

# Ch 12. 데이터 저장과 관리(SQLite)

모바일게임프로그래밍  
김지심교수

# 학습목표



데이터베이스의 기본 개념을 이해한다.  
SQLite를 이용해 앱을 개발할 수 있다.  
SQLite GUI 툴을 사용할 수 있다.

01

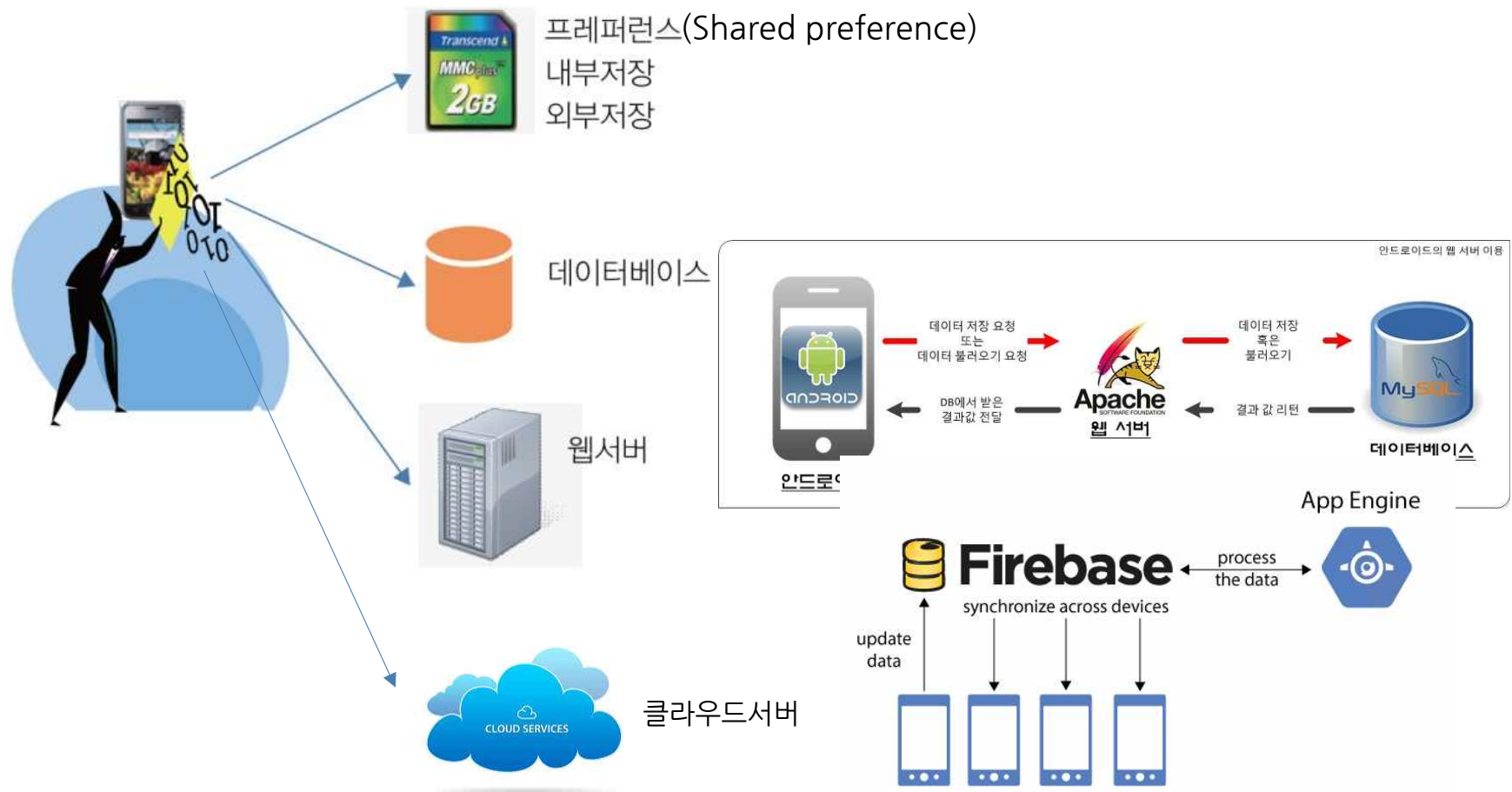


## SQLite 기본



# 1. SQLite 기본

## 안드로이드 데이터 저장 방법



# 1. SQLite 기본

## 데이터베이스(Database) 정의

대용량의 데이터 집합을 체계적으로 구성해놓은 것

## 데이터베이스 관리 시스템(DBMS)

데이터베이스는 여러 사용자나 시스템이 서로 공유할 수 있도록 데이터베이스를 관리해주는 시스템 또는 소프트웨어

크게 계층형(Hierarchical), 망형(Network), 관계형(Relational), 객체지향형(Object-Oriented), 객체관계형(Object-Relational) DBMS 등의 유형으로 나뉨

# 1. SQLite 기본

## 관계형 데이터베이스(RDBMS)

계층형, 망형, 관계형, 객체지향형, 객체관계형 DBMS 등의 유형  
DBMS 중 가장 많이 사용되는 것은 관계형 DBMS  
SQLite도 관계형 DBMS에 속함

## 관계형 데이터베이스의 장·단점

### 장점

업무가 변화할 경우에 다른 DBMS에 비해 변화에 쉽게 순응할 수 있는 구조  
유지보수 측면에서도 편리  
대용량 데이터 관리와 데이터 무결성(Integration)을 잘 보장

### 단점

시스템 자원을 많이 차지해서 시스템이 전반적으로 느려짐

# 1. SQLite 기본

## 데이터베이스 관련 용어

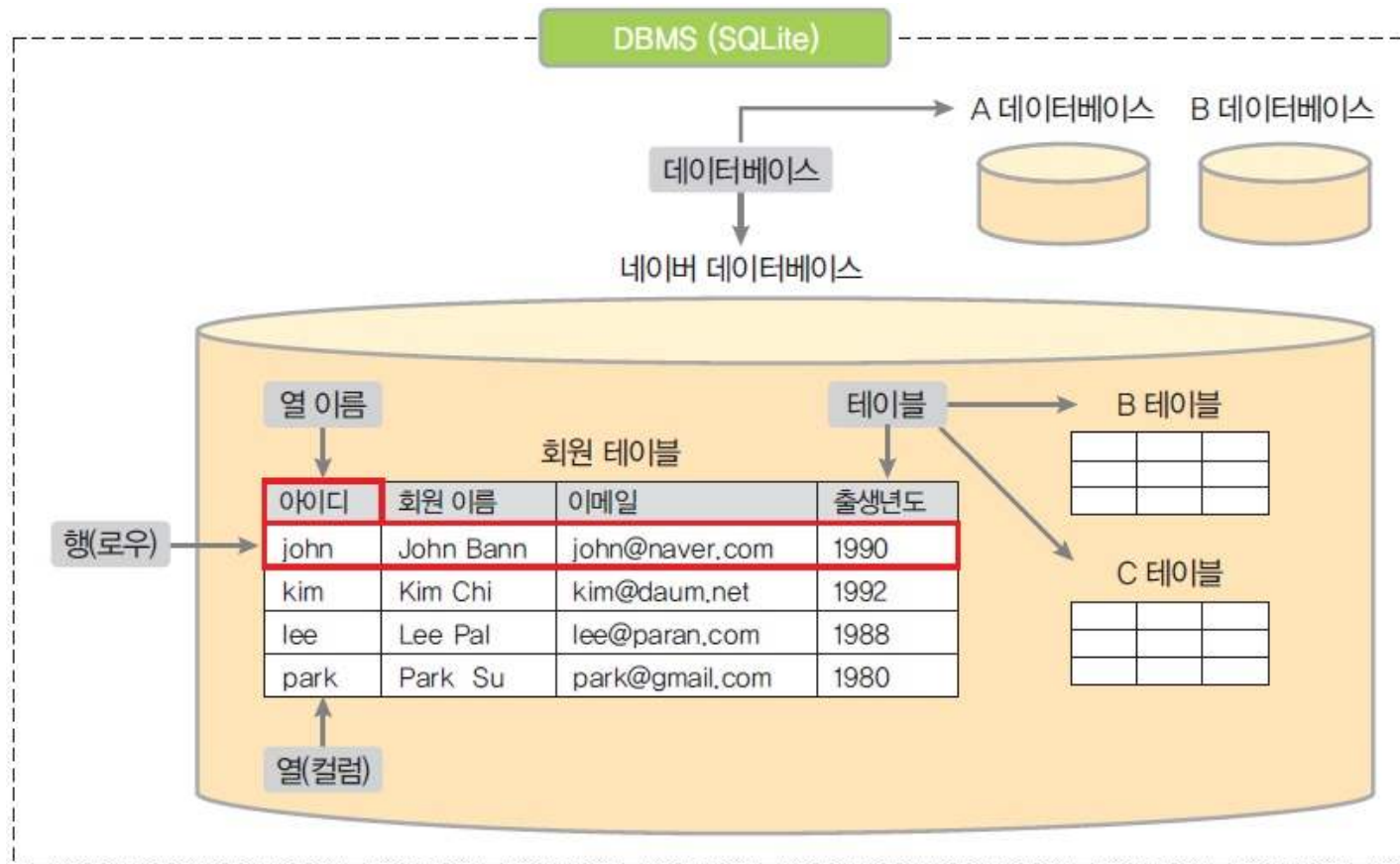


그림 12-1 DBMS 구성도



# 1. SQLite 기본

## 데이터베이스 관련 용어

데이터 : 하나하나의 단편적인 정보

테이블(table) : 회원 데이터가 표 형태로 표현된 것

데이터베이스(DB) : 테이블이 저장되는 장소로 주로 원통 모양으로 표현  
각 데이터베이스는 서로 다른 고유한 이름이 있어야 함

DBMS : 데이터베이스를 관리하는 시스템 또는 소프트웨어를 말함

### 안드로이드에 포함된 SQLite 소프트웨어가 이에 해당

열(field, column) : 각 테이블은 1개 이상의 열로 구성됨

열 이름 : 각 열을 구분하는 이름, 열 이름은 각 테이블 안에서는 중복되지 않아야 함

데이터 형식 : 열의 데이터 형식을 뜻함

테이블을 생성할 때 열 이름과 함께 지정해줘야 함

행(row) : 실제 데이터를 뜻함

SQL : 사용자와 DBMS가 소통하기 위한 언어



# 1. SQLite 기본

## SQL문

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정해야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. 이미 테이블이 만들어져 있는 경우가 많기 때문에 CREATE TABLE은 통상적으로 DML보다 적게 사용된다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	테이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지 지정한다.
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코드를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.

# 1. SQLite 기본

## SQLite

초경량급 DB로 C언어로 작성된 SQL 데이터베이스 엔진 기반의 DB  
데이터를 단말-side에 저장하며 플랫폼-독립적(로컬 DB)

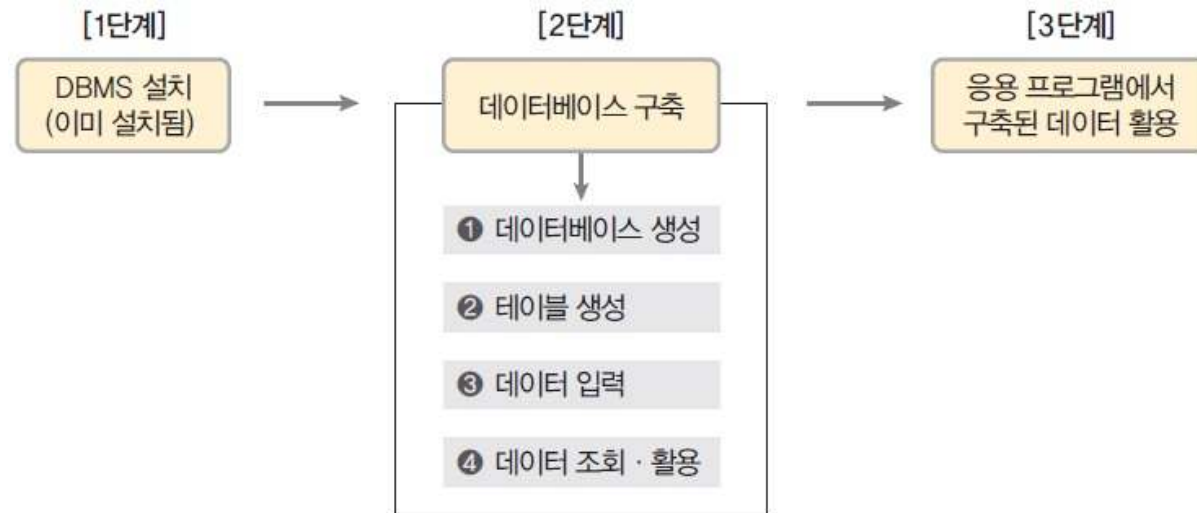
\*<http://www.sqlite.org>

### DB서버가 있는데 왜 로컬DB가 필요한가?

유선 네트워크보다 불안정한 네트워크의 상태에 대응하기 위해,  
서버 과부하를 줄이기 위해,  
속도 등을 고려해, ...

# 1. SQLite 기본

## SQLite에서 데이터베이스 구축



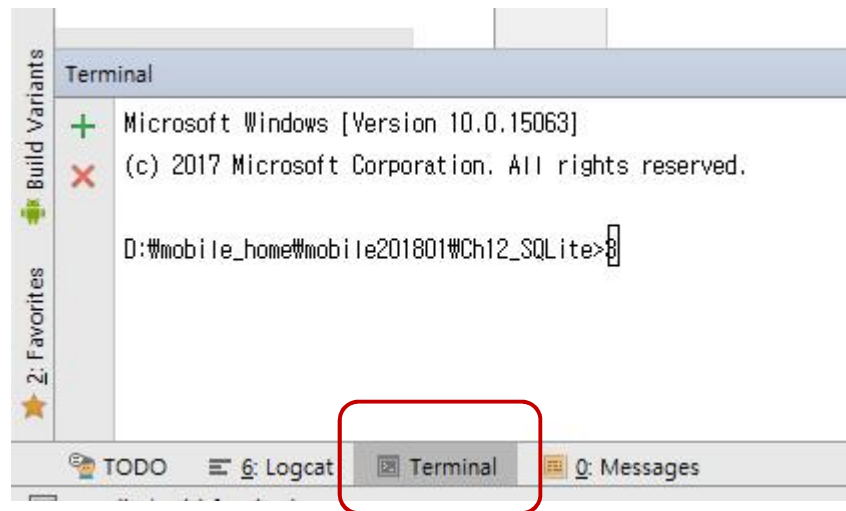
# 1. SQLite 기본

## 예제 12-1: Project12-1

ADB로 내장된 SQLite에 접속하여 DB를 생성  
(안드로이드 코드는 작성하지 않고 ADB에서 DB 브라우징)

- \* 교재와 같이 명령 프롬프트로 SQLite를 실행해도 되나, 본 실습에서는 안스의 터미널에서 실행
- \* SQLite로 DB를 브라우징하기 위해서는 AVD를 가동해야 함

안스에서 프로젝트를 생성하고 AVD를 실행한 후, 터미널 실행

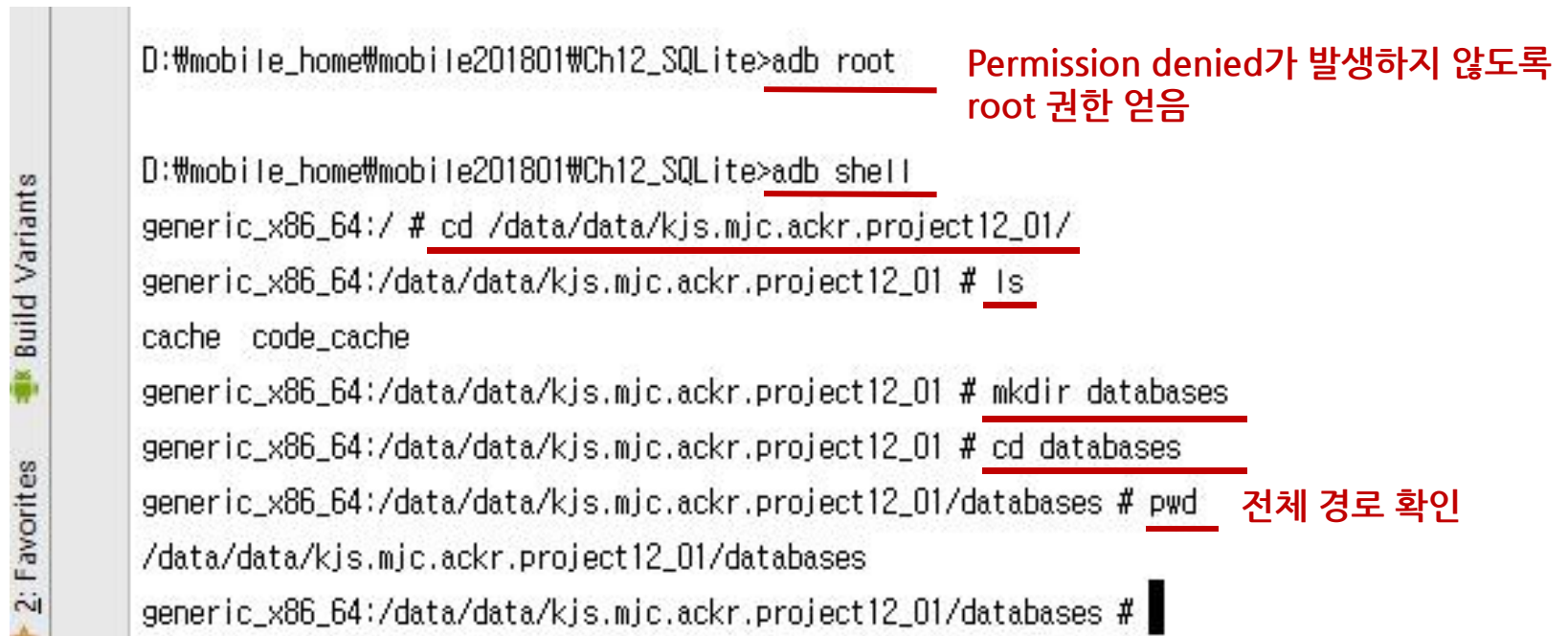


# 1. SQLite 기본

## 예제 12-1: Project12-1

다음 명령을 차례로 수행하여 DB 디렉토리 생성

DB 디렉토리 생성 경로: /data/data/본인 실습 패키지명



```
D:\mobile_home\mobile201801\Ch12_SQLite>adb root
D:\mobile_home\mobile201801\Ch12_SQLite>adb shell
generic_x86_64:/ # cd /data/data/kjs.mjc.ackr.project12_01/
generic_x86_64:/data/data/kjs.mjc.ackr.project12_01 # ls
cache  code_cache
generic_x86_64:/data/data/kjs.mjc.ackr.project12_01 # mkdir databases
generic_x86_64:/data/data/kjs.mjc.ackr.project12_01 # cd databases
generic_x86_64:/data/data/kjs.mjc.ackr.project12_01/databases # pwd
/data/data/kjs.mjc.ackr.project12_01/databases
generic_x86_64:/data/data/kjs.mjc.ackr.project12_01/databases #
```

Permission denied가 발생하지 않도록  
root 권한 얻음

전체 경로 확인

# 1. SQLite 기본

## 예제 12-1: Project12-1 데이터베이스 생성

SQLite 실행 및 데이터베이스 생성

```
1  sqlite3  naverDB
```

```
generic_x86_64:/data/data/kjs.mjc.ackr.project12_01/databases # sqlite3 naverDB
SQLite version 3.18.2 2017-07-21 07:56:09
Enter ".help" for usage hints.
sqlite>
```

# 1. SQLite 기본

## 예제 12-1: Project12-1 회원 테이블을 생성

```
CREATE TABLE 테이블이름 (열이름1 데이터형식, 열이름2 데이터형식 ...);
```

### 테이블 생성 및 확인

```
1 CREATE TABLE userTable (id char(4), userName char(15),  
2     email char(15), birthYear int);  
3 .table  
4 .schema userTable
```

```
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> 
```



# 1. SQLite 기본

저자  
한마디

## SQL문 규칙

우선 SQL문은 대·소문자를 구분하지 않는다. 또한 모든 SQL문의 끝에는 세미콜론(;)을 붙여야 한다. 예외적으로 SQLite 자체 명령은 소문자로 써야 하고 시작에 마침표(.)를 붙이며 끝에 세미콜론을 붙이지 않아도 된다. 자주 사용하는 SQLite 명령은 다음과 같다.

- .table : 현재 데이터베이스의 테이블 목록을 보여준다.
- .schema 테이블 이름 : 테이블의 열, 데이터 형식 등의 정보를 보여준다.
- .header on : SELECT문으로 출력할 때 헤더를 보여준다.
- .mode column : SELECT문으로 출력할 때 칼럼 모드로 출력해준다.
- .exit : SQLite를 종료한다.

일반적으로 SELECT문을 사용하기 전에 .header on과 .mode column을 설정하면 결과 화면이 보기 좋게 출력된다.

# 1. SQLite 기본

## 예제 12-1: Project12-1

회원 테이블에 4개의 행을 입력하는 SQL문

```
INSERT INTO 테이블이름 VALUES(값1, 값2, ...);
```

### 데이터 4건 입력

```
1 INSERT INTO userTable VALUES('john' , 'John Bann' , 'john@naver.com' , 1990);  
2 INSERT INTO userTable VALUES('kim' , 'Kim Chi' , 'kim@daum.net' , 1992);  
3 INSERT INTO userTable VALUES('lee' , 'Lee Pal' , 'lee@paran.com' , 1988);  
4 INSERT INTO userTable VALUES('park' , 'Park Su' , 'park@gmail.com' , 1980);
```

```
sqlite> INSERT INTO userTable VALUES('KIM', 'KIMJS', 'jisimkim@mjc.ac.kr', 2018);  
sqlite> INSERT INTO userTable VALUES('강', '강안드', 'kang@mjc.ac.kr', 1998);  
sqlite> INSERT INTO userTable VALUES('이', '이명지', 'lee@mjc.ac.kr', 1997);  
sqlite>
```

# 1. SQLite 기본

## 예제 12-1: Project12-1 데이터 조회 · 활용

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

### 데이터 조회 예

```
1 .header on
2 .mode column
3 SELECT * FROM userTable;
4 SELECT id, birthYear FROM userTable WHERE birthYear <= 1990;
5 SELECT * FROM userTable WHERE id = 'park';
```



```
sqlite> .head on
sqlite> .mode column
sqlite> select * from userTable;
id      userName  email      birthYear
-----
KIM      KIMJS     jisimkim@mj.ac.kr 2018
강       강안드     kang@mj.ac.kr    1998
이       이명지     lee@mj.ac.kr     1997

sqlite> SELECT id,birthYear FROM userTable WHERE birthYear <=1997;
id      birthYear
-----
이       1997

sqlite> SELECT userName, email FROM userTable WHERE id='이';
userName  email
-----
이명지     lee@mj.ac.kr

sqlite> 
```

02



## SQLite 활용



## SQLite 동작 방식

## SQLiteOpenHelper 클래스, SQLiteDatabase 클래스, Cursor 인터페이스 활용



### 그림 12-9 SQLite 관련 클래스 동작

## 2. SQLite 활용

### Reference

CLASS	DEFINITION
SQLiteDatabase	Sqlite DB의 핵심 클래스로서 이 클래스의 메소드를 이용하여 데이터의 CRUD 질의문을 수행
Cursor	쿼리 조건을 만족하는 레코드들의 결과집합에 대한 객체 Cursor 객체를 이용해 레코드를 선택하고 선택된 레코드의 필드 데이터를 얻는 구조로 핸들링
SQLiteOpenHelper	DB를 감싸고 있는 Helper 클래스로서, DB 관리만을 목적으로 하는 추상 클래스 SQLiteDatabase를 이용해 CRUD 작업을 하고, 관리 작업은(테이블 생성이나 스키마 변경 등) 이 클래스로 처리 → INSERT, SELECT 등을 수행할 때 테이블 생성 여부를 판단하지 않아도 되는 장점



## 2. SQLite 활용

### Reference

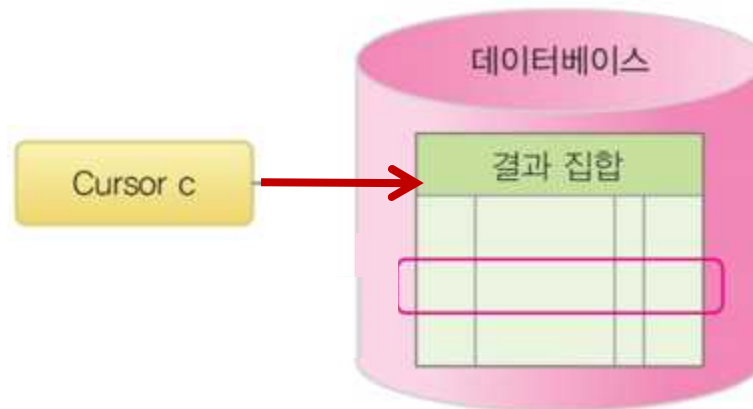
표 12-1 SQLite 관련 클래스 및 인터페이스와 메소드

클래스 또는 인터페이스	메소드	주 용도
SQLiteOpenHelper 클래스	생성자	DB 생성
	onCreate()	테이블 생성
	onUpgrade()	테이블 삭제 후 다시 생성
	getReadableDatabase()	읽기 전용 DB 열기, SQLiteDatabase 반환
	getWritableDatabase()	읽고 쓰기용 DB 열기, SQLiteDatabase 반환
SQLiteDatabase 클래스	execSQL()	SQL문(Insert/Update/Delete) 실행
	close()	DB 닫기
	query(), rawQuery()	Select 실행 후 커서 반환
Cursor 인터페이스	moveToFirst()	커서의 첫 행으로 이동
	moveToLast()	커서의 마지막 행으로 이동
	moveToNext()	현재 커서의 다음 행으로 이동



## 2. SQLite 활용

### 결과 집합(RESULT SETS) 및 커서(CURSOR)



현재 가리키는 레코드에서 컬럼의 값을 추출할 때는 `getString()`이나 `getInt()`와 같은 메소드를 호출하면 된다. 다음 메소드들 중에서 컬럼 타입에 맞게 사용하면 된다.

- `getDouble(int columnIndex)`
- `getFloat(int columnIndex)`
- `getInt(int columnIndex)`
- `long getLong(int columnIndex)`
- `short getShort(int columnIndex)`
- `String getString(int columnIndex)`

여기서 매개변수인 `columnIndex`는 컬럼의 번호를 나타낸다.

## 2. SQLite 활용

### 사용형식 예시

#### 테이블 생성

```
myDBHelper.getWritableDatabase();  
//SQLiteOpenHelper의 메소드를 이용해 DB를 R/W모드로 열어 SQLiteDatabase 객체로 리턴  
myDBHelper.onUpgrade(SQLiteDatabase, old version, new version);  
//onUpgrade()를 호출하여 Create 혹은 DROP 등 수행  
sqlDB.close();
```

#### 데이터 입력

```
myDBHelper.getWritableDatabase();  
sqlDB.execSQL(INSERT문);  
sqlDB.close();
```

#### 데이터 조회

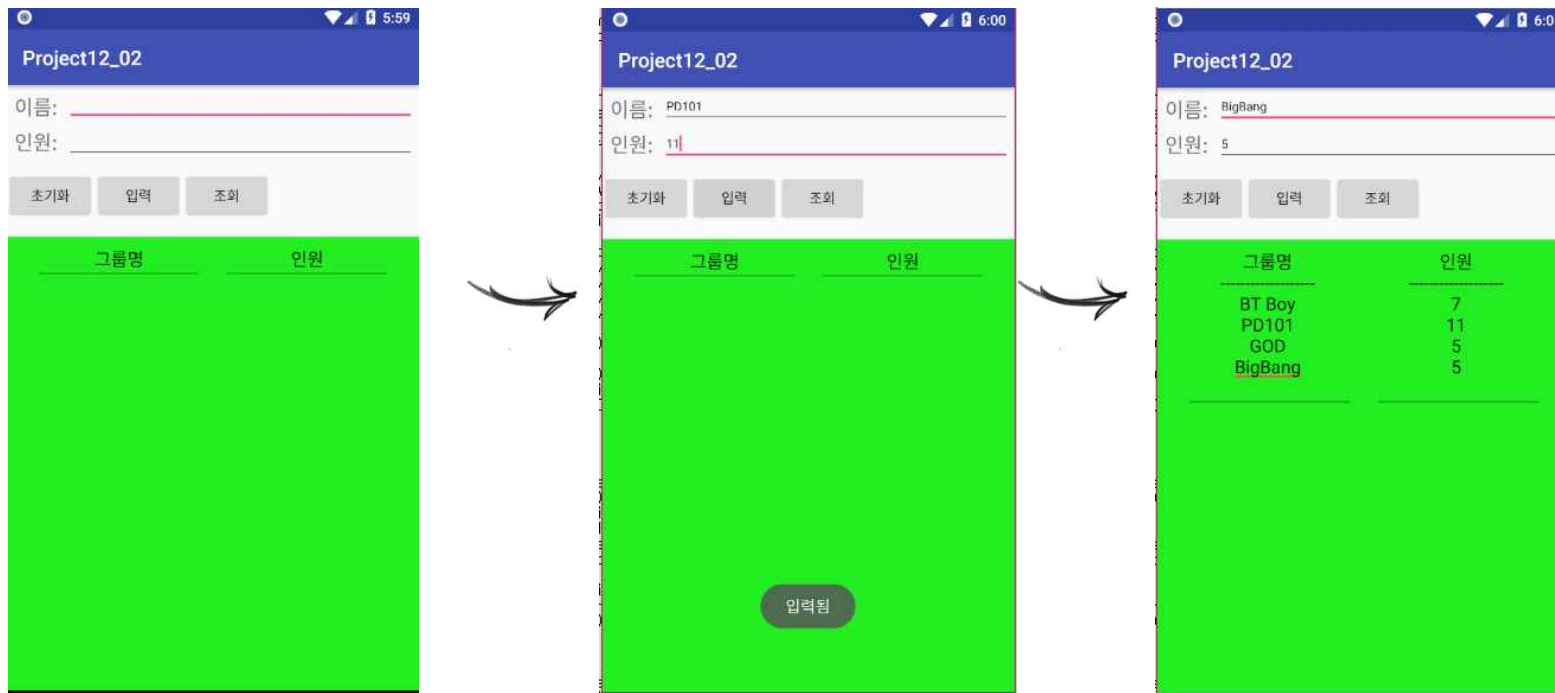
```
myDBHelper.getReadableDatabase();  
Cursor cursor = sqlDB.rawQuery(SELECT문);  
cursor.moveToNext(); //반복하여 데이터 operating  
sqlDB.close();
```

## 2. SQLite 활용

### 실습 12-2: Project12\_02

#### 가수그룹 DB 실습

\* 추후 DB를 로컬PC에 pull하여 결과 확인할 경우 Permission Denied 에러 발생한다면,  
API23(Marshmallow) 에뮬레이터에서 실행



## 2. SQLite 활용

### 실습 12-2: Project12\_02

#### activity\_main.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tvName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:text="이름: "
        android:textSize="20sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/tvNumber"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="8dp"
        android:text="인원: "
        android:textSize="20sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvName" />
    <EditText
        android:id="@+id/etName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="24"
        android:textSize="12sp"
        app:layout_constraintLeft_toRightOf="@id/tvName"
        app:layout_constraintRight_toRightOf="parent" />
    <EditText
        android:id="@+id/etNumber"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="24"
        android:textSize="12sp"
        app:layout_constraintLeft_toRightOf="@id/tvNumber"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/etName" />
```

```
<Button
    android:id="@+id/btnInit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="초기화"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@id/tvNumber" />
```

```
<Button
    android:id="@+id/btnInsert"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="입력"
    app:layout_constraintLeft_toRightOf="@id/btnInit"
    app:layout_constraintTop_toBottomOf="@id/tvNumber" />
```

```
<Button
    android:id="@+id/btnSelect"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="조회"
    app:layout_constraintLeft_toRightOf="@id/btnInsert"
    app:layout_constraintTop_toBottomOf="@id/tvNumber" />
```

## 2. SQLite 활용

### 실습 12-2: Project12\_02 activity\_main.xml

```
<android.support.constraint.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_percent="0.25" />

<LinearLayout
    android:layout_width="420dp"
    android:layout_height="480dp"
    android:background="#2e2"
    android:gravity="center|top"
    app:layout_constraintBaseline_toBaselineOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="@id/guideline">

    <EditText
        android:id="@+id/etNameResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="20dp"
        android:ems="8"
        android:gravity="center"
        android:text="그룹명" />

    <EditText
        android:id="@+id/etNumberResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="8"
        android:gravity="center"
        android:text="인원" />

</LinearLayout>

</android.support.constraint.ConstraintLayout>
```

## 2. SQLite 활용

### 실습 12-2: Project12\_02

#### MyDBHelper.java

SQLiteOpenHelper의 서브 클래스를 생성  
onCreate(), onUpgrade() 메소드를 오버라이딩

```
12 public class MyDBHelper extends SQLiteOpenHelper {
13
14     public MyDBHelper(Context context) {
15         super(context, "groupDB", null, 1);
16         DB를 생성하는 액티비티 전달    커서를 지정하는 파라미터(null: 표준커서)
17
18     @Override
19     public void onCreate(SQLiteDatabase sqLiteDatabase) {
20         sqLiteDatabase.execSQL("CREATE TABLE groupTBL (gName CHAR(20) PRIMARY KEY, gNumber INTEGER);");
21     }
22
23     @Override
24     public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
25         sqLiteDatabase.execSQL("DROP TABLE IF EXISTS groupTBL");
26         onCreate(sqLiteDatabase);
27     }
28 }
```

## 2. SQLite 활용

### 실습 12-2: Project12\_02

#### MainActivity.java

myDBHelper 클래스, SQLiteDatabase 클래스 변수 등 선언  
[입력], [조회] 클릭 이벤트 처리

```
ct12_01 public class MainActivity extends AppCompatActivity {  
12  
13     MyDBHelper myHelper;  
14     SQLiteDatabase sqlDB;  
15     EditText etName, etNumber, etNameResult, etNumberResult;  
16  
17     @Override  
18     protected void onCreate(Bundle savedInstanceState) {  
19         super.onCreate(savedInstanceState);  
20         setContentView(R.layout.activity_main);  
21  
22         etName = findViewById(R.id.etName);  
23         etNumber = findViewById(R.id.etNumber);  
24         etNameResult = findViewById(R.id.etNameResult);  
25         etNumberResult = findViewById(R.id.etNumberResult);  
26  
27         myHelper = new MyDBHelper(this);  
28  
29         (findViewById(R.id.btnInit)).setOnClickListener((v) -> {  
32             sqlDB = myHelper.getWritableDatabase();  
33             myHelper.onUpgrade(sqlDB, 1, 2);  
34             sqlDB.close();  
35         });  
36  
37     }  
}
```



## 2. SQLite 활용

### 실습 12-2: Project12\_02

#### MainActivity.java

[입력], [조회] 클릭 이벤트 처리

```
(findViewById(R.id.btnInsert)).setOnClickListener((view) -> {
    sqldb = myHelper.getWritableDatabase();
    sqldb.execSQL("INSERT INTO groupTBL VALUES ( '" + etName.getText().toString() + "' , " + etNumber.getText().toString() + " );");
    sqldb.close();
    Toast.makeText(MainActivity.this, "입력됨",
        Toast.LENGTH_SHORT).show();
});

(findViewById(R.id.btnSelect)).setOnClickListener((view) -> {
    sqldb = myHelper.getReadableDatabase();
    Cursor cursor;
    cursor = sqldb.rawQuery("SELECT * FROM groupTBL;", null);

    String strNames = "그룹명" + "\r\n" + "-----" + "\r\n";
    String strNumbers = "인원" + "\r\n" + "-----" + "\r\n";

    while (cursor.moveToNext()) {
        strNames += cursor.getString(0) + "\r\n";
        strNumbers += cursor.getString(1) + "\r\n";
    }

    etNameResult.setText(strNames);
    etNumberResult.setText(strNumbers);

    cursor.close();
    sqldb.close();
});
}
```

## 2. SQLite 활용

### 실습 12-2: Project12\_02

- 1) 프로젝트를 실행한 후, 생성된 DB를 로컬에 복사하여 DB Browser for SQLite 프로그램으로 확인

터미널을 실행하여 adb shell에서 groupDB를 확인하고 adb를 종료한 후 PC로 pulling

```
D:\mobile_home\mobile201801\Ch12_SQLite>adb root

D:\mobile_home\mobile201801\Ch12_SQLite>adb shell
generic_x86_64:/ # cd /data/data/kjs.mjc.project12_02/databases
generic_x86_64:/data/data/kjs.mjc.project12_02/databases # ls
groupDB groupDB-journal groupdb
generic_x86_64:/data/data/kjs.mjc.project12_02/databases # exit

D:\mobile_home\mobile201801\Ch12_SQLite>adb root

D:\mobile_home\mobile201801\Ch12_SQLite>adb pull /data/data/kjs.mjc.project12_02/databases/groupDB D:\work_home\01_inbox_home
\data/data/kjs.mjc.project12_02/databases/groupDB: 1 file pulled. 1.3 MB/s (20480 bytes in 0.015s)
```

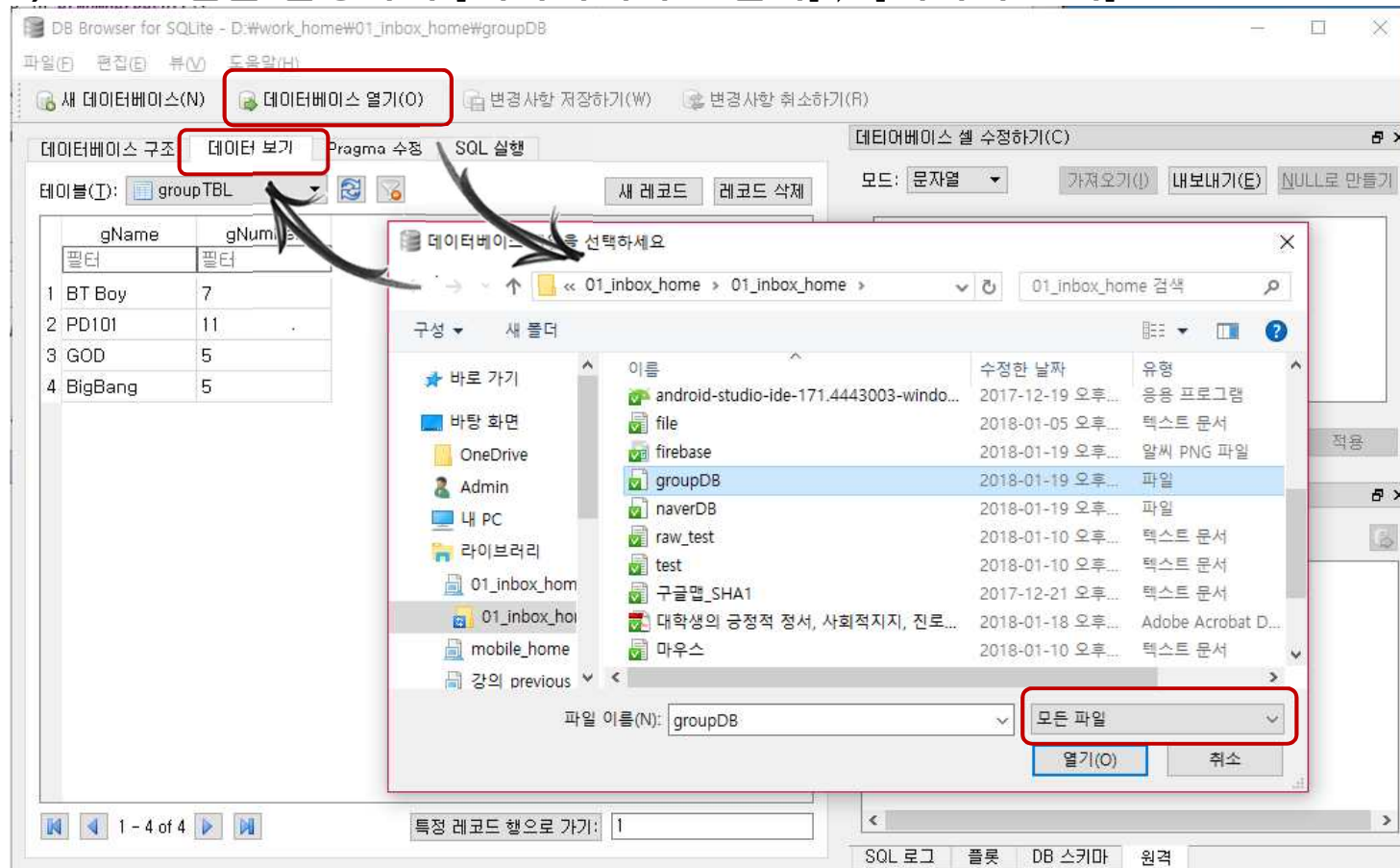
## 2. SQLite 활용

### 실습 12-2: Project12\_02

2) DB Browser for SQLite 프로그램을 다운받아 PC에 설치

\* <http://sqlitebrowser.org/>

3) 프로그램을 실행하여 [데이터베이스 열기] > [데이터 보기]



## 2. SQLite 활용

실습 12-2: Project12\_02

기 입력된 '이름'을 중복 입력하면 에러가 발생하니 해결해보자!

## 2. SQLite 활용

### 실습(직접 풀어보기) 12-2: Test12\_02

#### ▶ 직접 풀어보기 12-2

[실습 12-2]에 <수정>과 <삭제>를 추가하자.

- 이름에 그룹 이름과 변경된 인원을 입력한 후 <수정>을 클릭하면 해당 그룹의 인원이 변경되도록 한다.
- 이름에 그룹 이름을 입력하고 <삭제>를 클릭하면 해당 그룹의 행이 삭제되도록 한다.
- <입력>, <수정>, <삭제>를 클릭하면 그 결과가 즉시 화면에 보이도록 한다.

#### HINT

- 수정 SQL : `UPDATE groupTBL SET gNumber = 변경된_인원 WHERE gName = "변경할_그룹 이름";`
- 삭제 SQL : `DELETE FROM groupTBL WHERE gName = "삭제할_그룹 이름";`
- 입력/수정/삭제 후 즉시 보이게 하려면 `btnSelect.callOnClick()`을 호출한다.

- 이름 조회: 이름을 입력한 후 검색하면 해당 레코드가 조회됨 (해당 레코드가 없을 경우 Toast 출력, `getCount()` 사용)

Test12\_02

이름: 101

인원:

초기화   입력   수정   삭제   조회   이름 조회

그룹명	인원
101	11

## 2. SQLite 활용

### 실습: MyList2EntryDB

SimpleCursorAdapter 클래스를 이용하여 어댑터뷰에 DB 출력

activity\_main.xml

직접 만들자!



## 2. SQLite 활용

### 실습: MyList2EntryDB MainActivity.java

```
9
10 public class MainActivity extends AppCompatActivity {
11     MyDBHelper helper;
12     SQLiteDatabase db;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18
19         ListView list = findViewById(R.id.listView1);
20
21         helper = new MyDBHelper(this);
22         db = helper.getWritableDatabase();
23         Cursor cursor =
24
25         String[] columns = {"name", "tel"};
26         int[] viewForm = new int[]{android.R.id.text1, android.R.id.text2};
27
28         SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, android.R.layout.simple_list_item_2,
29         list.setAdapter(adapter);
30     }
31 }
```

완성하기!

완성해보기!



## 2. SQLite 활용

실습: MyList2EntryDB  
MydbHelper.java

```
11 public class MyDBHelper extends SQLiteOpenHelper {
12
13     @Override
14     public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
15         sqLiteDatabase.      완성하기! : 테이블 삭제 SQL문
16         onCreate(sqLiteDatabase);
17     }
18
19     public MyDBHelper(Context context) {
20         super(context, "mycontacts.db", null, 2);
21     }
22
23     @Override
24     public void onCreate(SQLiteDatabase sqLiteDatabase) {
25         sqLiteDatabase.      완성: 테이블 생성문(자동으로 부여되고 증가되는 id 필드 포함)
26
27         for (int i = 0; i < 20; i++) {
28             sqLiteDatabase.      완성하기! : INSERT SQL문
29         }
30     }
31 }
32 }
```

## 2. SQLite 활용

※ 이미지는 BLOB(Binary Large OBject)으로 저장  
바이트 배열 형식으로 변환하여 이진데이터로 저장  
조회한 후 Bitmap으로 변환하여 사용

```
0x0000 FFD8 FFE0 0010 4A46 4946 0001 0100 0001 0001 y0yà..JFIF.....
0x0012 0000 FFD8 0043 0001 0101 0101 0101 0101 ..yÜ.C.....
0x0024 0101 0101 0101 0101 0101 0101 0101 0101 .....
0x0036 0101 0101 0101 0101 0101 0101 0101 0101 .....
0x0048 0101 0101 0101 0101 0101 0101 0101 01FF .....
0x005A D800 4301 0101 0101 0101 0101 0101 0101 Ü.C.....
0x006C 0101 0101 0101 0101 0101 0101 0101 0101 .....
0x007E 0101 0101 0101 0101 0101 0101 0101 0101 .....
0x0090 0101 0101 0101 0101 0101 0101 0101 FFC0 0011 .....yÄ..
0x00A2 0000 9000 9003 0122 0002 1101 0311 01FF C400 .."......yÄ.
0x00B4 1F00 0001 0501 0101 0101 0100 0000 0000 .....
0x00C6 0001 0203 0405 0607 0809 0A0B FFC4 00B5 1000 .....yÄ.µ..
0x00D8 0201 0303 0204 0305 0504 0400 0001 7D01 0203 .....}...
0x00EA 0004 1105 1221 3141 0613 5161 0722 7114 3281 .....!1A..Qa."q.2
0x00FC 91A1 0823 42B1 C115 52D1 F024 3362 7282 090A ..,.#B±Ä.RÑð$3br ..
0x010E 1617 1819 1A25 2627 2829 2A34 3536 3738 393A .....%&'()*456789:
0x0120 4344 4546 4748 494A 5354 5556 5758 595A 6364 CDEFGHIJSTUVWXYZcd
0x0132 6566 6768 696A 7374 7576 7778 797A 8384 8586 efghijstuvwxyz
0x0144 8788 898A 9293 9495 9697 9899 9AA2 A3A4 A5A6 {~{|
0x0156 A7A8 A9AA B2B3 B4B5 B6B7 B8B9 BAC2 C3C4 C5C6 5~@*??µ§.i*ÄÄÄÄ
0x0168 C7C8 C9CA D2D3 D4D5 D6D7 D8D9 DAE1 E2E3 E4E5 ÇÈÉÊËÖÖÖÖ×@ÜÜäääää
0x017A E6E7 E8E9 EAF1 F2F3 F4F5 F6F7 F8F9 FAFF C400 ðçèéñòóóóó+@ÜÜyÄ.
0x018C 1F01 0003 0101 0101 0101 0101 0100 0000 .....
0x019E 0001 0203 0405 0607 0809 0A0B FFC4 00B5 1100 .....yÄ.µ..
0x01B0 0201 0204 0403 0407 0504 0400 0102 7700 0102 .....w...
0x01C2 0311 0405 2131 0612 4151 0761 7113 2232 B108 .....!1..AQ.øq."2.
0x01D4 1442 91A1 B1C1 0923 3352 F015 6272 D10A 1624 .B ¡±Ä.#3Rð.brÑ..$
0x01E6 34E1 25F1 1718 191A 2627 2829 2A35 3637 3839 4&ñ....&'()*56789
0x01F8 3A43 4445 4647 4849 4A53 5455 5657 5859 5A63 :CDEFGHIJSTUVWXYZc
```

SQLite Browser for SQLite

파일(F) 편집(E) 뷰(V) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장하기(W)

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

테이블(T): mail 새 레코드

	image	userName	title
필터	필터	필터	필터
1	BLOB	Kabin	hello
2	BLOB	hama	hahahahaha
3	BLOB	Dong	nice to meet ...
4	BLOB	Kabin	hello
5	BLOB	hama	hahahahaha
6	BLOB	Dong	nice to meet ...
7	BLOB	Kabin	hello
8	BLOB	hama	hahahahaha
9	BLOB	Dong	nice to meet ...

※ raw DB push하기

03



## SQLite GUI 툴 사용



### 3. SQLite GUI 툴 사용

#### DB Browser for SQLite

SQLite에 접근할 때 SQLite Database Browser라는 GUI 툴을 편리하게 사용 가능



그림 12-12 DB Browser for SQLite 화면

\* 다운로드: <https://github.com/sqlitebrowser/sqlitebrowser/releases>

### 3. SQLite GUI 툴 사용

#### 데이터베이스 및 테이블 생성

[파일]-[새 데이터베이스]를 선택하여 [저장하려는 파일명을 고르세요] 창에서 데이터베이스 파일이 저장될 경로와 파일명을 지정해주고 <저장>을 클릭

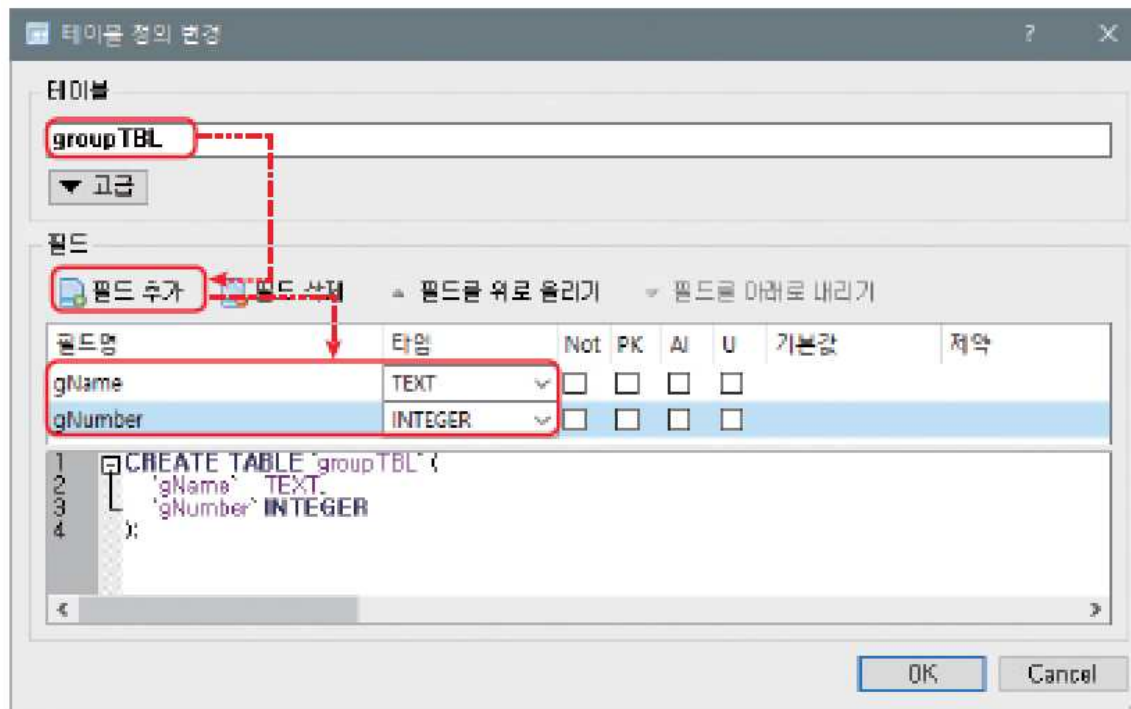


그림 12-13 DB Browser for SQLite에서의 데이터베이스 및 테이블 생성



### 3. SQLite GUI 툴 사용

#### 데이터 입력

데이터를 입력하려면 [데이터 보기] 탭 > [새 레코드]

데이터 입력 후 메뉴의 [파일]-[변경사항 저장하기]를 선택해서 변경 사항 저장  
생성한 데이터베이스 파일을 DDMS를 통해 AVD에 넣어서(Push) 사용

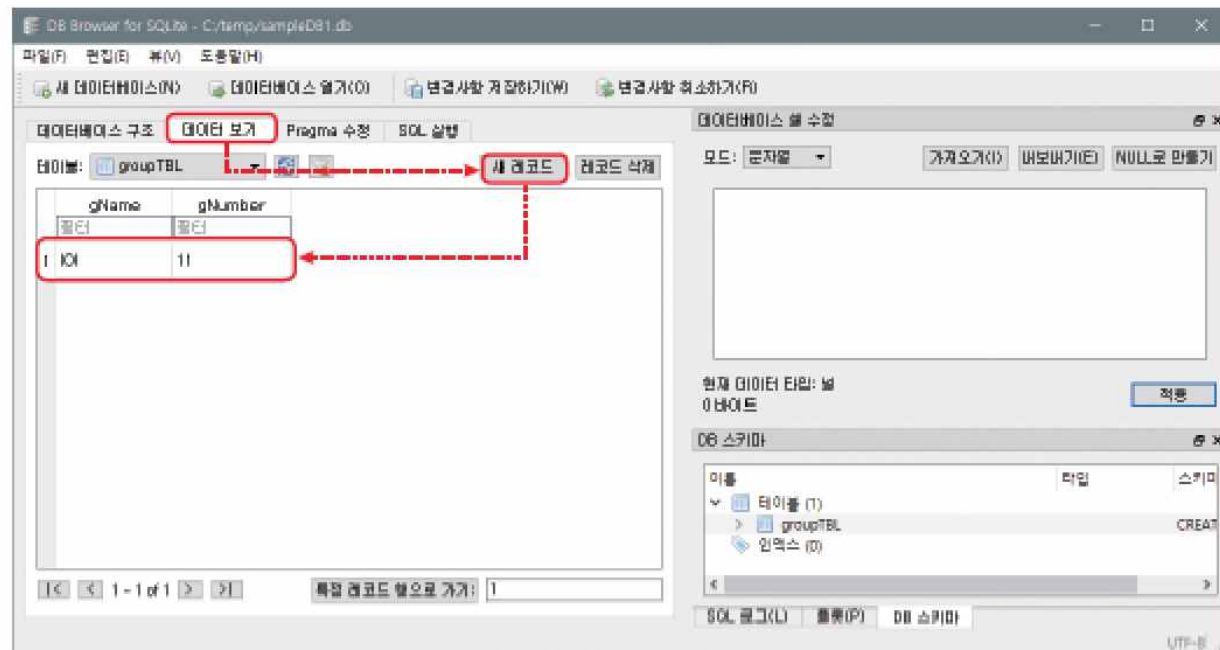
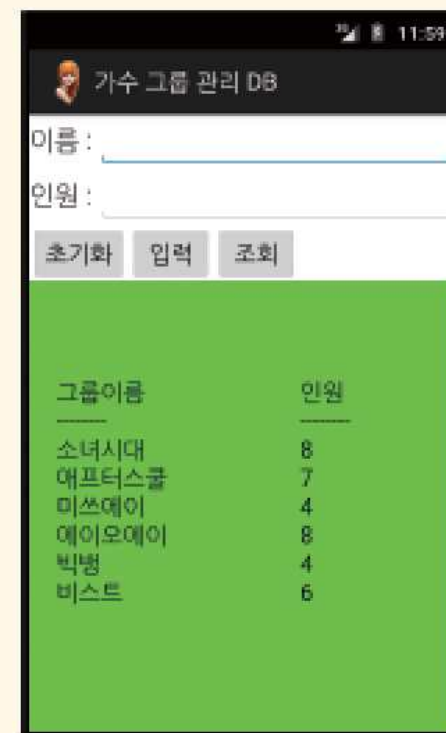
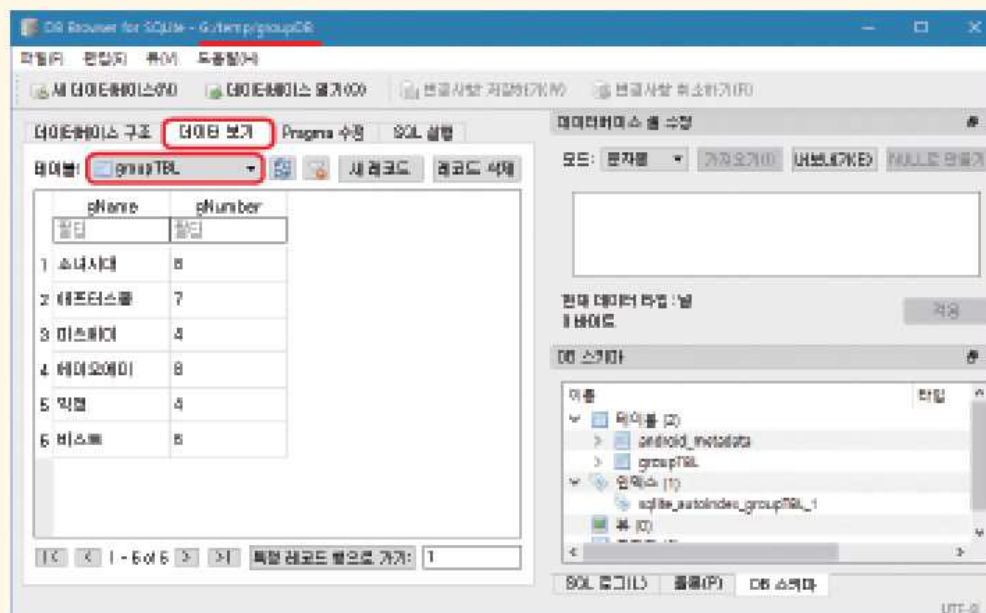


그림 12-14 DB Browser for SQLite에서의 행 데이터 입력

### 3. SQLite GUI 툴 사용

#### ▶ 작업 풀어보기 12-2

[실습 12-2]의 groupDB 데이터베이스 파일을 DDMS를 이용해서 AVD에서 PC로 가져온(Pull) 후 DB Browser for SQLite에서 가수 그룹 이름을 모두 한글로 고치고 가수 그룹을 몇 개 더 입력해보자. 그리고 AVD에 다시 넣은(Push) 후 [실습 12-2]를 실행해서 데이터가 잘 조회되는지 확인한다.



### 3. SQLite GUI 툴 사용

#### SQLite Developer

그래픽 화면에서 데이터베이스를 관리하기 위한 유사 툴  
[Database]-[Register Database]로 편집할 데이터베이스 파일을 선택 후 데이터 추가

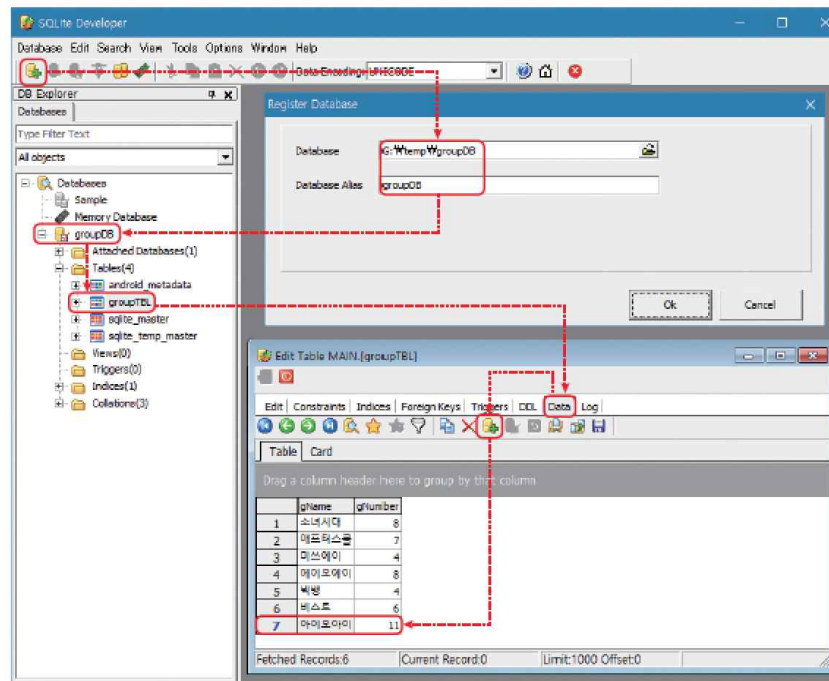


그림 12-15 SQLite Developer에서 행 데이터 처리

\* 다운로드: <http://www.sqlitedeveloper.com/download>