



SQLite

Update – 2018.07

Contents

- 데이터 베이스 Intro
- 데이터베이스 종류
- Keyword
- 데이터베이스 구축 과정
- SQLITE INTRO
- SQLITE CLIENT TOOL
- SQLITE3 설치
- SQLITE3 전용 명령어
- 파이썬 연동

DATABASE Intro

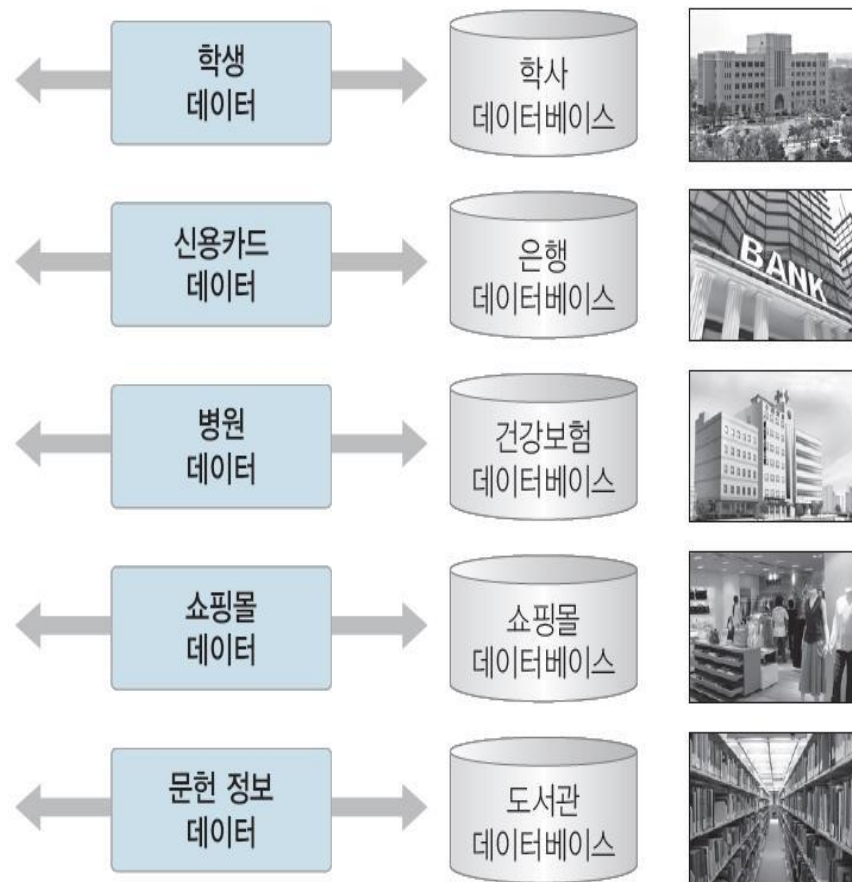
- 데이터 : 관찰의 결과로 나타난 정량적 혹은 정성적인 실제 값
- 정보 : 데이터에 의미를 부여한 것
- 데이터베이스란?

조직에 필요한 정보를 얻기 위해 논리적으로 연관된 데이터를 모아 구조적으로 통합해 놓은 것



DATABASE Intro

일상생활에서 생성되는 데이터베이스



DATABASE Intro

데이터베이스의 활용 분야

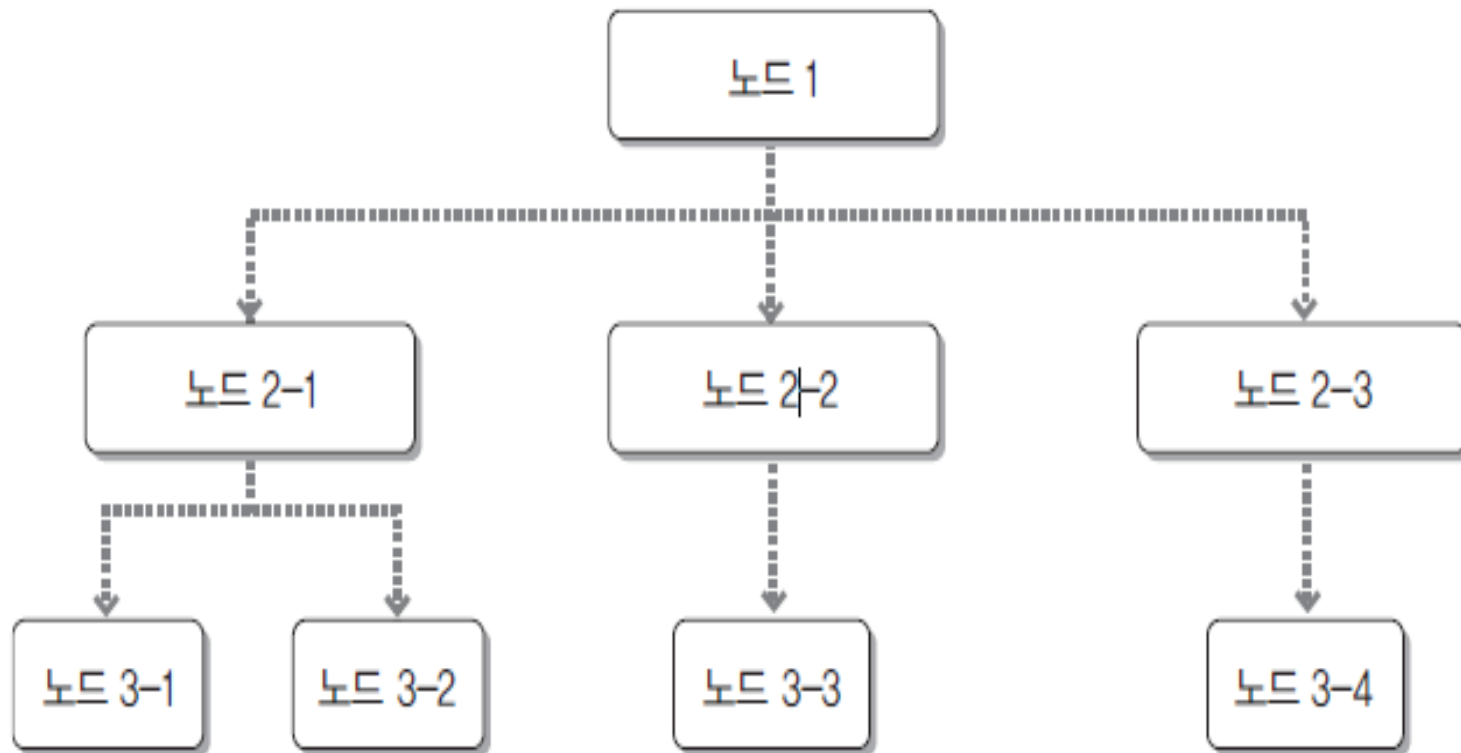
| 종류 | 특징 |
|--------|--|
| 생활과 문화 | <ul style="list-style-type: none">• 기상정보 : 날씨 정보를 제공• 교통정보 : 교통상황 정보를 제공• 문화예술정보 : 공연이나 인물에 관한 정보를 제공 |
| 비즈니스 | <ul style="list-style-type: none">• 금융정보 : 금융, 증권, 신용에 관한 정보를 제공• 취업정보 : 노동부와 기업의 채용 정보를 제공• 부동산정보 : 공공기관이나 민간의 토지, 매물, 세금 정보를 제공 |
| 학술정보 | <ul style="list-style-type: none">• 연구학술정보 : 논문, 서적, 저작물에 관한 정보를 제공• 특허정보 : 특허청의 정보를 기업과 연구자에게 제공• 법률정보 : 법제처와 대법원의 법률 정보를 제공• 통계정보 : 국가기관의 통계 정보를 제공 |

DB Kind

→ 계층형 DBMS

- 처음으로 나온 DBMS 개념 - 1960년대에 시작
- 각 계층은 트리Tree 형태, 1:N 관계
- 문제점
 - 처음 구축한 이후 그 구조를 변경하기가 상당히 까다로움
 - 주어진 상태에서의 검색은 상당히 빠름
 - 접근 유연성 부족해서 임의의 검색에는 어려움

DB Kind



DB Kind

→ 망형 DBMS

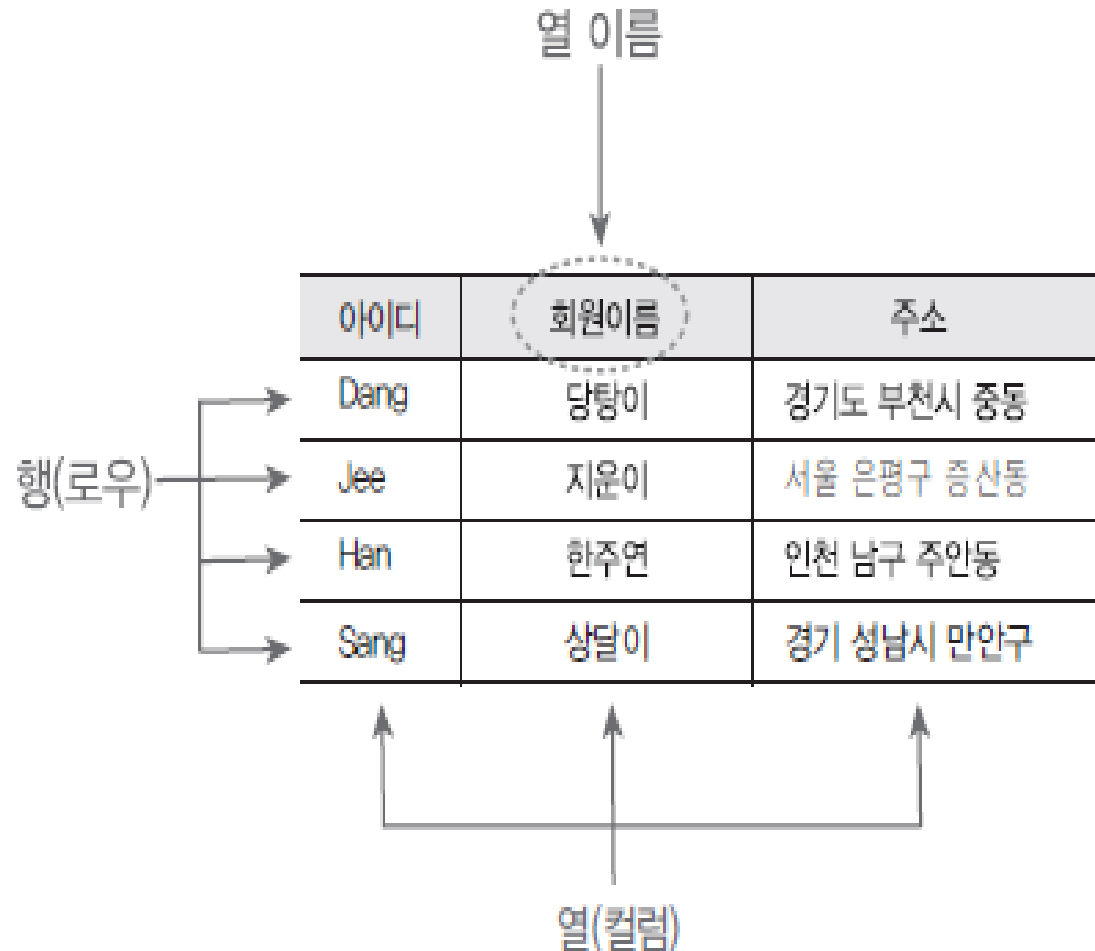
- 처음으로 나온 DBMS 개념 - 1960년대에 시작
- 1:1, 1:N, N:M(다대다) 관계 지원
- 문제점
 - 복잡한 내부 포인터
 - 프로그래머가 이 모든 구조를 이해해야만 프로그램의 작성 가능

DB Kind

→ 관계형 데이터베이스

- ◆ RDB (Relational DataBase)
- ◆ 2차원 표 이용한 데이터 목록화 관리
 - Excel, Google Docs 등 스프레드시트
- ◆ 자연스럽고 직관적인 이해 가능
- ◆ Oracle, MariaDB, mySQL, SQLite, PostgreSQL

DB Kind



Keyword : SQL

- 관계형 데이터베이스에서 데이터 조작하기 위한 언어
- 모국어 말하는 것처럼 데이터 조작
- 간단한 영어 문장과 유사
- 누구나 자연스럽게 데이터 조작 가능
- 간단한 기본 조작 명령어
 - SELECT(검색)
 - INSERT(등록)
 - UPDATE(갱신)
 - DELETE(제거)

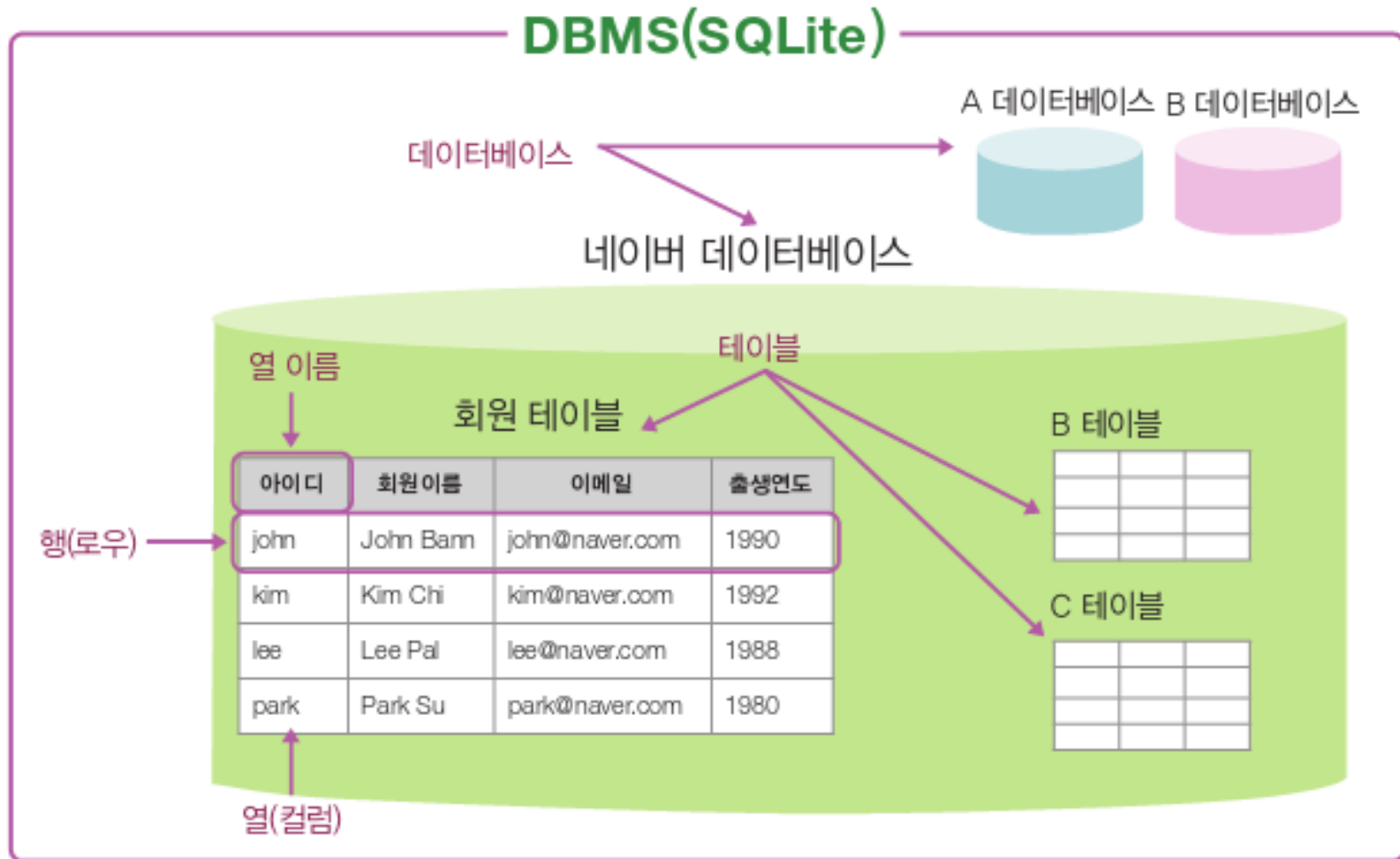
Ex) "주소가 서울시인 사람의 이름을 검색한다"

```
SELECT 이름
FROM 주소록
WHERE 주소 LIKE '서울시';
```

Keyword : TABLE

- 테이블, 행, 열
 - 관계용 데이터베이스와 SQL의 용어
 - 테이블(table)
 - 관계형 데이터베이스의 2차원 표
 - 열(column)과 행(row)
 - 가로 및 세로축
 - 열과 행 교차하는 부분을 셀(cell)이라 함

Keyword :Table



데이터베이스 구축 과정

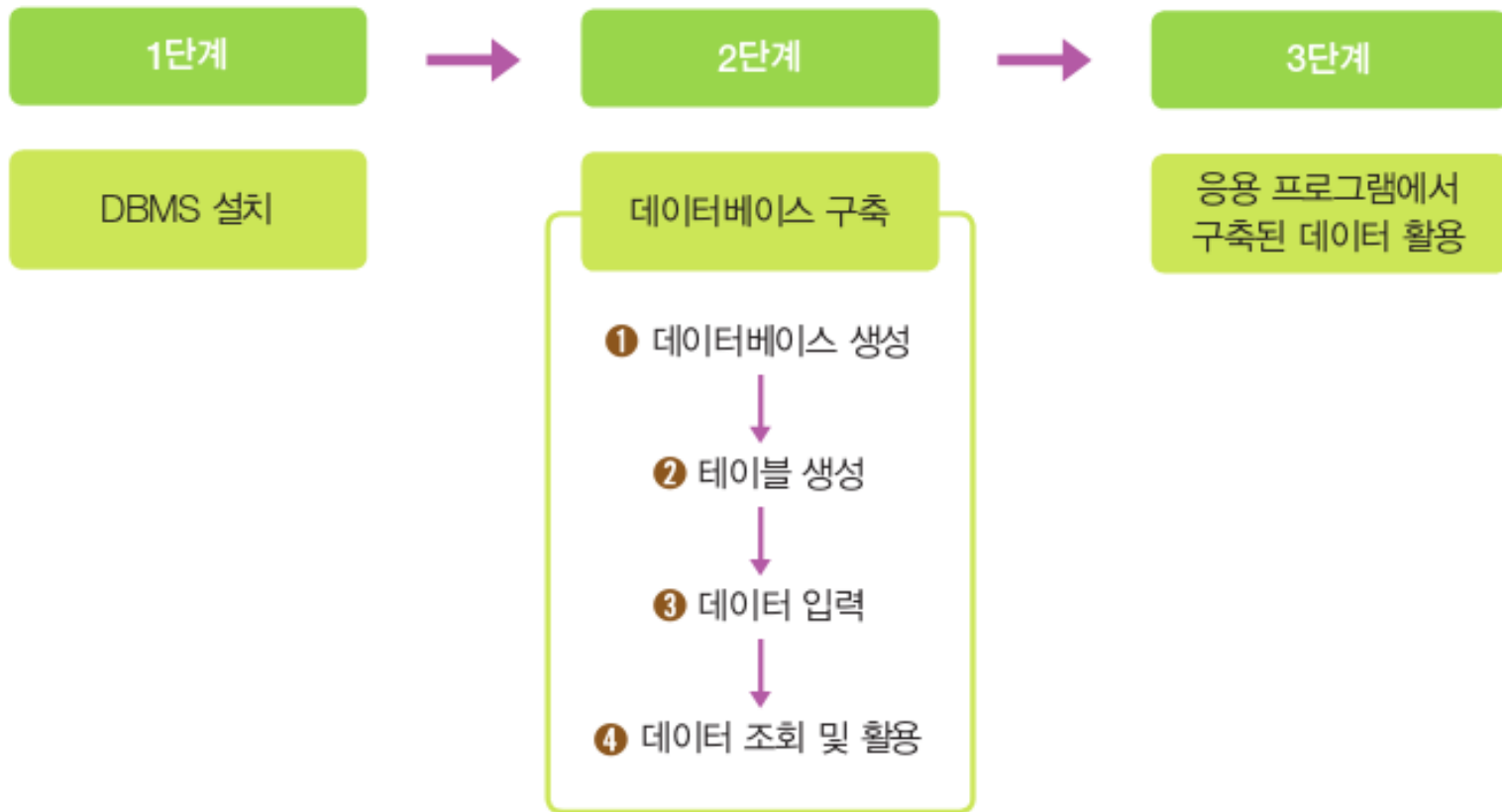


그림 13-2 데이터베이스 구축 및 운영 과정

SQLITE Intro

- 클라이언트 응용 프로그램에 임베디드되어 동작하는 오픈 소스 DBMS
- 안드로이드, iOS, macOS에 기본적으로 포함
- 데이터베이스 전체를 파일 하나에 저장



SQLite

[Home](#) [About](#) [Documentation](#) [Download](#) [License](#) [Support](#) [Purchase](#)

SQLite is a [self-contained](#), [high-reliability](#), [embedded](#), [full-featured](#), [public-domain](#), SQL database engine. SQLite is the [most used](#) database engine in the world.

[More Info](#)

Latest Release: [Version 3.24.0](#) (2018-06-04). [Download](#) [Prior Releases](#)

Sponsors

Ongoing development and support of SQLite is made possible in part by [SQLite Consortium](#) members, including:

 Navigation Data Standard

 Bloomberg

 Bentley
Sustaining Infrastructure

 mozilla

 Expensify

SQLITE Tool

■ DBMS 설치

■ SQLite 설치

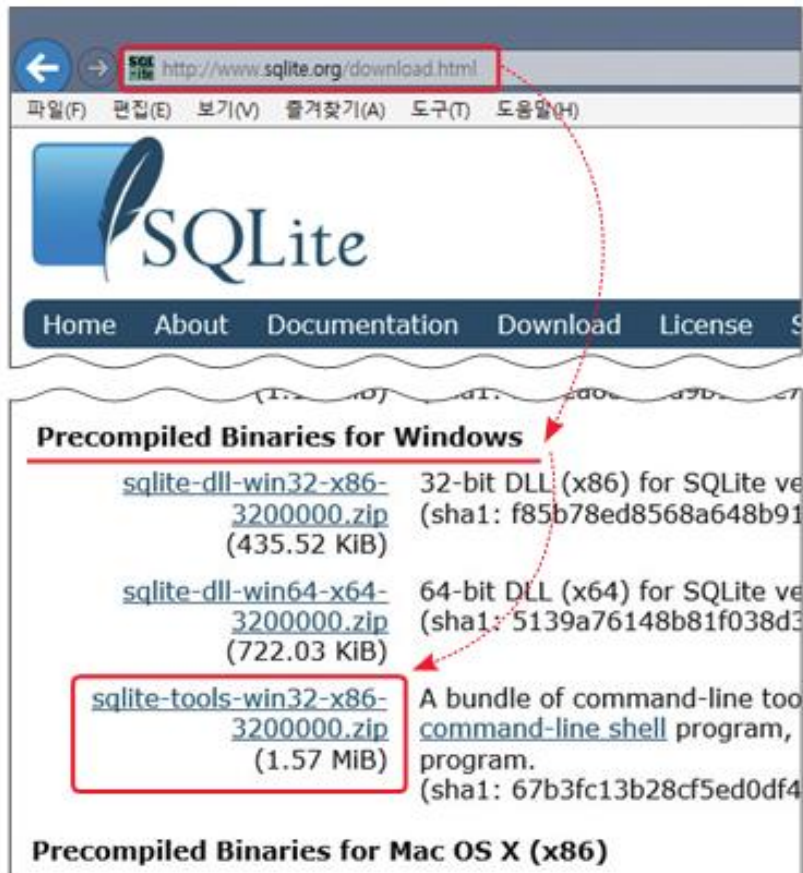


그림 13-3 sqlite 다운로드

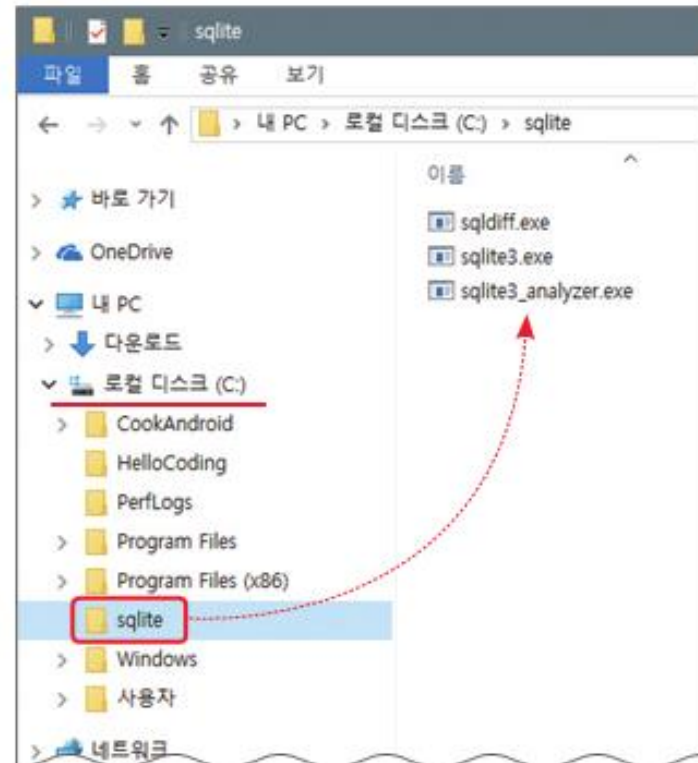
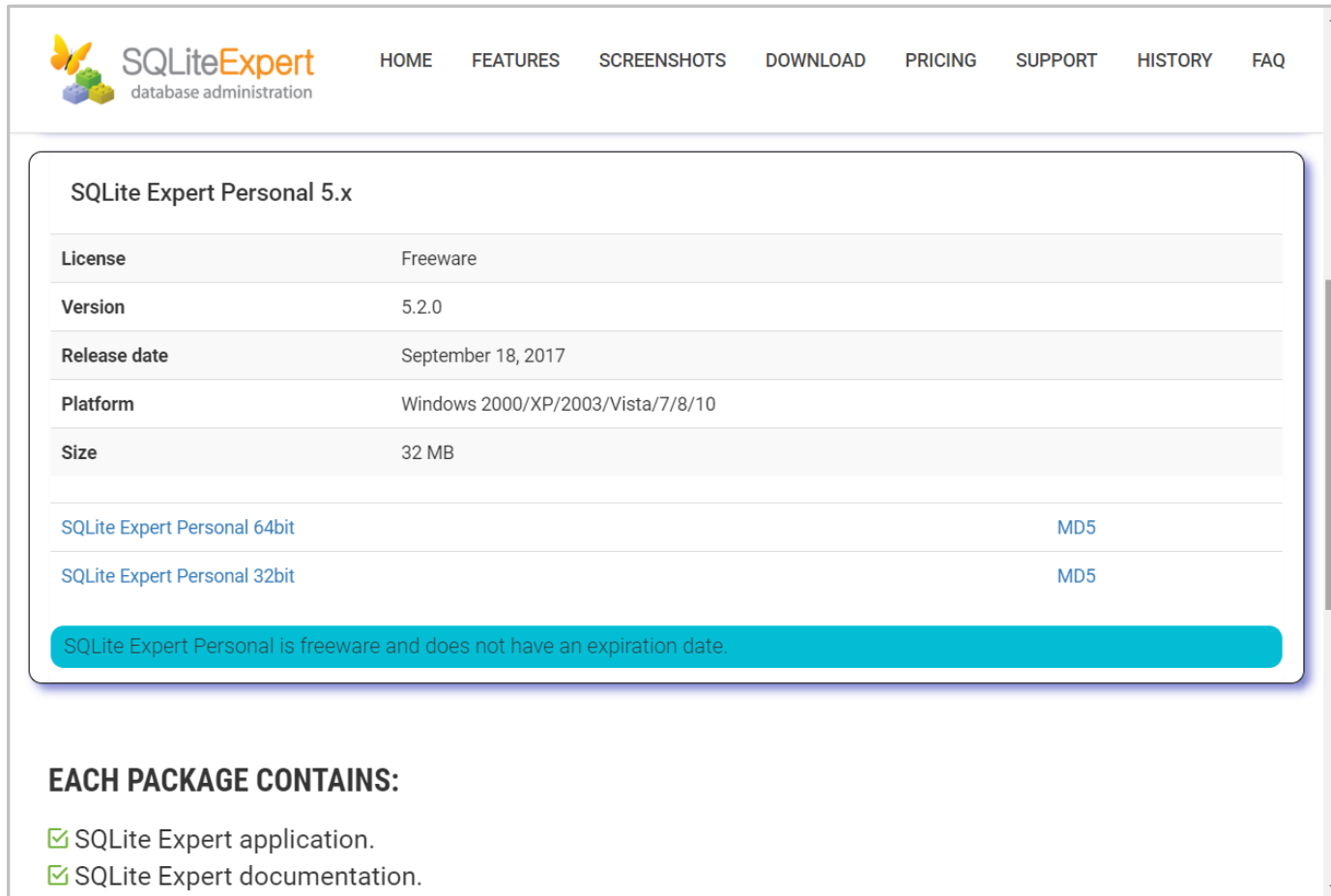


그림 13-4 sqlite 폴더

SQLITE Client Tool

- SQLITE Expert Tool

<http://www.sqliteexpert.com>



The screenshot shows the SQLite Expert website. The header includes the logo and navigation links: HOME, FEATURES, SCREENSHOTS, DOWNLOAD, PRICING, SUPPORT, HISTORY, and FAQ. The main content area is titled "SQLite Expert Personal 5.x" and contains a table with the following details:

| | |
|--------------|-----------------------------------|
| License | Freeware |
| Version | 5.2.0 |
| Release date | September 18, 2017 |
| Platform | Windows 2000/XP/2003/Vista/7/8/10 |
| Size | 32 MB |

Below the table, there are two links for downloading the software, each with an associated MD5 hash:

| | |
|--|---------------------|
| SQLite Expert Personal 64bit | MD5 |
| SQLite Expert Personal 32bit | MD5 |

A blue banner at the bottom of the product details section states: "SQLite Expert Personal is freeware and does not have an expiration date."

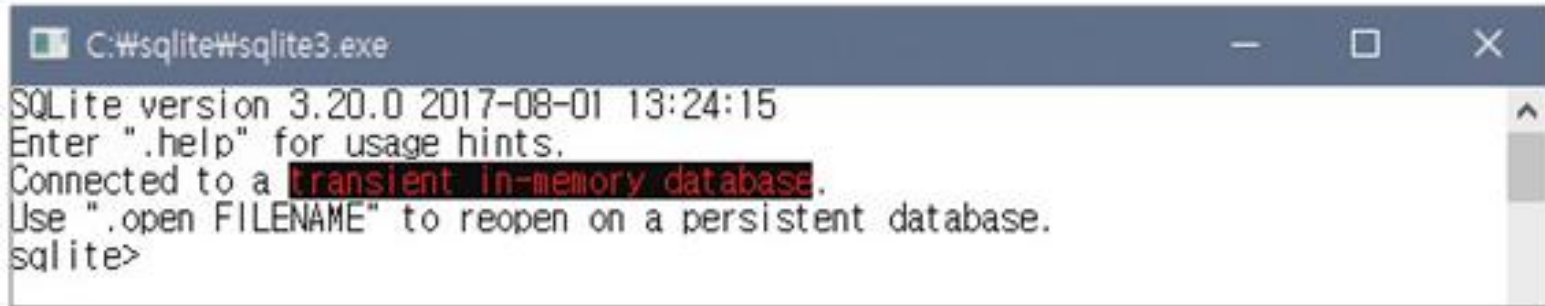
EACH PACKAGE CONTAINS:

- ✓ SQLite Expert application.
- ✓ SQLite Expert documentation.

SQLITE3 접속

■ 데이터베이스 명령어

- SQLite 에 접속

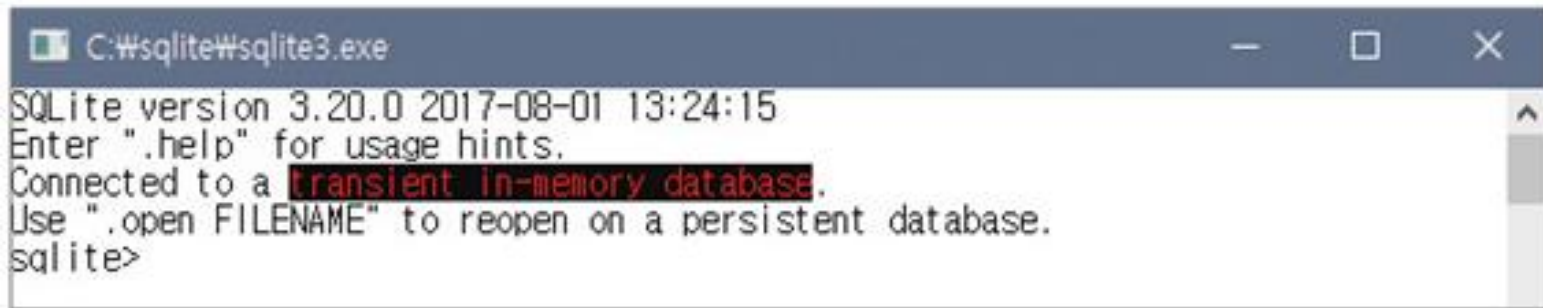


```
C:\sqlite#\sqlite3.exe
SQLite version 3.20.0 2017-08-01 13:24:15
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

SQLITE 명령어

■ 데이터베이스 명령어

■ SQLite 에 접속



```
C:\sqlite\sqlite3.exe
SQLite version 3.20.0 2017-08-01 13:24:15
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

자주 사용하는 SQLite 명령어

- .open 데이터베이스이름 : 데이터베이스를 열거나 생성한다
- . table : 현재 데이터베이스의 테이블 목록을 보여 준다.
- . schema 테이블이름 : 테이블의 열 및 데이터 형식 등 정보를 보여 준다.
- . header on : SELECT 문으로 출력할 때 헤더를 보여 준다.
- . mode column : SELECT 문으로 출력할 때 컬럼 모드로 출력한다.
- . quit : SQLite 를 종료한다.
- SELECT 문 사용 전
' . header on ' , ' . mode column ' 설정하면 결과 화면 보기 • 좋게 출력

SQLITE 명령어

■ 데이터베이스 명령어

```
sqlite> .open chinook.db
sqlite> .table
albums          employees      invoices       playlists
artists         genres         media_types    tracks
customers       invoice_items  playlist_track
sqlite> .schema albums
CREATE TABLE IF NOT EXISTS "albums"
(
  [AlbumId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  [Title] NVARCHAR(160) NOT NULL,
  [ArtistId] INTEGER NOT NULL,
  FOREIGN KEY ([ArtistId]) REFERENCES "artists" ([ArtistId])
    ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE INDEX [IFK_AlbumArtistId] ON "albums" ([ArtistId]);
sqlite> .head on
sqlite> .mode column
```

SQL 명령어

■ 데이터 조회하기 - SELECT

```
SELECT * FROM 테이블이름;
```

```
SELECT 열이름1, 열이름2, ... FROM 테이블이름 WHERE 조건;
```

```
sqlite> select * from artists;  
1|AC/DC  
2|Accept  
3|Aerosmith  
4|Alanis Morissette  
5|Alice In Chains  
6|Antonio Carlos Jobim  
7|Apocalyptica  
8|Audioslave  
9|BackBeat  
10|Billy Cobham
```

```
sqlite> .header on  
sqlite> .mode column  
sqlite> select * from artists;  
ArtistId      Name  
-----  
1             AC/DC  
2             Accept  
3             Aerosmith  
4             Alanis Mor  
5             Alice In C  
6             Antonio Ca  
7             Apocalypti  
8             Audioslave  
9             BackBeat
```

SQL 명령어

■ 데이터 조회하기 - SELECT

SELECT 열이름1, 열이름2, ... FROM 테이블이름 WHERE 조건;

```
sqlite> select name from artists;
```

```
Name
```

```
-----
```

```
AC/DC
```

```
Accept
```

```
Aerosmith
```

```
Alanis Mor
```

```
Alice In C
```

```
Antonio Ca
```

```
Apocalypsi
```

```
Audioslave
```

```
BackBeat
```

```
Black Sabbath
```

SQL 명령어

■ 데이터 조회하기 - SELECT

```
SELECT * FROM 테이블이름 limit 5;
```

```
sqlite> .open chinook.db
sqlite> .table
albums          employees      invoices      playlists
artists         genres        media_types   tracks
customers       invoice_items playlist_track
sqlite> select * from albums limit 5;
AlbumId      Title                                              ArtistId
-----
1      For Those About To Rock We Salute You          1
2      Balls to the Wall                               2
3      Restless and Wild                              2
4      Let There Be Rock                              1
5      Big Ones                                       3
sqlite>
```

SQL 명령어

■ 데이터 조회하기 - SELECT

SELECT 열이름1, 열이름2, ... FROM 테이블이름 WHERE 조건;

SQL WHERE AND, OR, NOT Clause

```
sqlite> select * from albums
...> where ArtistID = 12;
AlbumId      Title                ArtistId
-----
16           Black Sabbath       12
17           Black Sabbath       12
sqlite>
sqlite> select * from albums
...> where ArtistID = 12 or ArtistID = 2;
AlbumId      Title                ArtistId
-----
2            Balls to the Wall    2
3            Restless and Wild    2
16           Black Sabbath       12
17           Black Sabbath Vol    12
sqlite>
```


SQL 명령어

■ 데이터 조회하기 - SELECT

SELECT 열이름1, 열이름2, ... FROM 테이블이름 WHERE 조건;

SQL WHERE AND, OR, NOT Clause

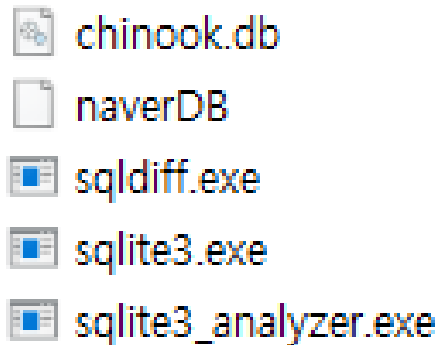
```
sqlite> .header on
sqlite> .mode column
sqlite> select *
...> from playlists
...> where Playlistid < 5;
PlaylistId  Name
-----
1           Music
2           Movies
3           TV Shows
4           Audiobooks
```

SQL 명령어

■ 데이터베이스 생성하기

`.open 데이터베이스이름`

```
sqlite>  
sqlite> .open naverDB  
sqlite>
```



A list of files and executables with their respective icons:

- chinook.db (database icon)
- naverDB (database icon)
- sqldiff.exe (executable icon)
- sqlite3.exe (executable icon)
- sqlite3_analyzer.exe (executable icon)

SQL 명령어

■ 테이블 생성하기

CREATE 테이블이름 (열1 데이터형식, 열2 데이터형식,...)

```
sqlite> .open naverDB
sqlite> create table
...> userTable(id char(4), userName char(15),
...> email char(15), birthYear int);
sqlite> .table
userTable
```

```
sqlite> .schema userTable
CREATE TABLE userTable(id char(4), userName char(15),
email char(15), birthYear int);
sqlite>
```

SQL 명령어

■ 테이블 생성하기 : 데이터타입

| Example Typenames From The CREATE TABLE Statement or CAST Expression | Resulting Affinity | Rule Used To Determine Affinity |
|---|--------------------|---------------------------------|
| INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8 | INTEGER | 1 |
| CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB | TEXT | 2 |
| BLOB <i>no datatype specified</i> | BLOB | 3 |
| REAL DOUBLE DOUBLE PRECISION FLOAT | REAL | 4 |

SQL 명령어

■ 데이터 삽입하기

INSERT 테이블 이름 VALUES (값1, 값2...);

```
sqlite> insert into userTable
...> values ('j001','김순돌','king@naver.com',2001);
sqlite> select * from userTable;
j001|김순돌|king@naver.com|2001
sqlite> .header on
sqlite> .mode column
sqlite> select * from userTable;
id          userName      email          birthYear
-----
j001        김순돌        king@naver.com 2001
sqlite>
```

SQL 명령어

■ 데이터 삽입하기

INSERT 테이블 이름 VALUES (값1, 값2...);

```
sqlite> insert into userTable
...> values ('m001', '이소연', 'lee@naver.com', 2008);
sqlite> insert into userTable
...> values ('sh890', '마동탁', 'mado@naver.com', 1999);
sqlite> insert into userTable
...> values ('guio0', '구하라', 'guif@naver.com', 1999);
sqlite> insert into userTable
...> values ('doraju', '이하니', 'hani@naver.com', 1989);
sqlite> select * from userTable;
```

| id | userName | email | birthYear |
|--------|----------|----------------|-----------|
| j001 | 김순돌 | king@naver.com | 2001 |
| m001 | 이소연 | lee@naver.com | 2008 |
| sh890 | 마동탁 | mado@naver.com | 1999 |
| guio0 | 구하라 | guif@naver.com | 1999 |
| doraju | 이하니 | hani@naver.com | 1989 |

```
sqlite>
```

SQL 명령어

■ 데이터 삭제하기

DELETE FROM 테이블 이름 WHERE 열이름=값;

```
sqlite> select * from userTable;
id          userName      email          birthYear
-----
m001        이소연        lee@naver.com  2008
sh890       마동탁        mado@naver.co  1999
guio0       구하라        guif@naver.co  1999
doraju      이하니        hani@naver.co  1989
sqlite> delete from userTable where id = 'j001';
sqlite> select * from userTable;
id          userName      email          birthYear
-----
m001        이소연        lee@naver.com  2008
sh890       마동탁        mado@naver.co  1999
guio0       구하라        guif@naver.co  1999
doraju      이하니        hani@naver.co  1989
sqlite>
```

SQL 명령어

■ 데이터 값 수정하기

UPDATE 테이블 이름 SET 열이름=수정값 WHERE 열이름=값;

```
sqlite> select * from userTable;
id          userName      email          birthYear
-----
m001        이소연        lee@naver.com  2008
sh890       마동탁        mado@naver.co  1999
guio0       구하라        guif@naver.co  1999
doraju      이하니        hani@naver.co  1989
sqlite> update userTable
...> set email='mado009@naver.co'
...> where userName='마동탁';
sqlite> select * from userTable;
id          userName      email          birthYear
-----
m001        이소연        lee@naver.com  2008
sh890       마동탁        mado009@naver  1999
guio0       구하라        guif@naver.co  1999
doraju      이하니        hani@naver.co  1989
sqlite>
```


SQL 명령어

■ 테이블 삭제하기

DROP TABLE 테이블이름;

```
sqlite> create table userAddr(id char(4), addr text, phone char(12));  
sqlite> .table  
userAddr    userTable
```

```
sqlite> drop table userAddr;  
sqlite> .table  
userTable
```

SQL 명령어

■ 퀴즈

SQLite에 다시 접속해서 naverDB에 다음 테이블(productTable)을 구축해 보자.

| 제품코드(pCode) | 제품명(pName) | 가격(price) | 재고수량(amount) |
|-------------|------------|-----------|--------------|
| p0001 | 노트북 | 110 | 5 |
| p0002 | 마우스 | 3 | 22 |
| p0003 | 키보드 | 2 | 11 |

힌트 제품코드와 제품명은 char형으로 지정하고, 가격과 재고수량은 int형으로 지정한다.

출력 결과

```
pCode  pName  price  amount
-----
p0001  노트북  110    5
p0002  마우스  3       22
p0003  키보드   2       11
```



파이썬 SQLite



■ 파이썬에서 데이터 입력하는 코딩 순서



파이썬 SQLite

- Squlit3 모듈 임포트 및 버전 확인

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3

# SQLITE3 모듈 버전
print(sqlite3.version)

# SQLITE3 버전
print(sqlite3.sqlite_version)
```

파이썬 SQLite

■ 데이터베이스 연결하기

```
import sqlite3
conn = sqlite3.connect('test.db')
cursor = conn.cursor() #커서 생성

cursor.execute('select * from album ')

print('전체 데이터 출력하기')
# 조회한 데이터 불러오기
result_list = cursor.fetchall()
# 데이터 출력하기
for i in result_list :
    print(i)
conn.close()
```

```
연결변수 = Sqlite3.connect('데이터베이스경로')
Cursor = 연결변수.cursor()
Cursor.execute('SQL 명령어')
리스트이름 = cursor.fetchall()
연결변수.close()
```

파이썬 SQLite

■ 테이블 조회하기

```
import sqlite3
conn = sqlite3.connect('test.db')
cursor = conn.cursor() #커서 생성
cursor.execute('select Title from albums limit 5')

result_list = cursor.fetchall()
print(result_list)
```

SQL 명령 :
SELECT 컬럼명또는* FROM 테이블

파이썬 SQLite

■ 데이터베이스 생성 및 테이블 생성하기

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3
```

```
# 데이터베이스 커넥션 생성
```

```
conn = sqlite3.connect('my_books.db') #DB 없다면 새로 생성
```

```
cursor = conn.cursor() #커서 생성
```

```
# 테이블의 제목, 출판일자, 출판사, 페이지수, 추천여부
```

```
# 테이블이 존재하지 않으면 생성
```

```
cursor.execute(""" create table if not exists books(
                    title text,
                    published_date text,
                    publisher text,
                    pages integer,
                    recommended integer
                ); """)
```

```
conn.commit() # DB 반영
```

SQL 명령 :

CREATE TABLE 테이블명 (컬럼명 데이터타입...)

파이썬 SQLite

- 데이터베이스 생성 및 테이블 생성하기

SQL 명령 :

CREATE TABLE 테이블명 (컬럼명 데이터타입...)

```
cursor.execute(' select * from books;')  
book_list = cursor.fetchall();  
print(book_list)  
conn.close() # 커넥션 닫기. DB 연결 종료
```


파이썬 SQLite

■ 테이블에 레코드 삽입하기 1

SQL 명령 :
INSERT INTO 테이블이름
VALUES(값1, 값2,...);

```
import sqlite3
# 데이터베이스 커넥션 생성
conn = sqlite3.connect('my_books.db') #DB 없다면 새로 생성
cursor = conn.cursor() #커서 생성
# 데이터 입력
cursor.execute("insert into books values('JAVA','2018-02-28','길벗',500,10);")
conn.commit()
# 데이터 조회 후 리스트로 변환
cursor.execute(' select * from books;')
book_list = cursor.fetchall();
print(book_list)
conn.close() # 커넥션 닫기. DB 연결 종료
```

파이썬 SQLite

■ 테이블에 레코드 삽입하기 2

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3
# 데이터베이스 커넥션 생성
conn = sqlite3.connect('my_books.db')
cursor = conn.cursor() #커서 생성
# 데이터 입력 SQL ? 파라미터 이용
sql = 'insert into books values (?, ?, ?, ?, ?)'
# 튜플을 이용한 데이터 입력
cursor.execute(sql, ('파이썬', '2018-03-04', '한빛', 584, 20))
conn.commit()
# 데이터 조회 후 리스트로 변환
cursor.execute(' select * from books;')
book_list = cursor.fetchall();
print(book_list)
conn.close() # 커넥션 닫기. DB 연결 종료
```

```
sql변수 = 'INSERT INTO 테이블명 VALUES(?,?,?)'
cursor.execute(sql변수,(값1,값2,값3))
```

파이썬 SQLite

■ 테이블에 레코드 삽입하기 3

```
sql변수 = 'INSERT INTO 테이블명 VALUES(?,?,?)'  
데이터리스트 = [(값1,값2,값3.),(값1,값2,값3.)... ]  
cursor.execute(sql변수,데이터리스트)
```

```
# sqlite3 파이썬 라이브러리 로딩  
import sqlite3  
# 데이터베이스 커넥션 생성  
conn = sqlite3.connect('my_books.db')  
cursor = conn.cursor() #커서 생성  
# 데이터 입력 SQL ? 파라미터 이용  
sql = 'insert into books values (?,?,?,?,?)'
```

파이썬 SQLite

■ 테이블에 레코드 삽입하기 3

책의 정보를 담고 있는 튜플들의 리스트

```
items = [  
    ('빅데이터','2014-12-01','삼성',296,11),  
    ('안드로이드','2010-10-01','영진',526,20),  
    ('스프링','2013-11-01','에이콘',248,15)  
]
```

여러 데이터 입력

```
cursor.executemany(sql,items)  
conn.commit()
```

데이터 조회 후 리스트로 변환

```
cursor.execute(' select * from books;')  
book_list = cursor.fetchall();  
print(book_list)  
conn.close() # 커넥션 닫기. DB 연결 종료
```

```
sql변수 = 'INSERT INTO 테이블명 VALUES(?,?,?)'  
cursor.execute(sql변수,(값1,값2,값3))
```

파이썬 SQLite

■ 테이블에 레코드 수정하기 1

UPDATE 테이블명 SET 컬럼명=새값 WHERE 컬럼명='값'

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3
# 데이터베이스 커넥션 생성
conn = sqlite3.connect('my_books.db')
cursor = conn.cursor() #커서 생성
# 데이터 수정
cursor.execute(" update books set recommended=100 where title='JAVA' ")
conn.commit()
# 데이터 조회 후 리스트로 변환
cursor.execute(' select * from books;')
book_list = cursor.fetchall();
print(book_list)
conn.close() # 커넥션 닫기. DB 연결 종료
```

파이썬 SQLite

■ 테이블에 레코드 수정하기 2

UPDATE 테이블명 SET 컬럼명=새값 WHERE 컬럼명='값'

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3
# 데이터베이스 커넥션 생성
conn = sqlite3.connect('my_books.db')
cursor = conn.cursor() #커서 생성
# 데이터 수정
sql = " update books set recommended=? where title=? "
cursor.execute(sql,(200,'스프링'))
conn.commit()
# 데이터 조회 후 리스트로 변환
cursor.execute(' select * from books;')
book_list = cursor.fetchall();
print(book_list)
conn.close() # 커넥션 닫기. DB 연결 종료
```

```
sql변수 = "UPDATE 테이블명 SET 컬럼명=?
WHERE 컬럼명=? "
cursor.execute(sql변수,(값1,값2))
```

파이썬 SQLite

■ 테이블에 레코드 삭제하기1

SQL 명령 :

delete from 테이블명 where 컬럼명=값;

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3
# 데이터베이스 커넥션 생성
conn = sqlite3.connect('my_books.db')
cursor = conn.cursor() #커서 생성
# 데이터 삭제
cursor.execute(" delete from books where publisher='영진' ")
conn.commit()
# 데이터 조회 후 리스트로 변환
cursor.execute(' select * from books;')
book_list = cursor.fetchall();
print(book_list)
conn.close() # 커넥션 닫기. DB 연결 종료
```

파이썬 SQLite

■ 테이블에 레코드 삭제하기2

```
# sqlite3 파이썬 라이브러리 로딩
import sqlite3
# 데이터베이스 커넥션 생성
conn = sqlite3.connect('my_books.db')
cursor = conn.cursor() #커서 생성
# 데이터 삭제
sql = " delete from books where publisher=? "
# 레코드 삭제시는 리스트 요소로 삽입
cursor.execute(sql, ['에이콘'])
conn.commit()
# 데이터 조회 후 리스트로 변환
cursor.execute(' select * from books;')
book_list = cursor.fetchall();
print(book_list)
conn.close() # 커넥션 닫기. DB 연결 종료
```

```
sql변수 = 'DELETE FROM 테이블명 WHERE 컬럼명=? '
cursor.execute(sql변수,[값]))
```


파이썬 SQLite

■ 퀴즈

| 과목번호 | 과목명 | 교수명 | 교수번호 | 강의실코드 | 강의실 선행 |
|---------|---------|-----|------|-------|---------|
| DBO101 | 오라클기초 | 서영학 | 21 | A301 | 공대301호 |
| DBO101 | 오라클기초 | 서영학 | 21 | A301 | 공대301호 |
| DBO101 | 오라클기초 | 서영학 | 21 | A301 | 공대301호 |
| DBO101 | 오라클기초 | 서영학 | 21 | A301 | 공대301호 |
| DBO101 | 오라클기초 | 김이교 | 22 | A302 | 공대302호 |
| DBM101 | MSSQL기초 | 서영학 | 21 | B301 | 인문대301호 |
| DBM101 | MSSQL기초 | 서영학 | 21 | B301 | 인문대301호 |
| JAVA102 | 자바중급 | 이민우 | 23 | A301 | 공대301호 |