SQL資料 SQLite用

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。

1.1 データベースの種類

・リレーショナルデータベース
・オブジェクトデータベース
・キーバリューストア（NoSQL）
・カード型データベース

リレーショナルデータベースでは複数の表を使ってデータを管理する。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。現在最も使われているデータベースの仕組みである。

1.2 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 Database
・Microsoft 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
（データ管理処理とロジックの分離）
・データ管理の高速化（分散処理）
・大量のデータ処理が可能（スケーラビリティ）

1.3 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

例：商品テーブル

商品 ID	商品名	単価
1	りんご	200
2	みかん	150
3	いちご	200

■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。上の例では「1 りんご 200」で１件のデータ（１レコード）になる。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。上の例では「商品ID」の列が主キーとなっている。

■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：売上テーブル

売上 ID	商品 ID	個数	日付
1	1	5	2020-09-30
2	2	3	2020-09-30
3	3	1	2020-10-01
4	1	1	2020-10-01
5	2	2	2020-10-01

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

1.4　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
・テーブルにデータを入れて扱えるようにする。
・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
・行と列の表の形でデータを入れられるようにする。
・繰り返されるデータが出てこないように別テーブルに分ける。
、ある列が決まれば他の列も決まる、というところを別テーブルに分ける
・各テーブルに主キーをつける。
・各テーブル間でリレーションができるように外部キーを設定する。
・計算で求められる項目は省く

例：売上データ
2020年9月30日
・りんご 単価100円×5個 金額500円
・みかん 単価150円×3個 金額450円
・いちご 単価200円×1個 金額200円

2020年10月1日
・りんご 単価100円×1個 金額100円
・みかん 単価150円×2個 金額300円

まず、以下のような形でテーブルにする。

例：売上テーブル

商品名	単価	個数	金額	日付
りんご	100	5	500	2020-09-30
みかん	150	3	450	2020-09-30
いちご	200	1	200	2020-10-01
りんご	100	1	100	2020-10-01
みかん	150	2	300	2020-10-01

・商品名が決まれば単価が決まる。そのセットは繰り返されている項目でもあり、別テーブルに分ける
・商品と売上の各テーブルに主キーを付ける
・商品テーブルと売上テーブルのリレーションが出来るように、売上テーブルに商品テーブルの主キーである商品IDを設定する（外部キー）。
・金額は計算できる項目なので省く

※Aが決まればBも決まる＝BはAに関数従属する、という。

■2　SQLiteの操作

2.1 SQLiteとは

・PHP、Python、祖マートフォンに内蔵されているリレーショナルデータベース
・簡易的にデータベースが使用可能
・ファイル１つで１つのデータベース

2.2 インストール

以下よりダウンロードする。

https://www.sqlite.org/download.html

Windowsの場合、「Precompiled Binaries for Windows」にある「sqlite-tools-win32-x86-3240000.zip」のようなファイルをダウンロード（数字はバージョンのため異なる）。

ダウンロードしたファイル内のsqlite3.exeを自分が作成したフォルダにコピーする

2.3 起動

コマンドプロンプトで

sqlite3 hanbai.db

を入力する。hanbai.dbは今回のデータベース名。

2.4 基本的な操作

現在のデータベースのテーブル一覧表示
.tables

終了
.exit

■3　SQLの基礎

3.1　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に依存する点も多い。

■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が変わっても同様のSQLが使用できる
・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

3.2 命令の種類

■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：行の追加
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

3.3 文法の基本

・半角英数を使用。原則として大文字小文字を区別しない。
※注：この資料では命令は大文字、それ以外は小文字で記述
・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
・コマンドラインでは命令の最後にはセミコロンを付ける

3.4 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。

■SQLiteのデータ型

●文字列　　TEXT
●整数　　INTEGER
●実数　　REAL

日付・時刻型は存在しないのでTEXTを使用。
データ型は省略することも可能。省略するとTEXTになる。
また、SQLite非対応のデータ型を指定した場合にもTEXTになる。

3.5 データの表記

SQLでデータを指定するときには以下のように書く。

●数値
そのまま書く　例：100　50.2

●文字列
''または""で囲む　例：'りんご'　"みかん"

●日付
''または""で囲む　例："2020-12-01"
区切り文字は何でも良いが、関数との互換性からは - を使う方が望ましい。

●時刻
''または""で囲む　例：'13:30:00'　"13:30"

●日時
''または""で囲む　例：'2020-09-17 13:30:00'

●NULL
何もデータが入っていないことを表す

■4 テーブル作成

4.1 テーブル作成と削除

■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   INTEGER,
    sname,
    tanka INTEGER);
```

■現在のデータベースのテーブル一覧表示
```
.tables
```

■テーブルの列情報を表示
```
.explain ON
```

■テーブルの削除
```
DROP TABLE テーブル名;
```

■制約
PRIMARY KEY：主キーとする
NOT NULL：NULL（データ無し）を許可しない
DEFAULT [デフォルト値]：初期値設定
UNIQUE：他と重複しない
AUTO_INCREMENT：自動連番

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

4.3 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：sidが１, snameがりんご、tankaが200を追加
```
INSERT INTO shouhin (sid , sname   , tanka )
             VALUES ( 1  , 'りんご', 200 );
```

◎行追加を確認するには
```
SELECT * FROM shouhin;
```

■5 行の検索(SELECT)

SELECT：行の抽出を行う

5.1 全行の表示

■全行、全列表示
```
SELECT * FROM テーブル名;
```

例：
```
SELECT * FROM shouhin;
```

■列を指定して表示(射影)
```
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;
```

5.2 条件による行の選択(WHERE)

■WHEREで条件を指定し行を絞り込む
```
SELECT 列名 FROM テーブル名 WHERE 条件;
```

例：sidが1の行を表示
```
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。
文字列や日付は''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。

■条件の範囲指定
値１～値２の範囲
WHERE 列名 BETWEEN 値1 AND 値2

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

■NULLの検索
NULLは = では検索できない。ISを使う。

WHERE 列名 IS NULL

■否定（NOT）
LIKE や BETWEEN 、NULL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

■6 変更と削除

6.1 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：sidが1の行の単価を100に
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

例：sidが4の行の単価を120に、snameを「かき」に
UPDATE shouhin SET tanka=120,sname='かき'
WHERE sid=4;

6.2 行削除（DELETE）

行を削除する

DELETE FROM テーブル名 WHERE 検索条件;

例：sidが4の行を削除
DELETE FROM shouhin WHERE sid = 4;

注意：
WHERE をつけない場合、全行が削除となる。

■7 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。省略時には昇順。

例：tankaの昇順で並べ替え。
SELECT * FROM shouhin ORDER BY tanka ASC;

複数のキーを指定することも可能。

例：sidの昇順、同じ場合にはhiの降順
SELECT * FROM uriage ORDER BY sid ASC,hi DESC;

・ORDER BYはSELECT文の最後に付ける

■8 計算と関数

8.1 計算

SQLでは計算（四則演算）を行うことが出来る。

例：tankaに0.08をかけて表示
```
SELECT tanka*0.08 FROM shouhin;
```

計算項目は列として表示される。

8.2 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列の名前を変えることが出来る。

計算式 As 別名

例：tanka*0.08 に tax という別名を付与。
```
SELECT tanka*0.08 AS tax FROM shouhin;
```

注意:ASは省略も可能。単に空白で区切って別名を書く。

例：tanka*0.08 に tax という別名を付与。
```
SELECT tanka*0.08 tax FROM shouhin;
```

8.3 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(値)　　絶対値
ROUND(値)　　四捨五入

例：tanka*0.08 を四捨五入し tax という列で表示
```
SELECT ROUND(tanka*0.08) AS tax FROM shouhin;
```

◎文字列関数

LENGTH(文字列)　　文字列の長さ
SUBSTR(文字列,開始,終了)　　開始～終了まで
LOWER(文字列)　　小文字化
UPPER(文字列)　　大文字化
REPLACE(文字列,置換前,置換後)　　置換

例：商品名の最初の２文字を表示
```
SELECT SUBSTR(sname, 1, 2) FROM shouhin;
```

例：日付の / を - に変える
```
UPDATE uriage SET hi = REPLACE(hi, '/', '-')
```

◎日付・時刻関数

現在の日付　　date('now','localtime')
現在の時間　　time('now','localtime')
現在の日時　　datetime('now','localtime')
年を取り出す　strftime('%Y',値)
月を取り出す　strftime('%m',値)
日を取り出す　strftime('%d',値)

例：今日の日付表示
```
SELECT date('now','localtime')
```

例：今日の日付でsidが2の商品が2個売れた
```
INSERT INTO uriage (sid,kosu,hi)
            VALUES(2,2, date('now','localtime'));
```

例：売り上げがあった日
```
SELECT strftime("%d", hi) FROM uriage;
```

8.4 集計関数

集計関数は対象行全てを集計した結果を表示する特殊な関数（集約関数ともいう）。

例：tankaの合計を取得

```
SELECT SUM(tanka) FROM shouhin;
```

SUM(列名)　　合計
AVG(列名)　　平均値
MAX(列名)　　最大値
MIN(列名)　　最小値
COUNT(*)　　全行数
COUNT(列名)　　NULLではない行数
COUNT(DISTINCT 列名)　同じものを数えない

■9 グループ化

9.1 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：hiごとにグループ分けし、kosu の合計を計算。
```
SELECT hi,SUM(kosu) FROM uriage GROUP BY hi;
```

・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。

・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

9.2 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。

■10 テーブルの結合

10.1 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、これは通常は意味が無い。

10.2 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;
```

10.3 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;
```

10.4 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 ;
```

10.5 外部結合

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(OUTER JOIN)を行う。例えば、LEFT OUTER 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（OUTERは省略可能）。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。

10.6 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;
```

■11 サブクエリ

WHERE句内でSELECT文を使うことが出来る。
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：単価が最高額の商品名
```
SELECT sname FROM shouhin WHERE tanka =
       (SELECT MAX(tanka) FROM shouhin);
```

※複数ある場合、IN 演算子を使う。

例：個数が３個以上売れた商品の商品名
```
SELECT sname FROM shouhin WHERE sid IN
       (SELECT sid FROM uriage WHERE kosu >= 3)
```

■12 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;
```

ビューの使用は通常のテーブルと同じ。
```
  SELECT * FROM slist;
```

ビューの削除はDROP VIEWを使う。
```
  DROP VIEW ビュー名;
```

■13 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。
（同一接続内でのみ有効）

BEGIN; でトランザクションの開始

INSERT、UPDATE、DELETE などの処理を行った後、

次のどちらかを選ぶ

ROLLBACK;
・結果をキャンセルしBEGIN以前の時点に戻す。

COMMIT;
・その結果を確定させる。

■14 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;

■列の追加
ALTER TABLE テーブル名 ADD COLUMN 列名 型 [制約];

例：ALTER TABLE shouhin ADD COULUMN cid INT;

■列名の変更
ALTER TABLE テーブル名 RENAME COLUMN 旧列名 TO 新列名

例：cidをcategoryに変更
ALTER TABLE shouhin RENAME COLUMN cid TO cateid

■列の削除
ALTER TABLE テーブル名 DROP COLUMN 列名;

例：ALTER TABLE shouhin DROP COLUMN cateid ;

■15 行数指定

SELECTで行を取得する際に、その行数を限定する。

LIMIT 行数

例：先頭から３行表示
SELECT * FROM shouhin LIMIT 3

行数だけで無く表示の開始位置を指定する。

LIMIT 開始位置,行数

例：2件目から3行表示
SELECT * FROM shouhin LIMIT 2,3
（０行目から数える）