サーバサイドプログラミング

3. SQL

コンテンツメディアプログラミング演習 II

2014年

菊池，斉藤

SQL概要

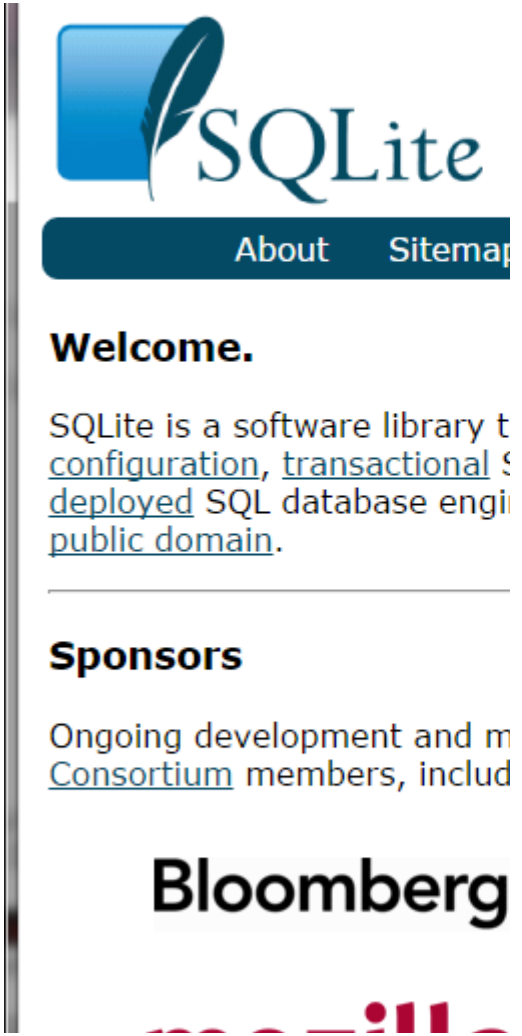
- SQL (Structured _____ Language)
  - リレーショナルデータベースの為のプログラミング言語.
  - IBMが提案し，1987にISO国際標準化.
  - データ定義，データベース操作（挿入，削除，選択），（トランザクション管理機能）
  - ケースインセンシティブ（大文字と小文字の区別なし）主に，予約語を大文字，変数を小文字で表記する.

SQLサーバの種類

サーバ名	特徴
Oracle	Oracle商用製品．シェアが高い．
Access	Microsoft社製品．Officeファミリー
	オープンソースRDB．Sun Micro開発．世界シェア1位．
PostgreSQL	オープンソースRDB．日本は利用が多い．
SQLite	オープンソースRDB データベースがファイル単位で構築されている．

SQLite 概要

- 概要
  - パブリックドメインの軽量なデータベースエンジン．現在はバージョン3 (3.8.6)
- 特徴
  - SQL92仕様に準拠
  - クライアントサーバ型ではない
    - » 他の言語から追加機能呼び出し
  - データ型は厳密ではない．



(1)コマンドラインツールでの実行

- sqlite3
  - コマンドプロンプトから起動する.

```
X:\> sqlite3　データベース名.sqlite
SQLite version 3.8.6 2014-08-15 11:46:33
Enter ".help" for usage hints.
sqlite>
```

  - .help　ヘルプの表示
  - .quit　終了
    (制御命令はドットから始まるSQLite固有)
  - 日本語文字コードは shift-JIS

(2) Nitrous.io での実行

データベース

- データベース
  - 一つのファイルで一つのデータベース
  - データベースに複数のテーブルが格納，（スキーマ，オート番号の最終値や統計情報）
  - 拡張子 .sqlite (他も許されている）
  - 例)Mts (Mountains) データベース

Mts.sqlite (sjis):
Mts0.sqlite (Utf-8)

列（カラム）→

NO	name	day	hour	height
1	谷川岳	0	5	1227
2	丹沢	0	6	1201
3	天城山	0	7	756
4	八ヶ岳	3	12	1409
5	那須岳	2	9	527
6	駒ヶ岳	0	4	316
7	燕岳	2	8	1313
8	奥穂岳	3	18	1685

行（レコード）→

2. データベースの検索

- SELECT文

  SELECT 列名 FROM テーブル名;

  ❑例1） テーブルの全行を出力

```
z:\> sqlite3 mts.sqlite
sqlite> select * from mts;
1|谷川岳|0|5|1227
2|丹沢|0|6|1201
3|天城山|0|7|756
4|八ヶ岳|3|12|1409
5|那須岳|2|9|527
6|駒ヶ岳|0|4|316
7|燕岳|2|8|1313
8|奥穂岳|3|18|1685
```

条件を付けた検索

- WHERE句

```
SELECT 列名 FROM テーブル名
WHERE 条件
```

  - 条件: 比較演算子(=, <, >, <>, >=),
    論理式 (AND, OR NOT)
  - 例2) IDが6以降を検索
    SELECT * FROM mts WHERE id >= 6;
  - 例3) IDが6以上で，日帰りが出来る(day が0)
    SELECT * FROM mts
    WHERE id >= 6 AND day = 0;

あいまい検索

- LIKE条件

```
SELECT  列名  FROM テーブル名
WHERE 列名 LIKE パターン
```

  - パターン: % = 任意の長さの文字列,
    _ = 任意の1文字
  - 例4) 「岳」で終わる山を検索
    SELECT * FROM mts where name like '%岳';
  - 例5) 2文字の山を検索
    SELECT * FROM mts
    where name like '_ _ _ _'; （注:間に空白が挿入）

列を選択して検索

- 射影 （指定された列のみのテーブル）

```
SELECT  列1,列2,..  FROM テーブル名
WHERE 条件
```

  - （数学的には，次元を下げる操作を指す）
  - 例6) 標高1300m以上の山名を検索せよ.

```
SELECT name,height  FROM mts
WHERE height > 1300;
八ヶ岳|1409
燕岳|1313
奥穂岳|1685
```

結果を加工

- 集合関数
  - » SUM　　総和
  - » AVG　　平均
  - » COUNT　　総数
  - » MAX,MIN　　最大値,最小値
- ❑ 例7) 標高の平均値を求めよ.
  SELECT AVG(height) FROM mts;
- ❑ 例8) 日帰りできる山の数を求めよ.
  SELECT COUNT(*) FROM mts WHERE day=0;

ソート

- ORDER

```
SELECT  列  FROM テーブル名
ORDER by 列 検索順;
```

  - 列でソートする.
    検索順 ASC = 昇順(小さな値から)
    _____ = 降順(大きな値から)
  - 例) IDで降順に並べる.
    SELECT * FROM mts ORDER by ID DESC;

グループ化

- GROUP

```
SELECT 列 FROM テーブル名
GROUP by 列;
```

  - 列で指定された行を束ねる.
  - 例) 日数ごとに山の数を数える.

    select day,count(*) from mts group by day;
  - 例）日数ごとに該当する山名を列挙する.

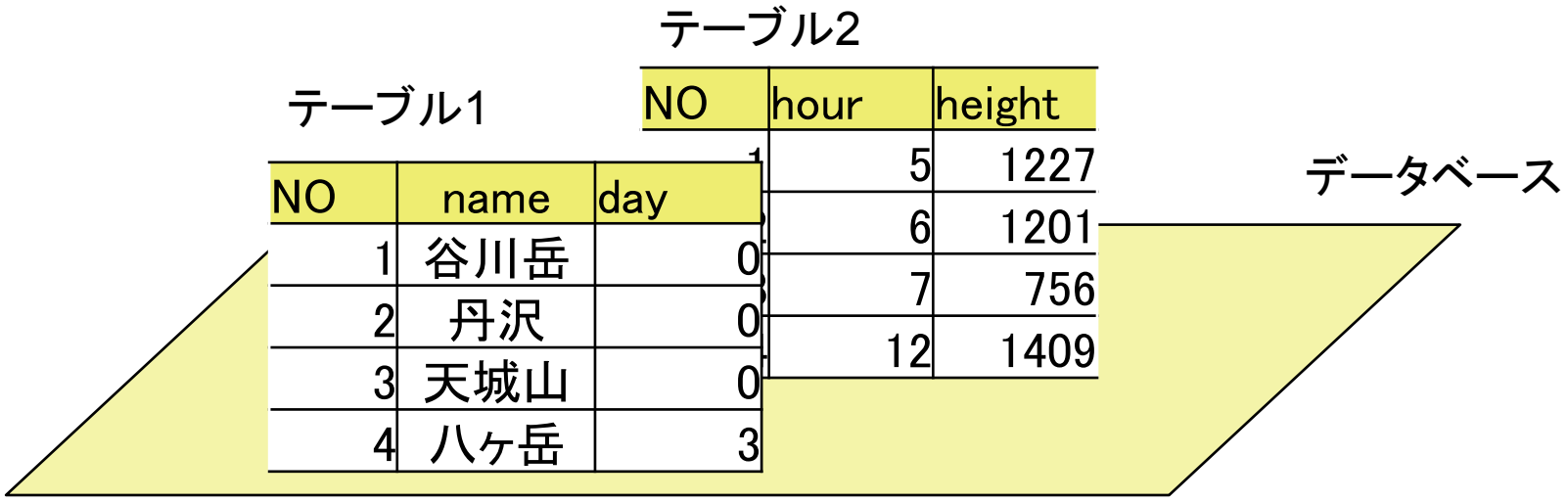
    select day,group_concat(name, ',') from mts group by day;

演習1

- mts.sqlite データベースを用いて，次を求めよ.
  (1) 日帰りできる山名と標高
  (2) 日帰りで，6時間以内で登山できる山の数
  (3) 標高が最も高い山の名前
  (4) 山名と標高のみの射影
  (5) 日数ごとの山の標高平均値

3. 複数のテーブルの結合

- データベースとテーブル
  - SQLite固有の管理コマンド
  - .databases　データベース一覧
  - .tables　格納されるテーブル一覧
  - .schema　スキーマ（列名と属性）表示



集合演算

- UNION, INTERSECT

  SELECT 列 FROM テーブル名1
  演算
  SELECT 列 FROM テーブル名2

  - 演算: UNION テーブル1 ∪ テーブル2
    INTERSECT テーブル1 ∩ テーブル2
  - 例) fms1 と fms2の和
    select * from fms1 UNION select * form fms2;
  - 例) fms1 と fms2の積
    select * from fms1 INTERSECT select * form fms2;

差集合（否定）

- EXCEPT

```
SELECT 列 FROM テーブル名1
EXCEPT
SELECT 列 FROM テーブル名2
```

  - テーブル1の要素でテーブル2の要素でないもの（テーブル1 – テーブル2)
  - 例) fms1 - fms2

    select * from fms1 EXCEPT select * form fms2;
  - 全体集合 U – A = Aの補集合（A以外）

結合 JOIN

- 結合
  - 複数の列を結合する操作．リレーショナルデータベースの基本機能．

第２正規化へのステップ（４）

受注顧客情報

受注ID	受注日	顧客ID	顧客名	住所
0001	11/10	100	斉藤 裕樹	中野区
0002	11/15	100	斉藤 裕樹	中野区
0003	11/20	200	小林 稔	国立市

受注明細

受注ID	商品ID	商品名	単価	数量
0001	S20	消しゴム	50	130
0001	T10	コピー用紙	450	50
0002	S20	消しゴム	50	80
0003	S10	えんぴつ	100	240

直積

- (Product)

  SELECT 列 FROM テーブル名1, テーブル名2

  - テーブル1x テーブル2 = 直積（テーブル1の各行にテーブル2の全行が結合）
  - 例) 受注顧客表 customersと受注表 ordersの直積

```
sqlite> select * from customers, orders;
1|11/10|100|斉藤裕樹|中野区|1|S20|消しゴム|50|130
1|11/10|100|斉藤裕樹|中野区|1|T10|コピー用紙|450|50
1|11/10|100|斉藤裕樹|中野区|2|S20|消しゴム|50|80
. . .
```

内部結合 (inner join)

- 列で行を対応させた新しい表を作る

```
SELECT 列 FROM テーブル名1, テーブル名2
WHERE テーブル1.列 = テーブル2.列
```

  - WHERE を として同じ.
  - 直積と異なり，列(ID)が同じ値の行同士のみを結合する.
  - 例) 受注IDで顧客と受注を結合する.
    sqlite> select * from customers, orders where customers.oid = orders.oid;

1|11/10|100|斉藤裕樹|中野区|1|S20|消しゴム|50|130
1|11/10|100|斉藤裕樹|中野区|1|T10|コピー用紙|450|50
2|11/15|100|斉藤裕樹|中野区|2|S20|消しゴム|50|80
3|11/20|200|小林稔|国立市|3|S10|えんぴつ|100|240

演習2

- stationary.sqlite データベースを用いて次を求めよ.
  (1) 全体の合計発注数
  (2) 発注者ごとの単価の合計
  (3) 商品ごとの発注者数の合計

  - 対応する列を自動的に決定する National Joinを用いてもよい.

  sqlite> select * from customers NATURAL JOIN orders;

4. テーブルの作成

- CREATE文

  CREATE TABLE テーブル名
  (列名1　データ型1，列名2 データ型2,…)

  - データ型:　　INTEGER　整数
    可変長文字列 (VARCHAR(n))
    CHARACTER(n)　　n文字固定長文字列
    (Date, time, float, decimal(m,n)など)
  - 例) ID, 氏名，住所からなる表fms1 を定義する.

```
CREATE TABLE fms1 (
  id integer,
  name varchar(20),
  addr   varchar(20)
);
```

行の挿入

- INSERT文

```
INSERT INTO テーブル名
VALUES(値1, 値2,..);
```

  - 値はテーブルの列の順番に指定する．文字列は'文字列' とする．欠損するところは，_____.
  - 例) 行を追加する.

```
INSERT INTO fms1 VALUES(100, '荒川 薫',
 '川崎市');
```

  - 挿入を確認するには，select文で検索する.
  - 例） 名前だけを指定して行を追加する.

```
INSERT INTO fms1(name) VALUES('斉藤裕樹');
```

行の削除

- DELETE文

```
DELETE FROM テーブル名
WHERE 条件;
```

  - 条件にマッチした行をテーブルから削除する.
  - 例) ID=100の行を削除する

```
DELETE FROM fms1 WHERE id=100;
```

Primary Key 主キー

- 主キー
  - 一つの表に一つだけ指定される列．行を一意に（同じキーの行が2個以上ない）決める．NULLは認められない．
  - 例）idを主キーに宣言する．

```
CREATE TABLE fms1 (
  id integer PRIMARY KEY,
  name varchar(20)
);
```

自動インクリメント

- AUTOINCREMENT
  - 主キーの_____を保障するために，IDを重複しないように自動生成する.
  - 例）名前だけでIDを自動生成.

```
sqlite> CREATE TABLE fms3 (
 id integer PRIMARY KEY AUTOINCREMENT,
name text );
sqlite> insert into fms3(name) values('菊池');
sqlite> insert into fms3(name) values('斉藤');
sqlite> select * from fms3;
1|菊池
2|斉藤
```

演習3

- 右の表をテーブル prof に格納したデータベース prof.sqlite を作れ.

(1) 斉藤の点数を90点に変更せよ(削除して挿入)

(2) 点数の高い順にソートせよ.

(3) 平均点を求めよ.

(4) 平均以下の学生名求めよ.

sid	学生名	点数
1	荒川	78
2	菊池	90
3	小林	95
4	小松	88
5	斉藤	80
6	鈴木	75
7	中村	85
8	宮下	92

演習4

- 次のExcelファイルをSQLデータベース pokemon.sqlite に登録せよ.
  - テーブル名 monsters
  - 主キー: No
  - name, typeは Text, 他は integer

	なまえ	ポイント	こうげき	ぼうぎょ	はやさ	タイプ
No	name	HP	offense	defense	speed	type
1	フシギダネ	45	49	49	45	L/P
2	ゼニガメ	44	48	65	43	W
3	ピカチュウ	35	55	30	90	E
4	ライチュウ	60	90	55	100	E
5	ピッピ	70	45	48	35	N
6	ニャース	40	45	35	90	N

課題5

- 次のデータベースから適当な表を選び，データベースに登録せよ.
  - 日本政府データカタログ Data.go.jp
  - 政府統計局 (県別面積、気象など) http://www.stat.go.jp/
  - 気象庁 (平均気温、桜開花など) http://www.jma.go.jp/
  - 少なくとも，列数5, 行数10の大きさのテーブルとする.
  - データベース名 mytable.sqlite

課題の提出

- 提出用フォルダー
  - CMP2\3SQL\2-組-番号
  - 課題3. prof.sqlite と 実行結果 ex3.txt
  - 課題4. pokemon.sqlite
  - 課題5. mydata.sqlite と説明書 mydata.doc

まとめ

- SQLは（　　）の為の標準化プログラミング言語．MySQLやSQLiteなどの各種サーバがある．
- データベースは複数の（　　）から成る．テーブルは行と列から成る．条件にあった行を取り出すには(　　)文を用いる．一部の列を取り出すことを（　　）という．
- 複数のテーブルを連結することを（　　）と呼ぶ．Primary Keyは主キーと呼ばれ，行を（　　）に決める列を指定する．
- テーブルの列を定義するには（　　）文を用いる．テーブルに行を追加するには（　　）文を用いる．

(参考)CSVファイルの入出力

- CSVファイルのインポート
  - CREATE文でテーブルを宣言.
    Create table mts(ID int, name text,…);
  - 区切りをカンマにする.
    sqlite> .separator ,
  - CSVファイルの読み込み
    sqlite> .import 'Mts.csv' mts
- CSVファイルへの書出し
  - 出力モードをCSVに変更し，ファイルmts2.csvに出力する.
    sqlite> .mode csv
    sqlite> .output mts2.csv
  - 元に戻す
    sqlite> .output stdout
  - (UTF-8になっているのでそのままではExcelで読めない)