# SQLite3

2020.11

# 1. SQL

## ■ SQL(Structured Query Language)

- 관계형 데이터베이스 관리 시스템(RDBMS)의 데이터를 관리/사용하기 위해 설계된 특수 목적의 프로그래밍 언어
- 대부분의 DB에서 사용이 가능한 표준이나, DBMS 종류마다 약간씩 상이함
- DCL(Data Control Language)
- DDL(Data Definition Language)
- DML(Data Manipulation Language)

## ■ 데이터 조작 언어(DML)

- 학습 사이트: https://www.w3schools.com/sql/default.asp
- CRUD operation

## ■ 주요한 SQL(Strucured Query Language) 구문

- SELECT - extracts data from a database
- INSERT INTO - inserts new data into a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database

- CREATE DATABASE - creates a new database
- CREATE TABLE - creates a new table
- ALTER DATABASE - modifies a database
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table

## 2. SQLite 란?

### ■ 경량 DBMS

- 별도의 서버가 필요 없음
- 모바일 기기에서 많이 활용되고 있음
- 파이썬3에 기본 내장되어 있음
- 파일 또는 메모리에 DB 생성
- 참고자료: SQLite로 가볍게 배우는 데이터베이스 (WikiDocs)

### ■ 데이터 타입

- 동적 데이터 타입
- Null, Integer, Real, Text, Blob 유형이 있음 (Boolean, Date, Time 없음)
- 다른 유형 데이터를 삽입해도 컬럼에 맞게 알아서 들어감.
- 다른 DB에서 사용하는 데이터유형 이름 그대로 사용해도 무방

### ■ DBMS 관리 툴

- DB Browser for SQLite(https://sqlitebrowser.org/dl)
- SQLite Expert(http://www.sqliteexpert.com/download.html)

## ■ 데이터베이스 접속

```
import sqlite3
conn = sqlite3.connect(':memory:')      # 메모리 DB 접속(일회성)
conn = sqlite3.connect('./test.db')     # 파일 DB 접속
. . .
데이터 쿼리 수행
. . .
conn.commit()           # 변경사항 저장
conn.close()
```

## ■ with 문 이용

```
import sqlite3
conn = sqlite3.connect('./test.db')

with conn:
    cur = conn.cursor()
    cur.execute('SELECT * FROM test_table')
    rows = cur.fetchall()
    for row in rows:
        print(row)
```

# 4. Data Definition Language(DDL)

## ■ 테이블 생성

```
cur = conn.cursor()
cur.execute('CREATE TABLE IF NOT EXISTS Eagles \
                (back_no INT NOT NULL, \
                 name TEXT, \
                 position TEXT, \
                 PRIMARY KEY(back_no));')
```

## ■ 테이블 구조/이름 변경

```
cur.execute('ALTER TABLE Eagles ADD COLUMN birth INTEGER')
cur.execute('ALTER TABLE Eagles RENAME TO Eagles.backup')
```

## ■ 테이블 삭제

```
cur.execute('DROP TABLE Eagles')
```

## ■ 데이터 삽입

```
# 기본 스트링 쿼리
cur = conn.cursor()
cur.execute("INSERT INTO Eagles VALUES(1, '하주석', '내야수');")
cur.execute("INSERT INTO Eagles VALUES
                (57, '정우람', '투수'), (8, '정근우', '내야수');")

# 파라메터: 튜플 사용
back_no = 50
name = '이성열'
position = '외야수'
cur = conn.cursor()
sql = 'INSERT INTO Eagles VALUES (?, ?, ?);'
cur.execute(sql, (back_no, name, position))

# 튜플 리스트 사용
players = ((22, '이태양', '투수'), (13, '최재훈', '포수'))
cur = conn.cursor()
sql = 'INSERT INTO Eagles VALUES (?, ?, ?);'
cur.executemany(sql, players)
```

## ■ 데이터 조회

```
# 순회 조회
cur = conn.cursor()
cur.execute('SELECT * FROM Eagles')
for row in cur:
    print(row)

# 단건 조회
cur = conn.cursor()
cur.execute('SELECT * FROM Eagles')
row = cur.fetchone()

# 다건 조회
rows = cur.fetchmany(2)

# 모두 조회
rows = cur.fetchall()
for row in rows:
    print(row)
```

## ■ 데이터 조회

```
# 필요한 column만 조회
cur = conn.cursor()
cur.execute('SELECT name FROM Eagles WHERE back_no > 10')
rows = cur.fetchall();
for row in rows:
    print(row)

# 원하는 순서 및 갯수
cur.execute('SELECT * FROM Eagles ORDER BY name')
cur.execute('SELECT * FROM Eagles ORDER BY name DESC')

cur.execute('SELECT * FROM Eagles ORDER BY name DESC LIMIT 1')
row = cur.fetchone()
print(row[1])           # '하주석'

# 집계 함수
cur.execute('SELECT count(*) FROM Eagles')
count = cur.fetchone()

max(column), min(column), sum(column), avg(column)
```

## ■ 데이터 검색

```
# 기본 스트링 쿼리
cur = conn.cursor()
cur.execute("SELECT * FROM Eagles WHERE position='내야수';")
rows = cur.fetchall();
for row in rows:
    print(row)

# Placeholder
cur = conn.cursor()
back_no = 50
cur.execute('SELECT * FROM Eagles WHERE back_no=?;', (back_no,))
player = cur.fetchone()
print(player[0])                # 50

# Grouping
sql = 'SELECT position, count(*) FROM Eagles GROUP BY position'
```

# 5. 데이터 조작 언어(Data Manipulation Language , DML)

## ■ 데이터 변경

```
position = '외야수'
back_no = 8
cur.execute('UPDATE Eagles SET position=? WHERE back_no=?;',
                (position, back_no))
cur.execute('SELECT * FROM Eagles WHERE back_no=?', (back_no,))
cur.fetchone()

테이블 구조 변경
cur.execute('ALTER TABLE Eagles ADD COLUMN birth INTEGER')

data = ((1995,1), (1986,57))
sql = 'UPDATE Eagles SET birth=? WHERE back_no=?'
cur.executemany(sql, data)
```

## ■ 데이터 삭제

```
cur = conn.cursor()
cur.execute('DELETE FROM Eagles WHERE back_no=1);')
```

# 6. 연습 문제

1. 투수들의 기록중에서 평균자책점(ERA), 투구인닝(IP), 탈삼진(SO) 기록을 찾아서 Pitcher_stats 란 테이블을 만들고, Eagles 테이블과 Join 하여 백넘버, 선수명, 포지션, 투구인닝, 평균자책점, 탈삼진 필드를 갖는 데이터 프레임을 만들어서 Join 한 결과를 입력하고, 그 결과를 보이시오.

2. 다음의 지시대로 DB 테이블을 만들고 이를 조회하는 프로그램을 만드시오.

   1) 국내의 대표적인 걸그룹 또는 보이그룹 5개 이상에 대하여 다음과 같은 정보를 갖는 테이블을 만드시오. *는 Primary Key
   id(*), group_name, 구성원 수, 데뷔일자, 소속사

   2) 이들이 불렀던 노래 또는 다른 사람이 불렀던 노래 10곡 이상에 대하여 다음의 정보를 갖는 테이블을 만드시오.
   song_id(*), song_name, 그룹 id, 발표년도, 작곡가, 도입부 가사

   3) 위 두개의 테이블을 조인한 결과를 가지고 다음의 필드를 갖는 데이터 프레임을 만드시오.
   그룹 이름, 구성원 수, 데뷔 일자, 노래 이름, 발표 년도

3. 사용자의 이름과, 비밀번호를 갖는 Users 테이블이 있다. 사용자의 이름과 비밀번호를 올바르게 입력하면 '성공'을 출력하고, 잘못 입력하면 '실패'를 출력하는 프로그램을 작성하시오.