DB연동

파이썬에서는 여러 종류의 DB를 지원하고 있다. 사용을 위해서는 각각의 DB에 해당하는 프로그램을 pip3를 통해 다운로드 및 설치해야 한다.

sqlite

sqlite3 모듈은 파이썬 표준 라이브러리로 별도의 설치없이 기본적으로 사용할 수 있다. 그러므로 import만 하면 데이터베이스를 사용할 수 있다.

sqlite는 작은 데이터를 저장할때 편리하고, 퍼블릭 라이센스이므로 어떤 목적으로든 사용할 수 있는 RDB(관계형데이터베이스) 이다.

관리툴 : DB Browser for SQLite [다운로드]

자료형

SQLite3 자료형	파이썬 자료형	용도	설명
NULL	None	공백	
INTEGER	int	정수형	별도로 크기를 지정하지 않는다. 두가지 자료형을 혼용해서 사용할 수 있다.
REAL	float	실수형	
TEXT	str, bytes	문자형	
BLOB	buffer	바이너리 데이터	이미지와 같은 파일 자체를 저장

DB연결 및 해제

```
# sqlite3 모듈을 불러온다.
import sqlite3

# "my_home.db" 파일을 데이터베이스로 연결하여 사용한다.
con = sqlite3.connect("my_home.db")

# 커서를 생성한다.
cur = con.cursor()

# 데이터베이스를 종료한다.
con.close()
```

- Connection 객체는 DB와의 연결을 주로 처리한다.
- Cursor 객체는 데이터를 저장 및 조회를 담당한다.

테이블생성

아래 2개의 테이블은 동일한 자료형을 가진 테이블이 된다.

```
#SQLite3 자료형을 사용
cur.execute("CREATE TABLE my_table1 (name TEXT, age INTEGER, money REAL)")

#파이썬의 자료형을 사용
cur.execute("CREATE TABLE my_table2 (name str, age int, money float)")

#기존에 생성된 테이블이 있는지 확인한 후 생성
cur.execute("CREATE TABLE IF NOT EXISTS my_table3 (name str, age int, money float)")
```

레코드 삽입

```
#기본형
cur.execute("INSERT INTO my_table1 VALUES ('홍길동', 10, 15000.99)")

#1개의 튜플 입력
cur.execute("INSERT INTO my_table1 VALUES (?,?,?)", ('홍길동', 10, 15000.99))

#2개이상의 튜플 입력
tu_record = (
        ('이순신', 10, 15000.99),
        ('성삼문', 20, 46000.99),
        ('류성용', 30, 99000.99)
)
cur.executemany("INSERT INTO my_table1 (name, age, money) VALUES (?,?,?)", tu_record)
```

- insert문 작성시 컬럼은 생략가능함
- 1개의 레코드 입력시 execute(), 여러개의 레코드 입력시 executemany() 사용

레코드 조회1

```
cur.execute("SELECT * FROM my_table1")
```

```
print(cur.fetchone())
print(cur.fetchall())
```

- 레코드를 한개씩 인출할때는 fetchone() 사용
- 레코드 전체를 한꺼번에 인출할때는 fetchall()을 사용한다. 단, 위와같이 fetchone()을 먼저 실행하였다면 첫번째 레코드는 이미 인출되었으므로 두번째 레코드부터 인출된다.

레코드 조회2

```
# 방법1
param1 = ('홍길동',)
cur.execute('SELECT * FROM my_table1 WHERE name=?', param1)
print('방법1', cur.fetchall())

# 방법2
param2 = '류성용'
cur.execute("SELECT * FROM my_table1 WHERE name='%s'" % param2)
print('방법2', cur.fetchall())

# 방법3
param3 = ('류성용', '이산')
cur.execute('SELECT * FROM my_table1 WHERE name IN(?,?)', param4)
print('방법4', cur.fetchall())

# 방법4
cur.execute("SELECT * FROM my_table1 WHERE name IN('%d','%d')" % ('류성용', '이산'))
print('방법5', cur.fetchall())

# 방법6
cur.execute("SELECT * FROM my_table1 WHERE name=:n1 OR name=:n2", {"n1":'류성용',
"n2":'이산'})
print('방법6', cur.fetchall()
```

- 서식문자는 Java와 동일하게 정수는 %d, 실수는 %f, 문자는 %s 를 사용한다.
- 방법3, 방법6은 딕셔너리를 사용한다.
- 대부분은 문자열 포매팅에서 다루었던 내용들이다.

레코드 수정

```
# 방법 1
c.execute("UPDATE my_table1 SET name=? WHERE age=?", ('김연아', 10))
# 방법 2
c.execute("UPDATE my_table1 SET name=:name WHERE age=:age", {"name": '이상화', 'age': 20})
# 방법 3
c.execute("UPDATE my_table1 SET name='%s' WHERE age=%d" % ('박찬호', 30))
```

레코드 삭제

```
# 방법 1
c.execute("DELETE FROM my_table1 WHERE age=?", (10,))
# 방법 2
c.execute("DELETE FROM my_table1 WHERE age=:age", {'age': 20})
# 방법 3
c.execute("DELETE FROM my_table1 WHERE age=%d" % 30)
```

예제] 17sqlite.py

```
import sqlite3

conn = sqlite3.connect('dbase1')
curs = conn.cursor()

tblcmd = 'create table people (name char, job char, pay int)'
curs.execute(tblcmd)

curs.execute('insert into people values (?,?,?)', ('이순신','장군',500))
```

```
curs.executemany('insert into people values (?,?,?)',
        [('강감찬','장군',700) , ('홍길동','의적',800)])

rows = [['곽재우','의병',1100],['유성룡','재상',1200],['임꺽정','의적',1500]]

for row in rows :
    curs.execute('insert into people values (?,?,?)', row)

conn.commit()
```

```
curs.execute('select * from people')
print(curs.fetchall())
print('---------------------------------')


curs.execute('select * from people')
for row in curs.fetchall():
    print(row)
print('---------------------------------')


curs.execute('select * from people')
for(name, job, pay) in curs.fetchall():
    print(name, ':', job, ':', pay)
```

MySQL(or MariaDB)

파이썬에서 MySQL 에 접속하기 위해서는 pip3 명령으로 PyMySQL 패키지를 설치해야한다.

pip에 대한 설명은 02.개발환경-JupyterNotebook 문서를 참조한다.

```
>>> pip3 install pymysql
```

MySQL접속

```
# root 계정으로 mysql에 접속한다.
c:\> mysql -u root -p mysql;
```

- mysql은 시스템 데이터베이스로 생성된 db, user, 권한 등을 관리한다.
- 오라클의 system계정과 동일한 역할을 한다.

DB계정 생성

DB명 : sample_db / 아이디 : sample_user / 패스워드 : 1234 로 변경하세요

```
# 데이터베이스 생성
MariaDB [mysql]> CREATE DATABASE 데이터베이스명;
# 데이터 베이스 확인
MariaDB [mysql]> SHOW DATABASES;
MariaDB [mysql]> Use 데이터베이스명;
```

- 오라클은 계정만 생성하여 접속하는 형태지만, MySQL은 DB계정을 생성하여 접속해야 한다.

사용자계정 생성

```
# 사용자확인
MariaDB [mysql]> SELECT host, user, Grant_priv FROM user; # y 인것 확인


select user, host from user;


# 사용자계정 생성
MariaDB [mysql]> CREATE USER '사용자계정명'@'localhost' IDENTIFIED BY '패스워드';
or
MariaDB [mysql]> CREATE USER '사용자계정명'@'%"  IDENTIFIED BY '패스워드';
```

- 사용자계정은 user테이블에서 관리된다.

- 앞에서 생성한 DB에 접근할 수 있는 사용자계정을 만든다.
- 로컬에서만 접속할때는 'id'@'localhost' 형태로 생성한다.
- 만약 외부에서도 접근해야 한다면 '%'를 사용한다. 맥에서는 이렇게 사용하는 것이 좋음.
- root 계정은 모든 DB를 접근할 수 있는 권한을 가진 관리자계정이다.

사용자 권한 주기

```
MariaDB [mysql]> GRANT ALL PRIVILEGES ON 데이터베이스명.* TO
'사용자계정명'@'localhost';
or
MariaDB [mysql]> GRANT ALL PRIVILEGES ON 데이터베이스명.* TO  '사용자계정명'@'%' with
GRNT OPTION;
```

- 데이터베이스명.* 은 해당DB의 모든 테이블을 의미한다.
- # with GRNT OPTION : 다른사용자에세 권한 부여

변경된 내용을 적용

```
MariaDB [mysql]> FLUSH PRIVILEGES;
```

DB 연결 및 해제

```
import pymysql


conn = pymysql.connect(host='localhost', user='user아이디', password='패스워드'
    ,db='db명', charset='utf8'
    , cursorclass=pymysql.cursors.DictCursor)
```

- cursorclass=pymysql.cursors.DictCursor 를
    - 기술하면 레코드가 딕셔너리 형태로 출력된다.
    - 제거하면 튜플 형태로 출력된다.

레코드 삽입, 조회, 수정, 삭제

사용법은 SQLite와 동일하므로 생략한다. 절차는 아래와 같다.

1. 패키지 임포트
2. MySQL 연결

3. 커서객체 준비
4. 쿼리문(CRUD)
5. 쿼리 실행
6. insert, update, delete인 경우
    a. commit() or rollback()
7. select인경우
    a. fetchone() or fetchall()
8. 연결 해제

```
#회원테이블(부모)
CREATE TABLE member
(
  id VARCHAR(30) NOT NULL,
  pass VARCHAR(30) NOT NULL,
  name VARCHAR(30) NOT NULL,
  regidate datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
);

#게시판테이블(자식)
CREATE TABLE board
(
  num INT NOT NULL AUTO_INCREMENT,
  title VARCHAR(100) NOT NULL,
  content TEXT NOT NULL,
  id VARCHAR(30) NOT NULL,
  postdate DATETIME DEFAULT current_timestamp,
  visitcount MEDIUMINT NOT NULL DEFAULT 0,
  PRIMARY KEY (num)
);

#외래키 설정
ALTER TABLE board ADD constraint fk_board_member
  FOREIGN KEY (id) REFERENCES member (id);



#더미데이터 삽입
INSERT INTO member (id, pass, NAME) VALUES
  ('korea', '1234', '대한민국');

#부모테이블인 member에 먼저 삽입한 후 자식테이블인 board에 삽입한다.
INSERT INTO board (title, content, id)
  VALUES ('제목입니다1', '내용입니다1', 'korea');
INSERT INTO board (title, content, id)
  VALUES ('제목입니다2', '내용입니다2', 'korea');
INSERT INTO board (title, content, id)
  VALUES ('제목입니다3', '내용입니다3', 'korea');
INSERT INTO board (title, content, id)
  VALUES ('제목입니다4', '내용입니다4', 'korea');
INSERT INTO board (title, content, id)
```

```
  VALUES ('제목입니다5', '내용입니다5', 'korea');


#전체 데이터 확인하기
SELECT * FROM member;
SELECT * FROM board;
```

예제] 18mysql01.py

DB명 : sample_db / 아이디 : sample_user / 패스워드 : 1234 로 변경하세요

```
import pymysql

conn = pymysql.connect(host='localhost', user='sample_user',
            password='1234', db='sample_db', charset='utf8')

curs = conn.cursor()

try:
    sql = "SELECT * FROM board"
    curs.execute(sql)

    rows = curs.fetchall()
    print(rows)
```

```
    print('출력1', '-'*40)
    for row in rows:
        print(row)

    print('출력2', '-'*40)
    for row in rows:
        print(row[0], row[1], row[2], end=" ")
        pdate = row[3]
        id = row[4]
        vcnt = row[5]
        print("%s, %s, %s" % (pdate, id, vcnt))

```

```
    print('출력3', '-'*40)
    sql = sql + " WHERE title like '%{0}%' ".
        format(input('검색어입력:'))
    curs.execute(sql)
    rows = curs.fetchall()
    print(rows)

except Exception as e:
    print("쿼리 실행시 오류발생", e)
```

```
print('-'*40)
conn.close()
print('자원반납')
```

예제] 18mysql02.py

```
import pymysql
conn = pymysql.connect(host='localhost', user='sample_user',
                       password='1234', db='sample_db',
                       charset='utf8')
curs = conn.cursor()

sql = f"""INSERT INTO board (title, content, id)
    VALUES ('{input('제목:')}', '{input('내용:')}', 'korea')"""
try:
    curs.execute(sql)
    conn.commit()
    print("1개의 레코드가 입력됨")
except Exception as e:
    conn.rollback()
    print("쿼리 실행시 오류발생", e)

conn.close()
```

예제] 18mysql03.py

```python
import pymysql

conn = pymysql.connect(host='localhost', user='sample_user',
                       password='1234', db='sample_db',
                       charset='utf8')

curs = conn.cursor()

#쿼리 작성시 format() 함수를 통해 문자열을 인덱스로 매칭한다.
sql = """update board
         set title='{1}', content='{2}'
         where num={0}
      """.format(input('수정할일련번호:'), input('제목:'), input('내용:'))
try:
    curs.execute(sql)
    conn.commit()
    print("1개의 레코드가 수정됨")
except Exception as e:
    conn.rollback()
    print("쿼리 실행시 오류발생", e)


conn.close()
```

여기까지 작성하세요.

예제] 18mysql04.py

```python
import pymysql

conn = pymysql.connect(host='localhost', user='sample_user',
                       password='1234', db='sample_db',
                       charset='utf8')

curs = conn.cursor()

#f-String으로 delete쿼리문 작성
sql = f"""delete from board where num='{input('삭제할일련번호:')}'"""

try:
    curs.execute(sql)
    conn.commit()
    print("1개의 레코드가 삭제됨")
except Exception as e:
    conn.rollback()
    print("쿼리 실행시 오류발생", e)

conn.close()
```