11.데이터베이스

contents

1. **SQLite**
2. **MySQL**
3. **Oracle**

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

- **table 생성**

```
import sqlite3
def create_table():
    conn=sqlite3.connect("bookdb") #데이터베이스 커넥션 생성
    cursor=conn.cursor()   #쿼리 명령을 위한 커서 생성
    #테이블 생성(제목,출판일자,출판사,페이지수,추천여부)
    sql="""create table if not exists books(
        title text,
        published_date text,
        publisher text,
        page integer,
        recommend integer
    )"""
    cursor.execute(sql) # 쿼리 명령 수행
    conn.commit() # 데이터베이스 변경내용 완료
    conn.close()  # 커넥션 닫기

create_table()
```

1. SQLite

- 데이터 추가

```
def insert_book1():
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="insert into books values('Java','2019-05-20','길벗',500,10)"
    cursor.execute(sql)
    conn.commit()
    conn.close()

insert_book1()
```

1. SQLite

- 데이터 조회(모든 데이터 조회)

```
def select_all():
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="select * from books"
    cursor.execute(sql)
    books=cursor.fetchall()  # 실행된 SQL 쿼리의 결과를 모두 가져오는 메서드
    # 한 행의 데이터를 튜플로 가져오며, 전체 결과는 list에 저장
    for book in books:
        print(book)
    conn.close()

select_all()
```

1. SQLite

- **데이터 추가(1행의 데이터를 파라미터로 전달)**

단수행의 데이터 추가

```
def insert_book2(data):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="insert into books values(?,?,?,?,?)"
    cursor.execute(sql,data)
    conn.commit()
    conn.close()

data=('Python','201001','한빛',600,20)
insert_book2(data)
select_all()
```

복수행의 데이터 추가

```
def many_insert_book(data):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="insert into books values(?,?,?,?,?)"
    cursor.executemany(sql,data)
    conn.commit()
    conn.close()

data=[('머신러닝','201001','한빛',600,20),
      ("안드로이드","202501",'골든래빗',550,10)]
many_insert_book(data)
select_all()
```

1. SQLite

- **데이터 조회**

쿼리 결과에서 1개의 데이터 출력

```
def one_book():
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="select * from books"
    cursor.execute(sql)
    # 쿼리결과에서 한개의 데이터를 가져옴
    books=cursor.fetchone()
    print(type(books))
    print(books)
    conn.close()

one_book()
```

쿼리 결과에서 지정된 갯수의 데이터 출력

```
def some_book(number):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="select * from books"
    cursor.execute(sql)
    # 쿼리결과에서 한개의 데이터를 가져옴
    books=cursor.fetchsome(number)
    for book in books:
        print(book)
    conn.close()

some_book(2)
```

1. SQLite

- 데이터 조회 (조건 검색)

```
def search_select(pages):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="select * from books where page>=? order by title desc"
    cursor.execute(sql,(pages,))
    books=cursor.fetchall()
    for book in books:
        print(book)
    conn.close()

search_select(600)
```

1. SQLite

- **update**

update 정보가 1개일 때

```
def update_book(data):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="update books set recommend=?, page=?where title=?"
    cursor.execute(sql, data)
    conn.commit()
    conn.close()

data=(50, 350, 'Java')
update_book(data)
select_all()
```

update 정보가 복수일 때

```
def update_books(data):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="update books set recommend=?, page=? where title=?"
    cursor.executemany(sql, data)
    conn.commit()
    conn.close()

data=[(50, 550, 'Java'),(20, 450,'python')]
update_books(data)
select_all()
```

1. SQLite

- **데이터 삭제**

삭제 조건이 단수일 때

```
def delete_book(data):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="delete from books where title=?"
    cursor.execute(sql, data)
    conn.commit()
    conn.close()

data1=("Java",)
delete_book(data1)
select_all()
```

삭제 조건이 복수일 때

```
def delete_books(data):
    conn=sqlite3.connect("bookdb")
    cursor=conn.cursor()
    sql="delete from books where title=?"
    cursor.executemany(sql, data)
    conn.commit()
    conn.close()

data1=[("머신러닝",), ("Python",)]
delete_books(data1)
select_all()
```

2. Mysql

- 라이브러리 설치

```
! pip install pymysql
```

- 라이브러리 임포트

```
import pymysql
```

- DB 연결

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

2. Mysql

- 데이터 입력

```
# 1개의 데이터 입력
def insert_data1(data):
    conn = pymysql.connect(  #pymysql 라이브러리 사용 DB연결
        host='localhost',
        user='pgm',
        password='1234',
        db='pydb',
        charset='utf8')
    cursor=conn.cursor()
    sql="insert into book(title,pub, pages, author) values(%s, %s, %s,%s)"
    cursor.execute(sql, data)
    conn.commit()
    conn.close()
```

0.0s

```
data=('파이썬프로그램','한빛',20,'홍길동')
insert_data1(data)
```

2. Mysql

- 데이터 입력

```
#복수의 데이터 입력
def insert_data2(data):
    conn = pymysql.connect(  #pymysql 라이브러리 사용 DB연결
        host='localhost',
        user='pgm',
        password='1234',
        db='pydb',
        charset='utf8')
    cursor=conn.cursor()
    sql="insert into book(title, pub, pages, author) values(%s, %s, %s,%s)"
    cursor.executemany(sql, data)
    conn.commit()
    conn.close()
```

```
data_list=[('자바','생능','500','황기태'),('데이터베이스','한빛',350,'서진수')]
insert_data2(data_list)
```

2. Mysql

- 데이터 조회

```
# 전체 데이터 조회
def select_all():
    conn=pymysql.connect(host='localhost',
    user='pgm',
    password='1234',
    db='pydb',
    charset='utf8')
    cursor=conn.cursor()
    sql="select * from book"
    cursor.execute(sql)
    for book in cursor:
        print(book)
    conn.close()

select_all()
```

```
# 조건 검색
def select_cond(title):
    conn=pymysql.connect(host='localhost',
    user='pgm',
    password='1234',
    db='pydb',
    charset='utf8')
    cursor=conn.cursor()
    sql="select * from book where title=%s"
    cursor.execute(sql,(title,))
    for book in cursor:
        print(book)
    conn.close()

select_cond('자바')
```

2. Mysql

- **update**

```
# 1개 데이터 업데이트
def update_book(data):
    conn=pymysql.connect(host='localhost',
                         user='pgm',
                         password='1234',
                         db='pydb',charset='utf8')
    cursor=conn.cursor()
    sql="update book set title=%s, pub=%s, pages=%s, author=%s where id=%s"
    cursor.execute(sql,data)
    conn.commit()
    conn.close()

data=('Java', '생능','700','홍길동',3)
update_book(data)
select_all()
```

2. Mysql

- **update**

```
# 복수개의 데이터 업데이트
def update_book(datas):
    conn=pymysql.connect(host='localhost',
                         user='pgm',
                         password='1234',
                         db='pydb',charset='utf8')
    cursor=conn.cursor()
    sql="update book set title=%s, pub=%s, pages=%s, author=%s where id=%s"
    cursor.executemany(sql,datas)
    conn.commit()
    conn.close()

datas=[('Java2', '생능','700','최주호',3),
       ('파이썬', '한빛','700','홍길동',4),
       ('자바', '길벗','700','박경미',5)]
update_book(datas)
select_all()
```

2. Mysql

- 삭제

```
# 단일 및 복수개의 데이터 삭제
def delete_book(data):
    conn=pymysql.connect(host='localhost',
                         user='pgm',
                         password='1234',
                         db='pydb', charset='utf8')
    cursor=conn.cursor()
    sql="delete from book where id=%s"
    #cursor.execute(sql,data)
    cursor.executemany(sql,data)
    conn.commit()
    conn.close()

#delete_book(4)
delete_book([5,6])
select_all()
```

3. Oracle

- **oracle 설치**

```
! pip install cx_Oracle
```

```
import cx_Oracle

# Oracle 데이터베이스 연결 정보
dsn = cx_Oracle.makedsn("호스트주소", 1521, service_name="서비스이름")
conn = cx_Oracle.connect(user="사용자이름", password="비밀번호", dsn=dsn)

cursor = conn.cursor()
print("Oracle DB 연결 성공!")
```

```
ex)
dsn = cx_Oracle.makedsn("localhost", 1521, service_name="xe")
conn = cx_Oracle.connect(user="pgm", password="1234", dsn=dsn)
```

3. Oracle

- 데이터 조회

```
def read_data():
    sql = "SELECT * FROM employees"
    cursor.execute(sql)

    for row in cursor.fetchall():
        print(row)

read_data()
```

3. Oracle

- 데이터 추가

```
def insert_data():
    sql = "INSERT INTO employees(id, name, age, salary) VALUES(:1, :2, :3, :4)"
    data = (101, '홍수민', 30, 70000)

    cursor.execute(sql, data)
    conn.commit()  # 변경 사항 저장
    print("데이터 삽입 완료")

insert_data()
```

3. Oracle

- 데이터 수정

```
def update_data():
    sql = "UPDATE employees SET salary = :1 WHERE id = :2"
    data = (60000, 101)

    cursor.execute(sql, data)
    conn.commit()
    print("데이터 업데이트 완료")

update_data()
```

3. Oracle

- **데이터 삭제**

```
def delete_data():
    sql = "DELETE FROM employees WHERE id = :1"
    data = (1,)

    cursor.execute(sql, data)
    conn.commit()
    print("데이터 삭제 완료")

delete_data()
```

3. Oracle

- **DB 연결 종료**

```
cursor.close()
conn.close()
print("Oracle DB 연결 종료")
```