983-1-12	110,000		
114	Inoue	Clerk	103	1982-1-23	130,000		10

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## データ内容(つづき)

dept表

deptno	dname	loc
10	Admin	5F
20	Plannning	2F
30	Sales	3F
40	Operations	4F

lic表

licno	lname	lvdr	licinc
11256	MCT	MS	100000
11263	MCSE	MS	30000
11563	MCSD	MS	30000
12101	Silver	Ora	10000
12102	Gold	Ora	50000
13640	CCNA	Cisco	10000
13642	CCNP	Cisco	50000
14101	CompM	Comp	3800
14102	CompO	Comp	2400
15303	KitSp	Int	8400
15606	TradeA	Jet	43000
15707	CFA	JCFA	6100
16106	BCM	CJSD	8800

## データ内容(つづき)

licctf表

empno	licno	ctfdate
101	11256	1996-08-10
101	11263	1995-01-05
101	14101	1992-12-03
102	11256	1997-04-03
102	11563	1996-12-08
102	14101	1996-05-30
103	11563	1997-04-03
103	14101	1997-02-03
104	11263	1995-06-18
104	14101	1998-05-08
105	11256	1995-06-18
105	14102	1996-07-08
106	12101	1997-09-10
106	12102	1997-09-30
106	14102	1994-08-23
107	11563	1997-11-05
107	12102	1998-03-10
107	14102	2000-11-11

empno	licno	ctfdate
108	14102	1999-08-09
108	15707	2001-09-08
109	14102	2002-03-14
109	15303	2002-06-06
109	15606	1998-10-23
110	13640	1995-04-18
110	13642	1999-05-29
110	14102	2003-05-09
110	15303	2001-06-04
111	14102	1992-08-05
112	12101	1992-05-09
112	12102	1998-03-02
112	14102	2004-03-16
112	15606	2002-08-23
112	15707	2000-12-06
113	14102	1995-09-30
113	15303	2000-02-14
114	14102	1999-08-14
114	15707	2000-11-25

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 付-4 参考書籍

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.

## 参考書籍

---

- コンピュータはなぜ動くのか(著:矢沢久雄、日経BP)
- 情報はなぜビットなのか(著:矢沢久雄、日経BP)

Copyright UCHIDA HUMAN DEVELOPMENT all rights reserved.