3. 데이터베이스

* 데이터베이스 기본 동작 순서

① 데이터베이스 만들기
② 테이블 만들기
③ 레코드 추가하기
④ 데이터 조회하기

* 사용 기본 API

- 데이터베이스를 열거나 만들기
  public abstract SQLiteDatabase **openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory)**
- 데이터베이스를 삭제하기
  public abstract boolean **deleteDatabase (String name)**
- 데이터베이스 확인하기
  public abstract String[] **databaseList()**
- create, insert, delete 등 결과데이터가 없는 SQL문 실행하기
  public void **execSQL(String sql)**
- select와 같이 조회에 따른 결과 데이터가 있는 SQL문 실행하기
  public Cursor **rawQuery(String sql)**

* 기본 SQL문

- 테이블 만들기
  CRREATE TABLE [IF NOT EXISTS] table_name(col_name column_definition, ...) [table_option] ...
- 테이블 없애기
  DROP TABLE IF EXISTS table_name
- 레코드 추가하기
  INSERT INTO table_name<(column list)> VALUES (value, ...)
- 테이블 내용 없애기
  DELETE FROM table_name [table_option] ...
- 데이터 조회하기
  SELECT column_name [,column_name2]
  FROM tablename
  WHERE [condition and|or condition...]
  [GROUP BY column-list]
  [HAVING conditions]
  [ORDER BY "column-list" [ASC | DESC]]

* Query 작성

SQLite에서 제공되는 메서드를 사용하거나 sql문을 작성하여 사용하는 방법이 있다.

▶INSERT
long SQLiteDatabase.insert(String table, String nullColumnHack, ContentValues values)

ex)
SQLiteDatabase db;
ContentValues value;
value.put("컬럼명", "값");
db.insert("테이블명", null, value)
db.execSQL("INSERT INTO table명 VALUES (null, '컬럼명', '값');");

▶DELETE
int delete (String table, String whereClause, String[] whereArgs)

ex)
```
db.delete("테이블명", null, null);
db.execSQL("DELETE FROM 테이블명;");
```

▶UPDATE
int update (String table, ContentValues values, String whereClause, String[] whereArgs)

ex)
```
db.update("테이블명", value, "컬럼명 = '변경할 문자열'", null);
db.execSQL("UPDATE 테이블명 SET 컬럼명 = '변경할 문자열' WHERE 컬럼 = '값'; ");
```

ex) 업데이트 할 컬럼이 여러개 일 때
```
db.execSQL("UPDATE 테이블명 SET update 할 컬럼명1 = '업데이트 할 문자열', "
                              + " update 할 컬럼명2 = '업데이트 할 문자열', "
                              + " update 할 컬럼명3 = 업데이트 할 정수값 "
                                                    + " WHERE 컬럼 = '값'; ");
```

▶SELECT
Cursor query (boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)

ex)
```
db.query("테이블명", new String[] {"쿼리할 컬럼명1", "쿼리할 컬럼명2"}, null, null, null, null,
        null);
db.rawQuery("SELECT 쿼리할 컬럼명1, 쿼리할 컬럼명2 FROM 테이블명", null);
```

* Cursor 객체

- SQL문을 이용해 쿼리를 실행한 후, 결과값으로 리턴받는 객체이다.
- 리턴값의 테이블에 들어 있는 각각의 레코드를 순서대로 접근할 수 있는 방법을 제공한다.
- moveToNext() : 그 다음 레코드를 가리키도록 한다. 레코드가 없으면 false 리턴.
- getCount() : 전체 레코드의 개수 구하기.
- getColumnCount() : 칼럼의 전체 개수 구하기.
- getColumnNames() : 모든 칼럼의 이름을 확인하기
- getColumnIndex() : 칼럼의 인덱스 값을 확인하기.
- close() : 커서 클래스 닫기

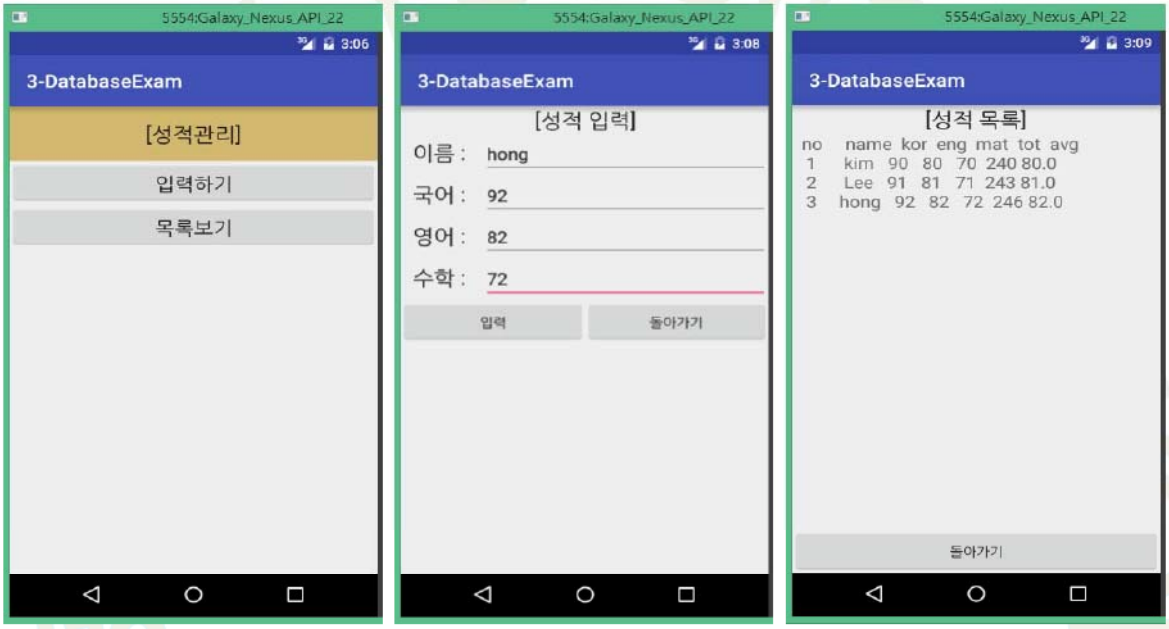
* SQLite에서 지원하는 칼럼 타입

칼럼 타입	설명
text, varchar	문자열
smallint, integer	정수 (2 byte or 4 byte)
real, float, double	부동소수 (4 byte or 8 byte)
boolean	true / false
date, time, timestamp	시간 (날짜, 시간, 날짜+시간)
blob, binary	이진수

예제3. DataBase로 학생 성적 관리하기

Application Name : 3-DatabaseExam
Company Domain : .example.com
Package Name : com.example.databaseexam
Project location : D:\Android\workspace\ch07\3-DBOpenExam

<실행 결과>



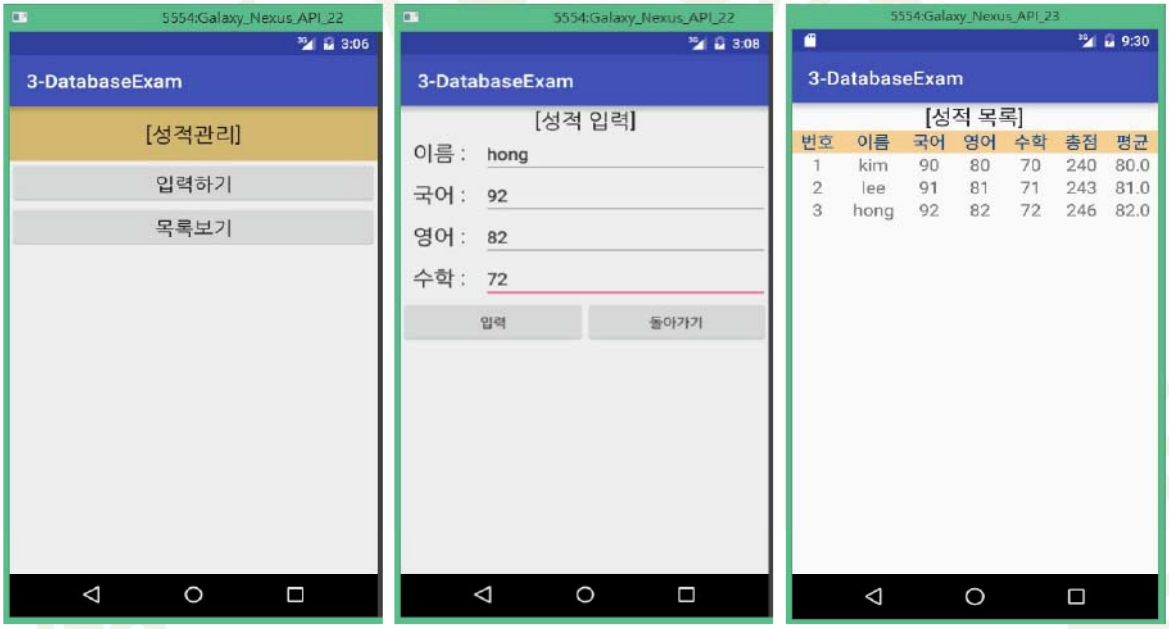
<작업 순서>

- app/res/layout 폴더에 Layout xml 파일 만들기
  => main.xml
  => input.xml
  => list.xml
- activity_main.xml 파일 기본 설정
  => <include> 로 main.xml, input.xml, list.xml Layout을 추가한다.
- MainActivity.java에서 이벤트 처리를 한다.

예제4. DataBase로 학생 성적 관리하기 - TableLayout으로 데이터 출력하기

Application Name : 3-DatabaseExam
Company Domain : .example.com
Package Name : com.example.databaseexam
Project location : D:\Android\workspace\ch07\3-DatabaseExam

<실행 결과>



<작업 순서>

- app/res/layout 폴더에 Layout xml 파일 만들기
  => main.xml
  => input.xml
  => list.xml 파일 수정하기
- app/res/layout 폴더에 table_row.xml Layout을 추가한다.
- activity_main.xml 파일 기본 설정
  => <include> 로 main.xml, input.xml, list.xml Layout을 추가한다.
- MainActivity.java에서 이벤트 처리를 한다.