データベース

概要

データベースは膨大な量のデータを蓄えることができ多数のユーザが同時に利用可能なため, 現在の企業情報システムでは欠かすことができないものとなっています.

しかしデータベースも正しい形で蓄えなければ意味がなく, また必要なデータをほしい形で取得したり, 適切に更新することができなければ, その真価を発揮することはできません.

ここでは, オープンソースの RDBM を利用してデータベースの理解を深め, 操作方法（SQL）やデータベース設計などのデータベースを利用した情報システムを構築するための技術を具体的な例や演習を用いて習得します.

キーワード

OSS, リレーショナルデータベース, SQL, 正規化, エンティティ.

目 次

図 目 次

表 目 次

データベース

第10回　代表的なOSS RDBMSとSQLの概要

10.1　OSS RDBMSの種類と特徴

ここ数年商用ソフトウェアに匹敵する性能と品質を持ったオープンソースソフトウェア (以下 OSS) が多くの場面で利用されるようになってきました. RDBMS も例外ではなく OSS の RDBMS が採用されるケースが増えてきています.[1][2][3]

日本国内のユーザにおいて代表格とされる OSS RDBMS と言えば「PostgreSQL」と「MySQL」がまずあげられます. これら以外にもさまざまな RDBMS が存在しており, 両 RDBMS と同じ C/C++ベースでは「Firebird」「Ingres」などがあり, Java ベースでは「Apache Derby」「HSQLDB」などがあります.

「PostgreSQL」「MySQL」「Firebird」「Apache Derby」の歴史・ライセンス・機能・特徴をみてみましょう.

10.1.1　PostgreSQL

歴史

PostgreSQL(ポストグレスエスキューエル：ポストグレス) は, 1970 年後半に米国カリフォルニア大学バークレー校 (UCB) で開発された「Ingres」を起源にもちます.

機能強化と改良が加えられ, 現在の PosgreSQL に至ります. 現在は PostgreSQL Global Development Group が中心となって開発を進めています.

日本での人気は高く, 多くの導入事例があります. 日本での普及と発展を主な目的とした「日本 PostgreSQL ユーザ会 (http://www.postgresql.jp/)」が活動を行っています.[4][5]

ライセンス

BSD ライセンスが採用されています. ソースコードが公開されていて無償で利用でき, 改良した場合はソフトのソースコードを公開することなく, 別の名前で販売可能です.[4]

機能, 特徴

- トランザクション処理に優れている
- 有償の RDBMS に劣らない本格的な機能を備えている
- 日本シェア率が高く, 日本語情報の提供が迅速である
- 開発やリリース自体が速い
- 有償サポートの選択肢が多い

10.1.2　MySQL

歴史

MySQL(マイエスキューエル) は, スウェーデンの MySQL 社により開発され 1995 年に MySQL1.0 が公開されました. 2008 年 MySQL 社がサン・マイクロシステムズ (Sun Microsystems) 社に買収されたことによって, サン・マイクロシステムズ社の所有となっています. 日本では, M y SQL の日本語化の検証や開発を行う「日本 MySQL ユーザ会 (http://www.mysql.gr.jp/)」が活動を行っています.[6]

ライセンス

フリーソフトウェア・オープンソースの GPL(GNU GeneralPublic License) ライセンス, もしくは商用ライセンスのどちらに従うかを選択します.[6]

機能, 特徴

- 高速性と堅牢性に定評がある
- マルチユーザ, マルチスレッドで動作する
- 世界シェアでは群を抜く利用実績 (Web のバックエンド DB としての実績・事例が非常に豊富)

10.1.3 Firebird

歴史

Firebird(ファイアーバード) は, 米 BorlandSoftware が 2000 年にオープンソース化した同社の製品「InterBase6」をベースとして開発されました. Firebird プロジェクトによって, 現在も InterBase 草創期からの開発メンバーや世界中の開発者有志により, 開発・保守・テストが行われています.

国内においても有志「Firebird 日本ユーザー会 (http://tech.firebird.gr.jp/)」によって製品の日本語環境における検証や技術情報の公開が行われています.[7]

ライセンス

OSI 承認の MPL(Mozilla Public License) のバリエーションである IPL(Interbase Public License) と, IDPL(Initial Developer's Public License) が採用されており, いずれも商用・非商用にかかわらず無料で使えます. IPL は InterBase 由来のソースコードに, IDPL は Firebird で新規に起こされたソースコードに適用されます.[7]

機能, 特徴

- メモリ効率に優れ, 軽快に動作する
- インストール・管理が容易である
- 適用形態によることのない 完全なフリーを実現している
- 大規模環境での運用には機能が十分ではない
- 日本語情報や技術者の数は現状では十分とは言えない

10.1.4 Apache Derby

歴史

Apache Derby(アパッチダービー) は, 2004 年 米国 IBM が Apache ソフトウェア財団に寄贈したデータベース「Cloudscape」がベースとなっています. 「Cloudscape」はもともとデータベース・ベンダーの米国クラウドスケープ社が開発したもので, 企業買収を経て IBM の手に渡ったものです.[8]

ライセンス

Apache Licence 2.0
(Apache ソフトウェア財団 (ASF) によるフリーソフトウェア向けライセンス規定です)[8]

機能, 特徴

- 優れた機能性を備え, なおかつ扱いやすい
- PureJavaDB のため, Java アプリケーションとのスムーズな統合が可能である
- XML 対応に強い
- 大規模環境での運用には機能が十分ではない
- 日本語情報や技術者の数は現状では十分とは言えない

10.1.5　まとめ

OSS RDBMS 製品は商用 RDBMS に比べても引けをとらないレベルに達しています.

OSS に限らず一旦稼動し始めた RDBMS を変更するのはなかなか難しいため, それぞれの特徴と, 利用する OS, ミドルウェア, プログラム言語などの環境や, 利用したい機能などを考慮し, よりマッチした RDBMS とそのバージョンを選定する必要があります.

10.2 標準 SQL(SQL92, SQL99)

本来データベースを管理しているのは RDBMS ですが, RDBMS 自体はユーザからの指示に従って管理を行ないます. このとき, ユーザはデータベース言語を使用して RDBMS を制御し, データベースを管理するのです. このデータベース言語が SQL(エスキューエル) です.

簡単にいうと, データベースに対して「こんなデータください」だとか「このデータをいれてね」といった会話をするために SQL を使用します.

SQL は国際標準化機構 (ISO) やアメリカ規格協会 (ANSI), 日本では日本規格協会 (JIS) で標準として規格化されています. 今現在一般的なのは「SQL92」と呼ばれる規格で, ANSI と ISO によって 1992 年に制定された規格です. 別名「SQL2」とも呼ばれています. また SQL の最新版は, 1999 年に規格化された「SQL99」(別名 SQL3) です.

現在のところほとんどの RDBMS は「SQL92」に準拠しています. 紹介した「PostgreSQL」「MySQL」「Firebird」「Apache Derby」も「SQL92」を標準にサポートしており「SQL99」も取り入れられてきています. 個々の RDBMS による独自拡張も多くありますが, 標準的な SQL を理解すれば多くの RDBMS を利用することができます.

10.3　DDL, DML, DCL

SQL は他のプログラミング言語に比べ直感的に理解しやすい構造になっています. 命令は大きく 3 つに分類することができます.

10.3.1　データ定義命令：DDL(Data Definition Language)

テーブルなどの定義を行う命令

例)

CREATE 文 … テーブルや制約条件などを定義する

DROP 文　… テーブルなどを削除する

など

10.3.2　データ操作命令：DML(Data Manipulation Language)

データの抽出や更新を行う命令

例)

SELECT 文 … レコードの抽出を行なう

INSERT 文 … テーブルにレコードを挿入する

DELETE 文 … レコードを削除する

UPDATE 文 … 特定のレコードのフィールドを更新する

など

10.3.3　データ制御命令：DCL(Data Control Language)

データベースの動作を制御する命令

例)

BEGIN 文　… トランザクション処理の開始を宣言する

COMMIT 文　… トランザクションの完了を指示する

ROLLBACK 文 … トランザクションを取り消す

など

10.4 データ型と演算子

SQL92/99 準拠の主なデータ型・演算子を紹介します.
※ただし RDBMS によって若干の差があるので注意が必要です.

10.4.1 データ型

文字列型

CHAR(n)

固定長文字列 (n 文字の文字列を格納できる).

VARCHAR(n)

可変長文字列 (n 文字以内で, 自由な長さの文字列を格納できる).

例)

郵便番号 123-4567 → CHAR(8) …文字 8 文字と決める
住所 山口県宇部市中区○○○ → VARCHAR(50) … 50 文字以内の文字と決める

数値型 (整数型, 実数型)

・整数型は整数を扱う時に指定するデータ型です.

INTEGER

符号付き整数 (-2147483648〜2147483647 までの整数を格納できる)

・実数型は, 小数点を含む数値を扱う時に指定するデータ型です.

NUMERIC(m(,n))

指定された桁数と小数点位置をもつ数値を定義する.

例)

商品コード 123 → INTEGER 整数値と決める
商品価格 12,345 → NUMERIC(18) 18 桁以内の整数値と決める
税率 0.05 → NUMERIC(3,2) 整数部 1 桁, 小数点以下 2 桁と決める

日付時刻型 (日付型, 時刻型)

DATE

日付 (年, 月, 日) を表現するデータ型.
YEAR, MONTH, DAY という三つの整数フィールドの集合.

TIME

時刻 (時, 分, 秒) を表現するデータ型.
HOUR, MINUTE, SECOND という三つの整数フィールドの集合.

TIMESTAMP

DATE と TIME の組み合わせ.

例)

仕入れ日 2010/04/01 → DATE 日付型と決める
仕入れ日時 2010/04/01 15:00:00 → TIMESTAMP 日付時刻型と決める

10.4.2　演算子

演算子とは, 演算内容を指示する記号のことです.
SQL にも他のプログラム言語と同様, 演算子が用意されています.

算術演算子

算術演算子は一般的にもっともよく使われる演算子です.
算術演算子と SQL を組み合わせることで, テーブルに登録されているデータに対して, 計算した結果を取得することができます.

+：加算
-：減算
*：乗算
/：除算
%：剰余

比較演算子

比較演算子は条件比較を行う際に使う演算子です. 値や数値を比較して条件式を評価した結果, 条件が成立した場合「TRUE(真)」成立しない場合は「FALSE(偽)] となります.

=：等しい

<> ， !=：等しくない

> ， >=：より大きい ，以上

< ， <=：より小さい ，以下

BETWEEN〜AND：範囲との比較

IN：値リストとの比較

IS NULL：データが格納されていない (NULL 値) ※

IS NOT NULL：データが格納されている

LIKE：文字列の曖昧検索

※ [NULL(ヌル) と”(空文字）の違い]

Null（ヌル)：何のデータも含まれない状態

’’（空文字)：長さ 0 の文字列

論理演算子

比較演算子を使えば, テーブルからレコードを抽出する選択条件を記述できます. しかし, ただ一つの条件式を指定するだけでは, 単純な条件式しか表現することができません. その様な場合に論理演算子を使って, 複数の条件式を一つの条件式としてまとめることができます.

AND：論理積

OR ：論理和

NOT ：論理否定

演算子の優先順位

以下, 優先順位の高い順

1	* , /	算術演算子
2	+ , -	算術演算子
3	= , >, <, >= , <= , <>, !=	比較演算子
4	IS NULL (IS NOT NULL)	比較演算子
	BETWEEN (NOT BETWEEN)	
	LIKE	
5	NOT	論理演算子
6	AND	論理演算子
7	OR	論理演算子

複数の条件を用いて演算を行う場合, 優先順位に基づいて行われます.
上表の同じ番号に書かれているものに順位の違いはなく, 先に書かれたものが優先されます.
優先順位を意図的に替えたい場合には括弧 ”（ ）” でくくります。
たとえ優先順位どおりであったとしても, 明示的に括弧をつけ計算内容を判り易くする事も必要です.

演算子の例

算術演算子, 比較演算子, 論理演算子を利用した例.

・a は 5 か 10

```
( a = 5 ) OR ( a = 10 )
 a IN ( 5, 10 )
```

・a は 5 以上かつ 10 以下

```
( a >= 5 ) AND ( a <= 10 )
 a BETWEEN 5 AND 10
```

・a に 50 をかけたものが 200 以上 かつ a が 10 未満

```
( a * 50 >= 200 ) AND ( a < 10 )
```

・a は NULL ではなく, かつ b は NULL

```
( a IS NOT NULL ) AND ( b IS NULL )
```

10.5 関数

関数とは, データを使って何らかの処理をし, その結果を返す仕組みのことです.
ここではよく利用する関数の一部を紹介します.
※関数も RDBMS によって若干の差があるので注意が必要です.

10.5.1 数値関数

ABS 関数：値の絶対値を返す
ROUND 関数：値を指定した小数点以下で四捨五入する
TRUNCATE 関数：値を指定した小数点以下で切り捨てる

10.5.2 文字列関数

LTRIM 関数：文字列の先頭のスペースをすべて取り除く
RTRIM 関数：文字列の末尾のスペースをすべて取り除く
SUBSTR 関数：文字列を部分的に切り出す
LENGTH 関数/LEN 関数：文字列の長さを返す

10.5.3 日付関数

DATE 関数：日付を取得する
YEAR 関数/MONTH 関数/DAY 関数：年を取得する/月を取得する/日を取得する
HOUR 関数/MINUTE 関数/SECOND 関数：時を取得する/分を取得する/秒を取得する
DATE_ADD 関数：日付などを加算（減算）する

10.5.4 集約関数

MAX 関数：最大値を取得する
MIN 関数：最小値を取得する
AVG 関数：平均値を取得する
SUM 関数：合計値を取得する
COUNT 関数：カウント数を取得する

10.6　まとめ

次からは実際に RDBMS を準備し, SQL を利用した学習をおこないます.

実際に SQL を実践することで, データ型・演算子・関数についてもより理解することができるでしょう.

第11回　SQL演習

11.1　検索 (SELECT)

表 (テーブル) から登録されているデータを抽出する操作を習得します.

次のような表 (データ入力済) があるとします.

表名：商品マスタ [shohin_master]

	商品コード	商品名	単価	販売終了日
列名	shohin_code	shohin_name	tanka	hanbai_end
データ型	char(4)	varchar(10)	numeric(18)	date
制約項目	NOT NULL	NOT NULL		
	A001	机	35,000	2011/12/31
	A002	椅子	10,000	
	A003	本棚	23,000	
	A004	脚立	10,000	2011/03/31
	B001	ペンケース	900	
	B002	鉛筆	100	

NOT NULL：必ず値が必要

SELECT 文の基本
[構文] SELECT 列名 1, 列名 2, … FROM 表名;

※ SQL 命令の末尾には, ピリオドでなくセミコロン (;) を置く.
※列名に「*」を指定することで全ての列が表示される.
　○ `select * from` 表名;
※ SQL 命令は大文字と小文字を区別しない.(ただし半角英数記号)
　○ `select` 列名 1，列名 2，… `from` 表名;
　○ `SELECT` 列名 1，列名 2，… `FROM` 表名;
　× ＳＥＬＥＣＴ 列名 1，列名 2，… ＦＲＯＭ 表名;（全角の場合）

※途中に改行をいれてもかまわない. SQL 文が長くなると改行をいれるほうが見やすくなる.

○ `SELECT 列名 1, 列名 2, …`
`FROM 表名;`

× `SELECT 列名 1, 列名 2, … FR (SQL 命令語途中の改行)`
`OM 表名;`

[**例題 1**] 商品マスタ表から, すべての行のすべての列を取得する.

「SELECT * FROM shohin_master;」

<結果>

商品コード	商品名	単価	販売終了日
A001	机	35,000	2011/12/31
A002	椅子	10,000	
A003	本棚	23,000	
A004	脚立	10,000	2011/03/31
B001	ペンケース	900	
B002	鉛筆	100	

6 行選択されました.

[**例題 2**] 商品マスタ表から, すべての行の商品名と単価を取得する.

「SELECT shohin_name, tanka FROM shohin_master;」

<結果>

商品名	単価
机	35,000
椅子	10,000
本棚	23,000
脚立	10,000
ペンケース	900
鉛筆	100

6 行選択されました.

[**演習 1**] 商品マスタ表から, すべての行の商品名と販売完了日を取得しなさい.

11.2 条件付き検索

SELECT 文では WHERE 句を用いることで条件を付けてデータを抽出することができます.

表から条件を指定してレコードを抽出する [構文] SELECT 表示する列名 FROM 表名 WHERE 抽出条件;

※ SQL 文の中で抽出条件などに数値や文字列のデータを記述する場合,
数値はそのまま記述し, 文字列はシングル・クォーテーション (') で囲む.

例)

123 → 数値
'123' → 文字列 (数値ではなく'123' という文字列)
' あいう' → 文字列

取得する値や抽出条件に, 算術演算子や比較演算子を利用することでより欲しいデータを取得することができます.

11.2.1 算術演算子

<算術演算子>

+ : 加算
- : 減算
* : 乗算
/ : 除算
% : 剰余

[**例題 3**] 商品マスタ表から, 商品名, 単価, 単価に 1.05 をかけた値を取得する.

-> 取得列 (値) : 商品名，単価，単価に 1.05 をかけた値
対象表 : 商品マスタ
抽出条件 : なし

「SELECT shohin_name, tanka, tanka*1.05 FROM shohin_master;」

<結果>

商品名	単価	単価× 1.05
机	35,000	36,750
椅子	10,000	10,500
本棚	23,000	24,150
脚立	10,000	10,500
ペンケース	900	945
鉛筆	100	105

6 行選択されました.

[**演習 2**] 商品マスタ表から, 商品名, 単価, 単価の 2 割引の値を取得しなさい.

11.2.2 比較演算子

<比較演算子>

= : 等しい

<> , != : 等しくない

> , >= : より大きい , 以上

< , <= : より小さい , 以下

[**例題 4**] 商品マスタ 表から, 単価が 10,000 円の商品名, 単価を取得する

-> 取得列 (値) : 商品名, 単価
対象表 : 商品マスタ
抽出条件 : 単価が 10,000 円

「SELECT shohin_name, tanka FROM shohin_master WHERE tanka = 10000;」

<結果>

商品名	単価
椅子	10,000
脚立	10,000

2 行選択されました.

[**演習 3**] 商品マスタ表から, 単価が 10,000 円以上のものを取得しなさい.

[**演習 4**] 商品マスタ表から, 単価が 10,000 円より大きいものを取得しなさい.

比較演算子 (BETWEEN〜AND：範囲との比較)

ある範囲内のデータを取得したいときに利用します.

`WHERE` 列名 `BETWEEN` ○○ `AND` △△：対象列の値が○○以上△△以下

[**例題 5**] 商品マスタ 表から, 単価が 500 円以上 10,000 円以下の 商品名, 単価を取得する.

-> 取得列 (値)：商品名，単価
　対象表　　：商品マスタ
　抽出条件　：単価が 500 円以上 10,000 円以下

「SELECT shohin_name, tanka FROM shohin_master
　WHERE tanka BETWEEN 500 AND 10000;」

<結果>

商品名	単価
椅子	10,000
脚立	10,000
ペンケース	900

3 行選択されました.

[**演習 5**] 商品マスタ表から, 単価に 1.05 かけた値が 10,000 円以上 30,000 円以下の商品名, 単価, 単価に 1.05 をかけた値を取得しなさい.

-> 取得列 (値)：商品名，単価，単価に 1.05 をかけた値
　対象表　　：商品マスタ
　抽出条件　：単価に 1.05 をかけた値が 10,000 円以上 30,000 円以下

比較演算子 (IN：値リストとの比較)

対象列の値が, 複数の値のうちいずれかに等しいという抽出条件を指定します.
WHERE 列名 IN(条件 1, 条件 2, …)：対象列の値が, 条件 1, 2, …のいずれか

[**例題 6**] 商品マスタ表から, 単価が 10,000 円または 23,000 円または 35,000 円のデータを取得する.

-> 取得列 (値)：すべて
対象表　　：商品マスタ
抽出条件　：単価が 10,000 円または 23,000 円または 35,000 円

「SELECT * FROM shohin_master
WHERE tanka IN (10000,23000,35000);」

<結果>

商品コード	商品名	単価	販売終了日
A001	机	35,000	2011/12/31
A002	椅子	10,000	
A003	本棚	23,000	
A004	脚立	10,000	2011/03/31

4 行選択されました.

[**例題 7**] 商品マスタ表から, 商品コードが B001 または B002 のデータを取得する.

-> 取得列 (値)：すべて
対象表　　：商品マスタ
抽出条件　：商品コードが 'B001' か 'B002'
※文字列の場合は '' でくくる

「SELECT * FROM shohin_master
WHERE shohin_code IN ('B001','B002');」

<結果>

商品コード	商品名	単価	販売終了日
B001	ペンケース	900	
B002	鉛筆	100	

2 行選択されました.

[**演習 6**] 商品マスタ表から, 商品名が’ 机’ または’ 本棚’ または’ 鉛筆’ のデータを取得しなさい.

比較演算子
(IS NULL：データが格納されていない, IS NOT NULL：データが格納されている)

対象列に値が格納されていない (NULL 値) かどうかを条件とします.
※ NULL 値と等しい値はなく, NULL 値に対して大小比較をおこなうこともできません.

○ `WHERE 列名 IS NULL`：対象列に値が格納されていないもの
× `WHERE 列名 = NULL`

[**例題 8**] 商品マスタ表から, 販売終了日が NULL のデータを取得する.

-> 取得列 (値)：すべて
対象表　　：商品マスタ
抽出条件　：販売終了日にデータが格納されていない (NULL)

「SELECT * FROM shohin_master
WHERE hanbai_end IS NULL;」

<結果>

商品コード	商品名	単価	販売終了日
A002	椅子	10,000	
A003	本棚	23,000	
B001	ペンケース	900	
B002	鉛筆	100	

4 行選択されました.

[**演習 7**] 商品マスタ表から, 販売終了日が NULL ではないデータを取得しなさい.

比較演算子 (LIKE：文字列の曖昧検索)

LIKE を利用すると文字列に対してあいまいなキーワードによる検索ができます.
ワイルドカード「%：任意の文字列」「_(アンダーバー) 任意の一文字」を利用します.

○ `WHERE` 比較する列名 `LIKE` 比較条件

LIKE 式			結果
'abc'	LIKE	'abc'	○
'abc'	LIKE	'a%'	○
'abc'	LIKE	'_b_'	○
'abc'	LIKE	'%ab_'	○
'abc'	LIKE	'_ab_'	×
'abc'	LIKE	'c'	×

[**例題 9**] 商品マスタ表から, 商品コードの頭 1 桁 が B のデータを取得する.

-> 取得列 (値)：すべて
　対象表　　：商品マスタ
　抽出条件　：商品コードの頭 1 桁が'B'

「SELECT * FROM shohin_master
　WHERE shohin_code LIKE 'B%';」

<結果>

商品コード	商品名	単価	販売終了日
B001	ペンケース	900	
B002	鉛筆	100	

2 行選択されました.

[**演習 8**] 商品マスタ表から, 商品コードに'2' を含むデータを取得しなさい.

11.2.3 論理演算子

論理演算子には AND と OR と NOT があります.

AND：論理積

OR ：論理和

NOT ：論理否定

論理演算子を利用し, 複数の条件式を組み合わせる事で複雑な条件を設定できます.

※複数の抽出条件がある場合

WHERE ○○ AND △△ AND … ← ○○ かつ △△ かつ …

WHERE ○○ OR △△ OR … ← ○○ または △△ または …

[**例題 10**] 商品マスタ表から, 以下の条件の商品名と単価を取得する.

単価が 10,000 円以下で販売終了日が NULL でないもの

-> 取得列 (値)：商品名, 単価

対象表 ：商品マスタ

抽出条件 ：単価が 10,000 円以下 かつ 販売終了日が NULL でないもの

「SELECT shohin_name,tanka FROM shohin_master

WHERE tanka <= 10000

AND hanbai_end IS NOT NULL;」

<結果>

商品名	単価
脚立	10,000

1 行選択されました.

[**演習 9**] 商品マスタ表から, 以下の条件の商品名, 単価, 単価× 1.05, 販売終了日を取得しなさい.

商品コードの頭 1 桁が'A'

単価に 1.05 をかけた値が 10,000 円以上

販売終了日が NULL でないもの

11.3　検索結果のソート

問い合わせた結果を並び替えて (ソート) 表示するには, ORDER BY 句を利用します.

```
[ 構文 ] SELECT 表示する列名 FROM 表名
        WHERE 抽出条件
        ORDER BY 並べ替えキー列 1 ASC[ DESC ],
            並べ替えキー列 2 ASC[ DESC ],
            …
```

ORDER BY 句には優先順位の高い列から記述します.

・小さいものから大きいもの「昇順」ASC (省略の場合 ASC になる)

・大きいものから小さいもの「降順」DESC

[**例題 11**] 商品マスタ表から, 単価を昇順, 商品コードを昇順にならべて表示する.

-> 取得列 (値)：すべて
　対象表　　：商品マスタ
　並び替え　：単価の昇順，商品コードの昇順

「SELECT * FROM shohin_master
　ORDER BY tanka ASC, shohin_code ASC;」

<結果>

商品コード	商品名	単価	販売終了日
B002	鉛筆	100	
B001	ペンケース	900	
A002	椅子	10,000	
A004	脚立	10,000	2011/03/31
A003	本棚	23,000	
A001	机	35,000	2011/12/31

6 行選択されました.

[**演習 10**] 商品マスタ表から, 商品名を降順にならべて取得しなさい.

11.4 集約関数とグループ化

次のような表 (データ入力済) があるとします.

表名：英語成績 [eigo_seiseki]

	学生番号	学生名	クラス	点数
列名	gakusei_no	gakusei_name	class_no	tensu
データ型	char(4)	varchar(20)	char(1)	integer
制約項目	NOT NULL	NOT NULL	NOT NULL	
	Z001	学生 A	1	70
	Z002	学生 B	1	98
	Z003	学生 C	1	50
	Z004	学生 D	2	100
	Z005	学生 E	2	80
	Z006	学生 F	3	65
	Z007	学生 G	3	69
	Z008	学生 H	1	45
	Z009	学生 I	2	60
	Z010	学生 J	3	55
	Z011	学生 K	2	90
	Z012	学生 L	3	70

11.4.1　集約関数

集約関数とは, 一つの列グループに対して施すことのできる演算機能をいいます. 集約関数は列グループ全体につき一つの値を生成します. 集約関数には次の五つがあります.

<集約関数>

MAX(引数)：引数 (列名など) の最大値を取得する.
MIN(引数)：引数 (列名など) の最小値を取得する.
AVG(引数)：引数 (列名など) の平均値を取得する.
SUM(引数)：引数 (列名など) の合計値を取得する.
COUNT(引数)：引数のカウント数を取得する.　引数は「列名」または「*」.
　　引数が列名の場合は NULL 値の行を除くすべての行が対象
　　引数が「*」の場合は NULL 値を含むすべての行が対象

[**例題 12**] 英語成績表から, 最高点の点数を取得する.

-> 取得列 (値)：最高点の点数
　対象表　　：英語成績
　抽出条件　：なし
　並び替え　：なし

「SELECT MAX(tensu) FROM eigo_seiseki; 」

<結果>

MAX(点数)
100

1 行選択されました.

[**例題 13**] 英語成績表の件数を取得する.

-> 取得列 (値) : 行の件数 (列指定なし)
対象表 : 英語成績
抽出条件 : なし
並び替え : なし

「SELECT COUNT(*) FROM eigo_seiseki;」

<結果>

COUNT(*)
12

1 行選択されました.

[**演習 11**] 英語成績表から, 最低点を取得しなさい.
[**演習 12**] 英語成績表から, 平均点を取得しなさい.
[**演習 13**] 英語成績表から, 点数をすべて足したもの (合計点) を取得しなさい.

11.4.2　グループ化

SQL におけるグループ化とは, 同列内の値の中で, 同じ値を持つデータごとに集合化することをいいます. グループ化には GROUP BY 句を用います.

[構文] SELECT 表示する列名 FROM 表名
　　WHERE 抽出条件
　　GROUP BY グルーピングする列名;

[**例題 14**] 英語成績表に存在しているクラスをグループ化で取得する.

-> 取得列 (値)：クラス
　対象表　　：英語成績
　抽出条件　：なし
　グループ化 ：クラス毎
　並び替え　：なし

「SELECT class_no FROM eigo_seiseki GROUP BY class_no;」

<結果>

クラス
2
3
1

3 行選択されました.

[**例題 15**] 英語成績表にある点数が 50 点以下の学生が所属するクラスをクラスのグループ化で取得する.

-> 取得列 (値)：クラス
対象表　　：英語成績
抽出条件　：点数が 50 点以下
グループ化 ：クラス毎
並び替え　：なし

「SELECT class_no FROM eigo_seiseki WHERE tensu <= 50 GROUP BY class_no;」

<結果>

クラス
1

1 行選択されました.

実際には 50 点以下の学生はクラス「1」に 2 名いますが, 欲しい情報は「誰が」ではなく「どのクラスか」なので, クラスでグループ化することで一目でわかります.

11.4.3　集約関数とグループ化

数値データを扱う際には, 集約関数とグループ化を用いることでより目的に合った検索を実行することができます.

[**例題 16**] 英語成績表から, クラス毎の人数, 最高得点を取得し, クラスの昇順に並べる.

-> 取得列 (値)：クラス，クラスの人数 (カウント)，クラスの最高得点
　対象表　　：英語成績
　抽出条件　：なし
　グループ化 ：クラス毎
　並び替え　：クラスの昇順

「SELECT class_no, count(gakusei_no), MAX(tensu) FROM eigo_seiseki
GROUP BY class_no ORDER BY class_no;」

<結果>

クラス	COUNT(学生番号)	MAX(点数)
1	4	98
2	4	100
3	4	70

3 行選択されました.

[**例題 17**] 英語成績表から, 点数が 80 点以上のクラス毎の人数を取得し, クラスの昇順に並べる.

-> 取得列 (値)：クラス，クラスの人数 (カウント)
対象表　　：英語成績
抽出条件　：点数が 80 点以上
グループ化 ：クラス毎
並び替え　：クラスの昇順

「SELECT class_no, count(gakusei_no) FROM eigo_seiseki
WHERE tensu >= 80
GROUP BY class_no ORDER BY class_no ;」

<結果>

クラス	人数
1	1
2	3

2 行選択されました.

[**演習 14**] 英語成績表から, クラス毎の平均点を取得し, クラスの昇順に並べなさい.

[**演習 15**] 英語成績表から, クラス毎の人数, 最低点, 最高点, 平均点を取得し, クラスの昇順に並べなさい.

～ここまでの基本 SELECT 構文のまとめ～

```
[ 構文 ] SELECT 列 1, 列 2, …      ← 複数の場合はカンマでつなぐ
         FROM 表名                 ← 対象表
         WHERE 抽出条件 1          ← 複数の場合は AND または OR でつなぐ
           AND 抽出条件 2
           …
         GROUP BY グルーピングする列 1,     ← 複数の場合はカンマでつなぐ
           グルーピングする列 2,
           …
         ORDER BY 並べ替えキー列 1 ASC[ DESC ],     ← 複数の場合はカンマでつなぐ
           並べ替えキー列 2 ASC[ DESC ],            (優先順位の高いものから記述)
           …
```

SELECT 文の組み立てのポイントは … まず以下を考えます.

- 取得する列 (値) は？ (SELECT)
- 対象となる表は？ (FROM 句)
- 抽出条件は？ (WHERE 句)
- グループ化する基準は？ (GROUP BY)
- どの並びで表示するか？ (ORDER BY)

その中でさらに, 目的にあった比較演算子や算術演算子, 集約関数などを組み合わせていきます.

11.5 内部結合と外部結合

いままでは一つの表からデータを抽出する方法について学習しました. ここでは, 複数の表の間に定義されたリレーションシップをもとに, 複数の表を結合してデータを引き出す方法を学習します.

11.5.1 内部結合

結合元と結合先で一致したデータだけが結合でき, 結合できなかったデータは結合結果に反映されません.

<内部結合の例>

表: 商品マスタ [shohin_master]

商品コード	商品名	単価	販売終了日
shohin_code	shohin_name	tanka	hanbai_end
char(4)	varchar(20)	integer	date
A001	机	35,000	2011/12/31
A002	椅子	10,000	
A003	本棚	23,000	
A004	脚立	10,000	2011/3/31
B001	ペンケース	900	
B002	鉛筆	100	

表: 受注 [juchu]

受注NO	受注日	商品コード	数量
juchu_no	juchu_date	shohin_code	suryo
char(12)	date	char(4)	integer
201001100001	2010/1/10	A002	2
201001200001	2010/1/20	A003	1
201001210001	2010/1/21	B001	3
201002210001	2010/2/21	B002	12
201002210001	2010/2/21	A002	1
201002220001	2010/2/22	A005	1

shohin_master.shohin_code = juchu.shohin_code

内部結合の場合

商品マスタ. 商品コード	商品マスタ. 商品名	受注. 受注日	受注. 数量
A002	椅子	2010/1/10	2
A003	本棚	2010/1/20	1
B001	ペンケース	2010/1/21	3
B002	鉛筆	2010/2/21	12
A002	椅子	2010/2/21	1

商品マスタと受注の
どちらにもあるものが抽出

表の結合をおこなう場合は, FROM 句で INNER JOIN を指定します.

[構文] SELECT 列名 1, 列名 2, …列名 n ← どの列を表示させるか
FROM 表名 1
INNER JOIN 表名 2
ON 表名 1. 列名 = 表名 2. 列名; ← 表の結合条件

どの表に属する列であるかを明示する場合は, 列名に「表名. 列名」と指定します.

FROM 句で INNER JOIN を指定するほかに　WHERE 句で結合条件を指定することもできます.

[構文] SELECT 列名 1, 列名 2, …列名 n ← どの列を表示させるか
FROM 表名 1, 表名 2
WHERE 表名 1. 列名 = 表名 2. 列名; ← 表の結合条件

[**例題 18**] 商品マスタ表と受注表を商品コードで結合させた内部結合で, 商品マスタ表の商品コードと商品名, 受注表の受注日と 数量を取得する.(説明図)

-> 取得列 (値)：商品マスタ表の商品コード，商品名，受注表の受注日，数量
対象表　　：商品マスタ，受注 (内部結合)
結合条件　：商品マスタ表の商品コードと受注表の商品コード
抽出条件　：なし

INNER JOIN を指定した場合

```
「SELECT shohin_master.shohin_code,
        shohin_master.shohin_name,
        juchu.juchu_date,
        juchu.suryo
 FROM shohin_master
 INNER JOIN juchu
 ON shohin_master.shohin_code = juchu.shohin_code;」
```

WHERE 句を利用した場合

```
「SELECT shohin_master.shohin_code,
        shohin_master.shohin_name,
        juchu.juchu_date,
        juchu.suryo
 FROM shohin_master,juchu
 WHERE shohin_master.shohin_code = juchu.shohin_code;」
```

[**例題 19**] 商品マスタ表と受注表からどの商品 (商品マスタ. 商品コード, 商品マスタ. 商品名) がどのくらい受注があったか受注のあった商品に対して, 商品コード毎の受注. 数量の合計を取得する.

-> 取得列 (値)：商品マスタ表の商品コード，商品名，商品の受注数 (合計値)
対象表　　：商品マスタ，受注 (内部結合)
結合条件　：商品マスタ表の商品コードと受注表の商品コード
抽出条件　：なし
グループ化 ：商品毎

```
「SELECT shohin_master.shohin_code,
        shohin_master.shohin_name,
        SUM(juchu.suryo)
 FROM shohin_master
 INNER JOIN juchu
 ON shohin_master.shohin_code = juchu.shohin_code
 GROUP BY shohin_master.shohin_code, shohin_master.shohin_name;」
```

<結果>

商品コード	商品名	受注の合計
B002	鉛筆	12
B001	ペンケース	3
A002	椅子	3
A003	本棚	1

4 行選択されました.

11.5.2　外部結合

結合先に一致するデータがなくても結合結果に反映させます．何を結合元にするかによって, 左結合と右結合, 完全外部結合があります．

表：商品マスタ［ shohin master ］

商品コード	商品名	単価	販売終了日
shohin_code	shohin_name	tanka	hanbai_end
char(4)	varchar(20)	integer	date
A001	机	35,000	2011/12/31
A002	椅子	10,000	
A003	本棚	23,000	
A004	脚立	10,000	2011/3/31
B001	ペンケース	900	
B002	鉛筆	100	

表：受注［ juchu ］

受注NO	受注日	商品コード	数量
juchu_no	juchu_date	shohin_code	suryo
char(12)	date	char(4)	integer
201001100001	2010/1/10	A002	2
201001200001	2010/1/20	A003	1
201001210001	2010/1/21	B001	3
201002210001	2010/2/21	B002	12
201002210001	2010/2/21	A002	1
201002220001	2010/2/22	A005	1

shohin_master.shohin_code = juchu.shohin_code

【左外部結合の場合】

商品マスタ. 商品コード	商品マスタ. 商品名	商品マスタ. 単価	受注. 受注日	受注. 数量
A001	机	35,000	NULL	NULL
A002	椅子	10,000	2010/1/10	2
A002	椅子	10,000	2010/2/21	1
A003	本棚	23,000	2010/1/20	1
A004	脚立	10,000	NULL	NULL
B001	ペンケース	900	2010/1/21	3
B002	鉛筆	100	2010/2/21	12

商品マスタにあるが
受注にはない行も抽出

【右外部結合の場合】

商品マスタ. 商品名	商品マスタ. 単価	受注. 商品コード	受注. 受注日	受注. 数量
椅子	10,000	A002	2010/1/10	2
本棚	23,000	A003	2010/1/20	1
ペンケース	900	B001	2010/1/21	3
鉛筆	100	B002	2010/2/21	12
椅子	10,000	A002	2010/2/21	1
NULL	NULL	A005	2010/2/22	1

受注にあるが
商品マスタにはない行も抽出

【完全外部結合の場合】

商品マスタ. 商品コード	商品マスタ. 商品名	商品マスタ. 単価	受注. 商品コード	受注. 受注日	受注. 数量
A001	机	35,000	NULL	NULL	NULL
A002	椅子	10,000	A002	2010/1/10	2
A002	椅子	10,000	A002	2010/2/21	1
A003	本棚	23,000	A003	2010/1/20	1
A004	脚立	10,000	NULL	NULL	NULL
NULL	NULL	NULL	A005	2010/2/22	1
B001	ペンケース	900	B001	2010/1/21	3
B002	鉛筆	100	B002	2010/2/21	12

片方になくても
どちらかにあれば抽出

<左外部結合>

[構文] SELECT 列名 1, 列名 2, …列名 n
FROM 表名 1
LEFT OUTER JOIN 表名 2
ON 表名 1 列名 = 表 2. 列名;

[**例題 20**] 商品マスタ表と受注表を商品コードで結合させた左外部結合で, 商品マスタ表の商品コードと商品名と単価, 受注表の受注日と数量を取得する.(説明図)

-> 取得列 (値)：商品マスタ表の商品コード，商品名，単価，受注表の受注日，数量
対象表　　：商品マスタ，受注 (外部結合)
結合条件　：商品マスタ表の商品コードと受注表の商品コード
抽出条件　：なし

「SELECT shohin_master.shohin_code,
shohin_master.shohin_name,
shohin_master.tanka,
juchu.juchu_date,
juchu.suryo
FROM shohin_master
LEFT OUTER JOIN juchu
ON shohin_master.shohin_code = juchu.shohin_code;」

<右外部結合>

[構文] SELECT 列名 1, 列名 2, …列名 n
　　FROM 表名 1
　　RIGHT OUTER JOIN　表名 2
　　ON 表名 1. 列名 = 表名 2. 列名;

[**例題 21**] 商品マスタ表と受注表を商品コードで結合させた右外部結合で, 商品マスタ表の商品名と単価, 受注表の商品コードと受注日と数量を取得する.(説明図)

-> 取得列 (値)：商品マスタ表の商品名，単価，受注表の商品コード，受注日，数量
　対象表　　：商品マスタ (外部結合)，受注
　結合条件　：商品マスタ表の商品コードと受注表の商品コード
　抽出条件　：なし

「SELECT shohin_master.shohin_name,
　　shohin_master.tanka,
　　juchu.shohin_code,
　　juchu.juchu_date,
　　juchu.suryo
　FROM shohin_master
　RIGHT OUTER JOIN juchu
　ON shohin_master.shohin_code = juchu.shohin_code;」

<完全外部結合>

[構文] SELECT 列名 1, 列名 2, …列名 n
FROM 表名 1
FULL OUTER JOIN 表名 2
ON 表名 1. 列名 = 表名 2. 列名;

※完全外部結合は処理に負担がかかります

[**例題 22**] 商品マスタ表と受注表を商品コードで結合させた完全外部結合で, 商品マスタ表の商品コードと商品名と単価, 受注表の商品コードと受注日と数量を取得する.(説明図)

-> 取得列 (値)：商品マスタ表の商品コード，商品名，単価，
受注表の商品コード，受注日，数量
対象表 ：商品マスタ，受注（完全外部結合）
結合条件 ：商品マスタ表の商品コードと受注表の商品コード
抽出条件 ：なし

「SELECT shohin_master.shohin_code,
shohin_master.shohin_name,
shohin_master.tanka,
juchu.shohin_code,
juchu.juchu_date,
juchu.suryo
FROM shohin_master
FULL OUTER JOIN juchu
ON shohin_master.shohin_code = juchu.shohin_code; 」

[**例題 23**] 商品マスタ表にあるすべての商品に対して, どの商品 (商品マスタ. 商品コード, 商品マスタ. 商品名) がどのくらい受注があったか (商品コード毎の受注. 数量の合計) を受注表より取得し, 商品コードの昇順で並べなさい.

-> 取得列 (値)：商品マスタ表の商品コード，商品名，商品の受注数 (合計値)
　対象表　　：商品マスタ，受注 (外部結合)
　結合条件　：商品マスタ表の商品コードと受注表の商品コード
　抽出条件　：なし
　グループ化：商品毎
　並び替え　：商品コードの昇順

```
「SELECT shohin_master.shohin_code,
        shohin_master.shohin_name,
        SUM(juchu.suryo)
  FROM shohin_master
  LEFT OUTER JOIN juchu
  ON shohin_master.shohin_code = juchu.shohin_code
  GROUP BY shohin_master.shohin_code, shohin_master.shohin_name
  ORDER BY shohin_master.shohin_code; 」
```

<結果>

商品コード	商品名	受注の合計
A001	机	
A002	椅子	3
A003	本棚	1
A004	脚立	
B001	ペンケース	3
B002	鉛筆	12

6 行選択されました.

(演習) 次のような表があるとします.

「取得資格」表…学生が取得している資格一覧を格納
「資格マスタ」表…資格コードと資格名を管理しているマスタ

表:取得資格 [shutoku_shikaku]

学生番号	資格コード
gakusei_no	shikaku_code
char(4)	char(4)
Z002	EC01
Z002	EK01
Z004	EC01
Z004	EK01
Z004	EK02
Z005	EC01
Z009	JS01
Z011	EC01
Z011	EK01
Z012	NC01

主キー:学生番号,資格コード

表:資格マスタ [shikaku_master]

資格コード	資格名
shikaku_code	shikaku_name
char(4)	varchar(10)
EC01	英語検定Ⅰ
EK01	英会話Ⅰ
EK02	英会話Ⅱ
JS01	情報処理Z
KC01	漢字検定A
NC01	日本語検定A

主キー:資格コード

[**演習 16**] 取得資格表と資格マスタ表を資格コードで内部結合し, 学生番号・資格コード・資格名を学生番号の昇順で取得しなさい.

<ヒント:検索結果>

学生番号	資格コード	資格名
Z002	EC01	英語検定 I
Z002	EK01	英会話 I
Z004	EC01	英語検定 I
Z004	EK01	英会話 I
Z004	EC02	英会話 II
Z005	EC01	英語検定 I
Z009	JS01	情報処理 Z
Z011	EC01	英語検定 I
Z011	EK01	英会話 I
Z012	NC01	日本語検定 A

10 行選択されました.

[**演習 17**] 資格マスタ表に存在する資格を元に, 取得資格表を外部結合し資格コード・資格名・学生番号を, 資格コードの昇順・学生番号の昇順で取得しなさい.

<ヒント：検索結果>

資格コード	資格名	学生番号
EC01	英語検定 I	Z002
EC01	英語検定 I	Z004
EC01	英語検定 I	Z005
EC01	英語検定 I	Z011
EK01	英会話 I	Z002
EK01	英会話 I	Z004
EK01	英会話 I	Z011
EK02	英会話 II	Z004
JS01	情報処理 Z	Z009
KC01	漢字検定 A	
NC01	日本語検定 A	Z012

11 行選択されました.

11.6 副問い合わせ

SQL 文の中に SELECT 文を入れ子にして埋め込むのが「副問い合わせ (サブクエリ)」です.

11.6.1 単一行副問い合わせ

結果として 1 行を戻す副問い合わせは「単一行副問い合わせ」と呼ばれます. 単一行副問い合わせでは, 単一行比較演算子 (=, <, <=, >, >=, !=) を使用することができます.

```
[ 構文 ] SELECT 列名 FROM 表名
        WHERE 列名 = (SELECT 列名 FROM 表名
                      WHERE 抽出条件);
```

[**例題 24**] 英語成績表から, 点数が最高点の行を取得する.

-> 取得列 (値) : すべて
対象表 : 英語成績
抽出条件 : 点数が最高点

「SELECT * FROM eigo_seiseki
WHERE tensu = (SELECT MAX(tensu) FROM eigo_seiseki);」

<結果>

学生番号	学生名	クラス	点数
Z004	学生 D	2	100

1 行選択されました.

[**演習 18**] 英語成績表から, 点数が平均点より高い行を取得しなさい.

11.6.2　複数行副問い合わせ

結果として複数行を戻す副問い合わせは「複数行副問い合わせ」と呼ばれます. 複数行副問い合わせでは, 複数行比較演算子 (IN) を使用します.

[構文] SELECT 列名 FROM 表名
　　WHERE 列名 IN (SELECT 列名 FROM 表名
　　WHERE 抽出条件);

[**例題 25**] 資格コード'EC01'(英語検定 I) を取得している学生 (取得資格表より) の英語成績を取得し, 学生番号の昇順にならべる.

-> 取得列 (値)：すべて
　対象表　　：英語成績
　抽出条件　：取得資格表より資格'EC01' を取得している学生
　並び替え　：学生番号の昇順

「SELECT * FROM eigo_seiseki
　WHERE gakusei_no IN (SELECT gakusei_no FROM shutoku_shikaku
　　WHERE shikaku_code='EC01')
　ORDER BY gakusei_no;」

<結果>

学生番号	学生名	クラス	点数
Z002	学生 B	1	98
Z004	学生 D	2	100
Z005	学生 E	2	80
Z011	学生 K	2	90

4 行選択されました.

[**演習 19**] 英語成績表より点数が 90 点以上の学生の学生番号と資格コードと資格名を, 取得資格表・資格マスタ表から取得しなさい.

-> 内部結合と副問い合わせを利用

<ヒント：検索結果>

学生番号	資格コード	資格名
Z002	EC01	英語検定 I
Z002	EK01	英会話 I
Z004	EC01	英語検定 I
Z004	EK01	英会話 I
Z004	EK02	英会話 II
Z011	EC01	英語検定 I
Z011	EK01	英会話 I

7 行選択されました.

11.7　挿入, 更新, 削除

ここまではデータのさまざまな抽出 (SELECT) の記述について学んできました. ここからはデータ挿入・更新・削除について学んでいきます. これらの操作は, 行単位で行われます。

11.7.1　挿入 (INSERT)

INSERT は, 表に行を追加するときに使用します.

[構文] INSERT INTO 表名 (列 A, 列 B, … 列 n) ← 追加対象の表名と列名の指定
VALUES (値 A, 値 B, … 値 n) ; ← 追加する行の各列の値

※ INTO で指定された表の列のデータ型と
VALUES で指定したデータのデータ型が一致していなくてはならない.

[**例題 26**] 商品マスタ表に, 以下のデータを追加します.

商品コード：C001
商品名　　：消しゴム
単価　　：50
販売終了日：NULL

「INSERT INTO shohin_master(shohin_code, shohin_name, tanka, hanbai_end)
VALUES('C001', ' 消しゴム', 50, NULL);」

SELECT 文で行が追加されているかどうかを確認します.
「SELECT * FROM shohin_master WHERE shohin_code = 'C001';」

[**演習 20**] 商品マスタ表に, 以下のデータを追加しなさい.
その後 SELECT 文で行が追加されていることを確認しなさい.

商品コード：学生番号の右 4 桁
商品名　　：ノート + 自分の名前の苗字
単価　　：200
販売終了日：2012/12/31

11.7.2 更新 (UPDATE)

UPDATE は, 行を更新するときに使用します.

[構文] UPDATE 表名	← どの表のデータの
SET 列名 = 値	← 更新させたい列と更新内容
WHERE 抽出条件;	← どのような条件のデータを

※複数行が該当する抽出条件を指定すれば, 複数行の列を一括更新できます.
※抽出条件を指定しない場合, 全行が対象となり全行一括更新されます.

[**例題 27**] 商品マスタ表の, 商品コード'C001' の単価を 60 円に更新する.

「UPDATE shohin_master
SET tanka = 60
WHERE shohin_code = 'C001';」

SELECT 文で値が更新されているかどうかを確認します.
「SELECT * FROM shohin_master WHERE shohin_code= 'C001';」

[**演習 21**] 商品マスタ表の, 商品コード "学生番号の右 4 桁" の
商品名を ' コンパス + 自分の名前の苗字' に更新しなさい.
その後 SELECT 文で行が更新されていることを確認しなさい.

11.7.3　削除 (DELETE)

DELETE は, 表から指定した行を削除するときに使用します.

[構文] DELETE FROM 表名 ← どの表のデータの
　　WHERE 抽出条件 ; ← どのような条件のデータを

※更新と同様に, 削除の場合も抽出条件に一致する複数の行をまとめて削除できます.
※抽出条件を指定しない場合, 全行が対象となり全行削除されます.

[**例題 28**] 商品マスタ表より, 商品コード'C001' の行を削除する.

「DELETE FROM shohin_master
WHERE shohin_code = 'C001';」

SELECT 文で確認します.

「SELECT * FROM shohin_master WHERE shohin_code= 'C001';」

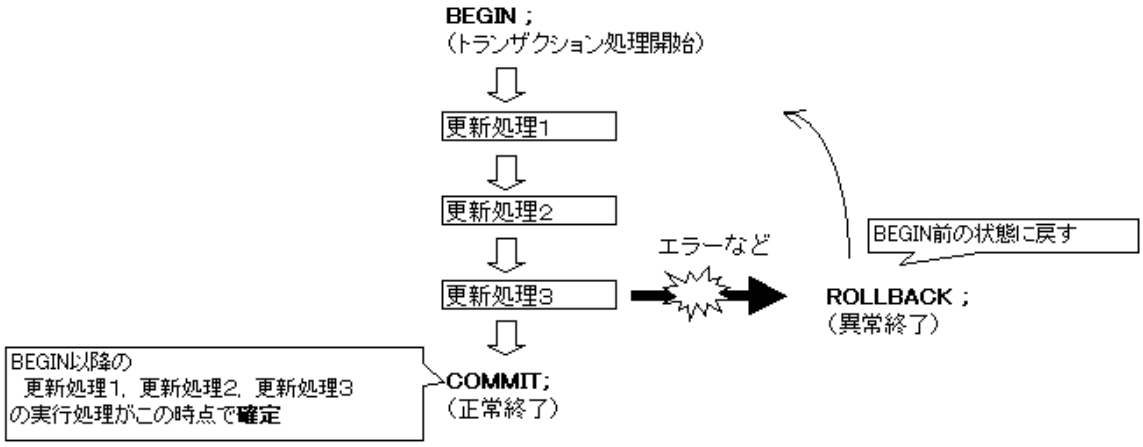
[**演習 22**] 商品マスタ表より, 商品コード"学生番号の右 4 桁" の行を削除しなさい.
その後 SELECT 文で行が削除されていることを確認しなさい.

11.8 トランザクション管理

データベースは, 多くの人が同時にアクセスし閲覧や更新をします. トランザクションは, 一連のデータベースに対する複数の操作 (抽出, 更新, 削除など) を一つの処理単位とすることで, 複数のユーザーが同時にデータベースを操作する状況において, データの整合性を確保するため, またデータの障害復旧といった目的に利用します.

トランザクションによるデータの変更処理が正常に終了した場合に, その変更処理を有効な結果と確定し, データベースに反映することをコミット (COMMIT) といいます. また, トランザクションによるデータの変更処理の途中で何らかの障害が発生した場合に, それまでの変更処理を無効なものとし, トランザクションが実行される前の状態にもどすことをロールバック (ROLLBACK) といいます.

これらは, データ制御命令：DCL(Data Control Language) の一部です.



※ PostgreSQL では「BEGIN」コマンドを利用しなければトランザクションは開始せず INSERT・UPDATE・DELETE が実行された時点で自動的にコミットが実行される.

トランザクション管理
[構文] BEGIN; ← トランザクションの開始
…(INSERT, UPDATE, DELETE などの更新処理)
COMMIT; ← データベース反映の確定
ROLLBACK; ← データベース反映の取消

[**例題 29**]「商品マスタ表の, 商品コードが'A001' の単価を 33300 円に更新する」という処理を, トランザクションを明示的に宣言 (BEGIN) し, COMMIT または ROLLBACK で処理が確定された場合と取り消された場合を確認する.

→ 処理取消の場合
「BEGIN;
UPDATE shohin_master SET tanka = 33300 WHERE shohin_code = 'A001';
ROLLBACK; 」

SELECT 文で単価が変更されていない事を確認
「SELECT * FROM shohin_master WHERE shohin_code = 'A001';」

→ 処理確定の場合
「BEGIN;
UPDATE shohin_master SET tanka = 33300 WHERE shohin_code = 'A001';
COMMIT;」

SELECT 文で単価が変更されている事を確認
「SELECT * FROM shohin_master WHERE shohin_code = 'A001';」

[**演習 23**]「商品マスタ表の, 商品コードが'B001' の単価を 500 円に更新する」という処理と「商品マスタ表の, 商品コードが'B002' の販売終了日を'2015/12/31' に更新する」という二つの処理を, トランザクションを明示的に宣言 (BEGIN) し, 二つの処理の後に COMMIT または ROLLBACK で二つの処理が確定された場合と二つとも処理が取り消された場合を確認しなさい.

11.9 表の定義と削除

データ定義命令：DDL(Data Definition Language)L で用いられる主な命令には, 新しく表などを定義 (作成) する「CREATE」や, 既存の表などを削除する「DROP」, 既存の表などに変更を加える「ALTER」などがあります.

11.9.1 表の定義 (CREATE TABLE)

データベースに空の表を定義するには CREATE TABLE を利用します.
表名を何にするか, どのような値を格納するのか (列名とそのデータの型など) や制約項目 (主キー,NOT NULL など）を定義する必要があります.

PostgreSQL では主キーに名前をつけて CREATE 文内に記述します.
※主キー：列のデータが重複せず NULL 値ではないことを保証するもので,
テーブル内の行を一意に識別できるデータを指す.

表の定義
[構文] CREATE TABLE 表名
(列名 1 データ型,
列名 2 データ型,
…,
CONSTRAINT 主キー名 PRIMARY KEY(主キーの列名))
);

[**例題 30**]

次のような表を作るとします.

表名：商品マスタ_学生番号下 4 桁 [shohin_master_XXXX]

	商品コード	商品名	単価	販売終了日
列名	shohin_code	shohin_name	tanka	hanbai_end
データ型	char(4)	varchar(10)	numeric(18)	date
制約項目	NOT NULL	NOT NULL		
主キー	○			

NOT NULL：必ず値が必要

主キー：データが一意となる項目

```
CREATE TABLE shohin_master_6789 (
        shohin_code char(4) NOT NULL,
        shohin_name varchar(20) NOT NULL,
        tanka numeric(18),
        hanbai_end date,
    CONSTRAINT shohin_master_6789_pkey PRIMARY KEY (
        shohin_code
    )
)
```

テーブルが作成されたら, 中身を SELECT 文で確認しながら, INSERT 文を作成してテーブル内にデータを追加したり DELETE 文を作成してデータを削除したりしてみましょう.

11.9.2 表の削除 (DROP TABLE)

データベースから表を削除するにはDROP TABLEを利用します.

表の削除
[構文] DROP TABLE 表名;

[**例題 31**] 先ほど作成した「商品マスタ_”学生番号下4桁”」表を削除します.

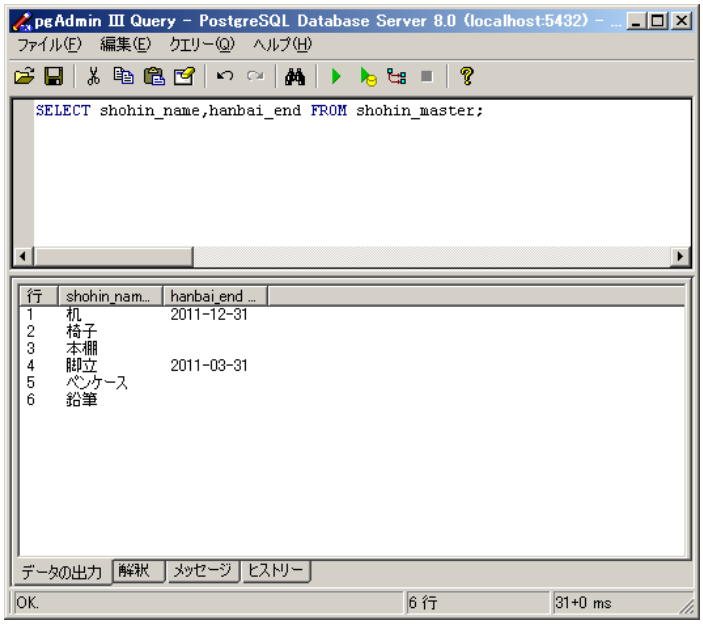
「DROP TABLE shohin_master_6789;」

SELECT文で確認してみましょう. 表がないためSELECT文はエラーとなります.

11.10　演習問題の解答例

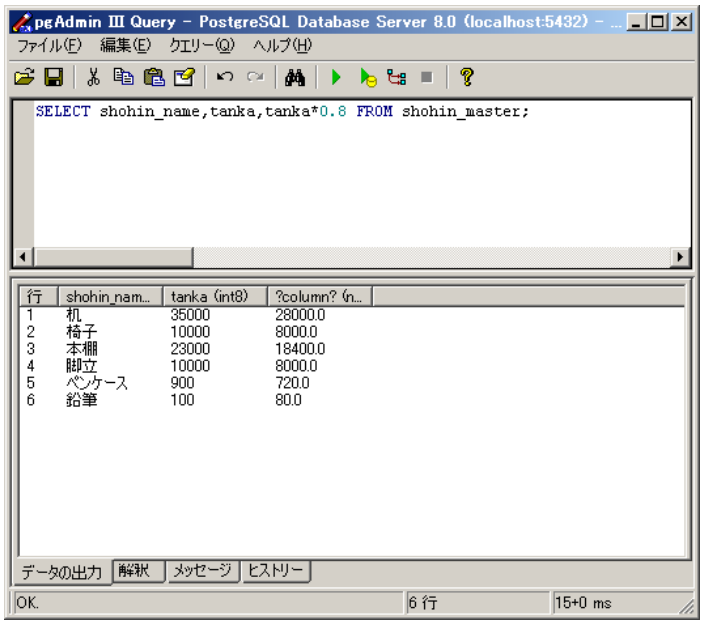
[演習 1]

SELECT shohin_name, hanbai_end FROM shohin_master;



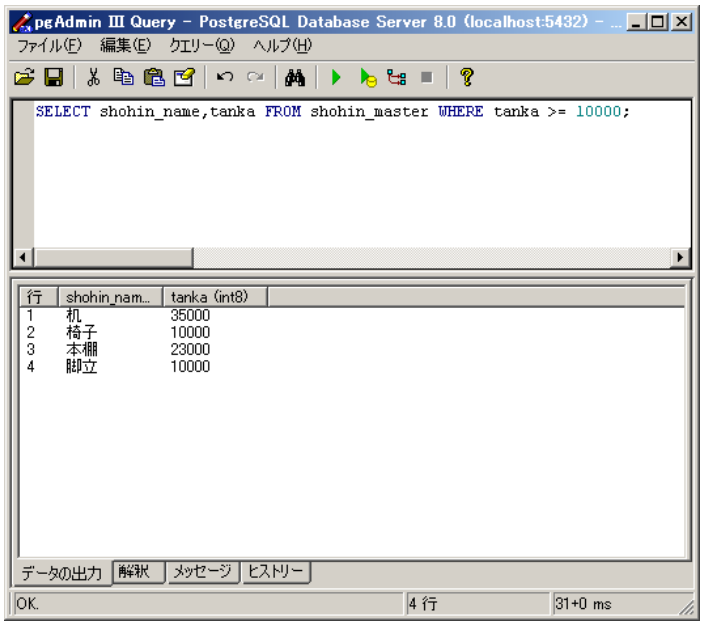
[演習 2]

SELECT shohin_name, tanka, tanka*0.8 FROM shohin_master;



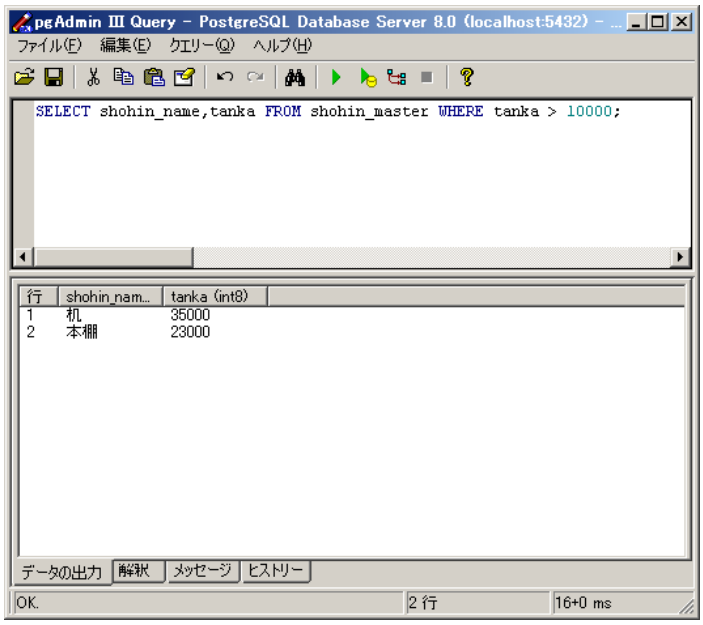
[演習 3]

SELECT shohin_name, tanka FROM shohin_master WHERE tanka >= 10000;



[演習 4]

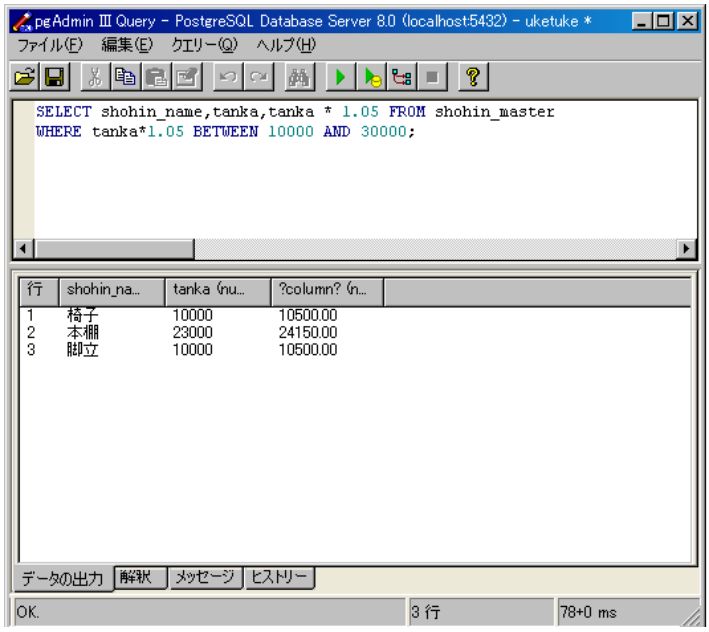
SELECT shohin_name, tanka FROM shohin_master WHERE tanka >10000;



[演習 5]

SELECT shohin_name, tanka, tanka*1.05 FROM shohin_master

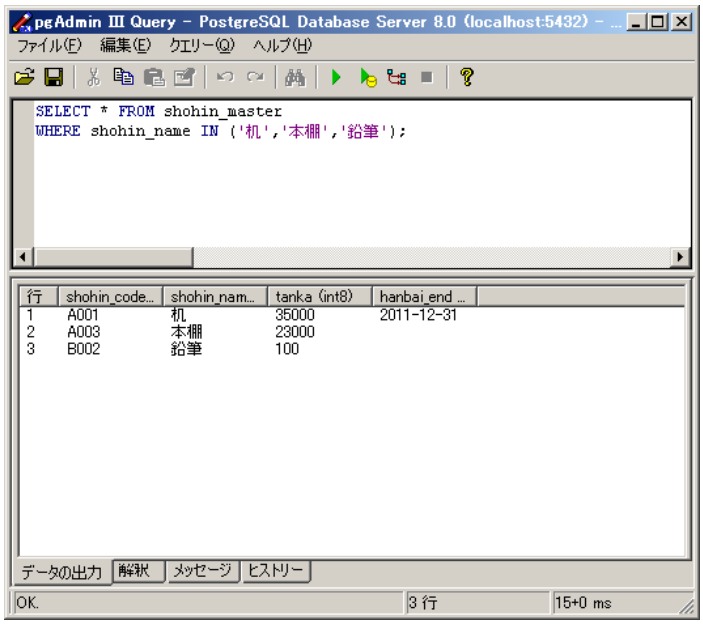
WHERE tanka*1.05 BETWEEN 10000 AND 30000;



[演習 6]

SELECT * FROM shohin_master

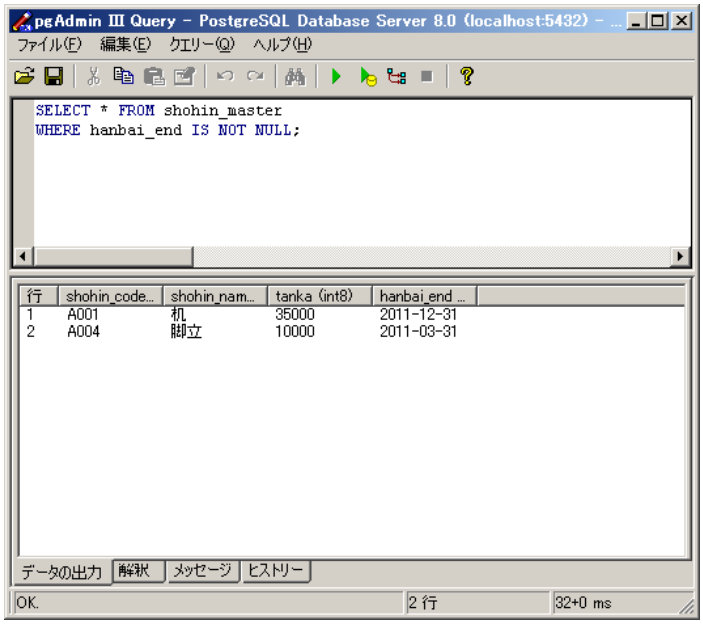
WHERE shohin_name IN (’机’,’本棚’,’鉛筆’);



[演習 7]

SELECT * FROM shohin_master

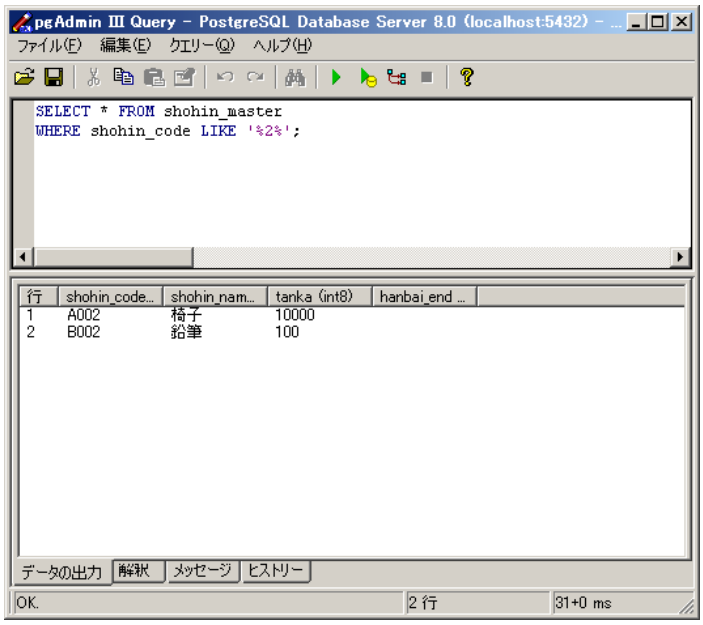
WHERE hanbai_end IS NOT NULL;



[演習 8]

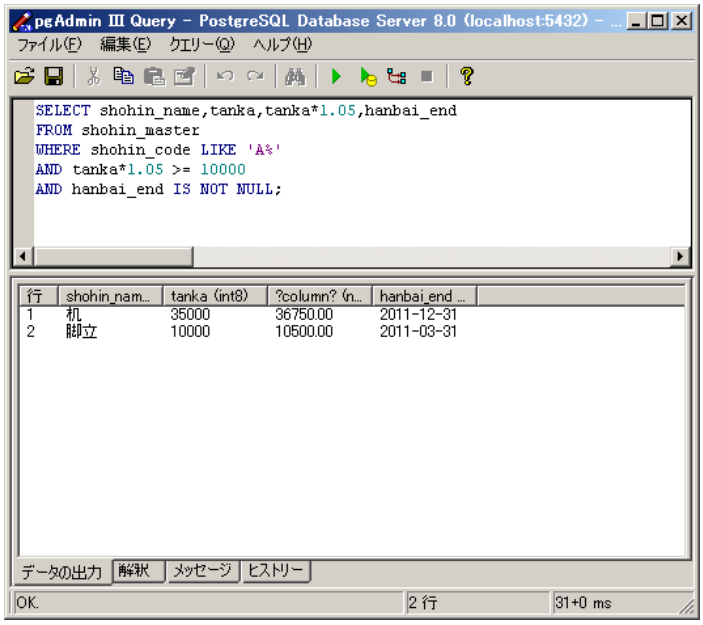
SELECT * FROM shohin_master

WHERE shohin_code LIKE '%2%';



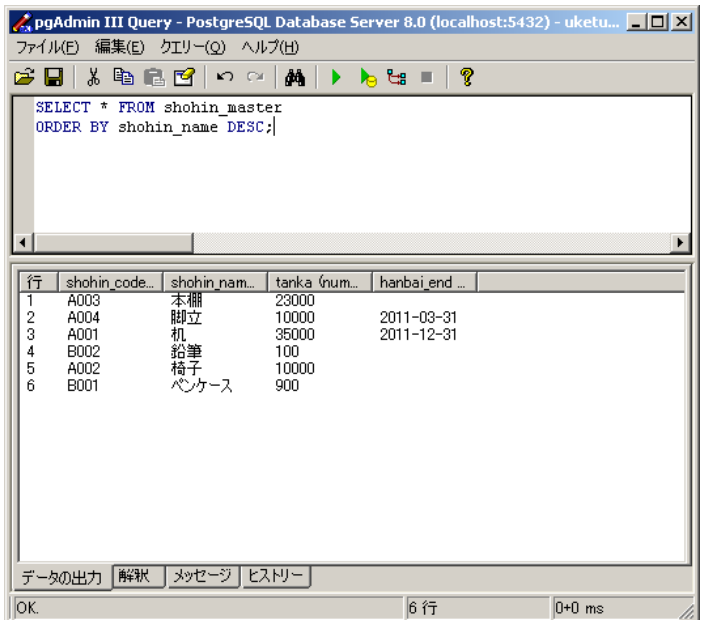
[**演習 9**]

```
SELECT shohin_name, tanka, tanka*1.05, hanbai_end
FROM shohin_master
WHERE shohin_code LIKE 'A%'
AND tanka * 1.05 >= 10000
AND hanbai_end IS NOT NULL;
```



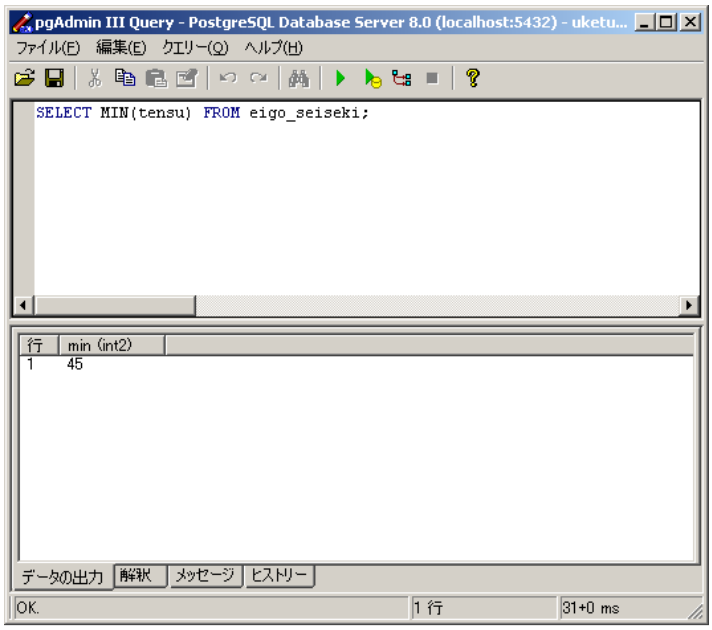
[**演習 10**]

```
SELECT * FROM shohin_master
ORDER BY shohin_name DESC;
```



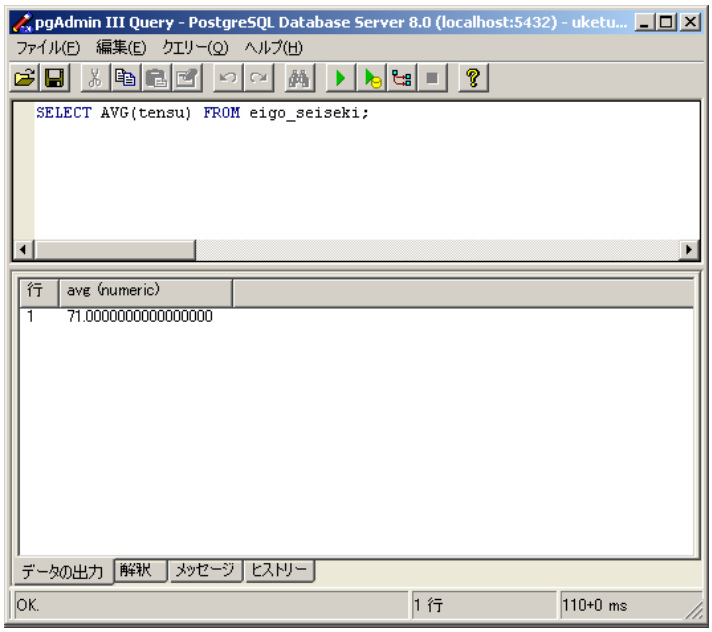
[演習 11]

SELECT MIN(tensu) FROM eigo_seiseki;



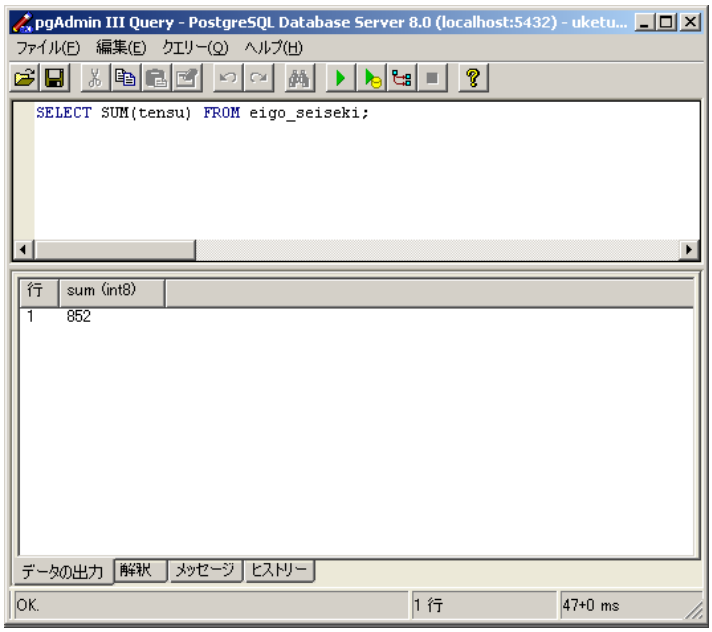
[演習 12]

SELECT AVG(tensu) FROM eigo_seiseki;



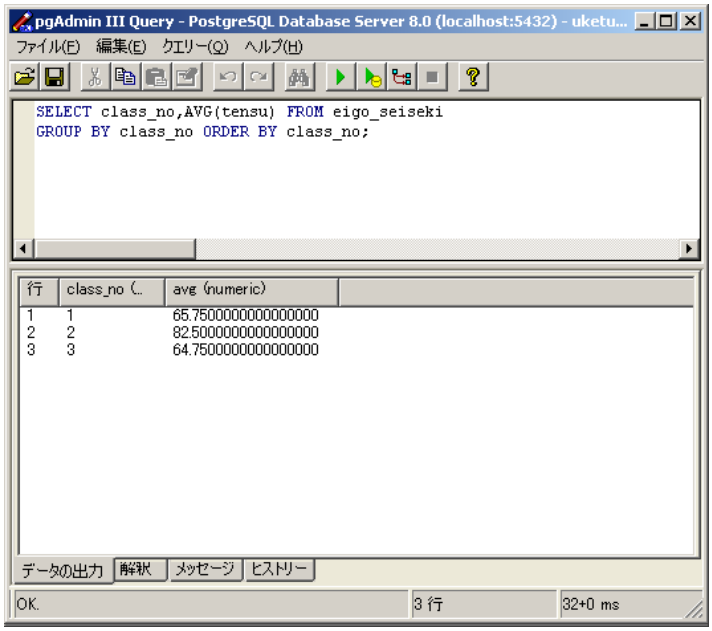
[演習 13]

SELECT SUM(tensu) FROM eigo_seiseki;



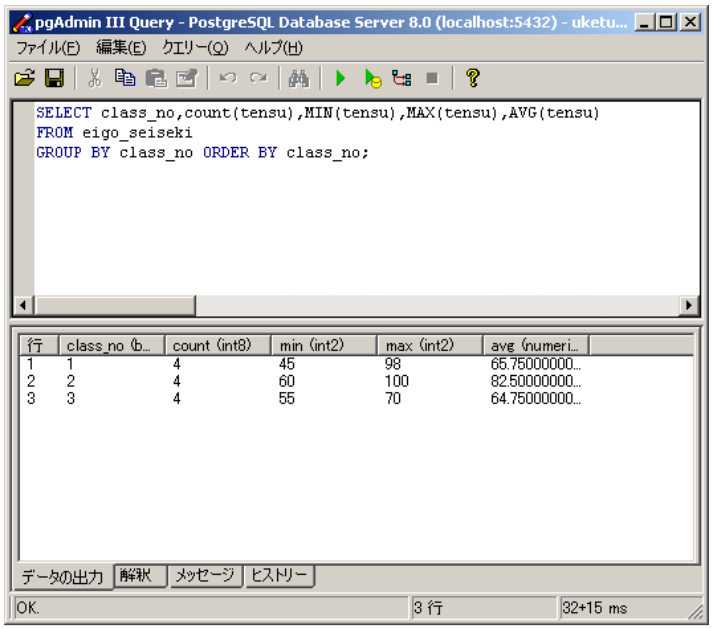
[演習 14]

SELECT class_no,AVG(tensu) FROM eigo_seiseki
GROUP BY class_no ORDER BY class_no;



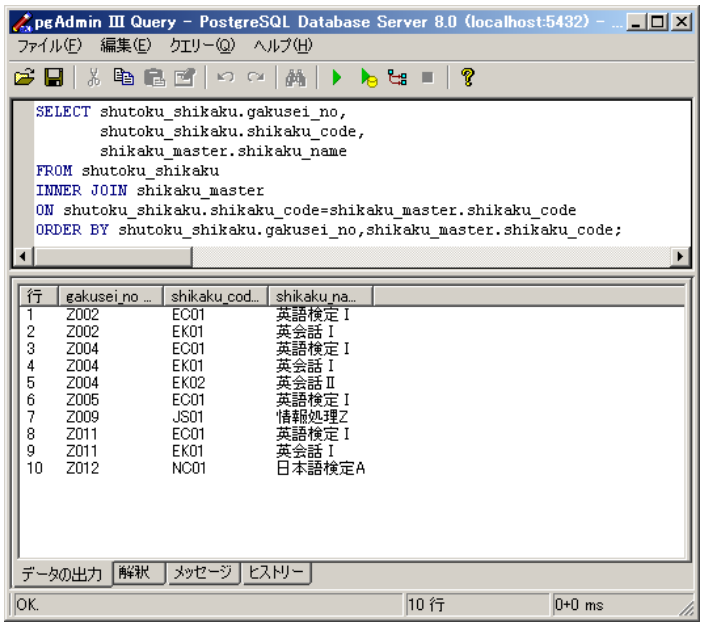
[**演習 15**]

```
SELECT class_no, count(tensu), MIN(tensu), MAX(tensu), AVG(tensu)
FROM eigo_seiseki
GROUP BY class_no ORDER BY class_no;
```



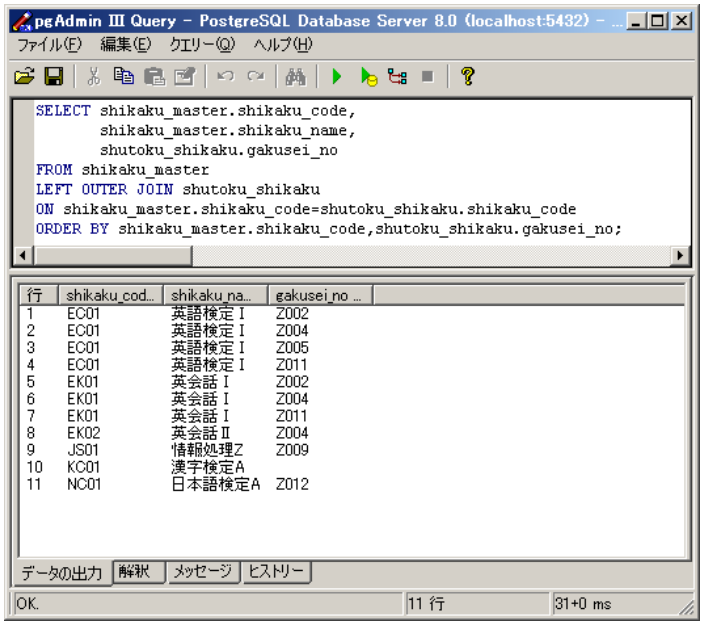
[**演習 16**]

```
SELECT shutoku_shikaku.gakusei_no,
       shutoku_shikaku.shikaku_code,
       shikaku_master.shikaku_name
FROM shutoku_shikaku
INNER JOIN shikaku_master
ON shutoku_shikaku.shikaku_code = shikaku_master.shikaku_code
ORDER BY shutoku_shikaku.gakusei_no, shikaku_master.shikaku_code;
```


pgAdmin III Query - PostgreSQL Database Server 8.0 (localhost:5432) -
ファイル(F) 編集(E) クエリー(Q) ヘルプ(H)
SELECT shutoku_shikaku.gakusei_no,
shutoku_shikaku.shikaku_code,
shikaku_master.shikaku_name
FROM shutoku_shikaku
INNER JOIN shikaku_master
ON shutoku_shikaku.shikaku_code=shikaku_master.shikaku_code
ORDER BY shutoku_shikaku.gakusei_no,shikaku_master.shikaku_code;
行 gakusei_no ... shikaku_cod... shikaku_na...
1 Z002 EC01 英語検定 I
2 Z002 EK01 英会話 I
3 Z004 EC01 英語検定 I
4 Z004 EK01 英会話 I
5 Z004 EK02 英会話 II
6 Z005 EC01 英語検定 I
7 Z009 JS01 情報処理Z
8 Z011 EC01 英語検定 I
9 Z011 EK01 英会話 I
10 Z012 NC01 日本語検定A
データの出力 解釈 メッセージ ヒストリー
OK.
10 行
0+0 ms

[演習 17]

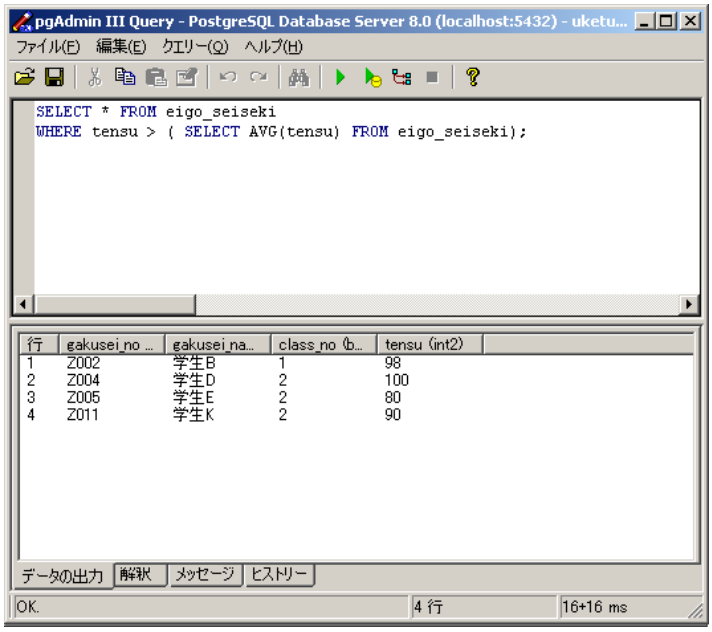
```
SELECT shikaku_master.shikaku_code,
       shikaku_master.shikaku_name,
       shutoku_shikaku.gakusei_no
FROM shikaku_master
LEFT OUTER JOIN shutoku_shikaku
ON shikaku_master.shikaku_code = shutoku_shikaku.shikaku_code
ORDER BY shikaku_master.shikaku_code, shutoku_shikaku.gakusei_no;
```



[**演習 18**]

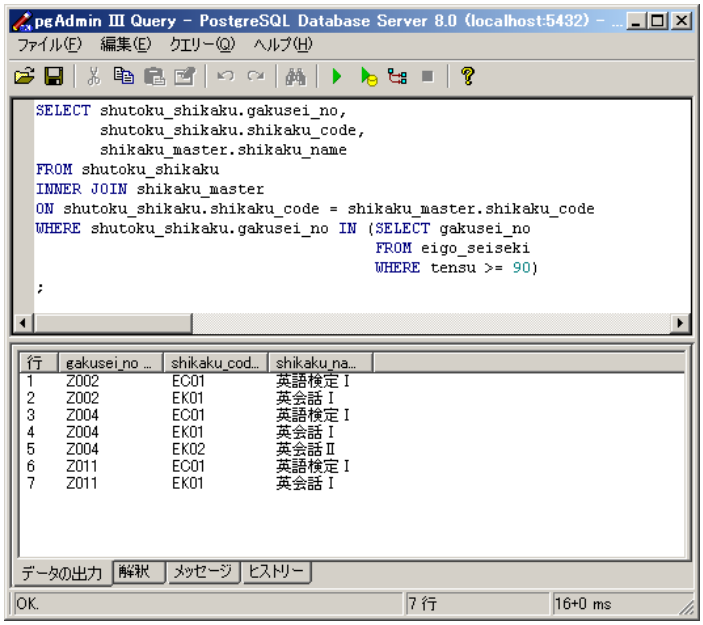
SELECT * FROM eigo_seiseki

WHERE tensu > (SELECT AVG(tensu) FROM eigo_seiseki);



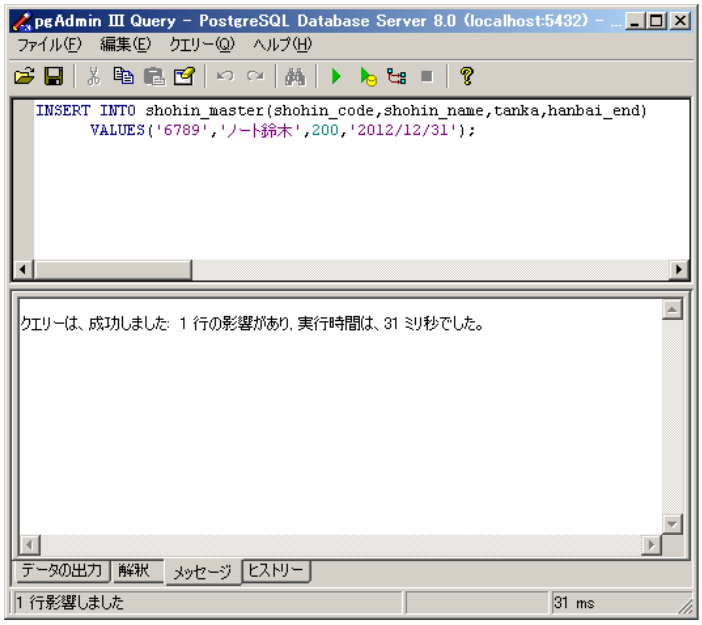
[**演習 19**]

```
SELECT shutoku_shikaku.gakusei_no,
       shutoku_shikaku.shikaku_code,
       shikaku_master.shikaku_name
FROM shutoku_shikaku
INNER JOIN shikaku_master
ON shutoku_shikaku.shikaku_code = shikaku_master.shikaku_code
WHERE shutoku_shikaku.gakusei_no IN
        (SELECT gakusei_no
         FROM eigo_seiseki
         WHERE tensu >= 90);
```

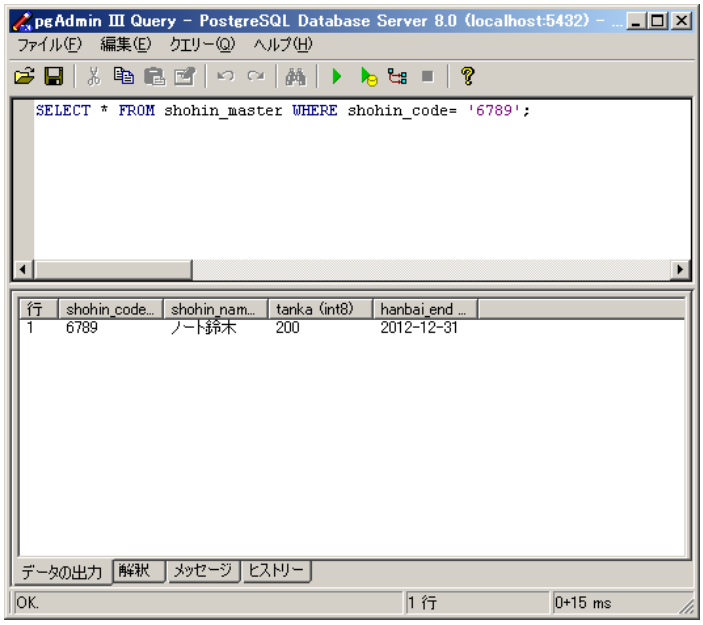


[**演習 20**]

INSERT INTO shohin_master(shohin_code, shohin_name, tanka, hanbai_end)
VALUES('6789', 'ノート鈴木', 200, '2012/12/31');



SELECT * FROM shohin_master WHERE shohin_code= '6789';

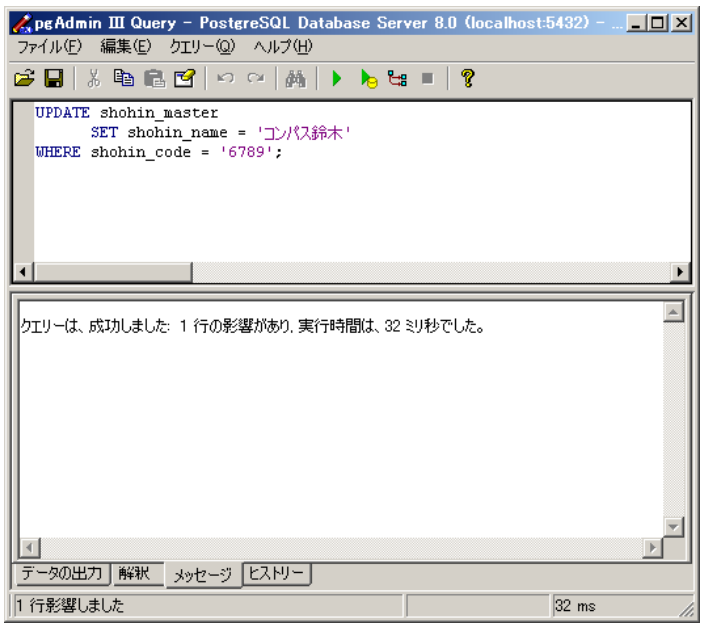


[**演習 21**]

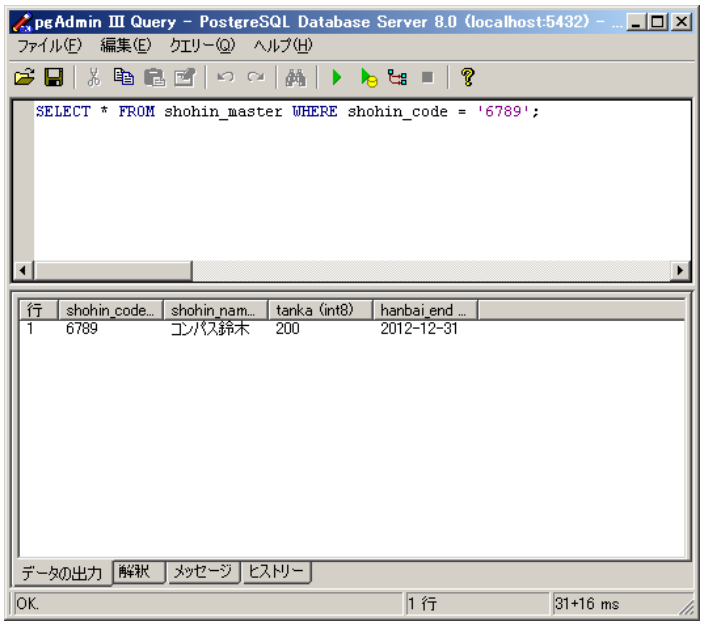
UPDATE shohin_master

SET shohin_name = ’コンパス鈴木’

WHERE shohin_code = ’6789’;

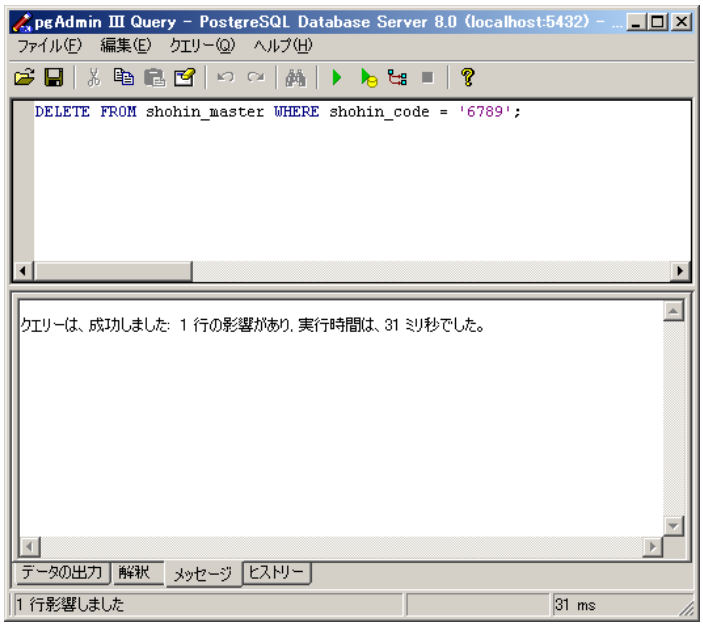


SELECT * FROM shohin_master WHERE shohin_code = ’6789’;

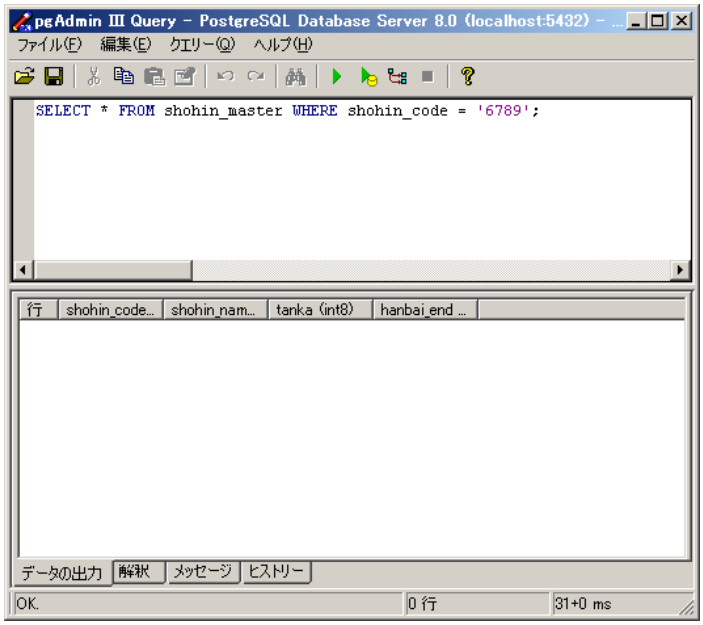


[**演習 22**]

DELETE FROM shohin_master WHERE shohin_code = ’6789’;



SELECT * FROM shohin_master WHERE shohin_code = ’6789’;



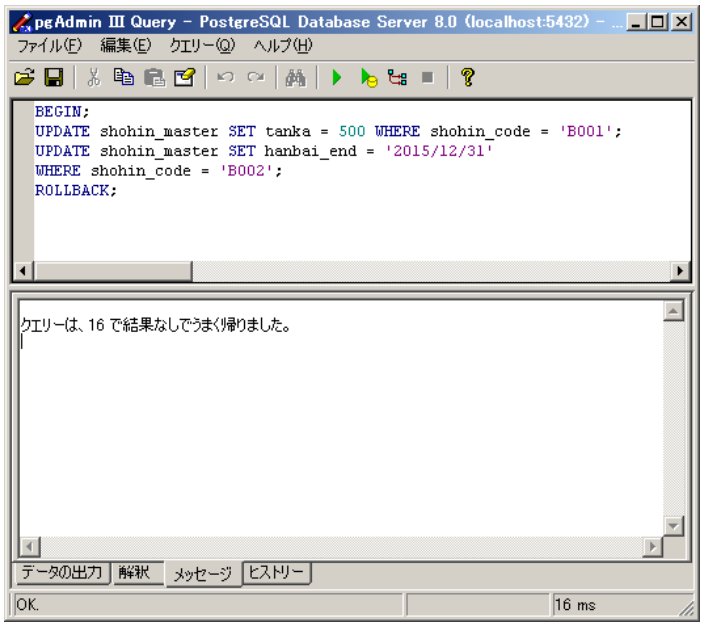
[**演習 23**]

BEGIN;

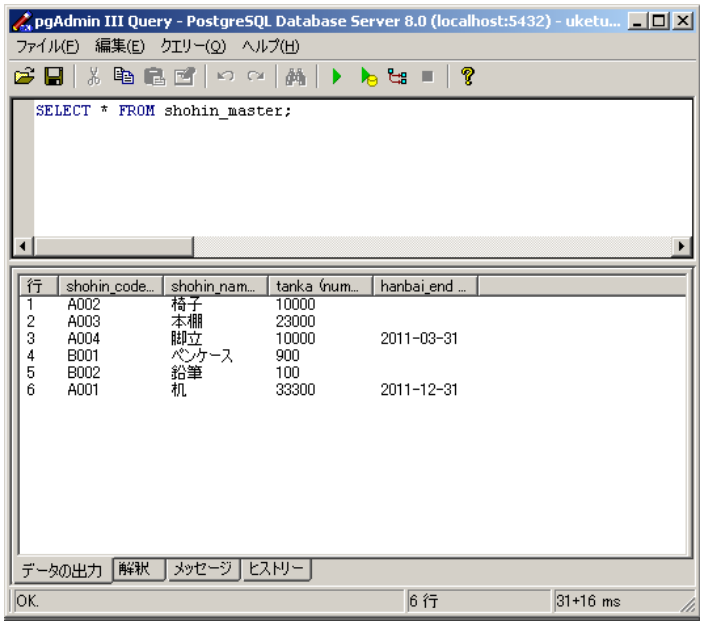
UPDATE shohin_master SET tanka = 500 WHERE shohin_code = 'B001';

UPDATE shohin_master SET hanbai_end = '2015/12/31' WHERE shohin_code = 'B002';

ROLLBACK;

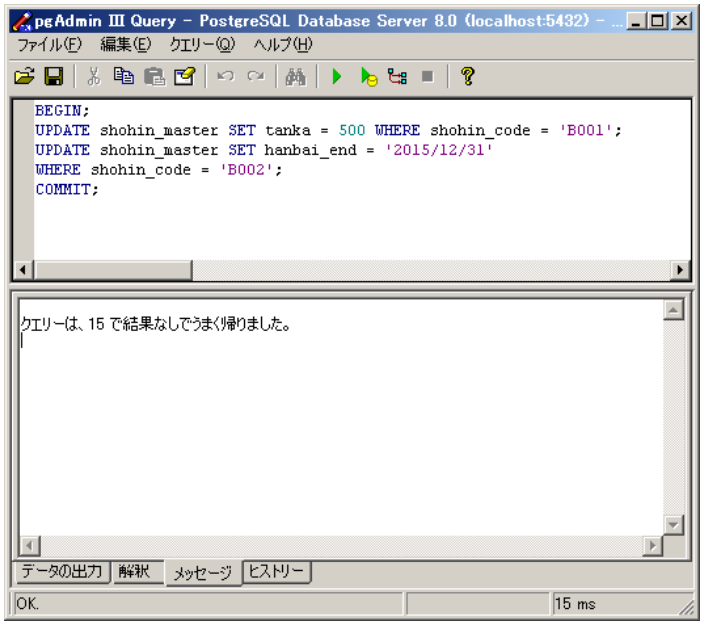


SELECT * FROM shohin_master;

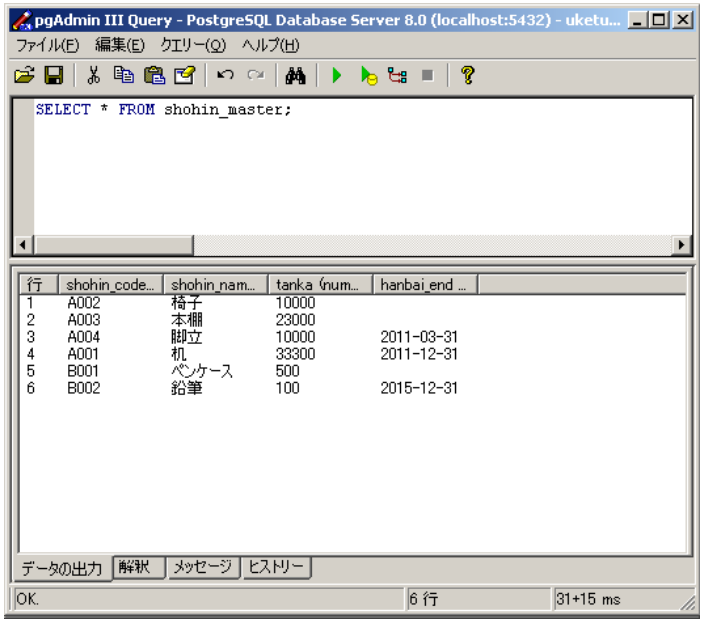


[※例題 29 実行後のデータ]

```
BEGIN;
UPDATE shohin_master SET tanka = 500 WHERE shohin_code = 'B001';
UPDATE shohin_master SET hanbai_end = '2015/12/31' WHERE shohin_code = 'B002';
COMMIT;
```




```
SELECT * FROM shohin_master;
```



[※例題 29 実行後のデータ]

第13回　DB設計実装演習

13.1　テーブル設計

リレーショナルデータベース (RDB) はデータをテーブル (表) という形で保存します. データベースを利用するには, データを入れるテーブルを前もって作成しておかなければなりません. テーブルを作成して, テーブル間に関連 (リレーションシップ) を設定することで, 複雑なデータもすっきりと表現できるのが RDB の特徴です.

そういった特徴を活かすためには, テーブルをどのように作成するかが非常に重要となります. これが「テーブル設計」です.

テーブルを設計するには, 次のステップを踏んでいきます.

1. **データ項目の洗い出し ～項目の収集～**
   管理・分析するために必要なデータ項目を洗い出していきます. 具体的には帳票や画面から項目を収集していきます.

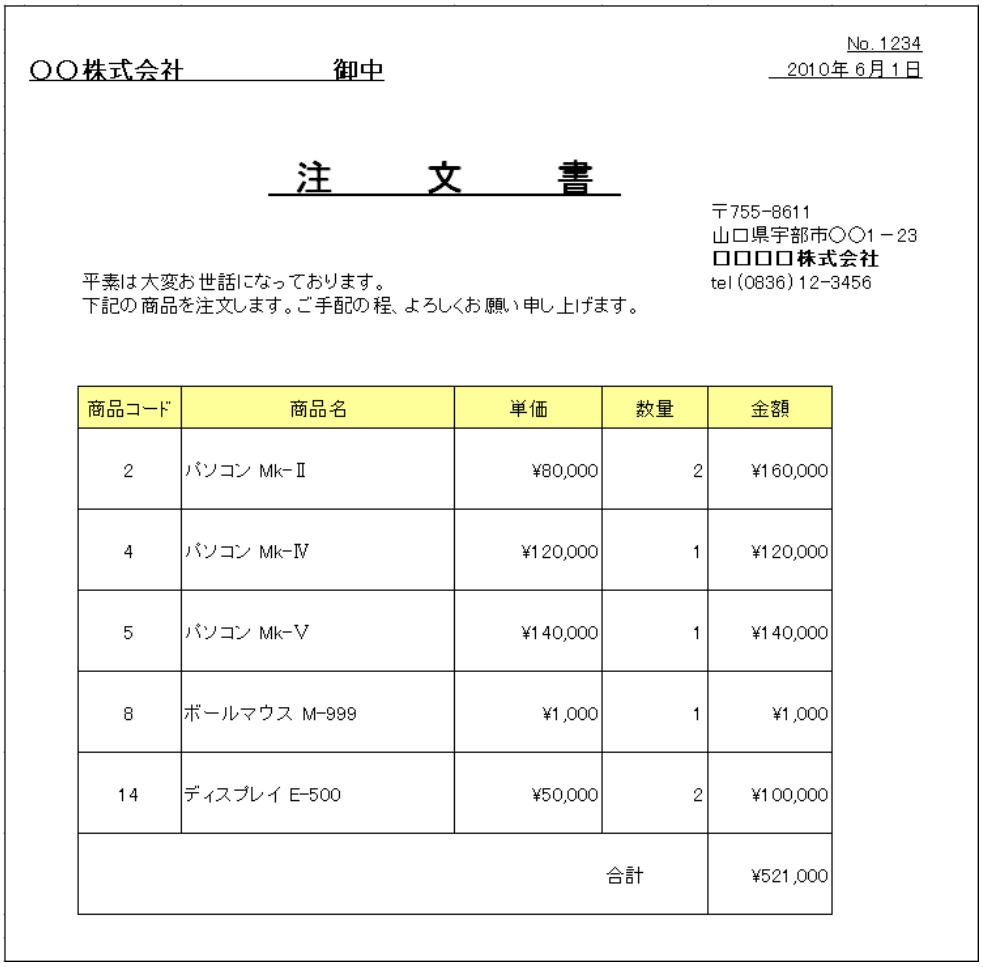
2. **データ項目の重複排除と分割 ～正規化～**
   正規化とはデータを管理しやすくするための手順で, データの重複を排除したりデータを分割したりします. データはできるだけ重複して持たせず１ヶ所で管理するようにすると, データを効率良く管理できます.

3. **リレーションの確認 ～エンティティ関連図 (ER 図) の作成～**
   データ正規化の結果を元にして, 関連 (リレーションシップ) と対応関係を明確に図にします. ER 図によって管理すべきデータの構造や対応関係が把握できます.

4. **データ型・制約の設定 ～エンティティ定義書の作成～**
   データ項目が, 文字列なのか数値なのか日付なのか, 何桁必要なのかといったデータの種類と大きさを決定し, テーブルの制約 (主キー, NOT NULL など) を明確に設定します.

以上の手順を具体的なサンプルを使って進めていきます.

■サンプル帳票 (注文書)



No. 1234
2010年 6月 1日

○○株式会社　　御中

注　文　書

〒755-8611
山口県宇部市○○1－23
□□□□株式会社
tel (0836) 12-3456

平素は大変お世話になっております。
下記の商品を注文します。ご手配の程、よろしくお願い申し上げます。

商品コード	商品名	単価	数量	金額
2	パソコン Mk-Ⅱ	¥80,000	2	¥160,000
4	パソコン Mk-Ⅳ	¥120,000	1	¥120,000
5	パソコン Mk-Ⅴ	¥140,000	1	¥140,000
8	ボールマウス M-999	¥1,000	1	¥1,000
14	ディスプレイ E-500	¥50,000	2	¥100,000
			合計	¥521,000

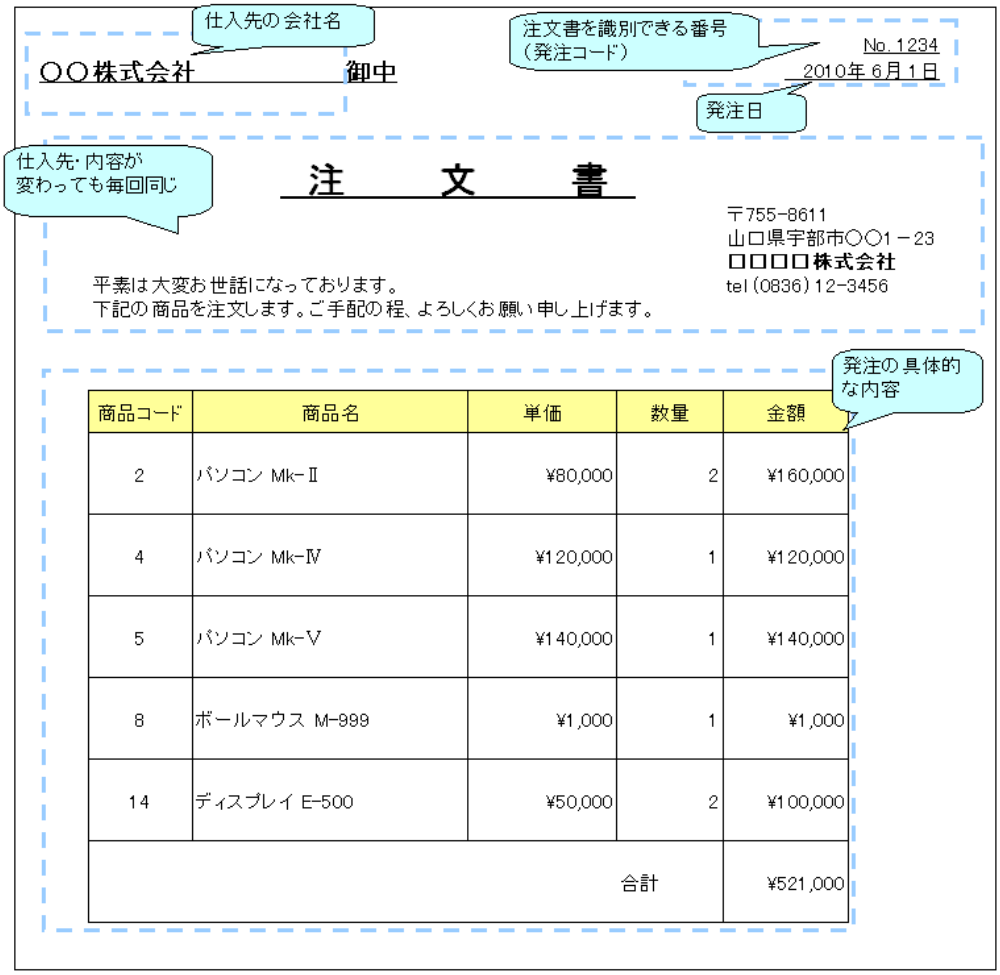
この帳票はタイトルから「注文書」で, 内容は「□□□□株式会社が, パソコンやマウスなどを○○株式会社から買うため, ○○株式会社へ発注したい内容を明記して送るもの」であることがわかります.

とある「受発注システム」を構築する中で, この注文書を発行するために必要なテーブルの設計をしてみましょう.

- 帳票とは …「帳簿」＋「伝票」＝「帳票」
  現在の企業で「帳票」といえば, 取引や業務などの内容を特定のフォーマットに基づいて記したものです.

13.1.1 データ項目の洗い出し ～項目の収集～

帳票から, 必要な項目を洗い出して収集していきます.



[備考]・発注コードは, 注文書が発生するごとに一意につけられる.
・金額は単価×数量で求められ, 合計金額はその金額をすべて足したものである.

この注文書から項目を列挙すると次のようになります.
※毎回同じもの (帳票タイトルや自社情報) は今回項目として挙げないこととします.

・発注コード ・発注日 ・仕入先名
・商品コード ・商品名 ・単価 ・数量 ・金額 ・合計金額

[演習 (受注伝票)1]

(13.2 演習 (受注伝票) へ)

13.1.2　データ項目の重複排除と分割 ～正規化～

正規化の手順

データを管理しやすくするため, データの重複を排除したりデータを分割するのが正規化です. まだ正規化されていないものを非正規形, 正規化されたものを正規形といいます. 正規形には正規化の程度により, 第一正規形から第五正規形まであります. ほとんどの場合, 第三正規形まで正規化すれば的確な正規化がなされたとしてよいとされていますので, ここでは第三正規形まで行います.

注文書を例に正規化を行っていきます.
まず収集した項目をテーブルに変換し, データをいれます. このときのテーブルは次のような形になります. これが非正規形の状態です.

・非正規形

発注コード	発注日	仕入先名	商品コード	商品名	単価	数量	金額	合計金額
1234	2010/6/1	○○株式会社	2	パソコン Mk-Ⅱ	¥80,000	2	¥160,000	¥52,100
			4	パソコン Mk-Ⅳ	¥120,000	1	¥120,000	
			5	パソコン Mk-Ⅴ	¥140,000	1	¥140,000	
			8	ボールマウス M-999	¥1,000	1	¥1,000	
			14	ディスプレイ E-500	¥50,000	2	¥100,000	

明細行分繰り返されている

・第一正規化 … 重複するデータが無い状態にする

テーブルを第一正規形にします. 繰り返し部分と繰り返されない部分とを切り離して別のテーブルにします. この注文書では発注の明細部分 (「商品コード」「商品名」「単価」「数量」「金額」) が繰り返されていますので切り離します. そのまま切り離したのでは, 発注と発注明細の関係が不明になりますので, 二つのテーブルを繋ぐ項目 (連結キー) として「発注コード」を設定します. 分離したテーブルは次のようになります.

「発注」

発注コード	発注日	仕入先名	合計金額
1234	2010/6/1	○○株式会社	¥52,100

「発注品目」

発注コード	商品コード	商品名	単価	数量	金額
1234	2	パソコン Mk-Ⅱ	¥80,000	2	¥160,000
1234	4	パソコン Mk-Ⅳ	¥120,000	1	¥120,000
1234	5	パソコン Mk-Ⅴ	¥140,000	1	¥140,000
1234	8	ボールマウス M-999	¥1,000	1	¥1,000
1234	14	ディスプレイ E-500	¥50,000	2	¥100,000

※行が一意に認識できる項目:「主キー (Primary Key：プライマリー キー)」
「発注」テーブルの主キー =「発注コード」

「発注品目」テーブルの主キー =「発注コード」と「商品コード」

・第二正規化 … 部分関数従属関係を分割して別表にする

主キーのうち, そのキーが決まれば値が決まるもの (部分関数従属関係) を分離して第二正規形にします.

ここでは「発注品目」の主キーの一つである「商品コード」が決まれば「商品名」「単価」は決まりますので「発注品目」テーブルから商品情報の部分を「商品マスタ」に分離します.

また「発注品目」テーブルの「金額」は単価と数量から計算され「発注」テーブルの「合計金額」はその金額の和で求められるので, この時点で取り除いておきます.

「発注」

発注コード	発注日	仕入先名
1234	2010/6/1	○○株式会社

「発注品目」

発注コード	商品コード	数量
1234	2	2
1234	4	1
1234	5	1
1234	8	1
1234	14	2

「商品マスタ」

商品コード	商品名	単価
2	パソコン Mk-Ⅱ	¥80,000
4	パソコン Mk-Ⅳ	¥120,000
5	パソコン Mk-Ⅴ	¥140,000
8	ボールマウス M-999	¥1,000
14	ディスプレイ E-500	¥50,000

・第三正規化 … 推移的関数従属関係を分割して別表にする

主キー以外の項目で, ある項目の値が決まれば値が決まるもの (推移的関数従属関係) を分離して第三正規形にします.

ここでは「発注」テーブルから仕入先情報の部分を「仕入先マスタ」テーブルに分離します.

この時「仕入先マスタ」テーブルに一意に識別する項目として「仕入先コード」を新たに設定し, これを主キーとします. 「発注」テーブルと「仕入先マスタ」テーブルの連結キーは「仕入先コード」となります.

※「発注コード」→「仕入先コード」→「仕入先名」が推移的関数従属です.

「発注」

発注コード	発注日	仕入先コード
1234	2010/6/1	003

「仕入先マスタ」

仕入先コード	仕入先名
003	○○株式会社

「発注品目」

発注コード	商品コード	数量
1234	2	2
1234	4	1
1234	5	1
1234	8	1
1234	14	2

「商品マスタ」

商品コード	商品名	単価
2	パソコン Mk-Ⅱ	¥80,000
4	パソコン Mk-Ⅳ	¥120,000
5	パソコン Mk-Ⅴ	¥140,000
8	ボールマウス M-999	¥1,000
14	ディスプレイ E-500	¥50,000

これで正規化は完了です. 注文書は四つのテーブルとなりました.

正規化のメリットとデメリット

今回の注文書の正規化についてメリットとデメリットを少し考えてみましょう.

メリット

・データに変更の必要が生じたときに, 変更する手間が大幅に削減される.
例）発注先の名称が変更されたときは「仕入先マスタ」の内容を変更するだけでよい.
例）商品の名称や単価が変更になったときは「商品マスタ」の内容を変更するだけでよい.

・無駄な列を削除することで, データ容量の削減やデータ処理の効率も上がる.
例）金額は単価×数量の計算で取得できるため, 無駄なデータ領域が必要なくなる.

・正規化されたデータは管理がしやすく, 他で利用できる可能性が高まる.
例）仕入先マスタは「注文書発行」以外の機能でも利用できる.

デメリット

システム上問題となる場合には, 正規化した内容を見直すことも必要です.

・分割によりテーブル数が増え, 処理時にテーブル結合が複雑になり逆に管理しにくくなる可能性もある.

・単価×数量=金額とは限らない (値引きやキャンペーン価格など) 場合に対応できない.
例)「発注品目」テーブルに「金額」項目を持たせておく

・会社名の変更や商品名・単価の変更があっても, 以前の値を保持したい場合は困る.
例)「発注品目」テーブルに「単価」項目を持たせて, 当時の単価を保持する.

正規化はテーブルの無駄を省いたり管理しやすくするために必ず必要な作業ですが, 以上のようなメリットとデメリットから, 正規化したものを状況に応じて崩すという考え方も大切です.

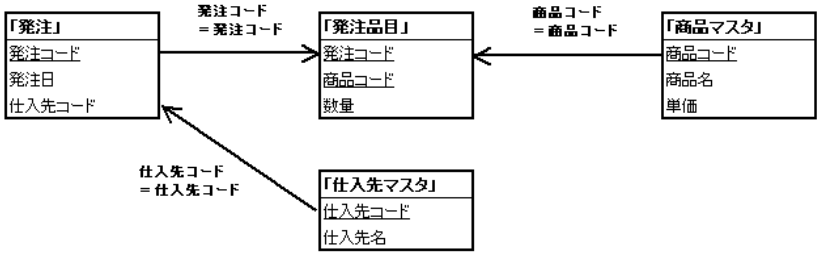
[演習 (受注伝票)2]

(13.2 演習 (受注伝票) へ)

13.1.3　リレーションの確認 ～エンティティ関連図の作成～

データベースとして表現すべき対象物をエンティティ (entity：実体) と呼びます. また, エンティティとエンティティの相互関係をリレーションシップ (relationship：関連) と呼びます. そして, この関係を図示し, エンティティとエンティティの間のリレーションシップを分析する技法を エンティティ関連図:ER(Entity-Relationship) 図といいます. 図式化により, テーブル構造やリレーション関係とその連結キーがよりわかりやすく明確になります.

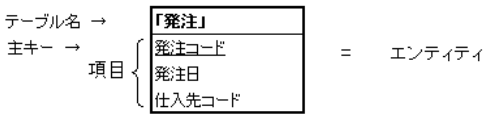
注文書の正規化した四つのテーブルを ER 図で表すと最終的には次のようになります.



具体的な作成手順をみてみましょう.

1. エンティティの作成と主キーの設定

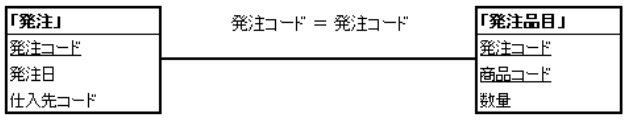
まず, テーブル名と項目を列挙したものを四角でくくります. 主キーにはアンダーバーをつけて表現します. すべてのテーブルについてエンティティを作成します.



2. リレーションシップの設定

主キーに着目しながら, エンティティ同士の関連 (リレーションシップ) があるものに線をひいて繋げます. 線には, 連結キーを明記しましょう.

「発注」テーブルと「発注品目」テーブルはリレーションの関係にあり, それぞれの「発注コード」同士で連結しますので, ER 図は次のようになります.

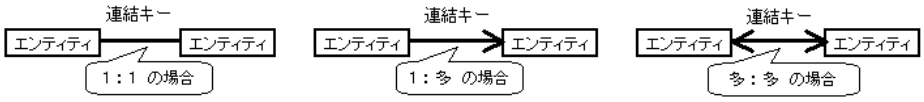


3. リレーションシップの連結キー関連関係設定

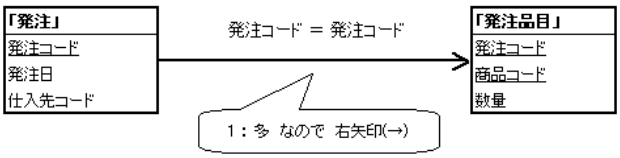
連結キーは「1 対 1」「1 対多」「多対多」のいずれかの関係となっています.

たとえば「発注」テーブル 1 行に対して「発注品目」は数行存在しますので「発注」:「発注品目」=「1：多」の関係といえます.

この関連関係はリレーションの線で表現します. 1 対 1 の関連は直線, 1 対多の関連は多の側方向を指す片方向矢印, 多対多の関連は両方向矢印になります.



「発注」テーブルと「発注品目」テーブルは 1：多 の関係のため矢印は次のようになります.



これをすべてのエンティティで設定すると, ER 図が完成です.

[演習 (受注伝票)3]

(13.2 演習 (受注伝票) へ)

13.1.4　データ型・制約の設定 ～エンティティ定義書の作成～

エンティティ定義書とはエンティティの属する項目 (データ項目) について具体的に説明したものです．データ項目の保持していく内容が文字列なのか数値なのか日付なのか何桁必要なのかといったデータの種類と大きさを決定したり, テーブルの制約 (主キー, NOT NULL など) を明確にします．

※定義書のフォーマットとして標準的なものはありません．

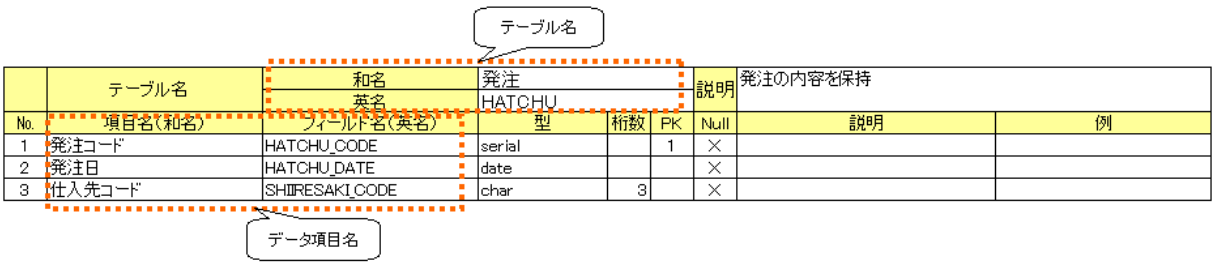
例えば「発注」テーブルのエンティティ定義書は以下のようになります．

	テーブル名	和名	発注			説明	発注の内容を保持	
		英名	HATCHU					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	発注コード	HATCHU_CODE	serial		1	×		
2	発注日	HATCHU_DATE	date			×		
3	仕入先コード	SHIIRESAKI_CODE	char	3		×		

1. テーブル名や項目名の設定

実際のデータベース内では, テーブル名や項目名は英字となりますので, 英字の項目名を設定していきます．(日本語対応のデータベースでは日本語での設定も可能ですが, 多くは英字で構築しているのが実情です)

英字の場合でも, 項目内容がわかりやすいように設定します．ある程度のネーミングルールを先に決めて統一すると良いでしょう．また予約語など (あらかじめ「予約」されていて, 変数名や関数名として定義できない単語) は使わないように注意が必要です．



例：○○番号 = ○○_NO, ○○_BANGO どちらかに統一し混在しないようにする．

例：「グループ」というデータ項目があっても, SQL に「GROUP BY」という構文がある (＝予約語) ため「GROUP」という名前設定できない．

2. データ型の設定

データ項目一つ一つについて, データ型を設定します.

PostgreSQL で利用できるデータ型は次を参照してください.[9]

■PostgreSQLでサポートされる主なデータ型

PostgreSQLにおけるデータ型	SQL標準におけるデータ型	コメント
CHAR	CHARACTER または CHAR	文字
CHAR(n)	CHARACTER(n) または CHAR(n)	固定長文字列
VARCHAR(n)	CHARACTER VARYING(n) またはCHAR VARYING(n) またはVARCHAR(n)	可変長文字列
BIT(n)	BIT(n)	固定長ビット型
BIT VARYING(n)	BIT VARYING(n)	可変長ビット型
サポートされていません	NCHAR	
サポートされていません	NATIONAL CHARACTER	
FLOAT4/8	FLOAT(p)	精度pの浮動小数点
FLOAT8	DOUBLE PRECISION	倍精度浮動小数点
FLOAT4	REAL	単精度浮動小数点
INT4	INTEGER または INT	符号つき整数
INT2	SMALLINT	小桁符号つき整数
NUMERIC(p,s)	NUMERIC(p,s)	任意精度の10進数数値
DECIMAL(p,s)	DECIMAL(p,s)	任意精度の10進数数値
DATE	DATE	日付(年月日)
TIME	TIME	時刻(時分秒)
TIMESTAMP	TIMESTAMP	日付と時刻
INTERVAL	INTERVAL	時間間隔
BOOL	BOOLEAN	真/偽

■PostgreSQL独自のデータ型

PostgreSQLにおけるデータ型	コメント	使用方法／説明
TEXT	可変長テキスト	
BYTEA	可変長バイナリデータ	
BIGINT(INT8)	8バイト整数	サポートされないプラットフォームもある
OID	オブジェクトID	データベース内のオブジェクトを識別する
SERIAL	順序数	自動的に連番を生成する特殊なデータ型
CIDR	ネットワークアドレス	CIDR表記のネットワークアドレス
INET	IPアドレス	ネットワークアドレス情報を含むIPアドレス
MACADDR	MACアドレス	イーサネットインターフェイスのアドレス

■PostgreSQLの数値データ型

データ型	データベース上のバイト数	値の範囲
INT2(SMALLINT)	2バイト	-32768～+32767
INT4(INTEGER)	4バイト	-2147483648～+2147483647
INT8(BIGINT)	8バイト	-9223372036854775808～+9223372036854775807
FLOAT4(REAL)	4バイト	-
FLOAT8(FLOAT)	8バイト	-
NUMERIC	可変長	1000桁までの精度の整数と小数
DECIMAL	可変長	1000桁までの精度の整数と小数
SERIAL	4バイト	-2147483648～+2147483647
SERIAL8	8バイト	-9223372036854775808～+9223372036854775807

テーブルを定義する際には, 格納される値を元に適切なデータ型を設定します. 標準 SQL 規格のデータ型であっても, 実際に使用できるかどうかは RDBMS によって異なります. また, 標準 SQL 規格には定義されていない RDBMS 独自のデータ型も少なくなく, そのようなデータ型を使用したほうが適切である場合もあります.

	テーブル名	和名	発注			説明	発注の内容を保持	
		英名	HATCHU					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	発注コード	HATCHU_CODE	serial		1	×		
2	発注日	HATCHU_DATE	date			×		
3	仕入先コード	SHIIRESAKI_CODE	char	3		×		

データ型・大きさ

例：「発注日」→ 日付が格納される = date 型

「発注コード」→ 一意に決まる数値の連番とする = serial 型

※ serial 型:PostgreSQL 独自のデータ型, 自動的に連番を生成.

3. 制約の設定

テーブルの制約にはいろいろな種類がありますが, ここでは「主キー (PRIMARY KEY)」と「NOT NULL」を設定し, エンティティ定義書で明確にしておきます.

・主キーとは, 列のデータが重複せず NULL 値ではないことを保証するもので, テーブルの中の行を一意に識別できる項目.

・NOT NULL とは, データに NULL 値を許可しない制約.

	テーブル名	和名	発注			説明	発注の内容を保持	
		英名	HATCHU					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	発注コード	HATCHU_CODE	serial		1	×		
2	発注日	HATCHU_DATE	date			×		
3	仕入先コード	SHIIRESAKI_CODE	char	3		×		

制約

その他注意事項, 初期値などの説明があれば記述する

「発注」テーブルでは「発注コード」が主キーですので「PK」列と「Null」列で制約を表現します. さらにこの例では「発注日」と「仕入れコード」も Null を許可しない項目と定義しています.

その他にも注意事項や説明があれば, エンティティ定義書の中で明確にしておきます.

注文書の四つのテーブルのエンティティ定義書は次のようになります.

	テーブル名	和名	発注			説明	発注の内容を保持	
		英名	HATCHU					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	発注コード	HATCHU_CODE	serial		1	×		
2	発注日	HATCHU_DATE	date			×		
3	仕入先コード	SHIIRESAKI_CODE	char	3		×		

	テーブル名	和名	発注品目			説明	発注の商品毎の発注数を保持	
		英名	HATCHU_HINMOKU					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	発注コード	HATCHU_CODE	integer		1	×		
2	商品コード	SHOHIN_CODE	integer		2	×		
3	数量	SURYO	integer			×		

	テーブル名	和名	商品マスタ			説明	商品情報を保持	
		英名	SHOHIN_MASTER					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	商品コード	SHOHIN_CODE	integer	4	1	×		
2	商品名	SHOHIN_NAME	varchar	40				
3	単価	SHIIRE_TANKA	numeric	18				

	テーブル名	和名	仕入先マスタ			説明	仕入先の情報を保持	
		英名	SHIIRESAKI_MASTER					
No.	項目名(和名)	フィールド名(英名)	型	桁数	PK	Null	説明	例
1	仕入先コード	SHIIRESAKI_CODE	char	3	1	×		
2	仕入先名	SHIIRESAKI_NAME	varchar	30				

[演習 (受注伝票)4]

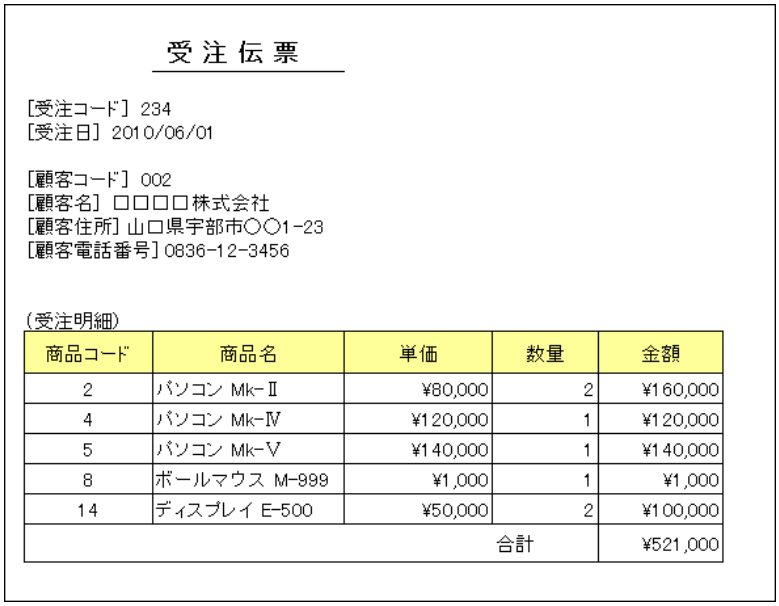
(13.2 演習 (受注伝票) へ)

以上でテーブル設計は完了です.

テーブル設計の成果物として, エンティティ関連図 (ER 図) とエンティティ定義書が作成されました. これらのデータベース設計書を元にシステムの設計を行っていくこととなります.

13.2　演習 (受注伝票)

次のような受注伝票から, テーブル設計をおこないなさい.



受 注 伝 票

[受注コード] 234
[受注日] 2010/06/01

[顧客コード] 002
[顧客名] □□□□株式会社
[顧客住所] 山口県宇部市○○1-23
[顧客電話番号] 0836-12-3456

(受注明細)

商品コード	商品名	単価	数量	金額
2	パソコン Mk-Ⅱ	¥80,000	2	¥160,000
4	パソコン Mk-Ⅳ	¥120,000	1	¥120,000
5	パソコン Mk-Ⅴ	¥140,000	1	¥140,000
8	ボールマウス M-999	¥1,000	1	¥1,000
14	ディスプレイ E-500	¥50,000	2	¥100,000
			合計	¥521,000

[演習 (受注伝票)1]

受注伝票から項目を列挙しなさい.

[演習 (受注伝票)2]

演習 1 で収集した項目をテーブルに変換し, 第三正規化までおこないなさい.

[演習 (受注伝票)3]

正規化したテーブルの ER 図を作成しなさい.

[演習 (受注伝票)4]

エンティティ定義書を作成しなさい.

[演習 (受注伝票)1 解答例]

・受注コード ・受注日
・顧客コード ・顧客名 ・顧客住所 ・顧客電話番号
・商品コード ・商品名 ・単価 ・数量 ・金額
・合計金額

[演習 (受注伝票)2 解答例]

＜非正規化＞

受注コード	受注日	顧客コード	顧客名	顧客住所	顧客電話番号	商品コード	商品名	単価	数量	金額	合計金額
234	2010/6/1	002	□□□□株式会社	山□県宇部市○○1-23	0836-12-3456	2	パソコン Mk-Ⅱ	¥80,000	2	¥160,000	¥521,000
						4	パソコン Mk-Ⅳ	¥120,000	1	¥120,000	
						5	パソコン Mk-Ⅴ	¥140,000	1	¥140,000	
						8	ボールマウス M-999	¥1,000	1	¥1,000	
						14	ディスプレイ E-500	¥50,000	2	¥100,000	

＜第一正規化＞

「受注」

受注コード	受注日	顧客コード	顧客名	顧客住所	顧客電話番号	合計金額
234	2010/6/1	002	□□□□株式会社	山□県宇部市○○1-23	0836-12-3456	¥521,000

「受注品目」

受注コード	商品コード	商品名	単価	数量	金額
234	2	パソコン Mk-Ⅱ	¥80,000	2	¥160,000
234	4	パソコン Mk-Ⅳ	¥120,000	1	¥120,000
234	5	パソコン Mk-Ⅴ	¥140,000	1	¥140,000
234	8	ボールマウス M-999	¥1,000	1	¥1,000
234	14	ディスプレイ E-500	¥50,000	2	¥100,000

＜第二正規化＞

「受注」

受注コード	受注日	顧客コード	顧客名	顧客住所	顧客電話番号
234	2010/6/1	002	□□□□株式会社	山□県宇部市○○1-23	0836-12-3456

「受注品目」

受注コード	商品コード	数量
234	2	2
234	4	1
234	5	1
234	8	1
234	14	2

「商品マスタ」

商品コード	商品名	単価
2	パソコン Mk-Ⅱ	¥80,000
4	パソコン Mk-Ⅳ	¥120,000
5	パソコン Mk-Ⅴ	¥140,000
8	ボールマウス M-999	¥1,000
14	ディスプレイ E-500	¥50,000

＜第三正規化＞

「受注」

受注コード	受注日	顧客コード
234	2010/6/1	002

「顧客マスタ」

顧客コード	顧客名	顧客住所	顧客電話番号
002	□□□□株式会社	山□県宇部市○○1-23	0836-12-3456

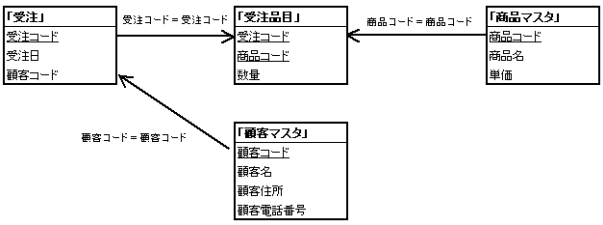
「受注品目」

受注コード	商品コード	数量
234	2	2
234	4	1
234	5	1
234	8	1
234	14	2

「商品マスタ」

商品コード	商品名	単価
2	パソコン Mk-Ⅱ	¥80,000
4	パソコン Mk-Ⅳ	¥120,000
5	パソコン Mk-Ⅴ	¥140,000
8	ボールマウス M-999	¥1,000
14	ディスプレイ E-500	¥50,000

[演習 (受注伝票)3 解答例]



[演習 (受注伝票)4 解答例]

	テーブル名	和名	受注			説明	受注伝票の内容を格納	
		英名	JHUCHU					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	受注コード	JHUCHU_CODE	serial		1	×		
2	受注日	JHUCHU_DATE	date			×		
3	顧客コード	KOKYAKU_CODE	char	6		×		

	テーブル名	和名	受注品目			説明	受注伝票の明細を格納	
		英名	JHUCHU_HINMOKU					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	受注コード	JHUCHU_CODE	integer		1	×		
2	商品コード	SHOHIN_CODE	integer		2	×		
3	数量	SURYO	integer					

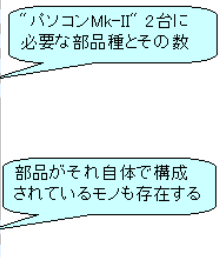
	テーブル名	和名	商品マスタ			説明	商品の情報を格納	
		英名	SHOHIN_MASTER					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	商品コード	SHOHIN_CODE	integer		1	×		
2	商品名	SHOHIN_NAME	varchar	40		×		
3	単価	TANKA	numeric	18				

	テーブル名	和名	顧客マスタ			説明	顧客の会社情報を格納	
		英名	KOKYAKU_MASTER					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	顧客コード	KOKYAKU_CODE	char	3	1	×		
2	顧客名	KOKYAKU_NAME	varchar	20		×		
3	顧客住所	KOKYAKU_ADDRESS	varchar	50				
4	顧客電話番号	KOKYAKU_TEL	char	11			-(ハイフン)なしの11桁	0831239876

13.3 演習 (受注部品一覧)

次のような受注部品一覧から, テーブルの正規化・ER 図・エンティティ定義書を作成しなさい。

受注商品					商品構成			
商品コード	商品名	受注数	販売単価	小計	商品コード	仕入商品	仕入単価	必要数
2	パソコンMk-II	2	¥80,000	¥160,000				
					7	光学式マウス M-444	¥1,000	2
					10	USBキーボード K-2525	¥2,000	2
					13	メモリ DDR-1GB PC-12000	¥4,000	8
					17	ハードディスク 500GB HD-9000	¥3,000	4
					20	CPU CPU-X3	¥11,000	2
					14	ディスプレイ E-500	¥45,000	2
:	:	:	:	:	:	:		:
8	ボールマウス M-999	1	¥1,000	¥1,000				
					8	ボールマウス M-999	¥700	1
14	ディスプレイ E-500	2	¥50,000	¥100,000				
					14	ディスプレイ E-500	¥45,000	2



[演習 (受注部品一覧) 解答例 1]

●正規化

「商品マスタ」

商品コード	商品名	(販売)単価
2	パソコン Mk-Ⅱ	¥80,000
8	ボールマウス M-999	¥1,000
14	ディスプレイ E-500	¥50,000

「部品表」

商品コード	部品コード	必要数
2	7	1
2	10	1
2	13	4
2	17	2
2	20	1
2	14	1
8	8	1
14	14	1

「部品マスタ」

商品コード	商品名	(仕入)単価
7	光学式マウス M-444	¥1,000
8	ボールマウス M-999	¥700
10	USBキーボード K-2525	¥2,000
13	メモリ DDR-1GB PC-12000	¥4,000
14	ディスプレイ E-500	¥45,000
17	ハードディスク 500GB HD-9000	¥3,000
20	CPU CPU-X3	¥11,000

●ER図

●エンティティ定義書

	テーブル名	和名	商品マスタ			説明	商品の情報を格納	
		英名	SHOHIN_MASTER					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	商品コード	SHOHIN_CODE	integer		1	×		
2	商品名	SHOHIN_NAME	varchar	40		×		
3	単価	TANKA	numeric	18				

	テーブル名	和名	部品表			説明	商品の構成部品を保持	
		英名	BUHIN_HYO					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	商品コード	SHOHIN_CODE	integer		1	×		
2	部品コード	BUHIN_CODE	integer		2	×		
3	必要数	HITSUYO_SU	integer					

	テーブル名	和名	部品マスタ			説明	部品の情報を管理	
		英名	BUHIN_MASTER					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	部品コード	BUHIN_CODE	integer		1	×		
2	部品名	BUHIN_NAME	varchar	40		×		
3	単価	TANKA	numeric	18				

[演習 (受注部品一覧) 解答例 2]

●正規化

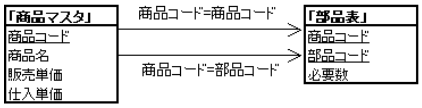
「商品マスタ」

商品コード	商品名	販売単価	仕入単価
2	パソコン Mk-Ⅱ	¥80,000	¥66,000
7	光学式マウス M-444		¥1,000
8	ボールマウス M-999	¥1,000	¥700
10	USBキーボード K-2525		¥2,000
13	メモリ DDR-1GB PC-12000		¥4,000
14	ディスプレイ E-500	¥50,000	¥45,000
17	ハードディスク 500GB HD-9000		¥3,000
20	CPU CPU-X3		¥11,000

「部品表」

商品コード	部品コード	必要数
2	7	1
2	10	1
2	13	4
2	17	2
2	20	1
2	14	1
8	8	1
14	14	1

●ER図



●エンティティ定義書

	テーブル名	和名	商品マスタ			説明	商品の情報を格納	
		英名	SHOHIN_MASTER					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	商品コード	SHOHIN_CODE	integer		1	×		
2	商品名	SHOHIN_NAME	varchar	40		×		
3	販売単価	HANBAI_TANKA	numeric	18				
4	仕入単価	SHIIRE_TANKA	numeric	18				

	テーブル名	和名	部品表			説明	商品の構成部品を保持	
		英名	BUHIN_HYO					
No.	項目名（和名）	フィールド名（英名）	型	桁数	PK	Null	説明	例
1	商品コード	SHOHIN_CODE	integer		1	×		
2	部品コード	BUHIN_CODE	integer		2	×		
3	必要数	HITSUYO_SU	integer					

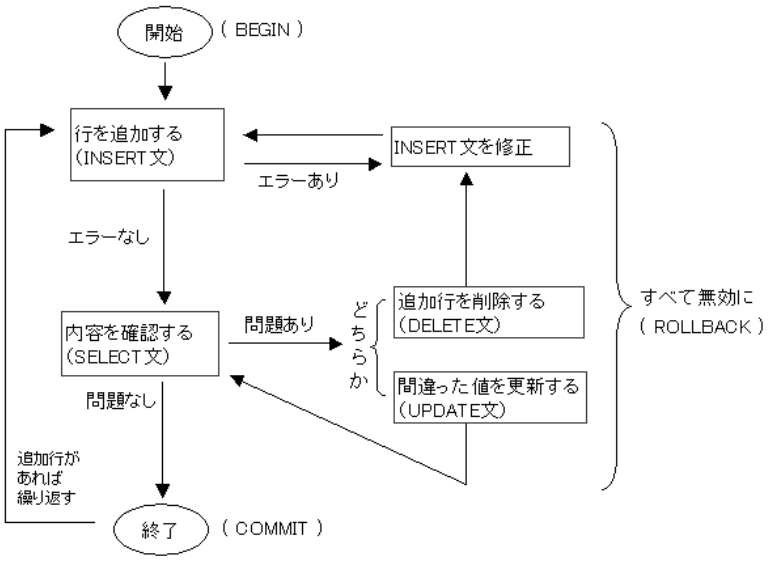
13.4　演習のためのデータ登録

「PC 組み立て販売」の演習を実施するために, 必要なデータを登録する作業（方法）について説明します.

13.4.1　登録の手順

まず登録したい値を行追加 (INSERT) します. 間違ったデータ型の値を登録しようとしたり, NOT NULL 制約が付加されているのに値がないなどの問題があれば, エラーとなって追加は実行されません. エラーが発生しなかった場合は, 正しく追加されたか抽出 (SELECT) で確認しましょう.

追加はされたが内容に間違いがあるような場合, トランザクション内であれば ROLLBACK を実行しやり直します. 既に確定している場合は行更新 (UPDATE) で間違いの値のみ更新するか, 一旦行削除 (DELETE) してから再度行追加 (INSERT) を実行します.



13.4.2　データ更新の注意点

行更新 (UPDATE) や行削除 (DELETE) は, 条件を間違えると対象行以外を更新/削除してしまいます. まず SELECT 文で対象行を抽出する条件の WHERE 句が正しいかどうかを確認してから, UPDATE 文や DELETE 文を組み立てると良いでしょう.

DELETE 文で WHERE 句を忘れた場合, テーブルの全行が削除されます.

例)「商品コード」が 3 の商品を「商品マスタ」テーブルから削除する

```
-> DELETE
   FROM shohin_master
   WHERE shohin_code = 3;
```

とするべきところを, WHERE 句の前にセミコロンを入力してしまったら…

```
-> DELETE
   FROM SHOHIN_MASTER;
```

エラーとはならずに実行してしまい,「商品マスタ」テーブルの全行が削除されてしまいます. まずは, SELECT 文で削除予定の対象行が抽出されるか確認をおこないます.

```
-> SELECT *
   FROM shohin_master
   WHERE shohin_code = 3;
```

確認後,「SELECT * 」の部分を「DELETE」に変更するという作業をすれば, 対象行以外を削除するような失敗を回避することができます.

UPDATE 文に対しても同じです. WHERE 句を忘れた場合, すべての行が対象になってしまいますので, まず SELECT 文で確認した後, その SELECT 文を UPDATE 文に加工するのが良いでしょう.

13.4.3 データ登録

実際に演習で必要なデータを登録しています.(登録するデータ：別紙参照)

<演習で利用するテーブル一覧>

商品マスタ (SHOHIN_MASTER)
自社マスタ (JISHA_MASTER)

仕入先マスタ (SHIIRESAKI_MASTER)
得意先マスタ (TOKUISAKI_MASTER)
在庫 (ZAIKO)
部品表 (BUHIN_HYO)
区分 (KUBUN)
発注 (HATCHU) *
発注品目 (HATCHU_HINMOKU) *
受注 (JUCHU) *
受注品目 (JUCHU_HINMOKU) *

*のテーブルは取引の途中に利用するもので初期状態ではデータがありません.

テーブルの詳細については「基本設計書」の「エンティティ一覧表」「エンティティ関連図 (ER 図)」「エンティティ定義書」を確認してください.

※「作成日時」「最終更新日時」

すべてのテーブルに, その行の作成された日時が格納される「作成日時」と最後に更新した日時が格納される「最終更新日時」があります. 現在の日時 (システム日付, システム時間) を INSERT 時に「作成日時」と「最終更新日時」両方に, UPDATE 時に「最終更新日時」にセットすることで, そのデータがいつ追加され, いつ最後の更新がされたかを管理することができます.

PostgreSQL に準備されている日付関数 (システム日付, システム時間) の一部

- current_date：システム日付を返す.
- current_time：システム時間を返す.
- current_timestamp：システム日付とシステム時間を返す.

「作成日時」「最終更新日時」は日付と時間の両方が必要ですから, current_timestamp を利用します.

SELECT 文で上記三つの関数を実行して結果を確認しましょう.

「select current_date;」
「select current_time;」

「select current_timestamp;」

データ（別紙参照）の登録作業を始めてください.

謝辞

本テキストは，IPA（独立行政法人 情報処理推進機構）の第 3 回 OSS モデルカリキュラム導入実証事業の一環として作成されたものである．

参考文献

[1] 可知　豊, 図解 オープンソースのことがわかる本, 日本実業出版社

[2] ずばりわかる！データベース SQL のルール 設計のルール, 日経ソフトウェア編

[3] ThinkIT-RDBMS 各々のメリットとデメリット比較-,
http://www.thinkit.co.jp/free/compare/8/1/

[4] 日本 PostgreSQL ユーザ会,
http://www.postgresql.jp/

[5] Let's Postgres,
http://lets.postgresql.jp/

[6] サン・マイクロシステムズ-MySQL-,
http://jp.sun.com/products/software/mysql/

[7] FireBird 日本ユーザー会,
http://firebird.gr.jp

[8] IT アーキテクト（軽量 RDBMS「Apache Derby」を使う）,
http://www.itarchitect.jp/enterprise/-/29481.html

[9] 石井 達夫, PostgreSQL 完全攻略ガイド, 株式会社技術評論社

ライセンス