SQL資料

古原作成

■1　データベースとは

データ管理の専用システムのことをデータベースと呼ぶ。データをさまざまな形で格納し、取り出しやすくしている。

■データベースの種類
・カード型データベース
・リレーショナルデータベース

カード型データベースはカードを単位としてデータを入力する。カード一枚に各項目があり、その内容を記述する。カードは表で言えば一行に該当する。

リレーショナルデータベースでは複数の表を使うことが出来る。表と表に関連（リレーション）を持たせて組み合わせて使用することが出来る。

個人がデータを管理するにはカード型データベースでも十分だが、本格的にデータを扱う場合、リレーショナルデータベース形式を使用する。

■2　DBMSとは?

DBMSはData Base Management Systemの略であり、データベース管理ソフトのことを言う。通常はアプリケーションソフトからDBMS経由でデータベースを扱う。

■主なDBMS
製品
・Oracle
・SQL Server

オープンソース
・MySQL / MariaDB
・PostgreSQL
・SQLite

■DBMSの役割
・データの整合性管理
・データの共有（マルチユーザ対応）
・トランザクション処理
・データベース操作のインタフェース提供

■DBMSを使う意味
・プログラムのデータ管理を楽にする
・データ管理の高速化（分散処理）
・データ管理処理とロジックの分離
・スケーラビリティ

■3　リレーショナルデータベースとは?

表（テーブル）を元にしてテーブルを組み合わせるデータ管理方法。複数の表が集まって一つのデータベースを構成する。

■表:テーブルの構造
・列（フィールド）項目を表す
・行（レコード）　項目の集まり１件を表す

■列:フィールド
列は「項目」を表し、その名前（列名）とどのようなデータを入れるのかを設定する。例えば、「商品名」の列は文字列を入れ、空欄が許されないなどの設定を行うことが出来る。

■行:レコード
行はデータ一件を表す。行を区別するためのキー（プライマリーキー）が必要になる。

■主キー:プライマリーキー
通常はどれかの列がプライマリーキー(主キー)を表す。プライマリーキーは各行を一意で示すことが出来るデータ。他の行と重複しない値を持つ。

■リレーション
プライマリーキーを組み合わせて表を関連づける。これをリレーションという。

例：
[商品テーブル]
sid：商品ID
sname：商品名
tanka：単価

[売り上げテーブル]
uid：売り上げID
sid：売り上げた商品のID
kosu：売り上げた個数
hi：売り上げた日

売り上げテーブルには売り上げの商品名や単価は含まれず、IDだけが含まれている。このIDを元に商品名、単価を求めることが出来る。

■4　テーブルの正規化

リレーショナルデータベースがテーブルを組み合わせてデータを扱えるように項目を分けることを正規化という。

■正規化する意味
・データの重複をなくす
・データの整合性を取りやすくする

■正規化の手順

・繰り返されるデータが出てこないように別テーブルにする
・計算で求められる項目は省く
・プライマリーキーをつける

例：
売り上げ表

商品名	単価	個数	金額	日付
りんご	100	5	500	2009/4/29
みかん	150	3	450	2009/4/29
いちご	200	1	200	2009/4/30
りんご	100	1	100	2009/4/30
みかん	150	2	300	2009/5/01
みかん	150	1	150	2009/5/02

・商品名と単価のセットは繰り返される項目なので別テーブルにする
・金額は計算できる項目なので省く
・商品と売り上げの各テーブルにプライマリーキーを付ける

■5　SQLとは?

SQL=Structured Query Language
データベース問い合わせ用の言語。
SQLでデータベースに命令を出し、その結果を受け取る。
命令を出すこと:SQLを発行する、クエリを発行する、問い合わせを行う、という言い方をする。

SQLはANSIやISOによって標準化されているが、細かい点ではDBMSに依存する点もある。

■SQLの例

```
SELECT 商品名 FROM 商品テーブル;
```

商品テーブルから「商品名」の列のみを取り出す。

■SQLの意義
・DBMSが変わっても原則としてSQLの変更の必要がない
・複雑な問い合わせが簡単に行える
・プログラム言語から発行可能

■6　MySQLの操作

6.1 MySQLとは

・世界中で広く利用されているDBMSの一つ。
・オープンソース（GNUライセンス）である。
・Webサイトでの利用が多い。
・LAMP（Linux, Apache, MySQL, PHP）の略称が広く知られるほど一般的に使用されている。

6.2 設定

my.ini を開き、以下の行のコメント（#）を外す

```
character_set_server=utf8
```

6.2 パスワードの設定

コマンドプロンプト（Shell）を開く

```
mysqladmin -u root password パスワード
```

を入力し、パスワードを設定する。

6.3 MySQLの操作方法

サービスを開始し、コマンドプロンプト（Shell）を開く

```
mysql -u root -p
```

を入力し、パスワードを入力する。

6.4 基本的な操作

データベースの一覧表示
```
SHOW DATABASES;
```

使用するデータベースの変更
```
USE データベース名;
```

現在のデータベースのテーブル一覧表示
```
SHOW TABLES;
```

文字コードの確認
```
SHOW VARIABLES LIKE 'char%';
```

■7　SQLの基礎

7.1 SQLの種類

■データ定義文（DDL=Data Definition Language）
CREATE：データベース、テーブルの作成
DROP：データベース、テーブルの削除
ALTER：データベース、テーブルの変更

■データ操作文（DML=Data Manipulate Language）
SELECT：行の検索
INSERT INTO：行の挿入・追加
UPDATE：行の変更
DELETE：行の削除

■データ制御文（DCL=Data Control Language）
COMMIT、ROLLBACKなどのトランザクション関連
GRANTなどのユーザ権限関連

7.2 SQLの基本

・命令の最後にはセミコロンを付ける
・半角英数を使用。（MySQLの場合、原則として大文字小文字を区別しない）
・文字列はシングルクォーテーション（'）またはダブルクォーテーション（"）で囲む
・複数のものを指定するときには原則としてカンマ（,）で区切る
・何も無い値は NULL と表現する

7.3 データ型

各フィールドにはデータ型が指定できる。
データ型はDBMSによって異なる。

■MySQLのデータ型

CHAR	固定長文字列(256文字まで)
VARCHAR	可変長文字列(65535文字まで)
TEXT	最大65535バイトまでの文字列
LONGTEXT	最大約42億文字（32bit）までの文字列
INT/INTEGER	整数（4バイト）
BIGINT	整数（8バイト）
FLOAT	浮動小数点数（4バイト）
DOUBLE	浮動小数点数（8バイト）
DATE	日付
DATETIME	日付と時刻
TIME	時刻

7.3 データベースの作成

■データベースの作成
CREATE DATABASE データベース名
例：
CREATE DATABASE hanbai;

■データベースの使用
コマンドの対象となるデータベースを切り替える。
USE データベース名;

例：
USE hanbai;

■データベースの削除
DROP DATABASE データベース名;

■８．テーブルの作成

■テーブルの作成
CREATE TABLE テーブル名
(列名1 データ型1 [制約],
 列名2 データ型2 [制約],
 列名3 データ型3 [制約],,,,);

例
```
CREATE TABLE shouhin (
    sid   INT,
    sname VARCHAR(40),
    tanka INT);
```

注：CHAR、VARCHARは文字数を指定する。

■現在のデータベースのテーブル一覧表示
SHOW TABLES;

■テーブルの列情報を表示
DESC テーブル名;

■テーブルの削除
DROP TABLE テーブル名;

■制約
PRIMARY KEY：プライマリキーとする
NOT NULL：NULL（データ無し）を許可しない
AUTO_INCREMENT：自動連番
DEFAULT [デフォルト値]：初期値設定

制約を付加した例
```
CREATE TABLE shouhin (
    sid   INT PRIMARY KEY,
    sname VARCHAR(40) NOT NULL,
    tanka INT DEFAULT 0);
```

8.1 行追加（INSERT）

INSERT INTO テーブル名 (列名 ,,,)
VALUES (値 ,,,)

列名と値の順番を対応させて記述する、

例：
INSERT INTO shouhin (sid , sname , tanka)
VALUES (1 , 'りんご', 200);

◎行追加の確認
SELECT * FROM shouhin;

■９ 行の検索(SELECT)

SELECT：行の抽出を行う

■全件、全列表示
SELECT * FROM テーブル名;

例：SELECT * FROM shouhin;

■列を指定して表示
SELECT 列名 FROM テーブル名;

例：SELECT sname,tanka FROM shouhin;

■重複を避けて表示
SELECT DISTINCT 列名 FROM テーブル名;

例：SELECT DISTINCT tanka FROM shouhin;

9.1 WHERE句

■条件を指定して表示
SELECT 列名 FROM テーブル名 WHERE 条件;

例：SELECT * FROM shouhin WHERE sid=1;

条件では等号や不等号を使うことが出来る。
文字列は''で囲む。

■比較演算子

A = B	AとBは等しい
A > B	AはBより大きい
A < B	AはBより小さい
A >= B	AはB以上
A <= B	AはB以下
A <> B	AとBは等しくない

■文字列の部分条件指定
LIKE演算子を使用し、文字列を部分検索できる。
例：
SELECT * FROM shouhin WHERE sname LIKE 'り%';

%は0文字以上の文字とマッチング
_は1文字の文字とマッチング

■どちらかの条件にあてはまる
WHERE 条件 OR 条件

例：SELECT * FROM shouhin WHERE sid = 1 OR sid =3;

sidが１または３があてはまる。

■両方の条件にあてはまる
WHERE 条件 AND 条件

例：SELECT * FROM shouhin WHERE sid >= 1 AND sid <=3;

sidが１以上でかつ３以下があてはまる。

■どれかと一致する条件
WHERE 列名 IN (値,値,,,);

例：SELECT * FROM shouhin WHERE sid IN (1,3);

sidが１または３があてはまる。
以下も同じ。

WHERE 列名 = ANY(値,値,,,);

例：SELECT * FROM shouhin WHERE sid = ANY (1,3);

■条件の範囲指定
WHERE 列名 BETWEEN 値 AND 値

例：
SELECT * FROM shouhin WHERE sid BETWEEN 2 AND 4;

２～４までがあてはまる。

■NULLの検索
NULLは = では検索できない

WHERE 列名 IS NULL

■否定（NOT）
LIKE や BETWEEN 、NOTを使用するときには、その前にNOTを付けて否定を表すことが出来る。

例：WHERE 列名 NOT LIKE 'り'

例：WHERE 列名 IS NOT NULL

■１０ 変更と削除

10.1 行の変更（UPDATE）

行の内容を変更する

UPDATE テーブル名 SET 列名= 値 , 列名 = 値,,,
WHERE 条件;

例：
UPDATE shouhin SET tanka = 100 WHERE sid = 1;

注意：
WHERE をつけない場合、全行が対象となる。

◎複数項目を対象とする場合、カンマで区切る

UPDATE shouhin SET tanka = 100,sname='なし'
WHERE sid = 2;

10.2 行削除（DELETE）

行を削除する

DELETE FROM テーブル名 WHERE 検索条件;

例：
DELETE FROM shouhin WHERE sid = 3;

注意：
WHERE をつけない場合、全行が対象となる。

■11 並べ替え

ORDER BY句を使用しSELECTの結果を並べ替えて表示

SELECT 列名 FROM テーブル名
ORDER BY 列名 並べ替え順;

[並べ替え順]には昇順の場合 ASC、降順の場合、DESC を指定する。注：ORDER BYはWHEREの後に付ける

例：
SELECT * from shouhin ORDER BY tanka ASC;

tankaの昇順で並べ替え。

例：複数キーを指定した場合
SELECT * from uriage ORDER BY sid ASC,hi ASC;

sidの昇順で並べ替え、同じ場合、hi の昇順

■12 計算と関数

12.1 計算

SQLでは計算（四則演算）を行うことが出来る。

例：SELECT tanka*2 FROM shouhin;

tankaを二倍にして表示 を行っている。
計算項目は列として表示される。

12.2 別名

計算式などでわかりにくい出力列の名前に別名を付けて変えることが出来る。

計算式 As 別名

例：
SELECT tanka*2 As nibai FROM shouhin;

tanka*2 に nibai という別名を付与。

注意:Asは省略も可能。単に空白で区切って別名を書く。

12.3 関数

SQLでは関数を使用することが出来る（関数はDBMSに依存するものが多い）。

◎算術関数

ABS　絶対値
FLOOR　切り捨て
CEIL　切り上げ
ROUND　四捨五入
MOD　剰余（ %演算子も使用可能）
POW　べき乗

例：
SELECT FLOOR(tanka*0.08) As TAX FROM shouhin;

tanka*0.08 を切り捨てした値をTAXという列で表示

◎文字列関数

CONCAT 文字列の連結
LENGTH 文字列の長さ
LOWER　小文字化
UPPER　大文字化

例：単価に"円"を付けて表示

SELECT CONCAT(tanka,"円") FROM shouhin;

◎日付・時刻関数
CURDATE 現在の日付
CURTIME 現在の時刻
NOW　　現在の日時
YEAR　　年を返す
MONTH　月を返す
DAY　　日を返す

例：今日の日付表示
SELECT CURDATE();

例：今日の日付で売り上げ挿入
INSERT INTO uriage (sid,kosu,hi)
　　VALUES(2,2,CURDATE());

例：売り上げがあった日
SELECT DAY(hi) FROM uriage;

◎その他
FORMAT カンマ区切り（値,小数点以下桁数）

12.4 集計関数

集計関数は対象行を計算した結果を表示する特殊な関数。

例：tankaの合計を取得

SELECT SUM(tanka) FROM shouhin;

SUM　　合計
MAX　　最大値
MIN　　最小値
AVG　　平均値
COUNT　個数（括弧内にDISTINCTをつけて重複排除）
　　列名指定：NULLではない行数をカウント
　　＊ ：全行数をカウント

■13 グループ化

13.1 グループ分けして集計

「GROUP BY 列名」と指定することで、その列の内容によって行をグループ分けし、そのグループごとに集計関数を使用することができる。

SELECT 集計関数（列名） FROM [テーブル名]
　　GROUP BY [グループ分けする列名];

例：SELECT hi,SUM(kosu) from uriage GROUP BY hi;

hi によってグループ分けし、kosu の合計を計算。

・GROUP BY はWHEREの後、ORDER BYの前に記述

・GROUP BY には複数項目を指定することが可能

・GROUP BY 指定時、SELECT で表示する列名は集計関数か、GROUP BY で指定された列しか使えない。

13.2 集計結果に条件付け

集計結果に条件付けを行うには HAVING を用いる。

SELECT 関数（列名） FROM テーブル名
GROUP BY グループ分けする列名 HAVING 条件;

※条件はWHEREと同様の書き方

例：SELECT hi,SUM(kosu) from uriage
　　GROUP BY hi
　　HAVING SUM(kosu) > 5

hi によってグループ分けし、kosu の合計が５より大きいものを表示。

・HAVING は GROUP BY の直後に書く

・HAVING と WHERE の違い

WHERE ：集計前に行を絞る
HAVING：集計後に行を絞る

処理の順番
WHERE → 集計 → HAVING

よって、集計関数をWHERE句に使うことは出来ないが、HAVING句に使うことは出来る。

■14 テーブルの結合

◎14.1 クロス結合

単純に２つのテーブルの全ての行の組み合わせを得るにはFROM にテーブル名を２つ書く。

```
SELECT 列名 FROM テーブル1,テーブル2
```

例：SELECT * FROM shouhin,uriage

この場合、全行の全組み合わせが表示される。
しかし、これは意味が無い。

◎14.2 WHEREを用いた結合

クロス結合に対し、WHERE を用いて意味がある行だけを絞り込む。

```
SELECT 列名 FROM テーブル1,テーブル2
WHERE テーブル1.列名 = テーブル2.列名;
```

```
例：SELECT * FROM shouhin,uriage
                WHERE shouhin.sid = uriage.sid;
```

両テーブルに同名の sid があるので、単に sid と書くとどちらのテーブルのsidか判断できない。このようなときには テーブル名.列名 のように書く。

行抽出の条件を付けるには AND でつなげる。

```
例：SELECT * FROM shouhin,uriage
        WHERE shouhin.sid = uriage.sid AND
              uriage.sid=1;
```

◎14.3 JOINを用いた結合

２つのテーブルをJOIN を用いて結合

```
SELECT 列名 FROM テーブル1
JOIN テーブル2
ON テーブル1.列名 = テーブル2.列名;
```

```
例：
SELECT * FROM shouhin JOIN uriage
                      ON shouhin.sid = uriage.sid;
```

行抽出の条件はWHERE で付ける

```
例：
SELECT * FROM shouhin JOIN uriage
                      ON shouhin.sid = uriage.sid;
                      WHERE uriage.sid=1;
```

◎14.4 JOINの便利な機能

■USING
USING を用いてキーを指定する結合も可能

```
例：
SELECT * FROM shouhin JOIN uriage USING (sid);
```

■NATURAL JOIN
JOINの代わりに NATURAL JOIN を用いることで自動的に両方の表にある同名の列をキーとみなしてくれる。

```
例：
SELECT * FROM shouhin NATURAL JOIN uriage ;
```

◎14.5 外部結合 (LEFT JOIN)

２つのテーブルをJOIN を用いて結合した際、両方にある行しか出力対象とされない。

片方のテーブルを常に出力する場合、外部結合を行う。例えば、LEFT JOIN を行うと左側に書いたテーブルにある行は全て出力される。右側のテーブルに対象がない場合、項目は全てNULLとされる。

```
SELECT 列名 FROM テーブル1
       LEFT JOIN テーブル2
       ON テーブル1.キー項目 = テーブル2.キー項目;
```

```
例：
SELECT * FROM shouhin LEFT JOIN uriage
                     ON shouhin.sid = uriage.sid;
```

なお、逆に右側のテーブルを常に出力する場合、RIGHT JOIN を用いる。

◎14.6 ３つのテーブルの結合

WHEREで３つのテーブルを結合するときには、WHEREでの条件が２つになりAND結合する。

```
SELECT hi,sname,cname FROM uriage,shouhin,category
       WHERE  uriage.sid = shouhin.sid AND
              shouhin.cid = category.cid;
```

JOINの場合、以下のようになる

```
SELECT hi,sname,cname FROM uriage
       JOIN shouhin ON uriage.sid = shouhin.sid
       JOIN category ON shouhin.cid = category.cid;
```

■15 サブクエリ

◎15.1 WHEREでのSELECTの使用

SELECT文をWHERE句内で使うことが出来る。
このときのSELECT文をサブクエリという。

SELECT 列名 FROM テーブル名
WHERE 列名 演算子 （SELECT 列名 FROM テーブル名2・・);

例：5/1に販売された商品名

```
SELECT sname FROM shouhin WHERE sid =
 (SELECT sid FROM uriage WHERE hi ='2016/05/01');
```

※複数ある場合、IN 演算子を使う。

例：5/1に販売された商品名

```
SELECT sname FROM shouhin WHERE sid IN
 (SELECT sid FROM uriage WHERE hi ='2016/05/01');
```

NOT IN を使うと、含まれない物を対象に出来る。

例：5/1に販売された商品以外の商品名

```
SELECT sname FROM shouhin WHERE sid NOT IN
 (SELECT sid FROM uriage WHERE hi ='2016/05/01');
```

■16 ビュー

複雑なSELECT文をビューとして保存しテーブルのように使用できる

```
  CREATE VIEW ビュー名 AS SELECT文
```

例：
以下のSQL文をビュー「slist」として作成

```
SELECT shouhin.sid,sname,tanka,kosu,tanka * kosu
AS kingaku,hi FROM shouhin,uriage
WHERE shouhin.sid=uriage.sid;
```

ビューの作成

```
CREATE VIEW slist AS SELECT
shouhin.sid,sname,tanka,kosu,tanka * kosu
AS kingaku,hi FROM shouhin,uriage
WHERE shouhin.sid=uriage.sid;
```

ビューの使用

```
  SELECT * FROM slist;
```

削除

```
  DROP VIEW ビュー名;
```

■17 トランザクション

トランザクション＝一連の処理
直訳＝処理・業務・取引

一連のSQL文をまとめて実行、キャンセルできる。

BEGIN; でトランザクションのスタート

ROLLBACK; でやりなおし（元に戻る）

COMMIT;　で確定

■18 テーブル定義の変更

■テーブルの削除
DROP TABLE テーブル名;

例：DROP TABLE shouhin;

■テーブル名の変更
ALTER TABLE テーブル名 RENAME 新テーブル名;

例：ALTER TABLE shouhin RENAME sho;

■テーブル列の追加
ALTER TABLE テーブル名 ADD 列定義;

例：ALTER TABLE shouhin ADD cid INT;

■テーブル：列の削除
ALTER TABLE テーブル名 DROP 列名;

■テーブル：PRIMARY KEYの追加
ALTER TABLE テーブル名 ADD PRIMARY KEY(列名);

例：ALTER TABLE shouhin ADD PRIMARY KEY(sid);

■テーブルの変更
ALTER TABLE テーブル名 CHANGE 旧列名 新列名 定義;

例：AUTO_INCREMENTに変更

```
ALTER TABLE shouhin CHANGE sid
                    sid INT AUTO_INCREMENT;
```

■19 MySQL特有の機能

◎19.1 最後に挿入したID

最後に挿入した行のIDは以下で取得できる。
SELECT LAST_INSERT_ID();

◎19.2 件数と開始位置の指定

SELECTで行を取得する際に、その件数を限定するには次のように指定する。

LIMIT 件数

例：SELECT * FROM shouhin LIMIT 10
先頭から１０件表示

また、件数だけで無く表示の開始位置を指定するには次のようにする。

LIMIT 開始位置,件数

例：SELECT * FROM shouhin LIMIT 20,10
２０件目から１０件表示
（０件目から数える）

◎19.3 ファイルのSQL実行

SOURCE ファイル名

例：
SOURCE c:¥work¥sql.txt

注：セミコロンはいらない

◎19.4 結果をファイル出力

SELECT の後に INTO OUTFILE ファイル名

例：
SELECT * FROM shouhin INTO OUTFILE
'c:¥work¥select.txt';

◎19.5 データのインポート

LOAD DATA INFILE ファイル名 INTO TABLE テーブル名

テーブルはあらかじめ作成しておく
ファイルはタブ区切り形式（utf-8）。
一行一レコード
注：改行コードを LF のみにする。

例：LOAD DATA INFILE 'c:¥¥work¥¥test.txt'
INTO TABLE shouhin;

◎19.6 バックアップと復元

ファイルにデータベースの内容をバックアップ
コマンドプロンプトから

mysqldump -u root -p データベース名 > ファイル名

例：
mysqldump -u root -p hanbai > hanbai.sql

■復元
mysql -u root -p データベース名 < ファイル名

例：
mysql -u root -p hanbai < hanbai.sql

注：
※復元先でデータベースはあらかじめCREATEしておく