헬로, 안드로이드

12주차 – SQL 활용하기 (1)

강대기
동서대학교 컴퓨터정보공학부

학습 목표

- SQLite가 뭔지 알고, 이를 사용할 줄 안다.
- SQL의 기본적인 사용법들을 안다.
- SQLite을 이용해 기본적인 데이터베이스 응용프로그램을 작성할 수 있다.
- 행을 추가하는 기본적인 데이터베이스 응용 프로그램을 작성할 수 있다.
- 쿼리를 실행하는 기본적인 데이터베이스 응용 프로그램을 작성할 수 있다.
- 쿼리 결과를 보여주는 기본적인 데이터베이스 응용 프로그램을 작성할 수 있다.

차례

- SQLite 소개
- SQL 기본
- 헬로, 데이터베이스
- 요약
- 퀴즈
- 연습문제

SQLite 소개

- 리처드 힙 박사가 2000년에 개발한 작지만 강력한 데이터베이스 엔진
- 안드로이드, 아이폰, 심비안폰, 파이어폭스 브라우저, 스카이프, PHP, 어도비 AIR, 맥 OSX, 솔라리스 등에 사용됨
- 인기 있는 이유
  - 무료, 작은 크기, 설치나 관리가 필요없음
- SQLite 데이터베이스는 하나의 파일임 – 실은 대부분의 임베디드 데이터베이스가 그러함
- /data/data/package_name/databases 에 파일 저장
- adb 나 이클립스의 파일 익스플로러 (창 > 뷰 보기 > 기타 > 안드로이드 > 파일 익스플로러) 에서 파일을 보고, 이동시키고, 삭제할 수 있음
- 프로그램에서 이 데이터베이스 파일을 액세스하려면 자바 입출력 루틴을 호출하는 대신, SQL 구문을 실행시키면 됨
- 안드로이드는 도우미 클래스와 메서드를 통해 SQL의 복잡한 부분을 숨기긴 하지만, 제대로 사용하려면 SQL을 알아야 함

감동적인 SQLite 라이선스

- 선을 행하고 악을 멀리하라
- 자신을 용서하고 남을 용서하라
- 서로 나누며 자신이 베푼 만큼만 취하라

SQL 기본

- SQL 구문에는 다음과 같은 것들이 있음
  - Data Definition Language (DDL)
  - 수정 (Modification)
  - 쿼리 (Query)

Data Definition Language (DDL)

- 데이터베이스에는 여러 개의 테이블들이 있음
- 테이블에는 여러 행들이 있음
- 행들에는 여러 열들이 있음
- 각 열들은 이름과 데이터 타입을 가짐
- DDL로 테이블과 열의 명칭들을 정의함
- 다음 구문은 열이 세 개인 테이블 생성

```
CREATE TABLE mytable (
  _id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT,
  phone TEXT
);
```

Data Definition Language (DDL)

- 한 열은 PRIMARY KEY 로 지정되는 데, 대부분의 데이터베이스 실제 응용에서는 그 행을 식별하는 고유 숫자가 사용됨
- AUTOINCREMENT 는 각 레코드의 키에 1을 더해 고유의 키 값을 가지게 해 줌
- 안드로이드에서 SQLite이 쓰일 때, 관습적으로 첫 열은 _id라고 명명함 – 안드로이드의 ContentProvider 에서 사용할 때 필요함
- SQLite의 열 타입은 강제적인 것이 아닌 힌트에 불과함 – 즉 타입 체킹을 안한다는 뜻으로 정수 열에 문자를 저장하거나, 그 반대도 아무 문제 없음 (이건 일부러 그렇게 한 게 아니라 기능을 구현하지 않은 것 뿐으로 보이나, 임베디드 데이터베이스에서 타입 체킹의 비용이 비싼 점을 감안하면 이해가 되기도 함)

수정 (Modification)

- SQL 데이터베이스에 레코드를 삽입, 삭제, 업데이트 하는 구문들을 의미함
- 몇 개의 전화 번호를 추가하는 구문은 다음과 같음
  - INSERT INTO TABLE mytable VALUES (null, '강대기', '320-1724');
  - INSERT INTO TABLE mytable VALUES (null, '강준서', '320-1725');
  - INSERT INTO TABLE mytable VALUES (null, 'Ian Kang', '320-1726');
- CREATE TABLE 구문에서 사용된 순서대로 값이 입력됨
- _id는 SQLite에서 직접 값을 찾아 넣을 것이므로 널(NULL) 값을 넣어 둠

쿼리 (Query)

- 데이터가 일단 테이블에 로드되고 나면, SELECT 구문을 사용해 쿼리를 테이블에 실행시킴
- 예를 들어 세번째 값을 원한다면, 다음과 같이 함
  - SELECT * FROM mytable WHERE (_id=3);
- 데이터베이스가 content addressable 한 특징이 중요하므로, 위와 같이 특정 위치보다는 이름으로 번호를 검색하게 됨
  - SELECT name, phone FROM mytable WHERE (name LIKE “%Ian%”);
- SQL은 대소문자 구분을 하지 않음

헬로, 데이터베이스

- 데이터베이스에 레코드를 저장한 뒤 이를 나중에 다시 보여주는 Events 라는 작은 응용 프로그램
- 프로젝트 생성
  - Events, org.example.events, Events, Events
- 데이터베이스를 설명하는 상수 – Constants 인터페이스
- SQLiteOpenHelper 도우미 클래스를 확장한 EventData 클래스로 데이터베이스의 생성과 버전 관리
- 이벤트를 저장하고 그 이벤트를 TextView로 보여주는 메인 프로그램 정의

Constants 인터페이스

- 데이터베이스를 설명하는 상수 몇 개를 저장할 장소로 Constants 인터페이스 사용 (자바에서 상수를 정의하는 방법)
- 각 이벤트는 events 테이블 안의 행으로 저장되며, 각 행은 _id, time, title 열이 있음
- _id는 primary key 이며 BaseColumns 인터페이스에 정의되어 있음
- time과 title은 타임스탬프와 이벤트 제목으로 각기 사용됨
- Java 5부터는 정적 임포트가 가능
- 정적 임포트에 대한 이클립스에서의 지원은 미약할 수도 있음

/src/org/example/events/Constants.java

```
import android.net.Uri;

import android.provider.BaseColumns;

public interface Constants extends BaseColumns {
  public static final String TABLE_NAME = "events";

  // Columns in the Events database
  public static final String TIME = "time";
  public static final String TITLE = "title";
}
```

SQLiteOpenHelper 사용하기

- EventsData 라는 helper 클래스를 만들어 데이터베이스 응용 프로그램을 구현 (데이터베이스 생성과 버전 관리)
- SQLiteOpenHelper 클래스를 확장
  - 생성자와 두 개의 메서드 구현
- DATABASE_NAME – 우리가 사용할 데이터베이스 이름
- DATABASE_VERSION – 우리가 사용할 데이터베이스 버전
- 최초에는 데이터베이스에 접근을 시도하면 SQLiteOpenHelper는 데이터베이스가 없다는 걸 감지하고 onCreate()를 호출해 데이터베이스를 만듦 – CREATE TABLE 구문 실행
- 접근한 데이터베이스가 버전 숫자에 근거해 오래된 것이면, onUpgrade() 메서드 호출 – 여기서는 간단히 테이블을 삭제해 버리지만, ALTER TABLE 명령으로 존재하는 테이블에 열을 추가할 수도 있음

/src/org/example/events/EventsData.java

```
public class EventsData extends SQLiteOpenHelper {
  private static final String DATABASE_NAME = "events.db";
  private static final int DATABASE_VERSION = 1;

  /** Create a helper object for the Events database */
  public EventsData(Context ctx) {
    super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
  }

  @Override
  public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + _ID
        + " INTEGER PRIMARY KEY AUTOINCREMENT, " + TIME
        + " INTEGER," + TITLE + " TEXT NOT NULL);");
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion,
      int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(db);
  }

}
```

메인 프로그램 정의하기 (Events.java)

- Events 프로그램은 로컬 SQLite 데이터베이스를 이용하여 이벤트를 저장하고 그 이벤트를 TextView 에 문자열로 보여줌
- 화면에 너무 많은 이벤트가 있는 경우에 대비해서 ScrollView로 감쌈

main.xml

```
<ScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:id="@+id/text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</ScrollView>
```

Events.java의 onCreate()

- setContentView로 메인 뷰 설정
- EventsData로 데이터베이스 인스턴스 설정
- addEvent()로 새 이벤트 추가
- getEvents()로 데이터베이스에서 이벤트 목록을 Cursor로 받음
- showEvents()로 이벤트들을 화면에 보여줌
- 데이터베이스에 행을 추가하고 읽어들여 보여주는 부분은 try-catch-finally 블록으로 묶음
- finally 블록에서 데이터베이스를 SQLiteDatabase.close()로 닫음

Events.java

```
public class Events extends Activity {
  private static String[] FROM = { _ID, TIME, TITLE, };
  private static String ORDER_BY = TIME + " DESC";
  private EventsData events;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    events = new EventsData(this);
    try {       addEvent("Hello, Android!");       Cursor cursor = getEvents();       showEvents(cursor);     } finally {       events.close();     }
  }

  private void addEvent(String string) {
    SQLiteDatabase db = events.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(TIME, System.currentTimeMillis());
    values.put(TITLE, string);
    db.insertOrThrow(TABLE_NAME, null, values);
  }

  private Cursor getEvents() {
    SQLiteDatabase db = events.getReadableDatabase();
    Cursor cursor = db.query(TABLE_NAME, FROM, null, null, null,   null, ORDER_BY);
    startManagingCursor(cursor);
    return cursor;
  }

  private void showEvents(Cursor cursor) {
    StringBuilder builder = new StringBuilder(   "Saved events:\n");
    while (cursor.moveToNext()) {
      long id = cursor.getLong(0);
      long time = cursor.getLong(1);
      String title = cursor.getString(2);
      builder.append(id).append(": ");
      builder.append(time).append(": ");
      builder.append(title).append("\n");
    }
    TextView text = (TextView) findViewById(R.id.text);
    text.setText(builder);
  }
}
```

행 추가하기 (addEvents())

- addEvents() 메서드를 통해 데이터베이스에 새로운 행을 추가함
- 시간은 현재 시간
- 제목은 주어진 string 변수
- SQLiteOpenHelper를 확장한 EventsData가 멤버로 가지고 있는 SQLiteDatabase 객체를 받아서, insertOrThrow() 메서드로 레코드를 추가함
- 다만 쓰기 작업을 해야 하므로, getWritableDatabase() 메서드로 객체를 받아야 함
- 레코드에 해당하는 ContentValues를 구성하는 방식은 동일함

쿼리 실행하기 (getEvents())

- 역시 행 추가하기와 비슷한 방법으로 SQLiteDatabase 객체를 받음. 다만 읽기 작업을 하므로, getReadableDatabase()로 받음
- SQLiteDatabase.query()를 통해 SELECT 문 수행. ORDER_BY로 레코드를 새 것부터 옛날 것 순으로 정렬해서 반환함
- Activity.startManagingCursor() 호출 – 액티비티의 수명주기에 맞춰 커서의 수명 주기도 같이 관리해 줌. 프로그래머가 액티비티의 수명주기에 따라 데이터 소스에 연결된 커서를 비활성화하거나 쿼리를 다시 호출할 필요가 없게 함
- Cursor는 Java Iterator, JDBC ResultSet, ASP.NET의 DataSet 과 비슷함

쿼리 결과 보여주기(showEvents())

- Cursor를 받아서, 사용자가 볼 수 있도록 출력함
- 이벤트를 큰 문자열로 만들어서 모든 이벤트를 저장하고 줄바꿈으로 구분함 – 모든 이벤트들을 하나의 문자열로 만듦 – 바보같은 방법
- Cursor.moveToNext() 메서드는 데이터베이스의 다음 행으로 넘어가게 함
- Cursor에 대해 getLong()과 getString()을 통해 해당 열의 데이터를 뽑아냄
- Cursor.getColumnIndexOrThrow()로 열 인덱스 숫자를 찾아낼 수 있으나, 실행 속도가 느려짐

요약

- SQLite가 뭔지 알아 보았고, 이를 사용할 줄 알았다.
- SQL의 기본적인 사용법들을 알았다.
- SQLite을 이용해 기본적인 데이터베이스 응용프로그램을 작성할 수 있었다.
- 행을 추가하는 기본적인 데이터베이스 응용 프로그램을 작성할 수 있었다.
- 쿼리를 실행하는 기본적인 데이터베이스 응용 프로그램을 작성할 수 있었다.
- 쿼리 결과를 보여주는 기본적인 데이터베이스 응용 프로그램을 작성할 수 있었다.

퀴즈

- 데이터베이스 테이블에 레코드를 삽입하는 메서드는 무엇인가?
- 데이터베이스에 대해 쿼리를 실행하는 메서드들을 열거해 보라. query()는 어떤 메서드이며, execSQL()은 어떤 메서드인가?
- ORDER BY는 어떤 옵션인가?
- 커서란 무엇인가?
- 데이터베이스에서 컬럼 인덱스 숫자를 찾아내는 방법은 무엇인가?

연습문제

- 학생 이름, 국어, 영어, 수학에 대한 데이터베이스 프로그램을 작성하라. 총점, 평균, Grade 등을 계산해서 디스플레이 하라.
- 자동차를 운전하는 사람을 위한 차계부 프로그램을 구상해 보라. 차계부 데이터베이스에 저장해야할 데이터는 무엇이 있는지 열거해 보라.
- 도서 대여점의 도서 관리 프로그램을 작성하라. 도서를 표현하고 저장하기 위해서는 어떤 데이터가 있어야 하고, 고객이 도서를 대여하기 위해서는 어떤 데이터가 필요한가?
- 점심이나 저녁 외식을 위한 음식점을 추천하는 서비스를 생각해 보자. 이러한 추천 서비스를 위해, 음식점을 데이터베이스에 저장해야 한다. 음식점을 저장하기 위해 기본적으로 어떤 데이터가 필요하고, 사용자에게 효과적인 서비스를 제공하려면 어떤 데이터가 더 필요할까?