8장. 데이터베이스 연동 (sqlite3)

데이터베이스(DataBase)

- **챗GPT 에게 물어보기**

+ 데이터베이스와 데이터베이스 언어인 SQL에 대해 알려줘

1️⃣ 데이터베이스(Database)란?

데이터베이스(DB)는

👉 *많은 데이터를 체계적으로 저장하고, 빠르게 찾고, 안전하게 관리하기 위한 공간*이에요.

엑셀과 비교하면 이해가 쉬워요:

- 엑셀: 파일 하나에 데이터 저장 (작을 때 OK)
- 데이터베이스:
  - 데이터가 수백만~수억 건이어도 처리 가능
  - 여러 사람이 동시에 사용 가능
  - 보안, 백업, 복구가 강력함

📌 예시:

- 쇼핑몰 → 회원정보, 주문내역, 결제정보
- 회사 → 직원정보, 급여, 프로젝트
- 앱 → 사용자 계정, 메시지, 로그 기록

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

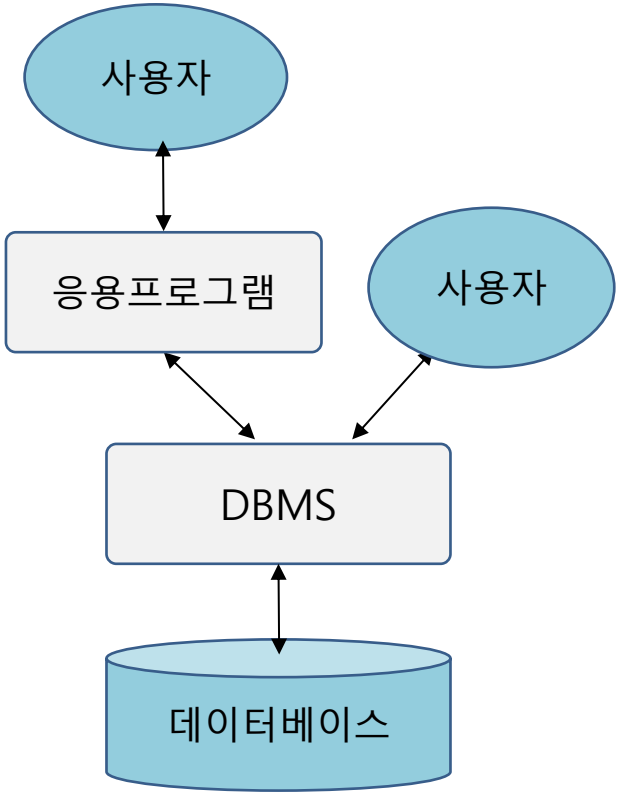
과목 테이블

과목번호	과목명	담당교수
0303	웹 프로그래밍	송미영
0116	데이터베이스	오용철

데이터베이스 관리 시스템

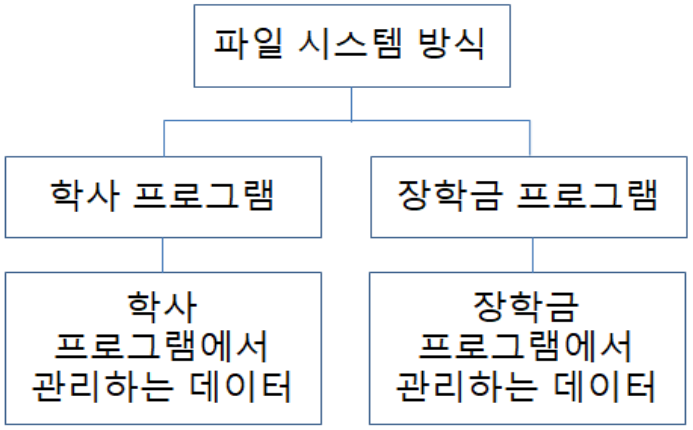
데이터베이스 관리 시스템(Database Managentment System)

- 많은 양의 데이터를 정교하게 구축하고 관리하는 소프트웨어이다.
- 데이터베이스의 정의, 데이터베이스 갱신, 질의 처리, 요지보수, 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.

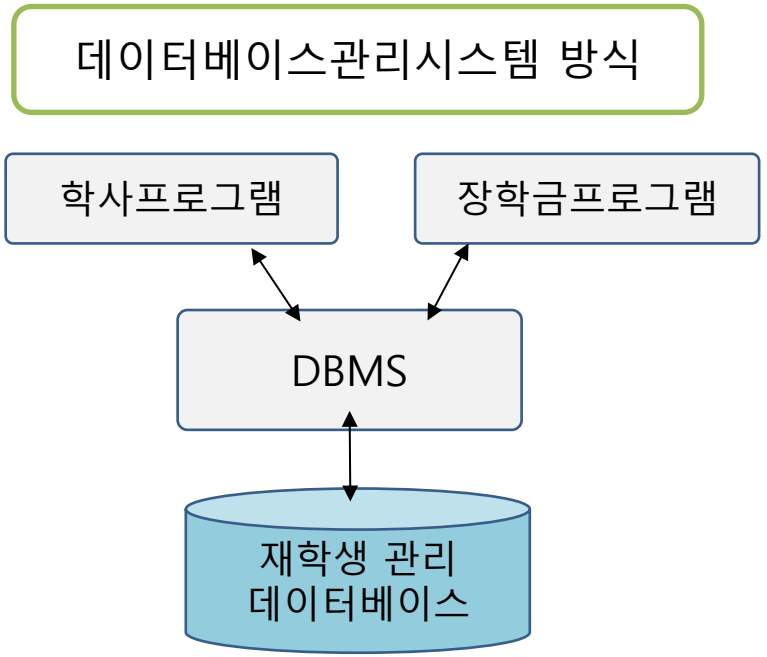


파일 시스템과 DBMS

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 논리모델을 구성하고 물리모델을 사용해 데이터베이스에 반영하는 작업
- 기본 요소

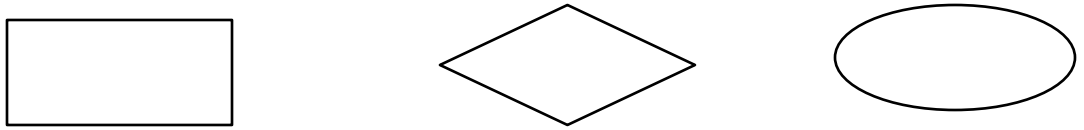
구분	개념	실제 예
엔티티(Entity)	물리적 개념에서는 테이블로 표현	고객, 상품, 주문
속성(Attribute)	물리적 개념에서는 칼럼(Column)으로 표현	고객아이디, 고객명, 주소
관계(Releationship)	기본키와 참조키로 정의 됨(일대일, 일대다)	고객과 주문과의 관계

데이터 모델링

◆ **개념적 설계**

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

→ **개체 관계도(E-R 다이어그램)** : Entity-Releationship 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**

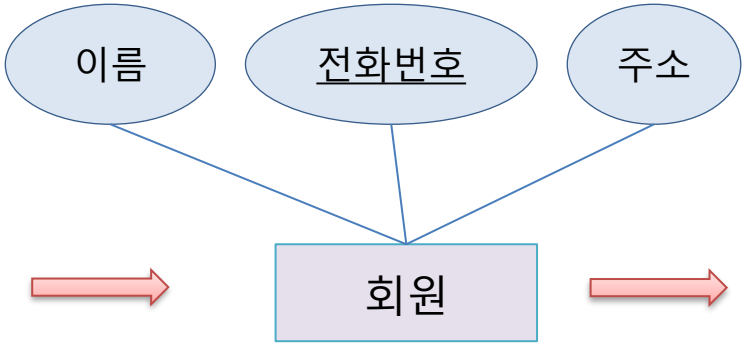
논리적 데이터베이스 구조를 실제 기계가 처리하기에 알맞도록 내부 저장 장치 구조와 접근 경로 등을 설계하는 과정

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

데이터 베이스 설계

현실 단계

개념 단계



논리 단계

회원

이름	전화번호	주소

물리 단계

```
name CHAR(20)
phone TEXT
address TEXT
```

개체	개체타입	레코드타입	자료형타입
특성	속성	특성	특성
값	값	값	값

관계형 데이터베이스

관계형 데이터 모델

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

  예) 회사의 사원정보, 소속된 부서정보

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

사원 정보
사원 번호
사원 이름
사원 직급
부서이름
위치

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

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

기본키

부서 코드	부서 이름	위치
10	전산팀	서울
20	총무팀	인천

기본키

외래키

사원 번호	사원 이름	나이	부서코드
501	이강	23	10
502	김산	24	20
503	남한강	31	10
504	북한강	35	20

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 Defiintion Language) - 데이터 정의어	테이블을 포함한 여러 객체를 생성, 수정, 삭제하는 명령어
DML(Data Manipulation Language) - 데이터 조작어	데이터를 저장, 검색, 수정, 삭제하는 명령어
DCL(Data Control Language) - 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 파일(모듈)이다.

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

❖ DB Browser for sqlite3

오픈소스 소프트웨어로 SQLite 데이터베이스를 GUI 기반으로 편리하게 조작할 수 있도록 해주는 tool이다. 데이터베이스 파일을 생성, 검색 및 편집하려는 사용자를 위해 설계된 고품질의 시각적 오픈 소스 도구입니다

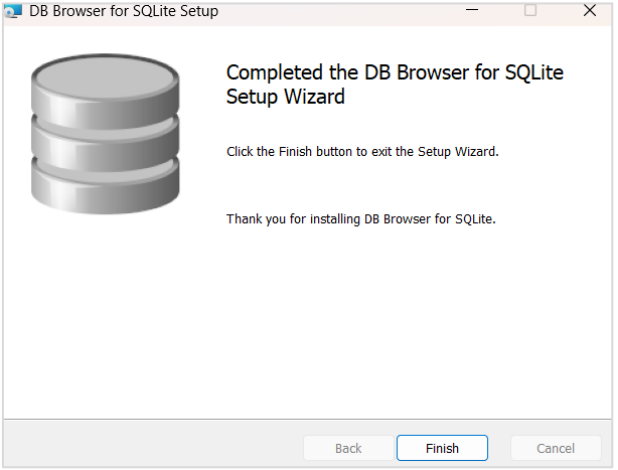
sqlite3

❖ DB 브라우저 설치

Windows

Our latest release (3.13.1) for Windows:

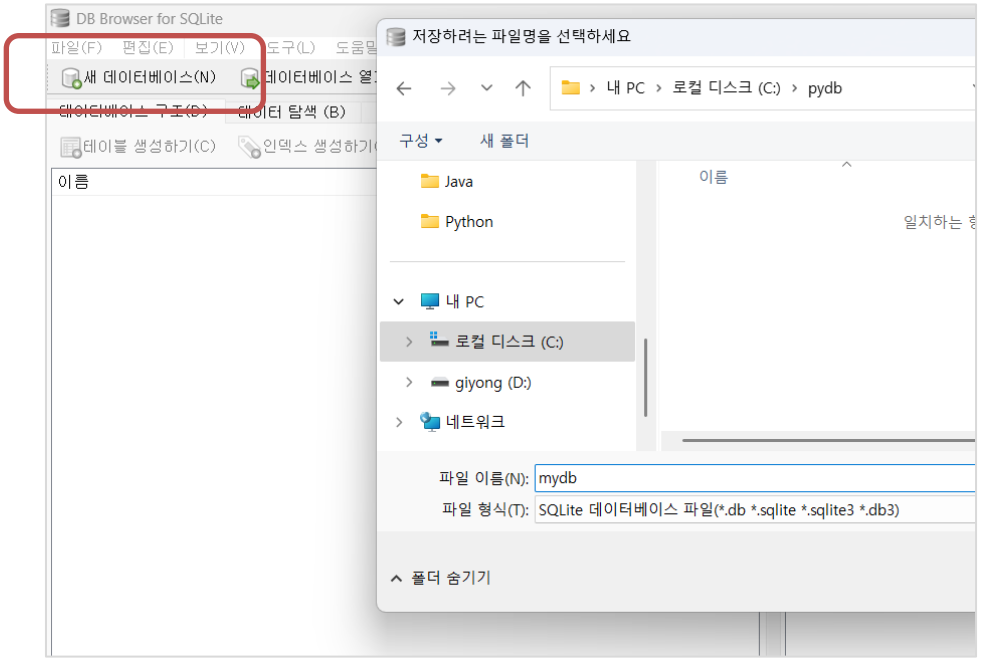
- DB Browser for SQLite - Standard installer for 32-bit Windows
- DB Browser for SQLite - .zip (no installer) for 32-bit Windows
- DB Browser for SQLite - Standard installer for 64-bit Windows
- DB Browser for SQLite - .zip (no installer) for 64-bit Windows

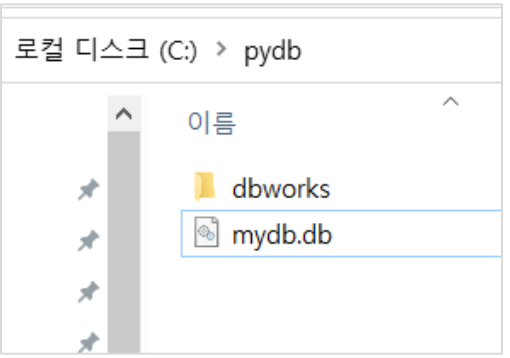


sqlite3

❖ 데이터베이스 생성

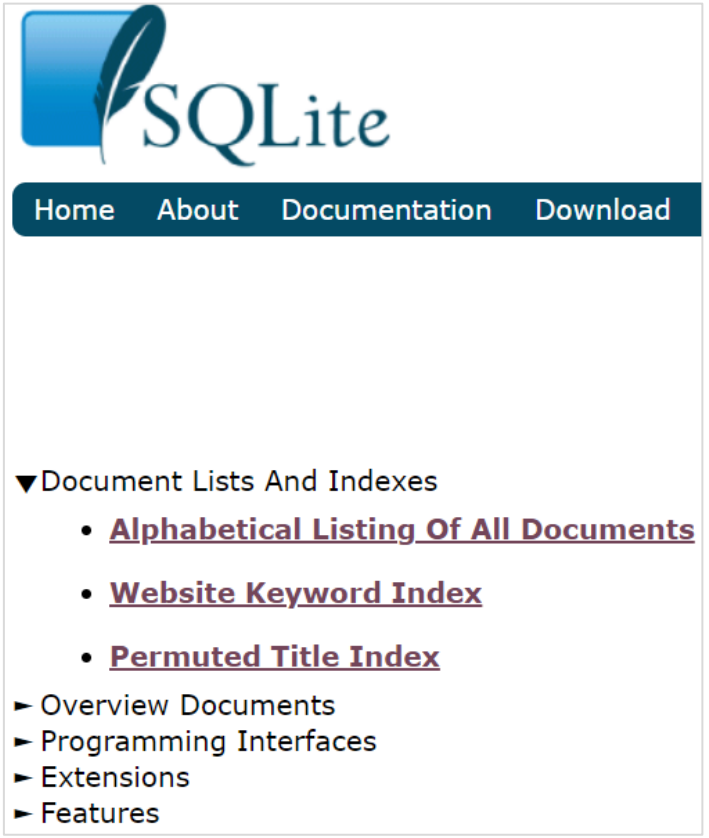
새 데이터베이스 > mydb.db 이름으로 저장

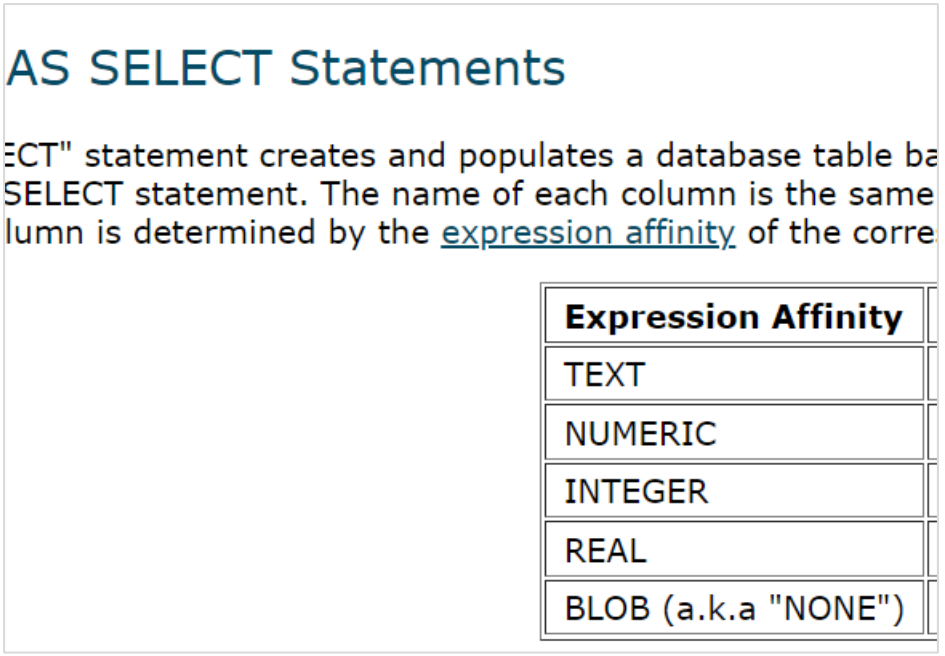




sqlite3

Sqlite3 Document > Alphabet.. > CREATE TABLE





sqlite3

- 데이터 타입 및 기초 문법

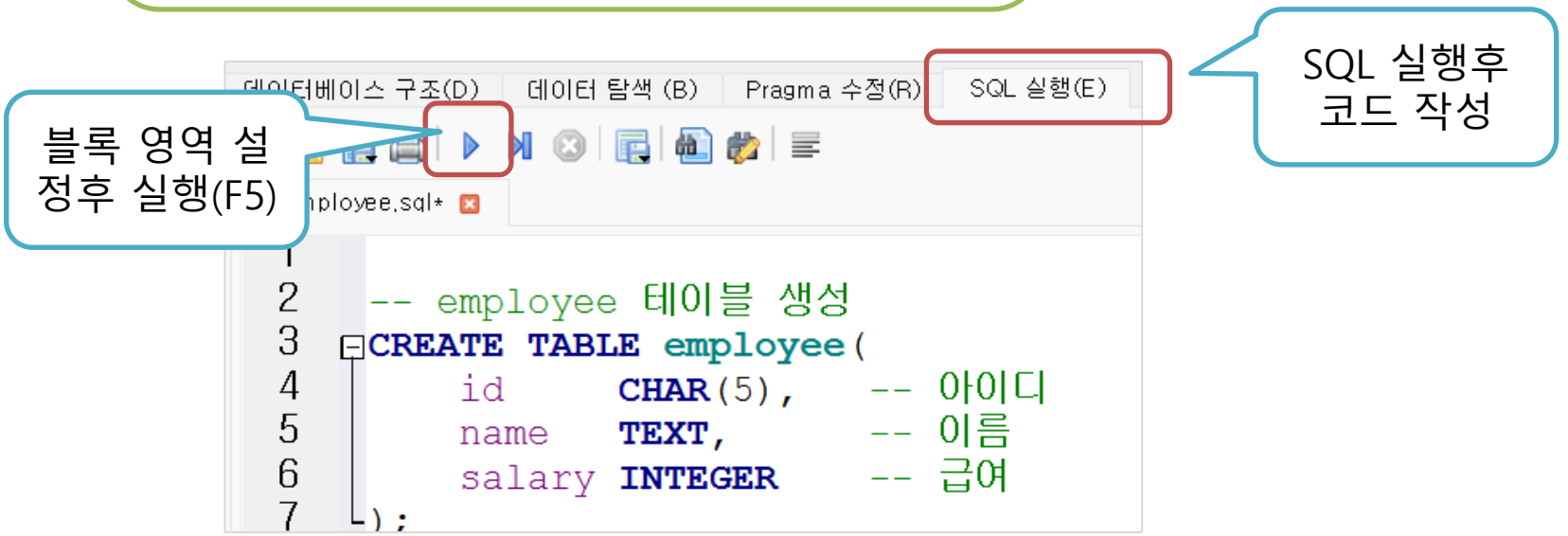
데이터 타입	설명
char	고정길이 문자
text	가변길이 문자(주로 사용)
integer	정수값
real	실수값
datetime	날짜와 시간

- 문자는 홑따옴표만 사용할 수 있음
- 한줄 주석 처리는 (--) , 여러 줄 주석은 /* ~ */
- 문장의 끝은 세미콜론(;) 사용

DDL – 테이블 생성

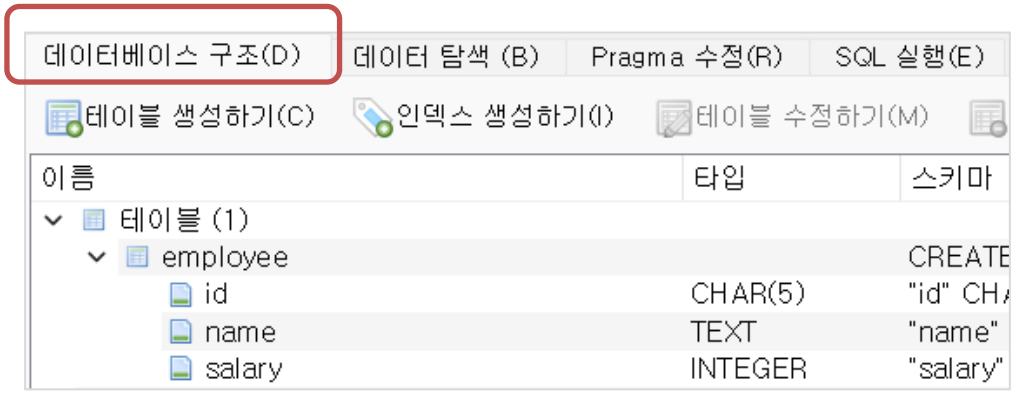
- 테이블 생성

```
CREATE TABLE 테이블이름(
        칼럼명 자료형
)
```



DDL – 테이블 생성

- DB 브라우저 사용하기
  - 데이터 베이스 구조 – 테이블 생성 확인
  - 데이터 탐색 – 표형태로 데이터 출력



DML – 자료 삽입

- 자료 삽입(Insert)

INSERT INTO 테이블이름(칼럼명) VALUES (값1, 값2)

```
-- 자료 추가
INSERT INTO employee(id, name, salary) VALUES ('e1001', '이정후', 3000000);
INSERT INTO employee(id, name, salary) VALUES ('e1002', '신유빈', 2500000);
INSERT INTO employee(id, name) VALUES ('e1002', '한강');

-- 커밋 실행(자료 추가, 변경, 삭제후)
COMMIT;
```

	id	name	salary
1	e1001	이정후	3000000
2	e1002	신유빈	2500000
3	e1002	한강	NULL

DML – 자료 검색

- 자료 검색(Select)

SELECT 칼럼명 FROM 테이블이름 (전체 자료 조회)

SELECT 칼럼명 FROM 테이블이름 WHERE 칼럼명 = 값 (특정 자료 조회)

```
-- 특정한 또는 모든 칼럼 출력
SELECT id, name FROM employee;

SELECT id, name, salary FROM employee;

SELECT * FROM employee;

-- 특정한 자료 조회 - where 절
SELECT * FROM employee WHERE id = 'e1001';

-- SELECT * FROM employee WHERE name = '신유빈';
SELECT * FROM employee WHERE name LIKE '신유빈';

-- 급여가 300만원 이상인 사원 검색
SELECT * FROM employee WHERE salary >= 3000000;

-- 급여가 없는 사원 조회
SELECT * FROM employee WHERE salary is NULL;
```

DML – 자료 검색

- 자료 검색(Select) - 정렬

SELECT 칼럼명 FROM 테이블이름 ORDER BY 칼럼명 DESC

```
-- 급여가 많은 순으로 정렬하기 - ORDER By
SELECT * FROM employee ORDER BY salary DESC;

-- 급여가 작은 순으로 정렬하기 - ORDER By
SELECT * FROM employee ORDER BY salary ASC;
```

DML – 자료 수정

- 자료 수정

UPDATE 테이블명 SET 칼럼명=값 WHERE 칼럼명=값

```
-- 이정후의 급여를 400만원으로 변경
UPDATE employee SET salary = 4000000 WHERE name LIKE '이정후';

-- id가 'e102'인 사원의 이름을 '임시현'으로 변경
UPDATE employee SET name = '임시현' WHERE id = 'e102';
```

	id	name	salary
1	e101	이정후	4000000
2	e102	임시현	2500000

DML – 자료 삭제

- 자료 삭제

DELETE FROM 테이블명 – 전체 자료 삭제

DELETE FROM 테이블명 WHERE 칼럼명=값 – 특정 자료 삭제

	id	name	salary
1	e101	이정후	3000000
2	e102	신유빈	2500000
3	e103	테스터	*NULL*

```
-- 이름이 '테스터'인 사원의 정보 출력
DELETE FROM employee WHERE name = '테스터';
```

무결성 제약

◆제약조건

테이블들은 각 속성(칼럼)에 대한 무결성을 유지하기 위한 다양한 제약 조건(Constraints)이 적용되어 있다.

제약 조건에는 NOT NULL, 기본키, 외래키, CHECK 등이 있다.

✓ NOT NULL

칼럼을 정의할 때 NOT NULL 제약 조건을 명시하면 반드시 데이터를 입력해야 한다.

```
칼럼명  데이터 타입 NOT NULL
```

무결성 제약

◆제약조건

✓기본키(PRIMARY KEY)

기본키는 Primay Key라고도 하며, UNIQUE와 NOT NULL 속성을 동시에 가진 제약 조건으로 테이블 당 1개의 기본키만 생성할 수 있다.

```
칼럼명  데이터 타입 PRIMARY KEY
또는
PRIMARY KEY(칼럼명, ...)
```

DDL – 테이블 삭제

- 테이블 삭제

DROP TABLE 테이블명

```
-- 테이블 삭제
DROP TABLE employee;
```

무결성 제약

◆ 제약조건

✓ id 중복저장 문제 발생 : 기본키 설정

✓ name : 비어있지 않도록 설정 – NOT NULL

```
-- employee 테이블 생성
CREATE TABLE employee_new(
    id CHAR(10) PRIMARY KEY,    -- 기본키
    name TEXT NOT NULL,         -- 이름
    salary INTEGER              -- 급여
);
```

무결성 제약

◆ 제약조건

```
-- 자료 삽입
INSERT INTO employee_new (id, name, salary) VALUES ('e101', '이정후', 3000000);
INSERT INTO employee_new (id, name, salary) VALUES ('e102', '신유빈', 2500000);
-- 아이디 중복 - 기본키 위배로 저장 안됨
INSERT INTO employee_new (id, name, salary) VALUES ('e102', '임시현', 3500000);
-- 이름 - NOT NULL 제약조거 위배
INSERT INTO employee_new (id, salary) VALUES ('e103', 2500000);
```

	id	name	salary
1	e101	이정후	3000000
2	e102	신유빈	2500000

회원 테이블 관리

◆ user 테이블 실습

```
CREATE TABLE user(
    u_id     TEXT PRIMARY KEY,
    u_passwd     TEXT NOT NULL,
    u_name   TEXT NOT NULL,
    u_joindate   TEXT DEFAULT (datetime('now', 'localtime'))
);
```

```
-- 유저 추가
INSERT INTO user(u_id, u_passwd, u_name) VALUES ('u101', '1234!@', '우상혁');
INSERT INTO user(u_id, u_passwd, u_name) VALUES ('u102', '1234#$', '신유빈');
INSERT INTO user(u_id, u_passwd, u_name) VALUES ('u103', '5678#$', '한강');

-- 유저 검색
SELECT * FROM user;

-- 최근 가입한 순으로 정렬하기
SELECT * FROM user ORDER BY u_joindate DESC;
```

	u_id	u_passwd	u_name	u_joindate
1	u103	5678#$	한강	2025-08-20 08:51:31
2	u102	1234#$	신유빈	2025-08-20 08:50:02
3	u101	1234!@	우상혁	2025-08-20 08:48:52

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)
```

DB 연결

- 파이썬으로 sqlite3에 연결

```
import sqlite3
# 데이터베이스 연결
try:
  conn = sqlite3.connect('c:/pydb/mydb.db')
  print(conn, "데이터베이스 연결 성공!")
except sqlite3.Error as e:
    print("데이터베이스 연결 실패:", e)
    conn = None
```

- 함수 실행 – main 영역

```
if __name__ == "__main__":

    # delete()
    # update()
    # select_one()
    # insert()
    select()
```

DB – 사원 관리

- 사원 추가

```
# 사원 추가
def insert():
    # with문이 알아서 close 처리
    with sqlite3.connect('c:/pydb/mydb.db') as conn:
        cursor = conn.cursor()
        sql = "INSERT INTO employee (id, name, salary) VALUES \
        ('e103', '김대리', 4000000)"
        cursor.execute(sql)
        conn.commit()
        print("사원 추가 완료!")
```

DB 연결

➢ 검색 함수 비교

구분	함수	사용 예
전체 검색	fetchall()	rows = cursor.fetchall()
1건 검색	fetchone()	row = cursor.fetchone()

DB – 사원 관리

➢ 전체 검색

```
# 사원 조회
def select():
    with sqlite3.connect('c:/pydb/mydb.db') as conn:
        cursor = conn.cursor()
        sql = "SELECT * FROM employee"
        cursor.execute(sql)
        rows = cursor.fetchall()
        for row in rows:
            print(row)
```

DB – 사원 관리

- 특정 사원 1명 검색

```
# 특정 사원 조회
def select_one():
    with sqlite3.connect('c:/pydb/mydb.db') as conn:
        cursor = conn.cursor()
        sql = "SELECT * FROM employee WHERE id = ?"
        cursor.execute(sql, ('e101',))
        row = cursor.fetchone()
        print(row)
```

DB – 사원 관리

- 사원 정보 수정

```
def update():
  with sqlite3.connect('c:/pydb/mydb.db') as conn:
    cursor = conn.cursor()
    sql = "UPDATE employee SET name=?, salary = ? WHERE id = ?"
    cursor.execute(sql, ('김대리', 3500000, 'e102'))
    conn.commit()
    print("사원 수정 완료!")
```

DB – 사원 관리

- 사원 정보 삭제

```
def delete():
    with sqlite3.connect('c:/pydb/mydb.db') as conn:
        cursor = conn.cursor()
        sql = "DELETE FROM employee WHERE id = ?"
        cursor.execute(sql, ('e103',))
        conn.commit()
        print("사원 삭제 완료!")
```

DB – 회원 관리

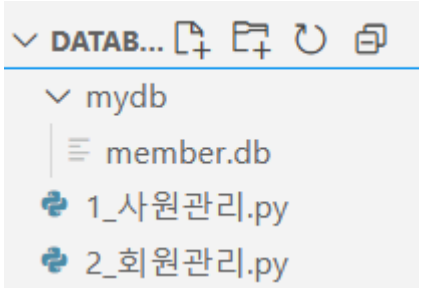
- **데이터 베이스 생성 – member.db**

  - 회원 테이블 생성 - member

칼럼 이름	자료형	설명
m_id	CHAR(5) PRIMARY KEY	회원번호
m_passwd	CHAR(10) NOT NULL	비밀번호
m_name	TEXT NOT NULL	이름
m_joindate	TEXT DEFAULT (datetime('now', 'localtime'))	가입일

DB – 회원 관리

- **DB 생성 – member db**




```
import sqlite3 as sq
# 데이터베이스 연결
try:
    conn = sq.connect('./mydb/member.db')
    print(conn, "데이터베이스 연결 성공!")
except sq.Error as e:
        print("데이터베이스 연결 실패:", e)
        conn = None
```

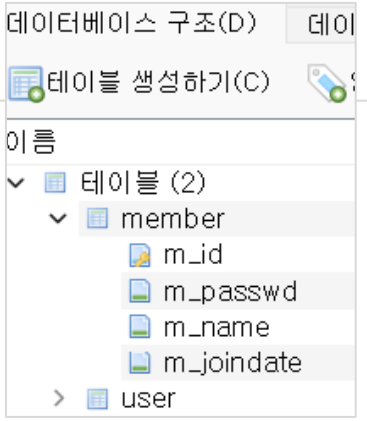
DB – 회원 관리

- DB 연결 함수 정의

```
def getconn():
    try:
        conn = sql.connect("./mydb/member.db")  # 연결 객체 생성
        return conn
    except sql.Error as e:
        print(f"DB 오류 발생: {e}")
        return None
```

DB – 회원 관리

- 테이블 생성 - member




```
def create_table():
    with getconn() as conn: # getconn() 함수 호출
        cursor = conn.cursor()
        sql = '''CREATE TABLE member (
                    m_id TEXT PRIMARY KEY,
                    m_passwd TEXT NOT NULL,
                    m_name TEXT NOT NULL,
                    m_join_date TEXT default (datetime('now','localtime')))'''
        cursor.execute(sql)
        conn.commit()
        print("테이블 생성 완료!")
        # conn.close()  # with문이 알아서 close 처리
```

DB – 회원 관리

- 회원 등록

```
def insert_member():
    with getconn() as conn:
        cursor = conn.cursor()
        sql = "INSERT INTO member (m_id, m_passwd, m_name) VALUES (?, ?, ?)"
        cursor.execute(sql, ('m1001', 'm1357@!', '안세영'))
        conn.commit()
        print("회원 추가 완료!")
```

DB – 회원 관리

- 회원 전체 검색

```
def selet_all():
    with getconn() as conn:
        cursor = conn.cursor()
        sql = "SELECT * FROM member"
        cursor.execute(sql)
        rows = cursor.fetchall()
        for row in rows:
            print(row)
```

DB – 회원 관리

- 회원 1명 검색

```
def select_one(m_id):
    with getconn() as conn:
        cursor = conn.cursor()
        sql = "SELECT * FROM member WHERE m_id = ?"
        cursor.execute(sql, (m_id,))
        row = cursor.fetchone()
        print(row)

# 메인 영역
if __name__ == "__main__":
    # create_table()
    # insert_member()
    # selet_all()
    select_one('m1002')
```

DB – 회원 관리

- 회원 수정

```
def update_member():
    with getconn() as conn:
        cursor = conn.cursor()
        sql = "UPDATE member SET m_name=? WHERE m_id = ?"
        cursor.execute(sql, ('김길리', 'm1001'))
        conn.commit()
        print("회원 수정 완료!")
```

DB – 회원 관리

- 회원 삭제

```
def delete_member():
    with getconn() as conn:
        cursor = conn.cursor()
        sql = "DELETE FROM member WHERE m_id = ?"
        cursor.execute(sql, ('m1002',))
        conn.commit()
        print("회원 삭제 완료!")
```

DB – 회원 관리

- 테이블 삭제

```
def drop_table():
    '''테이블 삭제'''
    conn = getconn()
    cur = conn.cursor()
    sql = "DROP TABLE member"
    cur.execute(sql)  # 실행
    conn.commit()     # 커밋(트랜잭션 완료)
    print("테이블 삭제")
```

DB – 회원 관리

- 회원 등록 – 리스트 사용

```python
# 회원을 여러명 등록(리스트 사용)
def insert_members():
    '''회원 등록'''
    conn = getconn()
    cur = conn.cursor()
    # 동적 바인딩 방식(? - 칼럼값 입력), 튜플 형태로 입력
    sql = "INSERT INTO member VALUES(?, ?, ?, datetime('now', '+9 hours'))"
    member_list = [
        ['m1003', "m0000", "콩쥐"],
        ['m1004', "m1111", "팥쥐"],
        ['m1005', "m2222", "심청"],
    ]
    cur.executemany(sql, member_list)
    conn.commit()
    print("회원 등록 완료!")
```

```
('m1001', 'm2468@!', '우영우', '2025-08-20 09:11:27')
('m1002', 'm1357@!', '오상식', '2025-08-20 09:18:38')
('m1003', 'm0000', '콩쥐', '2025-08-20 09:24:52')
('m1004', 'm1111', '팥쥐', '2025-08-20 09:24:52')
('m1005', 'm2222', '심청', '2025-08-20 09:24:52')
```