SQLite 로컬 데이터베이스

김용만

1. SQLite의 특징

- 오픈 소스
- 최소한의 라이브러리만으로 동작
- 관리유저나 설정파일 등이 불필요
- 원자성(atomicity), 일관성(consistency), 독립성(isolation), 영속성(durabitiy)이라는 데이터베이스에 필요한 특성을 갖추고 있다.
- SQL92에서 정의된 명령어의 대부분 이용가능, 하나의 데이터베이스가 하나의 파일에 저장

2. 어플리케이션 개발시 SQLite 사용

- 애플리게이션이 작성한 SQLite의 데이터베이스는 /data/data/<패키지명>/databases 디렉토리에 작성됨
- 하나의 애플리케이션에 대해서 하나의 디렉토리가 할당. 데이터베이스는 다른 애플리케이션이 읽고 쓰기는 할 수 없습니다. 그 경우는 Content provider의 기능을 이용
- 애플리케이션이 작성한 데이터베이스를 조작하고자 할 때는 /system/xbin 디렉터리에 있는 sqite3 명령어를 사용.

3. SQLite 사용시 기본 사항

1) 테이블의 작성과 삭제

Create [TEMP] TABLE <테이블명> (Column정의, [테이블제약]);

TEMP(또는 TEMPORARY) 키워드를 지정하면 일시적인 테이블을 작성한다. 이 테이블은 데이터베이스를 close했을 때에 자동적으로 삭제

SQLite 의 Storage Type

Storage Type	내용
INTEGER	최대 8바이트의 정수값
REAL	최대 8바이트의 부동소수점 수치
TEXT	UTF-8의 문자열

BLOB	Binary 데이터
NULL	값이 존재하지 않음

테이블의 작성 예

```
Create table contact_list(
        _id integer primary key autoincrement,
        name text not null,
        tel text not null default 'unknown',
        unique (name, tel),
        check(length(tel)>=3));
);
```

***외래키는 지원하지 않음**

테이블 삭제 Drop 명령어 사용

drop table 테이블명;

2) 테이블명의 변경과 컬럼 추가

테이블 명의 변경에는 ALTER 명령어의 RENATE TO를 사용

alter table 테이블명 renamte to 새로운 테이블명;

테이블의 column추가에는 alter 명령어의 add column을 사용

alter table 테이블명 add column 컬럼 정의;

3) SQLite의 내장 함수(Buit-in Function)

내장함수(Built-in Function)

함수	개요
min(X,Y,...,)	최소값을 반환한다.

max(X,Y,…,)	최대값을 반환한다.
typeof(X)	데이터의 형을 반환한다.
lenth(X)	문자열의 길이를 반환한다.
substr(X,Y,Z)	X의 문자열의 Y번째에서 Z문자의 길이의 문자열을 반환한다.
abs(X)	절대값을 반환한다.
round(X,Y)	X의 수치를 소수점 이하 Y에서 둥글게 하다.
upper(X)	대문자로 변환하다.
lower(X)	소문자로 변환하다.
coalesce(X,Y,…,)	최초의 NULL이 아닌 값을 반환한다.
hex(X)	문자열을 16진수 표기로 반환한다.
Ifnull00,Y,…(X)	최초의 NULL이 아닌 값을 반환한다.
random()	Radom한 값을 반환한다.
randomblob(X)	X바이트의 random한 값의 열을 반환한다.
nullif(X,Y)	2개의 값이 다르면 최초의 값을 반환한다. 같아면 NULL을 반환한다.
sqlite_version()	SQLite의 버전을 반환한다.
quote(X)	문자열을 SQL의 문자열로 삽입할 수 있도록 escape처리를 실행한다.
last_insert_rowid()	마지막에 삽입한 레코드의 ID를 반환한다.
sum(X,Y,…,)	NULL이 아닌 행의 값의 합계를 반환한다. 행이 없는 경우 NULL을 반환한다.
total(X,Y,…,)	NULL이 아닌 행의 값의 합계를 반환다. 행이 없는 경우 0를 반환한다.
avg(X,Y,…,)	NULL이 아닌 행의 값의 평균을 반환한다. 행이 없는 경우 NULL을 반환한다.
count(X)	그룹 안에서 X가 NULL이 아닌 행의 행수를 반환한다.
count(*)	그룹 별의 행수를 반환한다.

4. SQLite 데이터베이스 이용하기

1) SQLiteOpenHelper

데이터베이스를 생성하고 오픈하려면 SQLiteOpenHelper 객체를 사용한다. SQLiteOpenHelper 클래스는 애플리케이션에서 요구하는 내용에 따라 데이터베이스를 생성하거나 업그레이드하는 기능 제공

```
public class DatabaseHelper extends SQLiteOpenHelper{

  // 생성자
  public DatabaseHelper (Context context){
    super(context, "dbtest", null, 1);
    //context : Activity등의 Context 인스턴스
    //dbtest : 데이터베이스의 이름
    //null : 커서 팩토리(보통 null지정)
    //1 : 데이터베이스 스키마 버전
  }
  public void onCreate(SQLiteDatabase db){
     //테이블을 생성하고 초기 데이터를 추가
    String table_sql = "create Table test( _id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT
NOT NULL);";
    db.execSQL(table_sql);
  }
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
    // 버전 번호를 확인해서 새로운 스키마로 적절하게 업그레이드 함
    db.execSQL("DROP TABLE IF EXISTS test");
    onCreate(db);  //onCreate 메소드를 호출해서 새로 table 셋팅
  }
}
```

2) 데이터 추가

```
ContentValues cv = new ContentValues();
cv.put("title", "즐거운 하루");
cv.put("name", "홍길동");
db.insert("member", "title", cv);
```

3) 데이터 업데이트

```
ContentValues re = new ContentValues();
re.put("name", "홍길동");
```

String[] params = new String[] {"dragon"};
db.update("member", re, "userid=?", params);

4) 데이터 삭제

String[] params = new String[]{userid};
db.delete("member", "userid=?", params);

5) 데이터 불러오기

1) rawQuery() 메소드를 사용해 SELECT 구문을 직접 실행

Cursor c = db.rawQuery("SELECT name FROM sqlite_master where type='table' AND name='constants'", null);

2) query() 메소드를 인자로 각 부분의 값을 넘겨 실행

public Cursor query(String table, String[] columns, String selection, String[] selcetionArgs, String groupBy, String having, String orderBy, String limit)

- table : 대상 테이블 이름
- columns : 값을 가져올 컬럼 이름의 배열 (null : 모든 열)
- selection : WHERE 구문. 물음표를 사용해 인자의 위치를 지정할 수 있음 (null : 모든 레코드)
- selectionArgs : WHERE 구문에 들어가는 인자값
- groupBy : GROUP BY 구문 (null : 미사용)
- orderBy : ORDER BY 구문 (null : 미사용)
- limit : 레코드수 지정 (null : 미사용)

3) SQLiteQueryBuilder 클래스의 query() 메소드 이용

6) 커서의 활용

moveToFirst	가장 처음에 위치한 레코드로 커서를 이동
moveToNext	커서를 다음 행으로 옮김
moveToLast	가장 마지막에 위치한 레코드로 커서를 이동
moveToPrevious	이전 레코드로 커서를 이동
moveToPosition	특정 레코드로 커서를 이동
getPosition	커서가 현재 가리키고 있는 위치를 반환
getCount	전체 결과 건수가 몇 개인지 확인
getColumnCount	컬럼들의 전체 개수를 반환
getColumnIndex	특정 컬럼 번호 조회
getColumnName	특정 인덱스 값에 해당하는 컬럼 이름을 반환
getColumnNames	결과에 포함된 전체 컬럼 이름
isFirst	커서가 첫번째 레코드에 위치하는지를 반환
isLast	커서가 마지막 레코드에 위치하는지를 반환
isBeforeFirst	커서가 첫번째 레코드의 앞에 위치하는지를 반환
isAfterLast	커서가 마지막 레코드의 뒤에 위치하는지를 반환
getString() getInt()	컬럼값 반환
requery()	쿼리를 재실행
close()	커서가 확보한 자원을 모두 해제

7) 트랜잭션(Transaction) 다루기

여러 데이터베이스 연산이 반드시 모두 성공해야 하는 경우가 있다. 그런 경우, 연산들 중 하나라도 실패하면 연산 모두를 없던 일로 철회해야 한다. 하지만 모든 연산이 성공한다면, 비로소 연산 결과가 데이터베이스에 실제로 반영된다. 이와 같이 데이터베이스의 무결성이 보장되는 상태에서 요청된 연산들을 완수하기 위한 기본 작업 단위를 "트랜잭션(Transaction)"이라고 한다.

```
mDatabase.beginTransaction();
try{

  mDatabase.setTransactionSuccessful();
}catch(Exception e){
}finally{
```

```
  mDatabase.endTransaction();
}
```

5. ADB 셸을 이용한 데이터베이스 연결하기

1. 데이터베이스 연결

```
c:₩>adb shell
# sqlite3 /data/data/com.android.sql/databases/daily_memo.db
SQLite version 3.5.9
Enter ".help" for instructions
Sqlite>
```

2. 사용 가능한 데이터베이스 나열

```
sqlite>.databases
```

3. 사용 가능한 테이블 나열

```
sqlite>.table
```

4. 종료

```
sqlite>.quit 또는 sqlite>.exit
```