

# 10장. 데이터베이스(DB)



# 목 차

1

데이터 베이스란?

2

SQLite3

3

Member DB 관리

# 데이터베이스

## 데이터베이스(Database)

- 데이터들이 모여있는 데이터의 집합으로 **서로 관련 있는 데이터들의 모임이다.**

(메모장에 두서없이 적어 놓은 단어들의 모임은 데이터베이스가 아님)

- 데이터베이스와 생활

예) 학교 홈페이지에서 수강신청, 성적 조회

전산화된 도서관에서 도서 검색, 비행기나 기차 예매 등

학생 테이블

| 학번       | 이름  | 생년월일        | 학과명    |
|----------|-----|-------------|--------|
| 20150001 | 오상식 | 1987. 6. 10 | 컴퓨터공학과 |
| 20171010 | 최정보 | 1995. 5. 5  | 전자공학과  |
| 20182121 | 김나래 | 1993. 12. 1 | 기계공학과  |

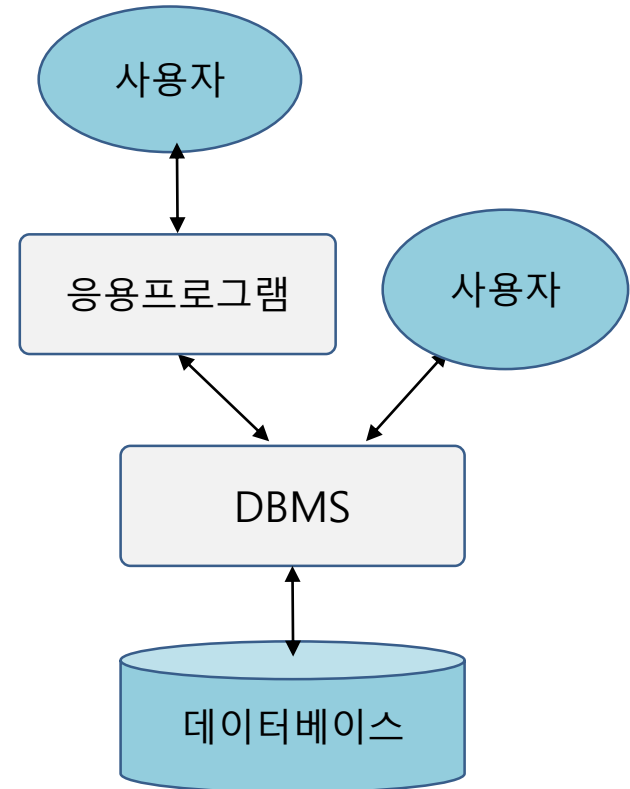
과목 테이블

| 과목번호 | 과목명     | 담당교수 |
|------|---------|------|
| 0303 | 웹 프로그래밍 | 송미영  |
| 0116 | 데이터베이스  | 오용철  |

# 데이터베이스 관리 시스템

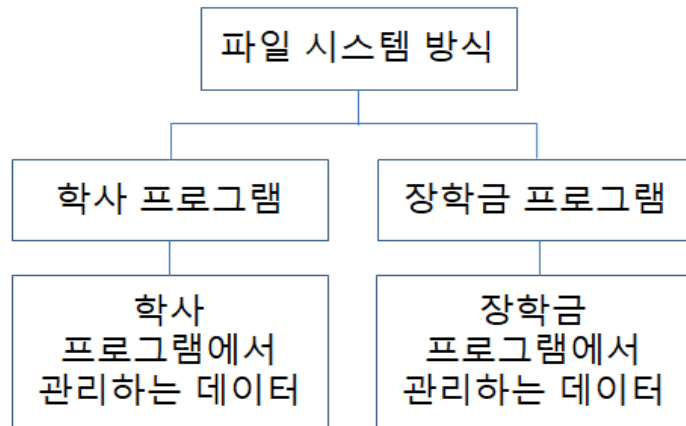
## 데이터베이스 관리 시스템(Database Management System)

- 많은 양의 데이터를 정교하게 구축하고 관리하는 소프트웨어이다.
- 데이터베이스의 정의, 데이터베이스 갱신, 질의 처리, 유지보수, 보안 등의 편리한 기능을 제공한다.
- 대표적으로 오라클 사의 Oracle 과 MySQL, 마이크로소프트사의 MSSQL등이 있다  
한편 sqlite3는 DB 서버가 아닌 응용 프로그램에 넣어 사용하는 가벼운(경량) 데이터베이스이다.

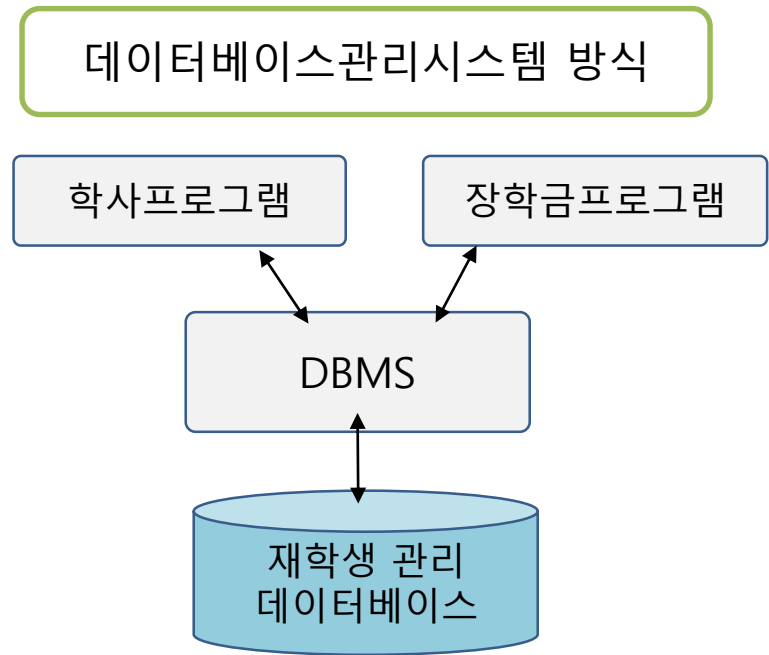


# 파일 시스템과 DBMS

## 파일시스템과 DBMS



파일 시스템은 서로 다른 여러 응용 프로그램이 제공하는 기능에 맞게 필요한 데이터를 각각 저장하고 관리한다. 따라서 각 파일에 저장한 데이터는 서로 연관이 없고 중복 또는 누락이 발생할 수 있다.



학생과 관련된 일련의 데이터를 한곳에 모아 관리하면 데이터의 오류, 누락, 중복 등의 문제를 해결할 수 있다.

# 파일 시스템과 DBMS

## 파일 시스템 방식의 문제점

이순신 학생이 졸업했는데 업데이트가 되지 않아 재학중으로 되어 있어 장학금 신청이 가능한 걸로 오류 발생

학사 프로그램

| 학번        | 이름  | 학과     | 상태  |
|-----------|-----|--------|-----|
| 2019-0001 | 홍길동 | 컴퓨터공학과 | 군휴학 |
| 2019-0002 | 이순신 | 경영학과   | 졸업  |
| 2019-0003 | 유관순 | 철학과    | 재학  |

장학금 신청 프로그램

| 장학금 | 이름  | 상태  | 가능여부 |
|-----|-----|-----|------|
| 국가  | 홍길동 | 군휴학 | 신청불가 |
| 성적  | 이순신 | 재학  | 신청가능 |
| 근로  | 유관순 | 재학  | 신청가능 |

# 데이터베이스 관리 시스템

## 데이터베이스 관리 시스템의 장점

- 데이터의 중복과 불일치 감소

데이터가 여러 곳에 분산되어 있으면 중복 저장될 수 있고, 같은 의미의 데이터가 다른 값을 갖게 되는 불일치가 생길 수 있다.

- 질의 처리에 효율적인 저장 구조

사용자는 질의(Query)를 통해서 데이터베이스에 접근하는데 시간이 소요되지만 DBMS는 시간을 줄이도록 저장 구조가 설계되어 있다.

- 백업(Backup)과 복구(Recovery)

데이터는 저장과 동시에 반드시 백업(따로 복사)되어야 한다. 복구는 트랜잭션(업무 단위)을 관리하여 데이터베이스가 피해를 보기 전 상태로 복구하는 것이다.

※ 단점 : 사용하는 자원이 많고 복잡하며 비싸다.

# 데이터 모델

## 데이터 모델

- 컴퓨터에 데이터를 저장하는 방식을 정의해 놓은 개념 모형이다.
- 계층형 데이터 모델, 네트워크형 데이터 모델, 관계형 데이터 모델, 객체 지향형

## 데이터 모델링(Data Modeling)

- 데이터 베이스의 설계시 클라이언트의 요구를 분석하여 논리모형을 구성하고 물리모형을 사용해 데이터베이스에 반영하는 작업
- 기본 요소

| 구분                | 개념                        | 실제 예           |
|-------------------|---------------------------|----------------|
| 엔티티(Entity)       | 물리적 개념에서는 테이블로 표현         | 고객, 상품, 주문     |
| 속성(Attribute)     | 물리적 개념에서는 칼럼(Column)으로 표현 | 고객아이디, 고객명, 주소 |
| 관계(Releationship) | 기본키와 참조키로 정의 됨(일대일, 일대다)  | 고객과 주문과의 관계    |

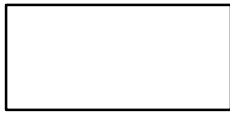


# 데이터 모델링

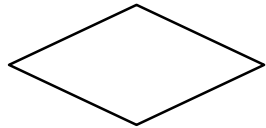
## ◆ 개념적 설계

현실세계를 추상화(특성화)하여 개체 타입과 관계를 파악하여 표현하는 과정

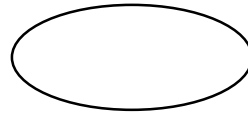
→ 개체 관계도(E-R 다이어그램) : Entity-Relationship Diagram



개체



관계



속성

## ◆ 논리적 설계

개념적 설계에서 만들어진 구조를 논리적으로 구현 가능한 데이터 모델로 변환하는 단계로 사용자가 알아볼 수 있는 형태로 변환하는 과정 -> 테이블(표) 형태

## ◆ 물리적 설계

논리적 데이터베이스 구조를 실제 기계가 처리하기에 알맞도록 내부 저장 장치 구조와 접근 경로 등을 설계하는 과정

예) name char(20) – name은 문자형 20Byte를 의미함

# 데이터 베이스 설계

현실 단계



개념 단계

이름

전화번호

주소

회원

논리 단계

회원

|    |      |    |
|----|------|----|
| 이름 | 전화번호 | 주소 |
|----|------|----|

물리 단계

name CHAR(20)  
phone TEXT  
address TEXT

개체

특성

값

개체타입

속성

값

레코드타입

특성

값

자료형타입

특성

값

# 관계형 데이터베이스

## 관계형 데이터 모델

- 데이터간의 관계에 초점을 둔 모델로 현재 가장 많이 사용하는 모델이다.

예) 회사의 직원정보, 소속된 부서정보

- 직원 정보와 부서 정보를 하나의 묶음으로 관리하면 데이터 구조가 간단해진다. 하지만 같은 부서 직원들은 부서 정보가 중복되므로 효율적인 관리가 어려워진다. 왜냐하면 부서 이름이 바뀌면 직원들의 부서 정보를 일일이 찾아서 수정해주어야 한다.

| 직원 정보 | 직원 번호 | 직원 이름 | 직원 직급 | 부서이름 | 위치 |
|-------|-------|-------|-------|------|----|
| 직원 번호 | 0001  | 홍길동   | 과장    | 회계팀  | 서울 |
| 직원 이름 | 0002  | 성춘향   | 대리    | 연구소  | 수원 |
| 직원 직급 | 0003  | 이몽룡   | 사원    | 영업팀  | 분당 |
| 부서이름  | 0004  | 심청이   | 사원    | 회계팀  | 서울 |
| 위치    |       |       |       |      |    |

데이터 중복발생

※ 정규화 전의 형태

# 관계형 데이터베이스

| 부서 정보 |
|-------|
| 부서 코드 |
| 부서 이름 |
| 위치    |

| 부서 코드 | 부서 이름 | 위치 |
|-------|-------|----|
| 10    | 회계팀   | 서울 |
| 20    | 연구소   | 수원 |
| 30    | 영업팀   | 분당 |

| 사원 정보 |
|-------|
| 사원 번호 |
| 사원 이름 |
| 사원 직급 |
| 부서 코드 |

| 사원 번호 | 사원 이름 | 사원 직급 | 부서코드 |
|-------|-------|-------|------|
| 0001  | 홍길동   | 과장    | 10   |
| 0002  | 성춘향   | 대리    | 20   |
| 0003  | 이몽룡   | 사원    | 30   |
| 0004  | 심청이   | 사원    | 10   |

※ 정규화 후의 형태 -> 1대 多의 구조로 변경된다.

한 부서에는 여러 명의 사원이 존재한다.

# 관계형 데이터베이스

## 관계형 데이터베이스의 구성 요소

### ● 테이블(Table)

표 형태의 데이터 저장 공간을 테이블이라고 한다. 2차원 형태로 행과 열로 구성

행(ROW) - 저장하려는 하나의 개체를 구성하는 여러 값을 가로로 늘어뜨린 형태다.

열(COLUMN) - 저장하려는 데이터를 대표하는 이름과 공통 특성을 정의

학생

| 학번       | 이름  | 생년월일        | 학과명    |
|----------|-----|-------------|--------|
| 20150001 | 오상식 | 1987. 6. 10 | 컴퓨터공학과 |
| 20171010 | 최정보 | 1995. 5. 5  | 전자공학과  |
| 20182121 | 김나래 | 1993. 12. 1 | 기계공학과  |

속성, 열, 칼럼, 애트리뷰트

튜플, 레코드, 행

# 관계형 데이터베이스

## 관계형 데이터베이스의 구성 요소

### ● 특별한 의미를 지닌 열 - 키

#### 기본키(Primary Key)

- 테이블의 지정된 행을 식별할 수 있는 유일한 값이어야 한다.
- 값의 중복이 없어야 한다.
- NULL값을 가질 수 없다.
- 주민등록번호, 학번, 사번 등

#### 보조키

- 대체키 또는 후보키라 하며 후보키 중에서 기본키로 지정되지 않은 열이다.

# 관계형 데이터베이스

## 외래키(FK : Foreign Key)

- 특정 테이블에 포함되어 있으면서 다른 테이블의 기본키로 지정된 키



# SQL이란?

## SQL(Structured Query Language)

- '에스큐엘', 또는 '시퀄'이라 부른다.
- 사용자와 데이터베이스 시스템 간에 의사 소통을 하기 위한 언어이다.
- 사용자가 SQL을 이용하여 DB 시스템에 데이터의 검색, 조작, 정의 등을 요구하면 DB 시스템이 필요한 데이터를 가져와서 결과를 알려준다.

| 구분   | 개념                               |
|--|----------------------------------|
| DDL(Data Definition Language)<br>- 데이터 정의어   | 테이블을 포함한 여러 객체를 생성, 수정, 삭제하는 명령어 |
| DML(Data Manipulation Language)<br>- 데이터 조작어 | 데이터를 저장, 검색, 수정, 삭제하는 명령어        |
| DCL(Data Control Language)<br>- 데이터 제어어      | 데이터 사용 권한과 관련된 명령어               |



# SQL – DDL

## DDL(데이터 정의어)

- ▷ 테이블 생성(만들기)  
**create table** 테이블이름(  
    name char(10),  
    age int  
)
- ▷ 테이블 삭제  
**drop table** 테이블 이름
- ▷ 테이블 변경  
**alter table** 테이블 이름 **add** 칼럼추가

# SQL – DML

## DML(데이터 조작어)

- ▷ 자료 삽입(insert)

**insert into** 테이블이름(칼럼명) **values** (값1, 값2)

- ▷ 자료 조회(select)

**select** 칼럼명 **from** 테이블이름

- ▷ 자료 수정(update)

**update** 테이블이름 **set='변경내용'** **where** 칼럼명

- ▷ 자료 삭제(delete)

**delete from** 테이블 이름

# SQL - DCL

## DCL(데이터 제어어)

### ▷ 커밋과 롤백

트랜잭션(작은 업무 단위) 완료를 의미하는 명령어 - **commit**

변경사항을 취소하고 원래대로 복구하는 명령어 - **rollback**

### ▷ 권한 부여와 해제

DB 권한을 부여하는 명령어 - **grant**

DB 권한을 해제하는 명령어 - **revoke**

# sqlite3

## sqlite3

Oracle나 MySql 같은 데이터베이스 관리 시스템이지만, 서버에 위치하지 않고 응용프로그램에 넣어 사용하는 파일이다.

Python에서는 내장된 라이브러리로 **import sqlite3**로 사용한다.

## DB Browser for sqlite3

오픈소스 소프트웨어로 SQLite 데이터베이스를 GUI 기반으로 편리하게 조작할 수 있도록 해주는 tool이다. – 다운로드 및 설치 필요

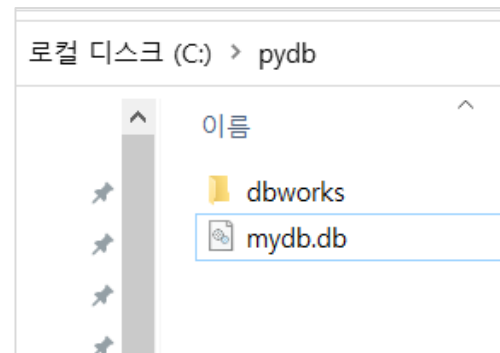
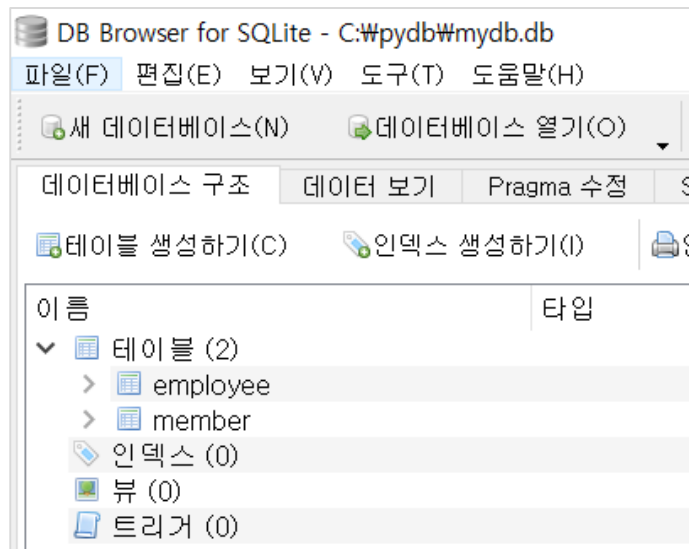
Our latest release (3.12.2) for Windows:

- [DB Browser for SQLite - Standard installer for 32-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 32-bit Windows](#)
- [DB Browser for SQLite - Standard installer for 64-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 64-bit Windows](#)

# sqlite3


## SQL 언어 실습

새 데이터베이스 > mydb.db 이름으로 저장



# sqlite3

## Sqlite3 Document > Alphabet.. > CREATE TABLE



Home About Documentation Download

▼ Document Lists And Indexes

- [Alphabetical Listing Of All Documents](#)
- [Website Keyword Index](#)
- [Permuted Title Index](#)

- Overview Documents
- Programming Interfaces
- Extensions
- Features

### AS SELECT Statements

"AS SELECT" statement creates and populates a database table based on a SELECT statement. The name of each column is the same as the column name in the SELECT statement. The affinity of each column is determined by the [expression affinity](#) of the corresponding expression in the SELECT statement.

| Expression Affinity |
|---------------------|
| TEXT                |
| NUMERIC             |
| INTEGER             |
| REAL                |
| BLOB (a.k.a "NONE") |

# sqlite3

## employee 테이블(Entity) 만들기

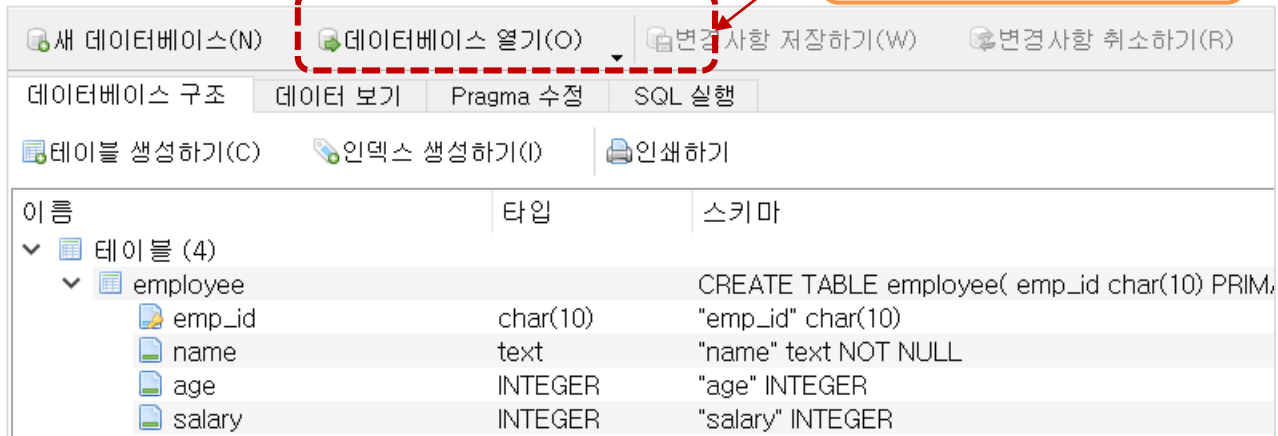
| 데이터 타입          | 설명      |
|-----------------|---------|
| <b>char</b>     | 고정길이 문자 |
| <b>text</b>     | 가변길이 문자 |
| <b>integer</b>  | 정수값     |
| <b>real</b>     | 실수값     |
| <b>datetime</b> | 날짜와 시간  |

```
-- 사원 테이블 생성 --  
CREATE TABLE employee(  
    emp_id char(10) PRIMARY key,  
    name text NOT NULL,  
    age INTEGER,  
    salary INTEGER  
);
```

# Member 테이블 생성하기

## ➤ 테이블 생성 확인

① pydb 연결하기



The screenshot shows a database management tool interface. At the top, there are buttons: '새 데이터베이스(N)', '데이터베이스 열기(O)', '변경사항 저장하기(W)', and '변경사항 취소하기(R)'. The '데이터베이스 열기(O)' button is highlighted with a red dashed box. An arrow points from the callout '① pydb 연결하기' to this button. Below the buttons, there are tabs: '데이터베이스 구조', '데이터 보기', 'Pragma 수정', and 'SQL 실행'. Under the '데이터베이스 구조' tab, there are buttons: '테이블 생성하기(C)', '인덱스 생성하기(I)', and '인쇄하기'. The main area displays a tree view of the database structure. Under '테이블 (4)', there is a folder 'employee'. Inside 'employee', there are four fields: 'emp\_id', 'name', 'age', and 'salary'. Each field has a corresponding data type and a SQL snippet.

| 이름       | 타입       | 스키마  |
|----------|----------|--|
| 테이블 (4)  |          |  |
| employee |          | CREATE TABLE employee( emp_id char(10) PRIM, |
| emp_id   | char(10) | "emp_id" char(10)                            |
| name     | text     | "name" text NOT NULL                         |
| age      | INTEGER  | "age" INTEGER                                |
| salary   | INTEGER  | "salary" INTEGER                             |



# sqlite3

## SQL 언어 실습

### 자료 추가

```
-- 자료 추가 --  
INSERT INTO employee VALUES ('e101', '추신수', 39, 10000)  
INSERT INTO employee VALUES ('e102', '박인비', 31, 20000)  
  
-- 자료 검색 --  
SELECT * from employee
```

문자 자료 – 홀따옴표, 쌍따옴표 사용가능

주석 처리 – 하이픈2개(-- )

# sqlite3

## SELECT 실습

```
-- 나이가 적은 순으로 오름차순 정렬 --  
SELECT * FROM employee ORDER BY age ASC  
  
-- 급여가 많은 순으로 내림차순 정렬 --  
SELECT * FROM employee ORDER By salary DESC
```

|   | emp_id | name | age | salary |
|---|--------|------|-----|--------|
| 1 | e103   | 손흥민  | 29  | 25000  |
| 2 | e102   | 박인비  | 31  | 20000  |
| 3 | e101   | 추신수  | 39  | 10000  |
| 4 | e104   | 이강   | 22  | 5000   |

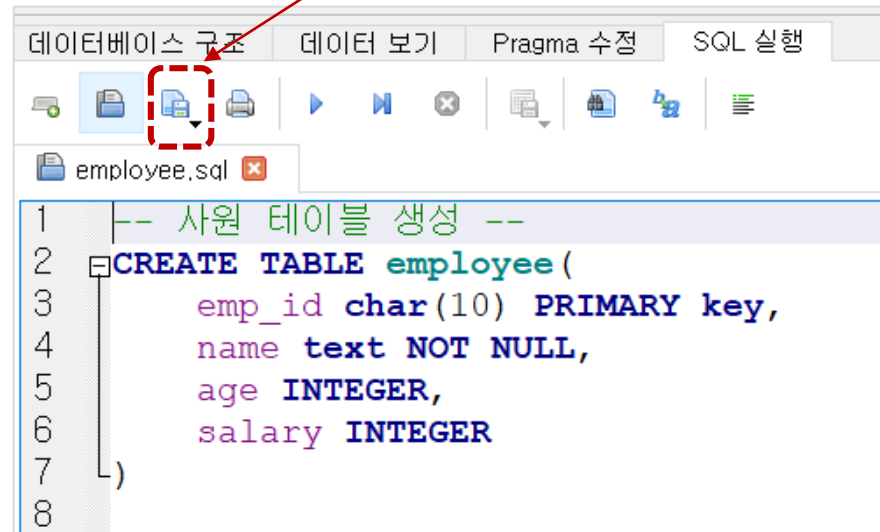
|   | emp_id | name | age | salary |
|---|--------|------|-----|--------|
| 1 | e104   | 이강   | 22  | 5000   |
| 2 | e103   | 손흥민  | 29  | 25000  |
| 3 | e102   | 박인비  | 31  | 20000  |
| 4 | e101   | 추신수  | 39  | 10000  |

# sqlite3

## sql 파일로 저장하기

sql 실행 탭 > sql 파일로 저장하기

SQL 파일 저장



# 파이썬으로 DB 관리

## ➤ DB 처리 프로세스

### 1. 데이터베이스 연결 객체 생성

```
conn = sqlite3.connect("c:/pydb/mydb.db")
```

### 2. CURSOR 객체 생성 – DB 테이블 작업 객체

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

### 3. execute() 함수 실행

```
cur.execute(sql)
```

# employee 테이블 자료 관리

## ➤ 전체 검색

```
import sqlite3

def select_emp():
    #conn = getconn()
    conn = sqlite3.connect("c:/pydb/mydb.db")
    cur = conn.cursor()
    sql = "select * from employee order by emp_id"
    cur.execute(sql)
    rs = cur.fetchall()    # rs=resultSet의 약자
    for i in rs:
        print(i)
    conn.close()
```

```
if __name__=="__main__":
    #insert_emp()
    #update_emp()
    #delete_emp()
    select_emp()
    #select_one()
```

# employee 테이블 자료 관리

## ➤ 사원 추가

```
def insert_emp():  
    #conn = getconn()  
    conn = sqlite3.connect("c:/pydb/mydb.db")  
    cur = conn.cursor()  
    sql = "insert into employee(emp_id, name, age, salary) values (?, ?, ?, ?)"  
    cur.execute(sql, ('e201', '장그래', 27, 10000))  
    conn.commit()  
    conn.close()
```

? 기호는 동적으로 메모리에 기억함  
칼럼 순서대로 매핑됨.

# employee 테이블 자료 관리

## ➤ 특정 회원 1명 검색

```
def select_one():  
    #conn = getconn()  
    conn = sqlite3.connect("c:/pydb/mydb.db")  
    cur = conn.cursor()  
    sql = "select * from employee where emp_id=?"  
    cur.execute(sql, ('e101',)) # 튜플은 1개일때 코머를 꼭 붙인다.  
    rs = cur.fetchone()  
    print(rs)  
    conn.close()
```

# employee 테이블 자료 관리

## ➤ 회원 정보 수정

```
def update_emp():  
    #conn = getconn()  
    conn = sqlite3.connect("c:/pydb/mydb.db")  
    cur = conn.cursor()  
    sql = "update employee set salary=? where emp_id=?"  
    #추신수의 급여를 10000에서 30000으로 변경  
    cur.execute(sql, (30000, 'e101'))  
    conn.commit()  
    conn.close()
```



# employee 테이블 자료 관리

## ➤ 회원 정보 삭제

```
def delete_emp():  
    #conn = getconn()  
    conn = sqlite3.connect("c:/pydb/mydb.db")  
    cur = conn.cursor()  
    # 사원 아이디가 'e102' 인 사원 정보를 삭제  
    sql = "delete from employee where emp_id=?"  
    cur.execute(sql, ('e201', ))  
    conn.commit()  
    conn.close()
```

# 파이썬으로 테이블 만들기

## 파이썬으로 member 테이블 생성하기

```
# 테이블 만들기
```

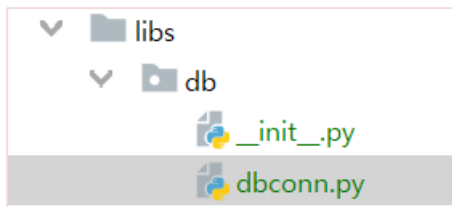
```
def create_table():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = """  
        CREATE TABLE member(  
            memberId CHAR(5) PRIMARY KEY,  
            passwd CHAR(8) NOT NULL,  
            name TEXT NOT NULL,  
            age INTEGER,  
            regDate DATETIME DEFAULT CURRENT_TIMESTAMP  
        )  
    """  
    cur.execute(sql)  
    conn.commit()  
    conn.close()
```

```
if __name__ == "__main__":  
    create_table()  
    #insert_member()  
    select_member()  
    #select_one()  
    #update_member()  
    #delete_member()
```

# 데이터베이스 연결하기

## ➤ 데이터베이스 연결하기

모듈로 만들기



```
import sqlite3

def getconn():
    conn = sqlite3.connect('c:/pydb/mydb.db')
    return conn
```

# 파이썬으로 테이블 만들기

## member 테이블에 자료 추가

```
# 데이터 추가
def insert_member():
    conn = getconn()
    cur = conn.cursor()
    #추가 방법 1 - ? 기호로 대입(동적 입력)
    #sql = "INSERT INTO member(memberId, passwd, name, age) VALUES (?, ?, ?, ?)"
    #cur.execute(sql, ('1001', 'a1234567', '콩쥐', 17))
    #추가 방법 2 - 각 행의 레코드를 직접 입력
    sql = "INSERT INTO member(memberId, passwd, name, age) VALUES ('1002', 'b1234567', '팔쥐', 18 )"
    cur.execute(sql)
    conn.commit()
    conn.close()
```

regDate(등록일)는 자동 입력된다.

# 파이썬으로 테이블 만들기

## member 테이블에 자료 검색 - 전체 검색

```
def select_member():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "SELECT * FROM member"  
    cur.execute(sql)  
    rs = cur.fetchall()  
    for i in rs:  
        print(i)  
  
    conn.close()
```

```
('10001', 'a1234567', '홍부', 35, '2021-09-23 20:24:23')  
( '10002', 'b1234567', '놀부', 40, '2021-09-23 20:25:37')  
( '10003', 'c1234567', '콩쥐', 19, '2021-09-23 20:26:16')
```

## 파이썬으로 테이블 만들기

member 테이블에 1명 검색 – 아이디와 비밀번호만 검색

```
def select_one():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "SELECT memberId, passwd FROM member WHERE memberId = ?"  
    cur.execute(sql, ('1002', ))  
    rs = cur.fetchone()  
    print(rs)  
    conn.close()
```

# 파이썬으로 테이블 만들기

## member 테이블에 자료 수정

```
def update_member():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "UPDATE member SET age = ? WHERE memberId = ?"  
    cur.execute(sql, (19, '1002'))  
    conn.commit()  
    conn.close()
```

# 파이썬으로 테이블 만들기

## member 테이블에 자료 삭제

```
def delete_member():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "DELETE FROM member WHERE memberId = ?"  
    cur.execute(sql, ('1003',))  
    conn.commit()  
    conn.close()
```



# member 로그인 구현

## member 로그인 구현

```
PS C:\pyworks\database> python t_member_login.py
```

```
아이디 : 1002
```

```
패스워드 :
```

```
로그인에 성공했습니다.
```

```
PS C:\pyworks\database> python t_member_l
```

```
아이디 : 1001
```

```
패스워드 :
```

```
아이디나 비밀번호가 일치하지 않습니다.
```

```
from getpass import getpass
import sqlite3
```

```
def login():
```

```
    memberId = input("아이디 : ")
```

```
    #passwd = input("패스워드 : ")
```

```
    passwd = getpass("패스워드 : ")
```

```
    conn = sqlite3.connect('c:/pydb/mydb.db')
```

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

```
    sql = "SELECT memberId, passwd FROM member"
```

```
    cur.execute(sql)
```

```
    rs = cur.fetchall()
```

# member 로그인 구현

## member 로그인 구현

```
access = False    # 로그인 성공 여부
for i in rs:
    if memberId == i[0] and passwd == i[1]:
        print("로그인에 성공했습니다.")
        access = True
        break
if access == False:
    print("아이디나 비밀번호가 일치하지 않습니다.")
conn.close()
```

```
login()
```

# AUTOINCREMENT(자동증가)

## tbl\_book 테이블 생성

```
CREATE TABLE tbl_book(  
    book_no INTEGER PRIMARY KEY AUTOINCREMENT,  
    title TEXT NOT NULL,  
    publisher TEXT NULL,  
    page INTEGER  
);  
  
SELECT * FROM tbl_book;  
  
INSERT INTO tbl_book(title, publisher, page) VALUES ('웹 표준의 정석', '고경희', 600);  
  
INSERT INTO tbl_book(title, publisher, page) VALUES ('점프 투 장고', '박응용', 350);  
  
COMMIT;
```

**sequence(순서) -> autoincrement(1씩 자동증가)**

자동 증가(book\_no)는 입력하지 않아야 자동 증가함.

# book 테이블 자료 관리

tbl\_book – 데이터베이스 연결 함수

```
import sqlite3

def getconn(): # 데이터베이스 연결
    conn = sqlite3.connect('c:/pydb/mydb.db')
    return conn
```

# book 테이블 자료 관리

## tbl\_book – 책 추가하기

```
def insert_book():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "INSERT INTO tbl_book(title, publisher, page) VALUES (?, ?, ?)"  
    cur.execute(sql, ('천개의 파랑', '천선란', 300))  
    conn.commit()  
    conn.close()
```

# book 테이블 자료 관리

## tbl\_book – 자료 전체 목록

```
def select_book():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "SELECT * FROM tbl_book"  
    cur.execute(sql)  
    rs = cur.fetchall()    # rs=resultSet의 약자  
    for i in rs:  
        print(i)  
    conn.close()
```

```
(1, '웹 표준의 정석', '고경희', 600)  
(2, '점프 투 장고', '박응용', 350)  
(3, '천개의 파랑', '천선란', 300)
```

# book 테이블 자료 관리

## tbl\_book – 책 1권 검색하기

```
def select_one():  
    conn = getconn()  
    cur = conn.cursor()  
    sql = "SELECT * FROM tbl_book WHERE book_no=?"  
    cur.execute(sql, (1,)) # 튜플은 1개일때 코머를 꼭 붙인다.  
    rs = cur.fetchone()  
    print(rs)  
    conn.close()
```

# book 테이블 자료 관리

## tbl\_book – 책 정보 변경 및 삭제하기

```
def update_book():  
    conn = getconn()  
    cur = conn.cursor()  
    # 1번 책의 페이지를 600 -> 650으로 변경  
    sql = "UPDATE tbl_book SET page=? WHERE book_no=?"  
    cur.execute(sql, (650, 1))  
    conn.commit()  
    conn.close()
```

```
def delete_book():  
    conn = getconn()  
    cur = conn.cursor()  
    # 3번책 삭제  
    sql = "DELETE FROM tbl_book WHERE book_no=?"  
    cur.execute(sql, (3,))  
    conn.commit()  
    conn.close()
```

(1, '웹 표준의 정석', '고경희', 650)

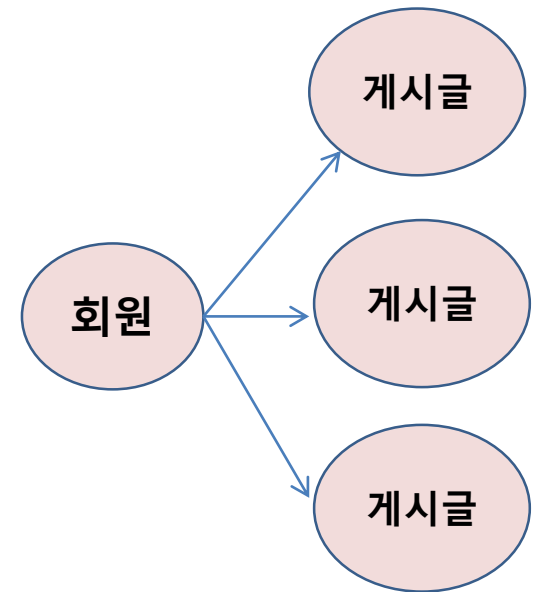
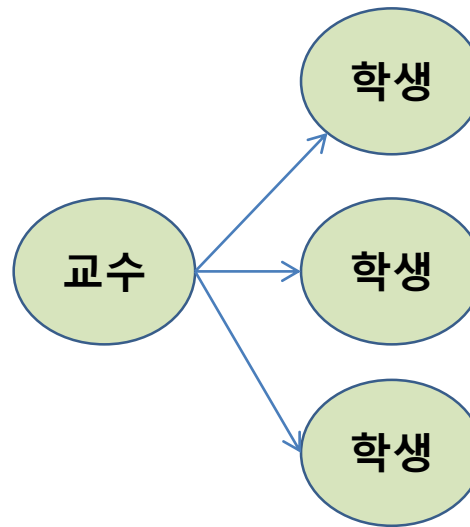
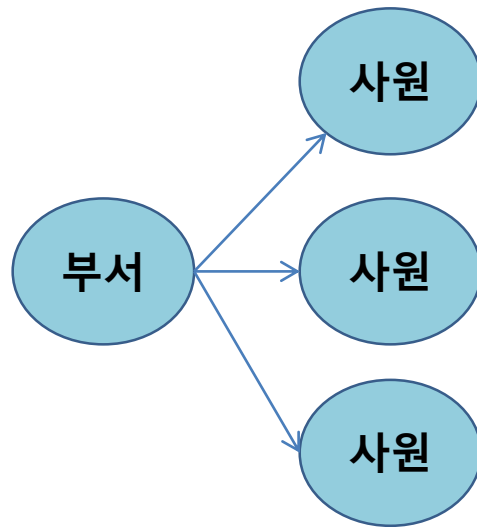
(2, '점프 투 장고', '박응용', 350)



# 관계(Releation)

## 엔티티(Entity) 관계(releation)

- 1대 多의 관계, 1대 1관계, 多 대 多 관계



# 관계(Releation)

## 외래키(FK : Foreign Key)

- 특정 테이블에 포함되어 있으면서 다른 테이블의 기본키로 지정된 키



# 데이터베이스 구축하기

## 부서와 직원 테이블 생성

- 사원(employee) 테이블에 Foreign Key 설정

```
-- 부서 테이블 --
CREATE TABLE department(
    deptid INTEGER,
    deptname TEXT NOT NULL,
    location TEXT NOT NULL,
    PRIMARY KEY(deptid)
);

-- 사원 테이블 --
CREATE TABLE employee(
    empid INTEGER,
    name TEXT NOT NULL,
    age INTEGER,
    deptid INTEGER,
    PRIMARY KEY(empid),
    FOREIGN KEY(deptid) REFERENCES department(deptid)
);
```

# 데이터베이스 구축하기

## 부서와 직원 자료 추가

```
-- 부서 자료 추가 --  
INSERT INTO department VALUES (10, '전산팀', '서울' )  
INSERT INTO department VALUES (20, '총무팀', '인천' )  
  
-- 사원 자료 추가 --  
INSERT INTO employee VALUES (501, '이강', 23, 10)  
INSERT INTO employee VALUES (502, '김산', 24, 20)  
INSERT INTO employee VALUES (503, '북한강', 25, 30)
```

에러가 발생하여 실행 중단됨.

결과: FOREIGN KEY constraint failed

19번째 줄:

```
INSERT INTO employee VALUES (503, '북한강', 25, 30)
```

|   | deptid | deptname | location |
|---|--------|----------|----------|
|   | 필터     | 필터       | 필터       |
| 1 | 10     | 전산팀      | 서울       |
| 2 | 20     | 총무팀      | 인천       |

| empid | name | age | deptid |
|-------|------|-----|--------|
| 필터    | 필터   | 필터  | 필터     |
| 501   | 이강   | 23  | 10     |
| 502   | 김산   | 24  | 20     |

부서 테이블에 부서코드 30이 없으므로, 사원 503을 추가 할 수 없다.

# 데이터베이스 구축하기

## 부서 자료 삭제

```
-- 부서 자료 삭제 --  
DELETE FROM department WHERE deptid = 20
```

```
에러가 발생하여 실행 중단됨.  
결과: FOREIGN KEY constraint failed  
17번째 줄:  
DELETE FROM department WHERE deptid = 20
```

사원 테이블에서 부서 테이블을 참조하고 있으므로 삭제할 수 없다.