(3) PythonでSQLite

専門演習B：Pythonによるデータ分析入門

データベース

データベース

- データベース
  - 再利用しやすいように管理されたデータの集合
  - 狭義：コンピュータ上でDBMSを用いて管理されるデータの集合
- データベース管理システム（DBMS）
  - データベースを管理するソフトウェア
- データモデル
  - データベースに格納するデータの構造
- データベース言語（↔ 問い合わせ言語）
  - DBMSに対して指示を与えデータベースを操作するための言語
  - 機能的にデータ操作・データ定義・データ制御の要素に分類される

データモデルに基づくデータベースの種類

- 関係データベース/リレーショナルデータベース（RDB）
- 階層型データベース
  - XMLデータベース（XMLDB）
- オブジェクト指向型データベース（OODB）
- NoSQL（Not only SQL）
  - ドキュメント指向データベース

関係データベース

- 表に似た構造でデータとその関係を管理する

学生

学籍番号	名前	所属
150101	山田	1
150151	伊庭	2
150021	佐藤	1
150059	松田	3
150037	鈴木	3
150083	佐藤	4

所属

講座番号	名前
1	情コミ
2	異文化
3	地域
4	現代

代表的なDBMS

- 関係データベース
  - 商用：Oracle DB，Microsoft SQL Server
  - オープンソース：MySQL，PostgreSQL，SQLite
- XMLデータベース
  - Xindice（Apache）
- NoSQL
  - MongoDB，HBase（Apache），Cassandra（Apache）
  - Google BigTable，Amazon DynamoDB（一般利用はできない）

SQLite

- オープンソースのRDBMS
- アプリケーションに組み込んで利用される
  - 他の多くのDBMSはクライアント・サーバ方式
  - コマンドライン版もある
- SQL92の機能の多くを実装

データベース言語SQL

- RDBMSのためのデータベース言語
- ISOによって言語仕様の標準化が行われている

よく使うSQL文

- データ定義
  - テーブルの作成：CREATE TABLE
  - テーブルの削除：DROP TABLE
  - テーブルの変更：ALTER TABLE
- データ操作
  - データの挿入：INSERT INTO
  - データの更新：UPDATE
  - データの削除：DELETE FROM
  - データの検索：SELECT

SQLite3を使ってみる

次のようなデータベースを作ってみる

学生

学籍番号	名前	所属
140101	山田	1
130151	伊庭	2
130021	佐藤	1
140037	鈴木	3
130083	佐藤	4
150170	田中	

所属

講座番号	名前
1	情報
2	異文化
3	地域
4	現代

コマンドライン版のSQLite3の起動と終了

```
$ sqlite3 student.db
SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter ".help" for usage hints.
sqlite> .quit
$
```

テーブルの作成

学生テーブルを作る

```
sqlite> create table student(
   ...> id integer primary key,
   ...> name text,
   ...> program integer);
```

主キーの指定（重複しない値）

講座テーブルを作る

```
sqlite> create table program(
   ...> id integer primary key,
   ...> name text);
```

どんなテーブルがあるか確認

テーブルの名前

```
sqlite> .tables
program  student
sqlite>
```

テーブルのスキーマの確認

```
sqlite> .schema student
CREATE TABLE student(id integer primary key, name text,
program integer);
sqlite>
```

データの挿入（学生テーブル）

学生データの挿入

```
sqlite> insert into student values(140101, "Yamada", 1);
sqlite> insert into student values(130151, "Iba",    2);
...
```

欠損のあるデータの挿入

【方法1】

```
sqlite> insert into student values(150170, "Tanaka", null);
```

【方法2】

```
sqlite> insert into student(id, name) values(150170, "Tanaka");
```

データの挿入（講座テーブル）

講座データの挿入

```
sqlite> insert into program values(1, "Joho");
sqlite> insert into program values(2, "Ibunka");
sqlite> insert into program values(3, "Chiiki");
sqlite> insert into program values(4, "Gendai");
```

データの検索(1)

全データの表示（学生テーブル）

```
sqlite> select * from student;
130021|Sato|1
130083|Sato|4
130151|Iba|2
...
```

学番(id)が140000より小さい学生の検索

```
sqlite> select * from student where id < 140000;
130021|Sato|1
130083|Sato|4
130151|Iba|2
```

データの検索(2)

学番が140000以上150000未満の学生の学番と名前

```
sqlite> select id,name from student
   ...> where 140000 <= id and id < 150000;
140037|Suzuki
140101|Yamada
```

表の統合(1)

所属講座がある全学生データの表示

```
sqlite> select * from student inner join program on
student.program == program.id;
130021|Sato|1|1|Joho
130083|Sato|4|4|Gendai
130151|Iba|2|2|Ibunka
140037|Suzuki|3|3|Chiiki
140101|Yamada|1|1|Joho
```

＊15年度生の「田中」は所属講座がないので表示されない。

表の統合(2)

全学生データの表示（学番，名前，講座）

```
sqlite> select student.id,student.name,program.name from
student left join program on student.program == program.id;
130021|Sato|Joho
130083|Sato|Gendai
130151|Iba|Ibunka
140037|Suzuki|Chiiki
140101|Yamada|Joho
150170|Tanaka|
```

SELECTに名前を設定

前ページのSELECT分に student_program という名前を設定

```
sqlite> create view student_program as
   ...> select student.id,student.name,program.name
   ...> from student left join program
   ...> on student.program == program.id;
```

＊ビューは概ねテーブルと同様に扱える。

PythonでSQLite3を使う

簡単なサンプルプログラム

```
#!/usr/bin/env LC_ALL=ja_JP.utf-8 python
# -*- coding: utf-8 -*-

import sqlite3
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

def insertItem(name, price):
    cur.execute(u"insert into 商品(品名, 単価) values(?, ?)", (name, price))

conn = sqlite3.connect("test_data.db")
cur = conn.cursor()

sql = u"""
create table 商品(
    id integer primary key autoincrement,
    品名 text,
    単価 integer
)"""
cur.execute(sql)
```

```
insertItem(u"リンゴ", 100)
insertItem(u"梨"   , 150)
insertItem(u"イチゴ", 300)

c = cur.execute(u"select * from 商品")
for id, name, price in c:
    print("%s: %s: %s"%(id, name, price))

conn.commit()
conn.close()
```

SQLite3利用の基本

【初めにやること】

・ SQLite3ライブラリのインポート

```
import sqlite3
```

・ データベースファイルへの接続

```
conn = sqlite3.connect(データベースファイル名)
```

【最後にやること】

・ データベースの更新（ファイルへの書き込み）

```
conn.commit()
```

・ データベースファイルとの接続の終了

```
conn.close()
```

SQLite3でのSQL文の実行

- カーソル（SQL実行用オブジェクト）の生成

```
cur = conn.cursor()
```

- SQL文の実行

```
cur.execute(SQL文)
```

← 日本語を含む場合は Unicode 文字列

SQLite3でのSQL文の使い方

・戻り値のない場合

```
cur.execute(u"insert into 商品 values(?,?,?),(1,u"りんご",100))
```

・戻り値のある場合（戻り値はカーソル）

```
c = cur.execute(u"select id,品名 from 商品 where 単価 < 200")
```

・戻り値を for 文で表示

```
for id,name in c:
    print(u"id:%s 品名:%s"%(id,name))
```

・戻り値をリストで取得

```
items = c.fetchall()
```