

12 데이터 저장과 관리



학습목표

- ❖ 데이터베이스의 기본 개념을 이해한다.
- ❖ SQLite 사용법을 익힌다.
- ❖ SQLite를 이용하여 앱을 개발한다.
- ❖ SQLite GUI 툴 사용법을 익힌다.

차례

1. SQLite의 기본
2. SQLite의 활용

01 데이터베이스의 기본 개념

■ 데이터베이스 정의

- 대용량의 데이터 집합을 체계적으로 구성해놓은 것

■ 데이터베이스 관리 시스템

- 데이터베이스는 여러 사용자나 시스템이 서로 공유할 수 있어야 함
- 데이터베이스 관리 시스템(DataBase Management System, DBMS) : 이러한 데이터베이스를 관리해주는 시스템 또는 소프트웨어
- DBMS의 유형
 - 계층형(hierarchical), 망형(network), 관계형(relational), 객체지향형(object-oriented), 객체관계형(object-relational)

01 데이터베이스의 기본 개념

■ 관계형 데이터베이스

- 계층형, 망형, 관계형, 객체지향형, 객체관계형 DBMS 등의 유형 중 실질적으로 가장 많이 사용됨
- SQLite도 관계형 DBMS 속함

■ 관계형 데이터베이스의 장단점

- 장점
 - 업무가 변화할 경우에 다른 DBMS에 비해 변화에 쉽게 순응할 수 있는 구조
 - 유지 및 보수 측면에서도 편리
 - 대용량 데이터 관리와 데이터 무결성(Integration)을 잘 보장
- 단점
 - 시스템 자원을 많이 차지해서 시스템이 전반적으로 느려짐

01 데이터베이스의 기본 개념

■ 데이터베이스 관련 용어

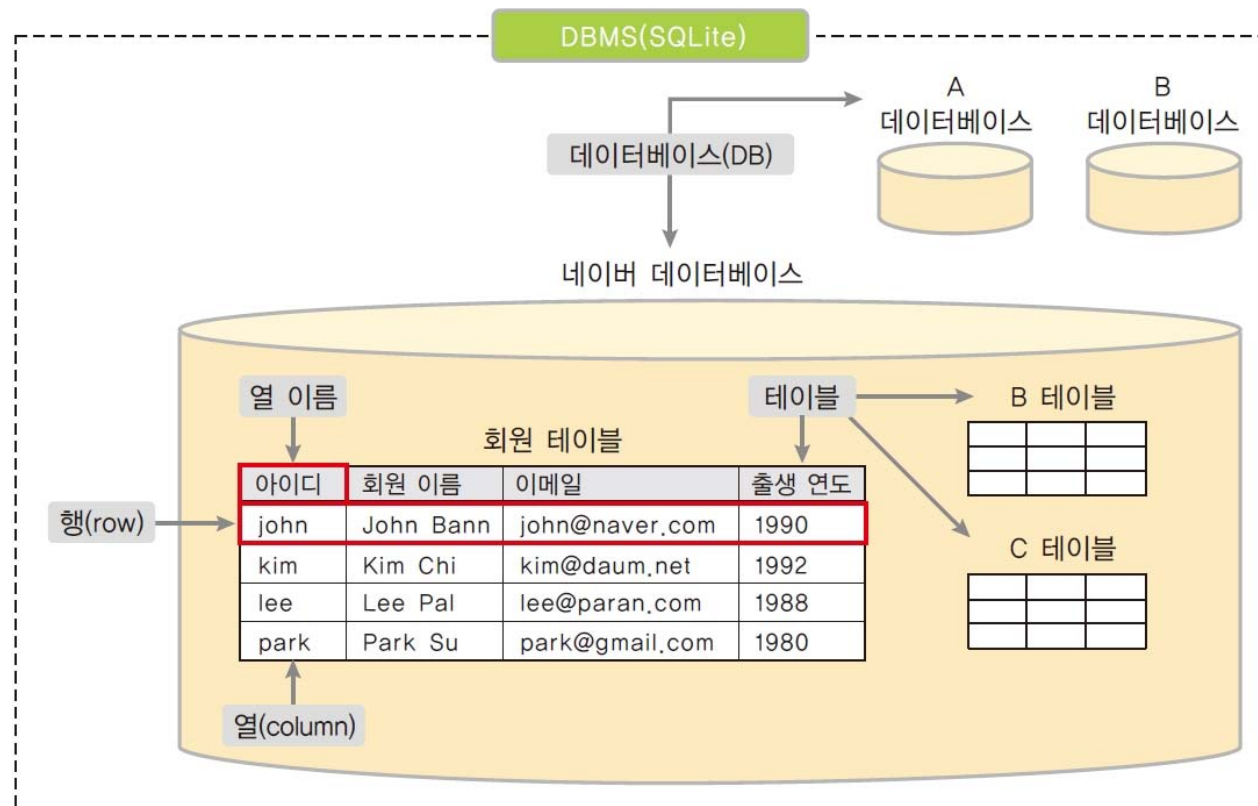


그림 12-1 DBMS의 구성

01 데이터베이스의 기본 개념

- **데이터** : 하나하나의 단편적인 정보를 뜻함
- **테이블** : 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(칼럼 또는 필드)** : 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 지정해야 함
- **행(로우)** : 실제 데이터
- **SQL** : 사용자와 DBMS가 소통하기 위한 언어

02 SQLite에서의 데이터베이스 구축

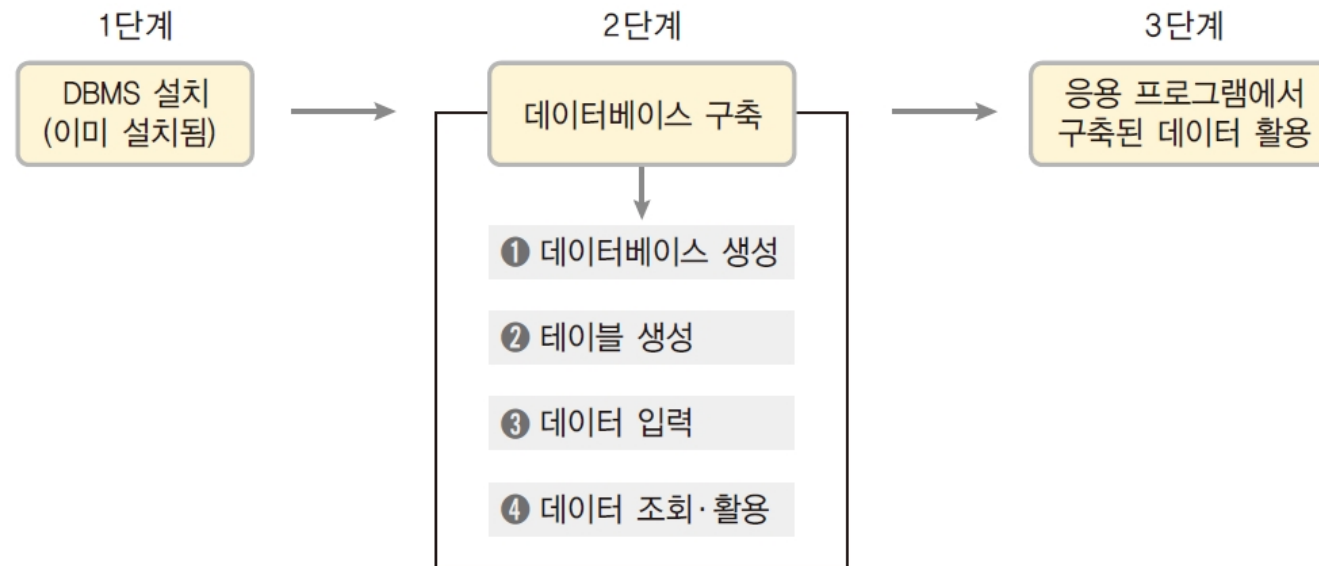


그림 12-2 데이터베이스 구축 및 운영 절차

02 SQLite에서의 데이터베이스 구축

- <실습 12-1> 네이버 데이터베이스 구축하기
 - 네이버 데이터베이스를 구축하면서 기본적인 SQL 문 익히기
- 데이터베이스 생성 전 작업
 - (1) 새 프로젝트 만들기
 - 프로젝트 이름 : 'Project12_1'
 - 패키지 이름 : 'com.cookandroid.project12_1'
 - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름
 - (2) 프로젝트를 실행하여 AVD 가동
 - (3) 명령 프롬프트를 실행하여 adb.exe가 있는 폴더로 이동

```

C:\> CD C:\CookAndroid\SDK\platform-tools#
C:\CookAndroid\SDK\platform-tools> DIR adb.exe
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 6E65-9828

C:\CookAndroid\SDK\platform-tools 디렉터리
2020-09-30 오전 12:18      3,233,280 adb.exe
                   1개 파일      3,233,280 바이트
                   0개 디렉터리 46,961,250,304 바이트 남음

C:\CookAndroid\SDK\platform-tools>
  
```

그림 12-3 adb.exe 파일 확인

02 SQLite에서의 데이터베이스 구축

- (4) 명령 프롬프트에서 다음 명령을 차례로 수행하면 SQLite에 접속할 준비가 된 것

셀을 이용한 데이터베이스 접속

```
1 adb root
2 adb shell
3 cd /data/data/com.cookandroid.project12_1
4 ls -l
5 mkdir databases
6 cd databases
7 pwd
```

The screenshot shows a terminal window titled '명령 프롬프트 - adb shell'. The commands and their outputs are as follows:

```
C:\CookAndroid\SDK\platform-tools> adb root
C:\CookAndroid\SDK\platform-tools> adb shell
generic_x86:/ #
generic_x86:/ # cd /data/data/com.cookandroid.project12_1
generic_x86:/data/data/com.cookandroid.project12_1 #
generic_x86:/data/data/com.cookandroid.project12_1 # ls -l
total 8
drwxrws--x 2 u0_a106 u0_a106_cache 4096 2020-10-08 09:05 cache
drwxrws--x 2 u0_a106 u0_a106_cache 4096 2020-10-08 09:05 code_cache
generic_x86:/data/data/com.cookandroid.project12_1 #
generic_x86:/data/data/com.cookandroid.project12_1 # mkdir databases
generic_x86:/data/data/com.cookandroid.project12_1 #
generic_x86:/data/data/com.cookandroid.project12_1 # cd databases
generic_x86:/data/data/com.cookandroid.project12_1/databases #
generic_x86:/data/data/com.cookandroid.project12_1/databases # pwd
/data/data/com.cookandroid.project12_1/databases
generic_x86:/data/data/com.cookandroid.project12_1/databases #
```

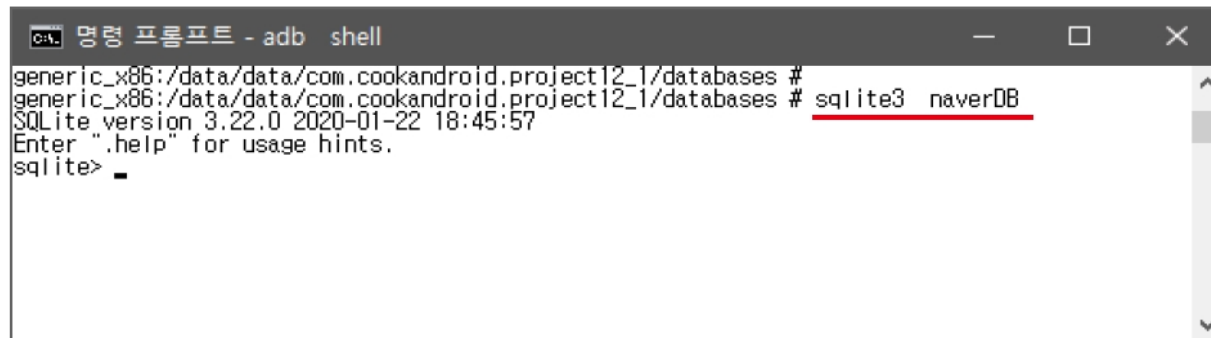
그림 12-4 데이터베이스용 폴더 생성

02 SQLite에서의 데이터베이스 구축

- 데이터베이스 생성
 - (5) sqlite3 명령을 실행하면서 데이터베이스 이름 지정
 - naverDB의 내부는 아직 비어 있음

SQLite 실행 및 데이터베이스 생성

```
1  sqlite3  naverDB
```



```
명령 프롬프트 - adb  shell
generic_x86:/data/data/com.cookandroid.project12_1/databases #
generic_x86:/data/data/com.cookandroid.project12_1/databases # sqlite3  naverDB
SQLite version 3.22.0 2020-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> _
```

그림 12-5 데이터베이스 생성

02 SQLite에서의 데이터베이스 구축

- 테이블 생성
 - (6) 테이블을 생성하는 SQL 문의 원형

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

테이블 생성 및 확인

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

The screenshot shows a terminal window titled '명령 프롬프트 - adb shell'. Inside, the following commands and outputs are visible:

```
sqlite>
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>
```

The commands and their corresponding outputs are underlined in the original image.

그림 12-6 테이블 생성 및 확인

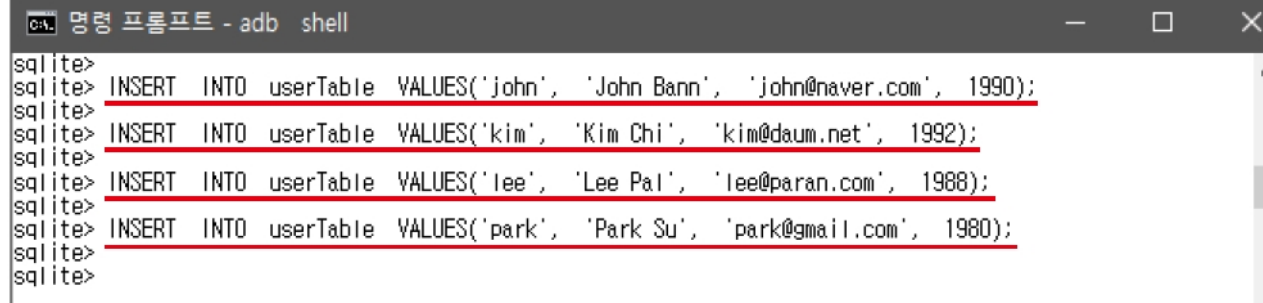
02 SQLite에서의 데이터베이스 구축

- 데이터 입력
 - (7) 생성한 회원 테이블(userTable)에 행 데이터를 입력

```
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);
```



The screenshot shows a terminal window titled '명령 프롬프트 - adb shell'. Inside, the SQLite command line interface is active. Four INSERT statements are entered and executed, each on a new line. The statements are: 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); Each statement is followed by a prompt 'sqlite>'.

```
C:\> 명령 프롬프트 - adb shell
sqlite>
sqlite> INSERT INTO userTable VALUES('john', 'John Bann', 'john@naver.com', 1990);
sqlite>
sqlite> INSERT INTO userTable VALUES('kim', 'Kim Chi', 'kim@daum.net', 1992);
sqlite>
sqlite> INSERT INTO userTable VALUES('lee', 'Lee Pal', 'lee@paran.com', 1988);
sqlite>
sqlite> INSERT INTO userTable VALUES('park', 'Park Su', 'park@gmail.com', 1980);
sqlite>
sqlite>
```

그림 12-7 데이터 4건 입력

02 SQLite에서의 데이터베이스 구축

- 데이터 조회·활용
 - (8) 데이터를 조회·활용하는 SQL 문은 SELECT로 주로 WHERE 절과 함께 사용함

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

데이터 조회의 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';
```

02 SQLite에서의 데이터베이스 구축

```

C:\ 명령 프롬프트 - adb shell
sqlite>
sqlite> .header on
sqlite>
sqlite> .mode column
sqlite>
sqlite> SELECT * FROM userTable;
id      userName      email      birthYear
-----
john    John Bann     john@naver.com 1990
kim     Kim Chi      kim@daum.net   1992
lee     Lee Pal      lee@paran.com  1988
park    Park Su      park@gmail.com  1980
sqlite>
sqlite> SELECT id, birthYear FROM userTable WHERE birthYear <= 1990;
id      birthYear
-----
john    1990
lee     1988
park    1980
sqlite>
sqlite> SELECT * FROM userTable WHERE id = 'park';
id      userName      email      birthYear
-----
park    Park Su      park@gmail.com 1980
sqlite>
sqlite>

```

그림 12-8 데이터 조회의 예

- (9) 모든 실습을 마쳤으면 .exit와 exit로 SQLite와 셸을 종료함

02 SQLite에서의 데이터베이스 구축

▶ 작업 풀어보기 12-1

myDB를 생성한 후 그 내부에 다음과 같은 제품 테이블(product)을 생성하라.

제품 이름	가격	제조 일자	제조 회사	남은 수량
TV	100	2017.7.22	Samsung	55
Computer	150	2019.5.5	LG	7
Monitor	50	2021.9.9	Daewoo	33

01 SQLite 프로그래밍

- 안드로이드 앱 개발을 위한 SQLite 동작 방식
 - SQLiteOpenHelper 클래스, SQLiteDatabase 클래스, Cursor 인터페이스 활용

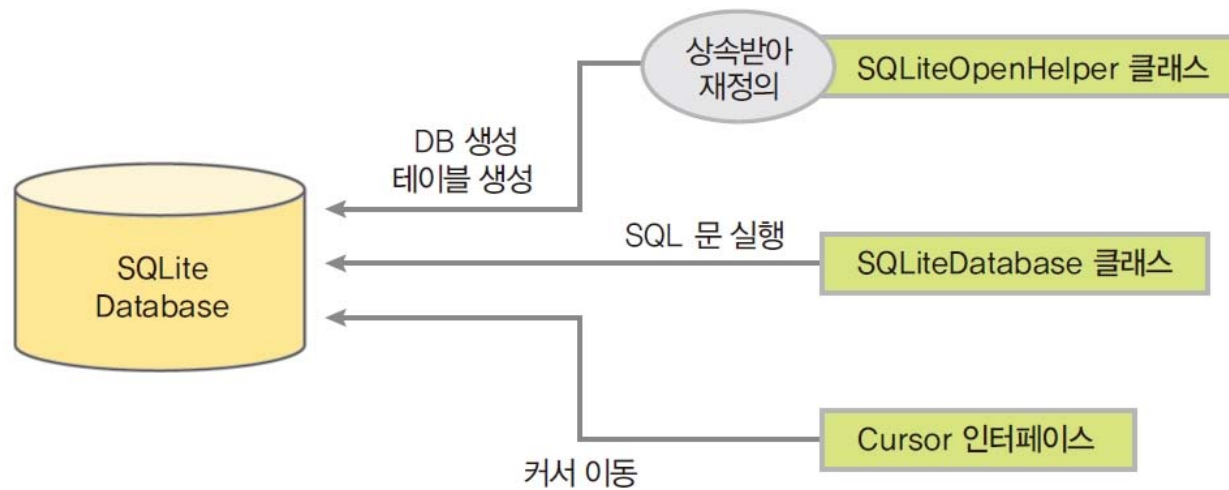


그림 12-9 SQLite 관련 클래스의 동작

01 SQLite 프로그래밍

■ 각 클래스에서 주로 사용되는 메소드

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 쓰이용 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

01 SQLite 프로그래밍

■ <실습 12-2> 가수 그룹 관리 DB 만들기

- 가수 그룹의 이름과 인원을 데이터베이스에 입력하고 조회하는 응용 프로그램 작성

■ 안드로이드 프로젝트 생성

- (1) 새 프로젝트 만들기
 - 프로젝트 이름 : 'Project12_2'
 - 패키지 이름 : 'com.cookandroid.project12_2'
 - 그 외 규칙은 [실습 2-4]의 (1)~(4)를 따름



그림 12-10 가수 그룹 관리 DB 결과 화면

01 SQLite 프로그래밍

- 화면 디자인 및 편집
 - (2) activity_main.xml의 바깥 리니어레이아웃 안에 4개의 리니어레이아웃(수평)을 만들고 화면을 다음과 같이 구성하기
 - 리니어레이아웃1~3은 layout_weight을 1로 리니어레이아웃4는 8로 설정
 - 리니어레이아웃1 : 텍스트뷰 1개, 에디트텍스트(edtName) 1개
 - 리니어레이아웃2 : 텍스트뷰 1개, 에디트텍스트(edtNumber) 1개
 - 리니어레이아웃3 : 버튼 3개(btnInit, btnInsert, btnSelect)
 - 리니어레이아웃4 : 에디트텍스트 2개 (edtNameResult, edtNumberResult)

01 SQLite 프로그래밍

예제 12-1 activity_main.xml

```

1  <LinearLayout>
2      <LinearLayout
3          android:layout_width="fill_parent"
4          android:layout_height="@dip"
5          android:layout_weight="1"
6          android:orientation="horizontal" >
7      <TextView
8          android:text="이름 : " />
9      <EditText
10         android:id="@+id/edtName"
11         android:layout_weight="1" />
12  </LinearLayout>
13  <LinearLayout>
14      ~~~~ 중간 생략(텍스트뷰, 에디트텍스트) ~~~~
15  </LinearLayout>
16  <LinearLayout>
17      <Button
18          android:id="@+id/btnInit"
19          android:text="초기화" />
20      ~~~~ 중간 생략 (버튼 2개) ~~~~
21  </LinearLayout>
22  <LinearLayout>
23      <EditText
24          android:id="@+id/edtNameResult"
25          android:layout_weight="1" />
26      ~~~~ 중간 생략 (에디트텍스트) ~~~~
27  </LinearLayout>
28 </LinearLayout>

```

01 SQLite 프로그래밍

- Kotlin 코드 작성 및 수정
 - (3) SQLiteOpenHelper 클래스에서 상속받은 클래스를 정의한 후 생성자 수정

예제 12-2 Kotlin 코드 1

```
1  override fun onCreate(savedInstanceState: Bundle?) {  
2      super.onCreate(savedInstanceState)  
3      setContentView(R.layout.activity_main)  
4  }  
5  
6  inner class myDBHelper(context: Context) : SQLiteOpenHelper(context,  
7      "groupDB", null, 1) {  
8  
9      override fun onCreate(p0: SQLiteDatabase?) {  
10         TODO("not implemented")  
11     }  
12  
13     override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {  
14         TODO("not implemented")  
15     }  
16 }
```

01 SQLite 프로그래밍

- (4) [예제 12-2] myDBHelper 클래스의 onCreate()와 onUpgrade() 메소드 코딩
 - onCreate() 메소드 : 테이블을 생성하는 기능을 코딩
 - onUpgrade() 메소드 : 테이블을 삭제한 후 다시 생성

예제 12-3 Kotlin 코드 2

```
1  override fun onCreate(p0: SQLiteDatabase?) {  
2      p0!!.execSQL("CREATE TABLE groupTBL(gName CHAR(20) PRIMARY KEY,  
3          gNumber INTEGER);")  
4  }  
5  
6  override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {  
7      p0!!.execSQL("DROP TABLE IF EXISTS groupTBL")  
8      onCreate(p0)  
9  }
```

01 SQLite 프로그래밍

- (5) 메인 액티비티 클래스에 다음 같은 변수를 선언하고 onCreate()에서는 위젯 변수에 activity_main.xml의 7개 위젯을 대입
 - 새로 생성한 myDBHelper 클래스 변수
 - 에디트텍스트에 대응할 변수 4개
 - 버튼에 대응할 변수 3개
 - SQLiteDatabase 클래스 변수

01 SQLite 프로그래밍

예제 12-4 Kotlin 코드 3

```

1  class MainActivity : AppCompatActivity() {
2
3      lateinit var myHelper : myDBHelper
4      lateinit var edtName : EditText
5      lateinit var edtNumber : EditText
6      lateinit var edtNameResult : EditText
7      lateinit var edtNumberResult : EditText
8      lateinit var btnInit : Button
9      lateinit var btnInsert : Button
10     lateinit var btnSelect : Button
11     lateinit var sqlDB : SQLiteDatabase
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         setContentView(R.layout.activity_main)
16         title = "가수 그룹 관리 DB"
17
18         edtName = findViewById<EditText>(R.id.edtName)
19         edtNumber = findViewById<EditText>(R.id.edtNumber)
20         edtNameResult = findViewById<EditText>(R.id.edtNameResult)
21         edtNumberResult = findViewById<EditText>(R.id.edtNumberResult)
22
23         btnInit = findViewById<Button>(R.id.btnInit)
24         btnInsert = findViewById<Button>(R.id.btnInsert)
25         btnSelect = findViewById<Button>(R.id.btnSelect)
26
27     }
28     ~~~ 생략 ~~~

```

01 SQLite 프로그래밍

- (6) <초기화>를 클릭했을 때 동작하는 리스너 코딩

예제 12-5 Kotlin 코드 4

```
1 myHelper = myDBHelper(this)
2
3 btnInit.setOnClickListener {
4     sqlDB = myHelper.writableDatabase
5     myHelper.onUpgrade(sqlDB, 1, 2)
6     sqlDB.close()
7 }
```

01 SQLite 프로그래밍

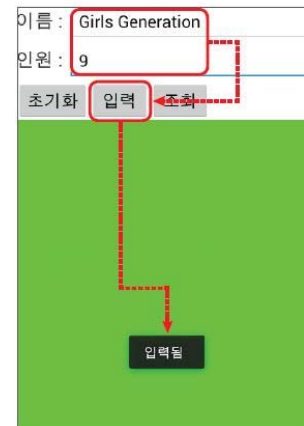
- (7) <입력>을 클릭하면 에디트텍스트의 값이 입력되는 리스너 코딩
 - 실행하여 데이터를 몇 건 입력했을 때 '입력됨' 메시지가 나옴

예제 12-6 Kotlin 코드 5

```

1 btnInsert.setOnClickListener {
2     sqlDB = myHelper.writableDatabase
3     sqlDB.execSQL("INSERT INTO groupTBL VALUES ( '"
4         + edtName.text.toString() + "' , '"
5         + edtNumber.text.toString() + "');"
6     sqlDB.close()
7     Toast.makeText(applicationContext, "입력됨", Toast.LENGTH_SHORT).show()
8 }

```



01 SQLite 프로그래밍

- (8) <조회>를 클릭할 때, 테이블에 입력된 내용이 모두 아래쪽 에디트텍스트에 출력되는 리스너 코딩

예제 12-7 Kotlin 코드 6

```

1  btnSelect.setOnClickListener {
2
3      sqlDB = myHelper.readableDatabase
4      var cursor: Cursor
5      cursor = sqlDB.rawQuery("SELECT * FROM groupTBL;", null)
6
7      var strNames = "그룹이름" + "\r\n" + "———" + "\r\n"
8      var strNumbers = "인원" + "\r\n" + "———" + "\r\n"
9
10     while (cursor.moveToNext()) {
11         strNames += cursor.getString(0) + "\r\n"
12         strNumbers += cursor.getString(1) + "\r\n"
13     }
14
15     edtNameResult.setText(strNames)
16     edtNumberResult.setText(strNumbers)
17
18     cursor.close()
19     sqlDB.close()
20 }

```

01 SQLite 프로그래밍

- 프로젝트 실행 및 결과 확인
 - (9) 프로젝트를 실행한 후 데이터를 입력하고 조회
 - (10) groupDB의 내용을 명령 프롬프트에서 확인

명령 프롬프트에서의 데이터베이스 접근

```
1 CD \CookAndroid\sdk\platform-tools\
2 adb root
3 adb shell
4 # cd /data/data/com.cookandroid.project12_2
5 # cd databases
6 # sqlite3 groupDB
7 sqlite> .header on
8 sqlite> .mode column
9 sqlite> SELECT * FROM groupTBL;
```

```
C:\#> CD #CookAndroid#SDK#platform-tools#
C:#CookAndroid#SDK#platform-tools> adb root
C:#CookAndroid#SDK#platform-tools> adb shell
generic_x86:/ # cd /data/data/com.cookandroid.project12_2
generic_x86:/data/data/com.cookandroid.project12_2 # cd databases
generic_x86:/data/data/com.cookandroid.project12_2/databases # sqlite3 groupDB
SQLite version 3.22.0 2020-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM groupTBL;
gName          gNumber
-----
Girls Generation 9
After School   7
Miss A         4
AOA            8
sqlite>
```

그림 12-11 명령 프롬프트에서의 데이터베이스 접근

01 SQLite 프로그래밍

▶ 직접 풀어보기 12-2

[실습 12-2]에 <수정>과 <삭제>를 추가하라.

- '이름'에 그룹 이름과 변경된 인원을 입력한 후 <수정>을 클릭하면 해당 그룹의 인원이 변경된다.
- '이름'에 그룹 이름을 입력하고 <삭제>를 클릭하면 해당 그룹의 행이 삭제된다.
- <입력>, <수정>, <삭제>를 클릭하면 그 결과가 즉시 화면에 출력된다.

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 결과가 보이게 하려면 btnSelect.callOnClick()을 호출한다.



그림 12-12 수정된 가수 그룹 관리 DB 앱

02 SQLite GUI 툴 활용

■ DB Browser for SQLite

- SQLite에 접근할 때 명령 프롬프트를 이용했지만 DB Browser for SQLite라는 GUI 툴을 사용하면 좀 더 편리



그림 12-13 DB Browser for SQLite 화면

02 SQLite GUI 툴 활용

- DB Browser for SQLite에서 DB 및 테이블 생성
 - [파일]-[새 데이터베이스]를 선택
 - [저장하려는 파일명을 고르세요] 창에서 데이터베이스 파일이 저장될 경로와 파일명을 지정하고 저장

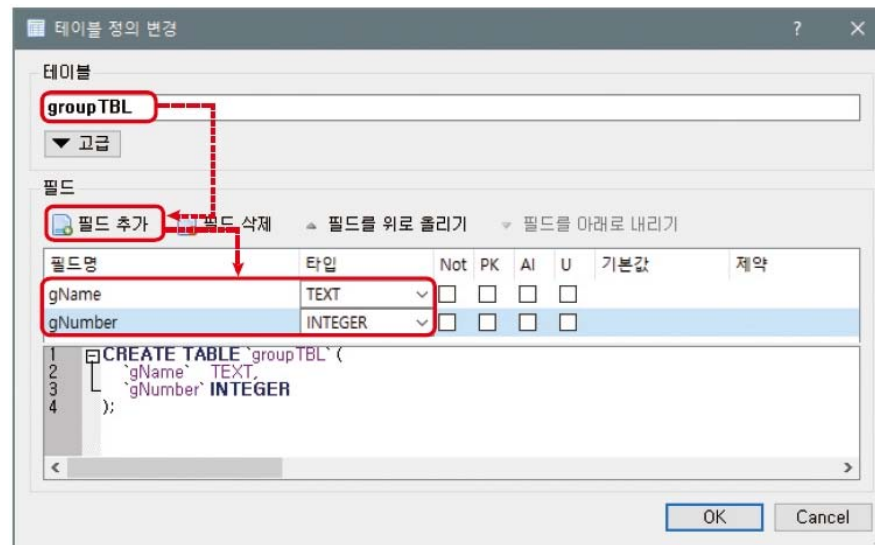


그림 12-14 DB Browser for SQLite에서의 데이터베이스 및 테이블 생성

02 SQLite GUI 툴 활용

- DB Browser for SQLite에서 데이터 입력
 - [데이터 보기] 탭 클릭 후 <새 레코드> 클릭하고 데이터 입력

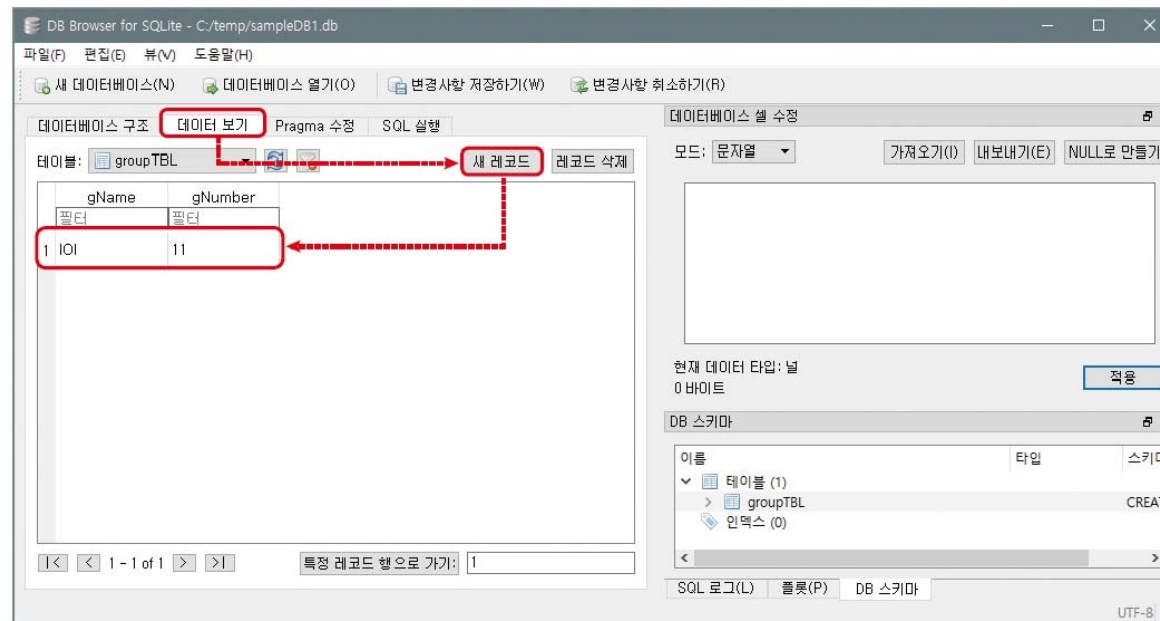


그림 12-15 DB Browser for SQLite에서의 행 데이터 입력

02 SQLite GUI 툴 활용

▶ 직접 풀어보기 12-3

[실습 12-2]의 groupDB 데이터베이스 파일을 Device File Explorer를 이용하여 AVD에서 PC로 가져온(Save As) 후, DB Browser for SQLite에서 가수 그룹 이름을 모두 한글로 고치고 가수 그룹을 몇 개 더 입력하라. 그런 다음 AVD에 다시 넣고(Upload) [실습 12-2]를 실행하여 데이터가 잘 조회되는지 확인하라.

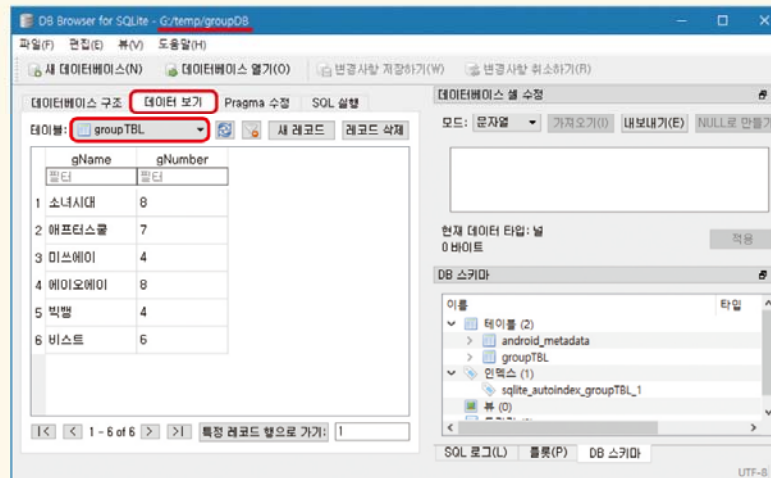


그림 12-16 SQLite 활용한 가수 그룹 관리 DB 앱

02 SQLite GUI 툴 활용

- **SQLite Developer**

- DB Browser for SQLite와 마찬가지로 그래픽 화면에서 데이터베이스를 관리하기 위한 툴
- <http://www.sqlitedeveloper.com/download>에서 다운로드하여 설치
- [Database]-[Register Database]로 편집할 데이터베이스 파일을 선택 후 데이터 추가

02 SQLite GUI 툴 활용

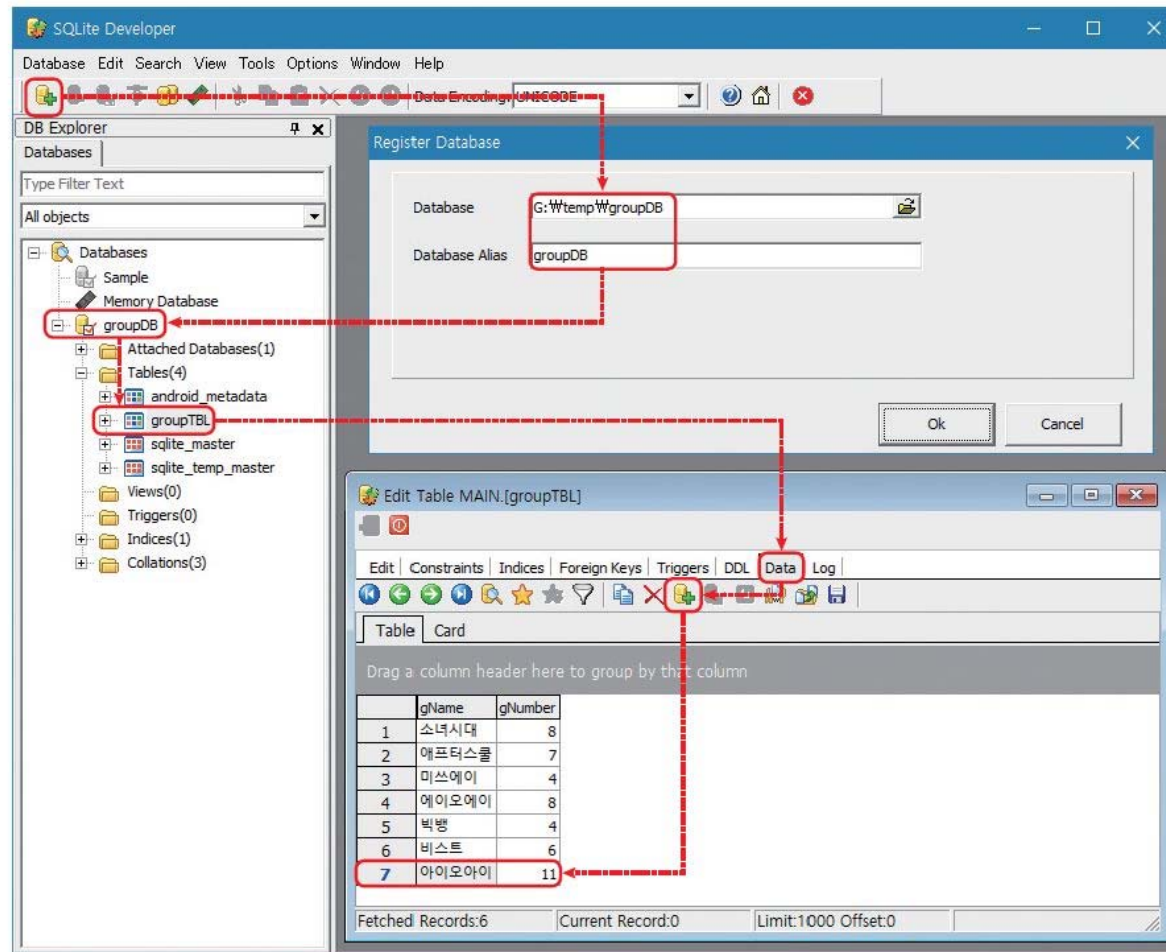


그림 12-17 SQLite Developer에서의 행 데이터 처리

01 SQLite 프로그래밍

- 직접 풀어보기 (학생 목록 예제)



01 SQLite 프로그래밍 (MainActivity.kr)

```

3  import android.content.Intent
4  import androidx.appcompat.app.AppCompatActivity
5  import android.os.Bundle
6  import android.view.View
7  import android.view.ViewGroup
8  import android.widget.BaseAdapter
9  import android.widget.Toast
10 import kotlinx.android.synthetic.main.activity_main.*
11 import kotlinx.android.synthetic.main.rowitem.view.*
12 class MainActivity : AppCompatActivity() {
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         setContentView(R.layout.activity_main)
17         var studentList : ArrayList<StudentC> = arrayListOf<StudentC>()
18
19
20
21         var myHelper = MyDBHelper( context: this)
22         var sqldb : SQLiteDatabase = myHelper.readableDatabase
23         var cursor : Cursor = sqldb.rawQuery( sql: "select * from studentTBL", selectionArgs: null)
24         while (cursor.moveToNext()) {
25             studentList.add( StudentC(cursor.getString(0), cursor.getString(1), cursor.getString(2), cursor.getString(3), cursor.getString(4), cursor.getString(5)) )
26         }
27         cursor.close()
28         sqldb.close()
29
30         var adapter1 = ListViewAdapter(studentList)
31         listView1.adapter = adapter1
32         listView1.setOnItemClickListener { adapterView, view, i, l ->
33             Toast.makeText(applicationContext, studentList.get(i).name, Toast.LENGTH_SHORT).show()
34             var intent1 = Intent(applicationContext, JasehiActivity::class.java)
35             intent1.putExtra( name: "name", studentList.get(i).name)
36             intent1.putExtra( name: "id", studentList.get(i).id)
37             intent1.putExtra( name: "hdong1", studentList.get(i).hdong1)
38             intent1.putExtra( name: "hdong2", studentList.get(i).hdong2)
39             startActivity(intent1)
40         }
41     }
42 }

```

01 SQLite 프로그래밍(MyDBHelper.kr, StudentC.kr)

- studentDB, studentTBL(학번(PK), 이름, 학과, 사진, 학생설명1, 학생설명2)

```
package aca.hankook.customlistview

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class MyDBHelper(var context: Context) : SQLiteOpenHelper(context, name: "studentDB", factory: null, version: 1) {
    override fun onCreate(p0: SQLiteDatabase?) {
        var tableCreateSql = "create table studentTBL (id char(4) primary key, name varchar(20), dept varchar(20), sajin varchar(5), hdong1 varchar(200), hdong2 varchar(600))"
        p0!!.execSQL(tableCreateSql)
        p0!!.execSQL("insert into studentTBL values('2101', '배수지', '멀티미디어과', 's2101', '대한민국의 가수겸배우', 'missA 멤버로 데뷔, 영화<건축사> 출연')")
        p0!!.execSQL("insert into studentTBL values('2102', '한지민', '컴퓨터소프트웨어과', 's2102', '활동1', '활동2')")
        p0!!.execSQL("insert into studentTBL values('2103', '남주혁', '스타트업과', 's2103', '대한민국의 모델이자 배우', '경남중학교 농구선수로 활동')")
        p0!!.execSQL("insert into studentTBL values('2201', '조정석', '간담체외과', 's2201', '대한민국의 뮤지컬 배우 출신', '연기가 본업, 예능방송, 영화 출연')")
        p0!!.execSQL("insert into studentTBL values('2202', '유연석', '소아외과', 's2202', '대한민국배우', '활동2')")
        p0!!.execSQL("insert into studentTBL values('2203', '정경호', '흉부외과', 's2203', '활동1', '활동2')")
        p0!!.execSQL("insert into studentTBL values('2204', '김대명', '산부인과', 's2204', '활동1', '활동2')")
        p0!!.execSQL("insert into studentTBL values('2205', '전미도', '신경외과', 's2205', '활동1', '활동2')")
    }

    override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {
        p0!!.execSQL("drop table if exists studentTBL")
        onCreate(p0)
    }
}

package aca.hankook.customlistview

class StudentC(var id: String, var name: String, var dept: String, var sajin: String, var hdong1: String, var hdong2: String) {
```

01 SQLite 프로그래밍 (ListViewAdapter.kr)

```

9  class ListViewAdapter(var studentList: ArrayList<StudentC>) : BaseAdapter() {
10      override fun getView(position: Int, view: View?, parent: ViewGroup?): View {
11          var rowItemView : View? = view
12          if (rowItemView == null) {
13              rowItemView = View.inflate(parent?.context, R.layout.rowitem, root: null)
14          }
15          val student : StudentC = studentList[position]
16
17          var sajinResid : Int = parent!!.resources.getIdentifier( name: "aca.hankook.customlistview:" + "drawable/" + student.sajin, defType: null, defPackage: null)
18          rowItemView!!.itemImage.setImageResource( sajinResid)
19          rowItemView.itemName.text = student.name
20          rowItemView.itemDept.text = student.dept
21          rowItemView.itemId.text = student.id.toString()
22          return rowItemView
23      }
24      override fun getItem(p0: Int): Any {
25          return studentList[p0]
26      }
27      override fun getItemId(p0: Int): Long {
28          return p0.toLong()
29      }
30      override fun getCount(): Int {
31          return studentList.size
32      }
33  }
34  }

```


01 SQLite 프로그래밍 (/res/layout/rowitem.xml)

