12. 데이터베이스

1. SQLite
2. Oracle
3. MySQL

1. SQLite

- **SQLit**
  - 공식사이트 : http://sqlite.org
  - 개발자 : 리처드 힙(Richard Hipp)이 2000년 8월 발표 C언어로 개발
- **특징**
  - 파일기반 DBMS, 저메모리, 빠른 처리 속도
  - 오픈소스
  - 별도의 DB 서버가 없어도 쉽고 편리하게 사용할 수 있는 Embedded SQL 엔진
  - 안드로이드, 아이폰 등의 스마트폰에 내장된 DB
  - 표준 SQL 지원

1. SQLite

- **SQLite에서 지원하지 않는 기능 (https://www.sqlite.org/omitted.html)**
  - RIGHT and FULL OUTER JOIN : left outer join은 가능함
  - Complete ALTER TABLE support
  - Complete trigger support
  - Writing to VIEWs : 읽기 전용 뷰만 가능
  - GRANT and REVOCK
- **SQLite 클라이언트 툴**
  - http://www.sqliteexpert.com/
  - Personal 64bit 버전 다운로드 및 설치

1.SQLite

- **SQLite 사용**
  - table 생성

```
import sqlite3
print(sqlite3.version)
print(sqlite3.sqlite_version)
```

```
import sqlite3 #SQLite3 라이브러리 로딩
# 테이블 생성
def create_table():
    conn=sqlite3.connect('my_books.db') #데이터베이스 커넥션 생성
    cursor=conn.cursor() # 커서 생성
    #my_books 테이블 생성, 제목, 출판일자,출판사, 페이지수, 추천여부
    cursor.execute('''create table if not exists books(
        title text,
        published_date text,
        publisher text,
        pages integer,
        recommend integer
        )''')
    conn.commit()
    conn.close()
create_table()
```

1. SQLite

데이터 입력

```
import sqlite3
#데이터 입력 함수
def insert_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("insert into books values('Java','2019-05-20','길벗',500,10)")
    sql='insert into books values(?,?,?,?,?)'
    cursor.execute(sql, ('Python','201001','한빛',584,20))
    items=[
        ('빅데이터','2014-07-02','삼성',296,11),
        ('안드로이드','2010-02-02','영진',526,20),
        ('Spring','2013-12-02','삼성',248,15)
    ]
    cursor.executemany(sql, items)
    conn.commit()
    conn.close()
insert_books()
```

1. SQLite

전체 데이터 출력

```
import sqlite3
def all_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select * from books")
    print('[1] 전체 데이터 출력하기')
    books=cursor.fetchall()
    print(type(books))
    print(len(books))

    for book in books:
        print(book)
    conn.close()

all_books()
```

1. SQLite

레코드 개수 정하여 출력

```
import sqlite3
# 데이터 개수 지정하여 출력
def some_books(number):
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select * from books")
    books=cursor.fetchmany(number)
    for book in books:
        print(book)
    conn.close()
some_books(3)
```

```
import sqlite3
1개의 데이터 출력
def one_book():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select * from books")
    book=cursor.fetchone()
    print(type(book))
    print(book)
    conn.close()
one_book()
```

1. SQLite

조건 지정 및 정렬하여 검색

```
import sqlite3
def big_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    cursor.execute("select title,pages from books where pages>300 order by
pages desc")
    books=cursor.fetchall()
    for book in books:
        print(book)
    conn.close()
big_books()
```

1. SQLite

수정

```
import sqlite3
def update_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    sql='update books set recommend=? where title=?'
    cursor.execute(sql,(200,'Java'))
    conn.commit()
    conn.close()
update_books()
one_book()
```

1. SQLite

삭제

```
import sqlite3
def delete_books():
    conn=sqlite3.connect("my_books.db")
    cursor=conn.cursor()
    sql="delete from books where publisher='한빛'"
    cursor.execute(sql)
    conn.commit()
    conn.close()
delete_books()
all_books()
```

2.Oracle

- **cx_Oracle 설치**
  - console창에서 설치
  - pip install cx_Oracle

```
CREATE TABLE PRODUCT(
product_id number,
product_name varchar2(50),
price number 0,
description clob,
picture_url varchar2(500),
primary key(primary_id));


CREATE TABLE cart(
cart_id number primary key,
userid varchar2(50) not null,
product_id references product(product_id),
amount number defalut 0)
```

```
insert into product values(1, '사과',10000,'
청도 꿀사과 당도가 매우 높음','apple.jpg');
insert into product values(2, '배',20000,'나
주배 입니다','pear.jpg');
insert into product values(3, '포도',10000,'
칠래산 적포도입니다','grap.jpg');
```

2. Oracle

Python Oracle 연동

```
#pip install cx_Oracle
#전체 레코드 조회
import cx_Oracle
conn=cx_Oracle.connect("pgm/1234@localhost:1521/xe")
cursor=conn.cursor()
sql="select * from product"
cursor.execute(sql)
for row in cursor:
    description=row[3].read() #CLOB 필드 읽는 방법
    print(row)
cursor.close()
conn.close()
```

2.Oracle

Python Oracle 연동

```
# 1개 레코드 삽입
import cx_Oracle
conn=cx_Oracle.connect("pgm/1234@localhost:1521/xe")
cursor=conn.cursor()
sql="insert into product(product_id,product_name,price,description,picture_url)
values(:1,:2,:3,:4,:5)"
data=(1, '레몬', 1500, '레몬에 포함된 구연산은 피로회복에 좋습니다.', 'lemon.jpg')
cursor.execute(sql,data)
cursor.close()
conn.commit()
conn.close()
```

2.Oracle

Python Oracle 연동

```
# 복수개 레코드 삽입
import cx_Oracle
conn=cx_Oracle.connect("pgm/1234@localhost:1521/xe")
cursor=conn.cursor()
items=[
(2, '오렌지', 2000, '비타민 C가 풍부합니다. 생과일 주스로 마시면 좋습니다.', 'orange.jpg'),
(3, '키위', 3000, '비타민 C가 매우 풍부합니다. 다이어트나 미용에 좋습니다', 'kiwi.jpg'),
(4, '포도', 5000, '폴리페놀을 다량함유하고 있어 항산화 작용을 합니다,', 'grape.jpg'),
(5, '딸기', 8000, '비타민 C나 플로보노이드를 다량 함유하고 있습니다,', 'strawberry.jpg'),
(6, '귤', 7000, '시네피린을 함유하고 있어 감기 예방에 좋다고 합니다,', 'tangerine.jpg')
]
sql="insert into product values(:1,:2,:3,:4,:5)"
#data=(7,'토마토',5000,'채소입니다','tomato.jpg')
for row in items:
    cursor.execute(sql,row)
#cursor.close()
conn.commit()
conn.close()
```

2.Oracle

Python Oracle 연동

```
# 레코드 1개 조회
#pip install cx_Oracle
import cx_Oracle
conn=cx_Oracle.connect("pgm/1234@localhost:1521/xe")
cursor=conn.cursor()
sql="select count(*) from product"
cursor.execute(sql)
count=cursor.fetchone()
print("상품갯수:",count[0])
cursor.close()
conn.close()
```

2.Oracle

Python Oracle 연동

```
# 일부레코드 삭제
cursor=conn.cursor()
sql="delete from product where product_id=:product_id"
cursor.execute(sql,{'product_id':5)

# 모든 레코드 삭제
cursor=conn.cursor()
sql="delete from product"
cursor.execute(sql)
```

```
# 모든 레코드를 한꺼번에 insert
sql="insert into product values(:1,:2,:3,:4,:5)"
cursor.binarraysize=len(items) # 리스트 개수 전달
cursor.executemany(sql,items) #
```

2.Oracle

Python Oracle 연동

```
import cx_Oracle
conn=cx_Oracle.connect("pgm/1234@localhost:1521/xe")
cursor=conn.cursor()
price = int(input("Input : "))
product_id = str(input("Input : "))

sql = "UPDATE product SET price ="+price+" WHERE product_id ="+product_id
curs.execute(sql)
conn.commit

sql="update product set price=:1 where product_id=:2"
data=(7000,1)
cursor.execute(sql,data)
conn.commit()

cursor.close()
conn.close()
```

3. MySQL

MySQL 테이블 작성

```
create database pydb;
use pydb
create table pages(
id int not null auto_increment primary key,
title varchar(200),
content text,
reg_date datetime default now()
);
```

PyMySQL 설치

콘솔에서 다음 명려어 입력

```
pip install PyMySQL
```

3. MySQL

DB연결

```
conn = pymysql.connect(  #pymysql 라이브러리 사용 DB연결
            host= ' localhost ' ,
            user= ' pgm ' ,
            password= ' 1234 ' ,
            db= ' pydb′,
            charset=′utf8′)

#conn=MySQLdb.connect( # MySQLdb 라이브러리 사용 DB연결
            "localhost",
            "pgm",
            "1234",
            "pydb",
            charset='utf8')

cursor = conn.cursor()
```

3. MySQL

- **insert**

```
sql="insert into pages(title,content) values(%s,%s)" %(title,content)
cursor.execute(sql)
conn.commit()
conn.close()
```

```
sql="insert into pages(title,content) values('%s','%s')"
data=tuple('title', 'content')
cursor.execute(sql, data)
conn.commit()
conn.close()
```

```
sql="insert into pages(title,content) values(%s, %s)"
data=[('title1', 'content1'),('title2, 'content2'),('title3', 'content3')]
cursor.executemany(sql, data)
conn.commit()
conn.close()
```

3. MySQL

insert example

```
def insert_book(data):
    conn=pymysql.connect(host='localhost',user='pgm',password='1234', db='pydb1', charset='utf8')
    cursor=conn.cursor()
    sql="insert into book(title,pub,page,author) values(%s,%s,%s,%s)"
    cursor.execute(sql,data)
    conn.commit()
    conn.close()

data=('안드로이드','한빛',650,'홍길동')
insert_book(data)
```

```
def insert_book_list(datas):
    conn=pymysql.connect(host='localhost',user='pgm',password='1234',db='pydb1',charset='utf8')
    cursor=conn.cursor()
    sql="insert into book(title,pub,page,author) values(%s,%s,%s,%s)"
    cursor.executemany(sql,datas)
    conn.commit()
    conn.close()

datas=[('안드로이드','한빛',650,'홍길동'),
       ('Oracle DB','한빛',650,'홍길동'),
       ('Spring','영딘',650,'홍길동')]
insert_book_list(datas)
```

3. MySQL

update, delete

```
conn = pymysql.connect(host='localhost', user='tester',
        password='7890', db='testdb', charset='utf8')
curs = conn.cursor()
sql = "update customer set region = '서울특별시' where region = '서울'"
curs.execute(sql)

sql = "delete from customer where id=%s"
curs.execute(sql, 6)
conn.commit()
conn.close()
```

3. MySQL

update example

```
def update_book(data):
    conn=pymysql.connect(host='localhost',user='pgm',password='1234',db='pydb1',charset='utf8')
    cursor=conn.cursor()
    sql="update book set title=%s, pub=%s, page=%s, author=%s where no=%s"
    cursor.execute(sql,data)
    conn.commit()
    conn.close()

data=('Java', '생능','700', ' 홍길동',1)
update_book(data)
```

```
def update_book(datas):
    conn=pymysql.connect(host='localhost',user='pgm',password='1234',db='pydb1',charset='utf8')
    cursor=conn.cursor()
    sql="update book set title=%s, pub=%s, page=%s, author=%s where no=%s"
    cursor.executemany(sql,datas)
    conn.commit()
    conn.close()

datas=[('Java2', '생능','700','최주호',2),
       ('Java3', '생능','700','최주호',3),
       ('Java4', '생능','700','최주호',4)]
update_book(datas)
```

3. MySQL

- **delete example**

```
def delete_book(data):
    conn=pymysql.connect(host='localhost',user='pgm',password='1234',
            db='pydb1', charset='utf8')
    cursor=conn.cursor()
    sql="delete from book where no=%s"
    cursor.executemany(sql,data)
    conn.commit()
    conn.close()

delete_book([1,2])
```

3. MySQL

- **try와 with문**

```
import pymysql
conn = pymysql.connect(host='localhost', user='tester', password='7890',
                       db='testdb', charset='utf8')
try:
    # INSERT
    with conn.cursor() as curs:
        sql = "insert into customer(name,category,region) values (%s, %s, %s)"
        curs.execute(sql, ('이광수', 1, '서울'))
    conn.commit()

    # SELECT
    with conn.cursor() as curs:
        sql = "select * FROM customer"
        curs.execute(sql)
        rs = curs.fetchall()
        for row in rs:
            print(row)
finally:
    conn.close()
```