모바일 프로그래밍

14 데이터베이스

2017 2학기

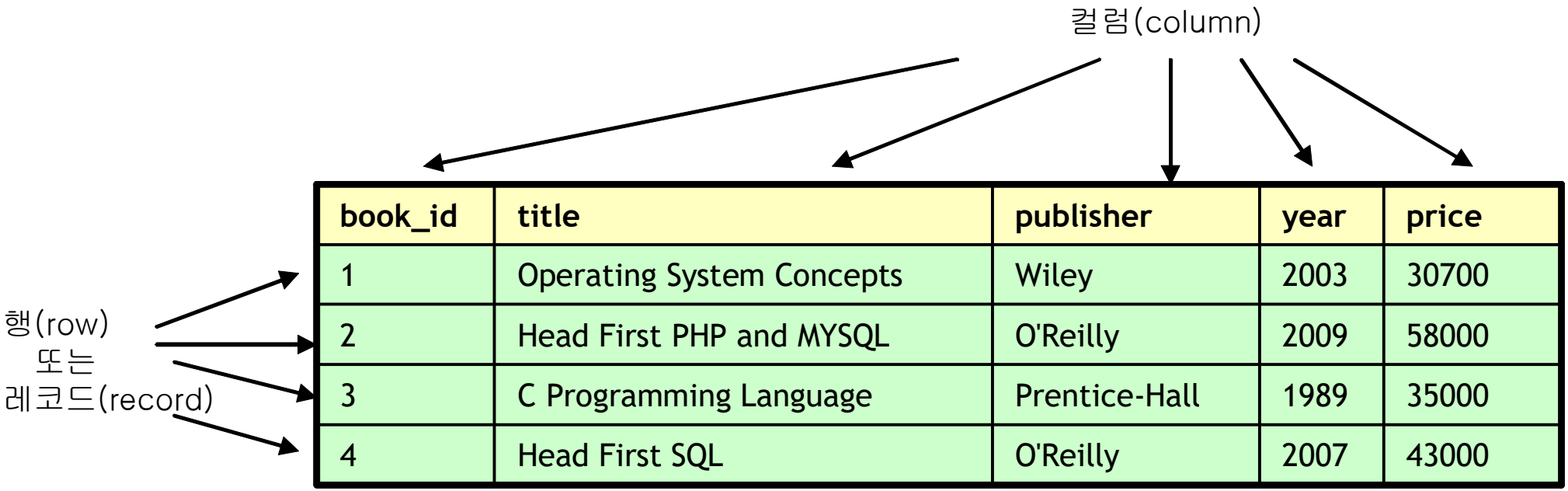
강승우

Computer Science and Engineering, KOREATECH

안드로이드에서 데이터베이스 이용

- SQLite
  - 안드로이드 플랫폼에서 지원하는 경량 데이터베이스 엔진
  - http://www.sqlite.org/
  - 특징
    - 별도의 서버 프로세스 없이 동작
    - 일반 디스크 파일 형태로 데이터를 관리 (하나의 파일이 완전한 SQL DB를 포함)
    - ANSI-C로 구현, 경량 라이브러리 (500KiB 이하)
    - Android, iOS 등의 모바일 플랫폼이나 임베디드 기기에서 많이 사용 (이외에도 Linux, Mac, Solaris 등 다양한 플랫폼에서 사용 가능)
    - SQL표준 SQL92의 대부분 지원
    - ACID (Atomic, Consistent, Isolate, Durable) 보장 (Transaction)
    - 소스가 public domain으로 공개 (어떤 용도로도 사용할 수 있음)
- 데이터베이스 이용 앱
  - 연락처, 메모장, 가계부, …

데이터베이스



book_id	title	publisher	year	price
1	Operating System Concepts	Wiley	2003	30700
2	Head First PHP and MYSQL	O'Reilly	2009	58000
3	C Programming Language	Prentice-Hall	1989	35000
4	Head First SQL	O'Reilly	2007	43000

SQL (Structured Query Language)

- RDBMS (Relational Database Management System)의 데이터를 관리하기 위해 설계된 언어
  - 관계형 데이터베이스에서 데이터베이스 스키마 생성, 수정, 데이터 검색 등을 하는데 이용
  - SQL 표준: 1986년 ANSI에 의한 최초 표준 SQL-86 이후 SQL-92에서 메이저 개정(ISO 9075)이 있었고, 현재 SQL:2011까지 개정본 나옴

- 데이터 정의 명령어
  - 관계형 데이터베이스의 구조 정의 (테이블 생성, 변경, 삭제 등)
- 데이터 조작 명령어
  - 데이터 검색, 삽입, 삭제, 갱신
- 데이터 제어 명령어
  - 데이터에 대한 액세스 제어 (권한 부여, 박탈 등)

SQL 명령어

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	데이블의 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 레코들를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.

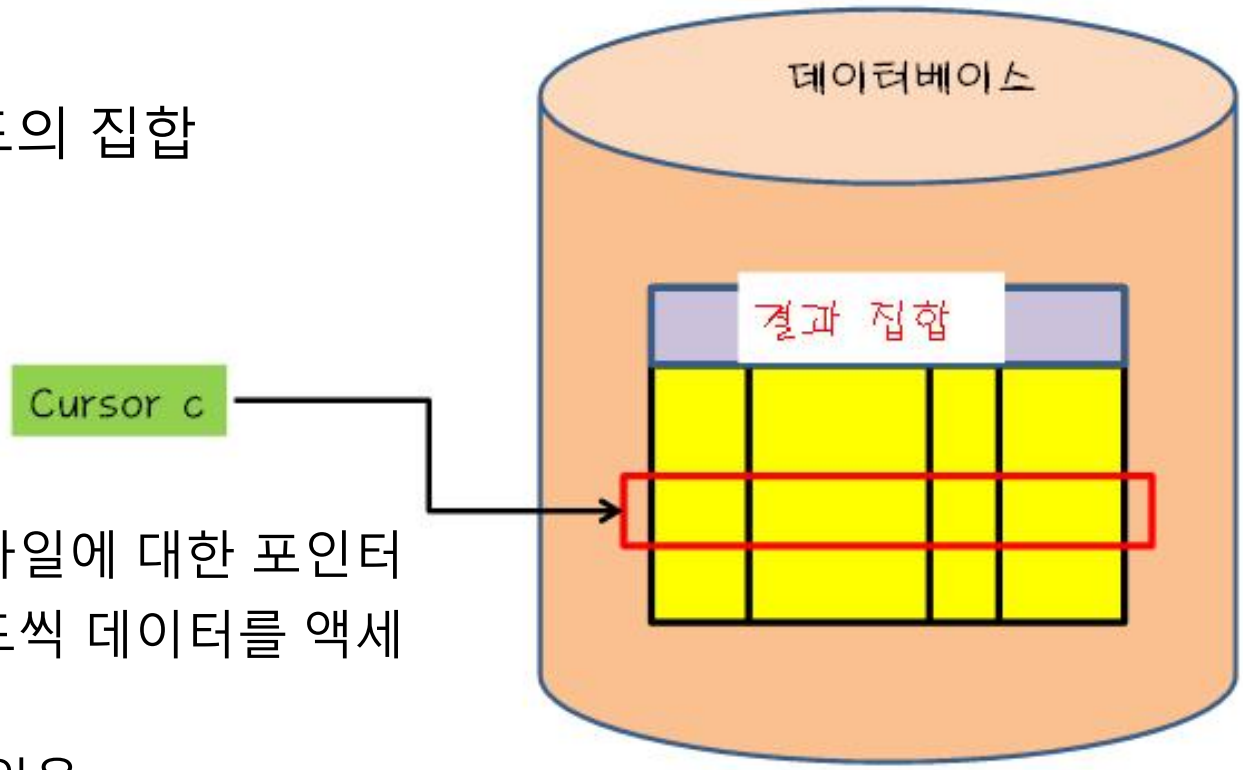
SQL 명령어 사용 예

```
INSERT INTO books (title, publisher, year, price)
      VALUES('Operating System Concepts', 'Wiley', '2003', 30700);
INSERT INTO books (title, publisher, year, price)
      VALUES('Head First PHP and MYSQL', 'OReilly', '2009', 58000);
INSERT INTO books (title, publisher, year, price)
      VALUES('C Programming Language ', 'Prentice-Hall', '1989', 35000);
INSERT INTO books (title, publisher, year, price)
      VALUES('Head First SQL', 'OReilly', '2007', 43000);
```

```
mysql> SELECT title, publisher, price FROM books;
+--------------------------+--------------+-------+
| title                    | publisher    | price |
+--------------------------+--------------+-------+
| Operating System Concepts | Wiley        | 30700 |
| Head First PHP and MYSQL | OReilly      | 58000 |
| C Programming Language   | Prentice-Hall | 35000 |
| Head First SQL           | OReilly      | 43000 |
+--------------------------+--------------+-------+
```

결과 집합(Result Sets), 커서(Cursors)

- 결과 집합
  - 질의의 조건을 만족하는 레코드의 집합
- 커서
  - 결과 집합의 레코드를 포함하는 파일에 대한 포인터
  - 커서를 이용하여 한번에 한 레코드씩 데이터를 액세스 할 수 있음
  - 레코드를 따라 커서를 이동할 수 있음
  - 결과 집합이 생성될 때 커서가 자동으로 만들어짐



SQLite 데이터베이스 생성

- SQLiteOpenHelper 클래스를 사용하여 데이터베이스를 생성하는 방법
- Context 클래스의 openOrCreateDataebase() 메소드로 데이터베이스 객체를 직접 생성하는 방법
  - SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory)

SQLiteOpenHelper 클래스를 이용하는 방법

- SQLiteOpenHelper
  - 데이터베이스 생성과 버전 관리를 위한 헬퍼 클래스
- SQLiteOpenHelper 클래스를 상속받는 클래스를 정의하고 아래 두 개의 메소드를 재정의하여 구현한다
  - onCreate(SQLiteDatabase db)
  - onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

- https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html?hl=ko

SQLiteOpenHelper 클래스 이용 과정 예

- SQLiteOpenHelper 클래스를 기반으로 DB를 생성하고 이용하는 과정
  1. SQLiteOpenHelper 클래스를 상속받는 클래스(예: DBHelper)를 정의하고, onCreate, onUpgrade 메소드 구현
  2. Activity (or Service) 클래스에서 이 클래스(DBHelper)의 객체를 생성
  3. 이 객체의 getWritableDatabase() 혹은 getReadableDatabase() 메소드를 호출하여 SQLiteDatabase 객체를 반환 받음
  4. 이 SQLiteDatabase 객체의 execSQL(), rawQuery() 메소드 등을 이용하여 SQL 문 실행

SQLiteOpenHelper 클래스 이용 – 서브클래스 정의

```
class DBHelper extends SQLiteOpenHelper {
        private static final String DATABASE_NAME = "mycontacts.db";
        private static final int DATABASE_VERSION = 1;

        public dbHelper(Context context) {
                super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        public void onCreate(SQLiteDatabase db) {
                db.execSQL("CREATE TABLE contact ( _id INTEGER PRIMARY KEY
                        AUTOINCREMENT, " + "name TEXT, tel TEXT);");
        }

        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                db.execSQL("DROP TABLE IF EXISTS contact");
                onCreate(db);
        }
}
```

SQLiteOpenHelper 클래스 이용 – 서브클래스 정의

- SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)
  - context: 데이터베이스를 생성하는데 이용할 context 객체 (activity 객체를 전달하거나 getApplicationContext()로 application의 context 객체 전달)
  - name: 데이터베이스 파일 이름 or null (in-memory DB)
  - factory: 커서 객체를 생성하는데 사용하는 매개변수 or null이면 default 커서 이용
  - version: 데이터베이스 버전
- onCreate(SQLiteDatabase db)
  - 데이터베이스가 처음 생성될 때 호출됨
  - SQL 명령어를 이용하여 테이블을 생성하고, 초기화를 수행한다
- onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
  - 데이터베이스가 업그레이드 될 필요가 있을 때 호출됨
  - 기존 테이블을 삭제하고 새로운 테이블을 생성한다

SQLiteOpenHelper 클래스 이용 - Activity에서 DB 생성 및 SQL 문 실행 예

```
public class DatabaseTest01 extends Activity {
    DBHelper helper;
    SQLiteDatabase db;

    public void onCreate(Bundle savedInstanceState) {
        ...
        helper = new DBHelper(this);
        try {
            db = helper.getWritableDatabase();
        } catch (SQLiteException ex) {
            db = helper.getReadableDatabase();
        }
        // 이제 필요할 때마다 db 객체를 통하여 SQL문장을 실행하면 된다.
        db.execSQL("INSERT INTO contact VALUES (null, '" + name
                        + "', '" + tel + "');");
    }
}
```

SQLiteOpenHelper 클래스 이용 – DB 생성

- public SQLiteDatabase getWritableDatabase()
  - 읽기/쓰기 가능한 데이터베이스를 생성하고 오픈
  - 처음으로 호출되는 경우 onCreate(SQLiteDatabase db) 메소드가 호출됨
- public SQLiteDatabase getReadableDatabase()
  - 읽기 전용 모드로 데이터베이스를 생성하고 오픈

데이터 추가, 삭제, 검색 방법 1

- SQLiteDatabase 클래스
  - SQLite 데이터베이스를 관리하기 위한 메소드를 제공
  - 이를 통하여 테이블 생성, 삭제, SQL 문 실행 (데이터 추가, 삭제, 검색), 그 외에 관리 태스크 수행할 수 있음

  - void execSQL(String sql)
    - SELECT 문을 제외한 모든 SQL 문장을 실행 (하나의 SQL 문만 실행)
    - CREATE TABLE, DELETE, INSERT 등
  - Cursor rawQuery(String sql, String[] selectionArgs)
    - SELECT 문을 실행할 수 있음
    - Cursor 객체를 반환 받아서 결과 집합의 레코드를 액세스 할 수 있음

  - https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html?hl=ko
  - https://developer.android.com/reference/android/database/Cursor.html?hl=ko

데이터 추가, 삭제, 검색 방법 2

- SQLiteDatabase 클래스에 정의된 전용 메소드 이용
  - long insert(String table, String nullColumnHack, ContentValues values)
    - 데이터베이스에 하나의 행(레코드) 추가
  - Int delete(String table, String whereClause, String[] whereArgs)
    - 데이터베이스에서 조건에 맞는 행 삭제
  - int update(String table, ContentValues values, String whereClause, String[] whereArgs)
    - 데이터베이스에서 조건에 맞는 행 갱신
  - Cursor query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)
    - 쿼리를 실행하고 결과 집합에 대한 커서를 반환

✓교재 내용과 SQLiteDatabase 클래스에 대한 설명 링크 참고

예제 – 연락처 저장 및 검색

- 예제 프로젝트 이름
  - 13_Database\Ch14DatabaseTest
- Java 소스
  - MainActivity.java
    - DBHelper 객체를 이용해 DB 생성
    - EditText에 입력된 데이터를 DB 테이블에 입력
    - EditText에 입력된 이름으로 테이블에서 데이터 검색하여 전화번호 표시
  - DBHelper.java
    - DB 테이블 생성하는 부분

예제 코드 – DBHelper.java

```
public class DBHelper extends SQLiteOpenHelper {
    private static final String DBName = "mycontacts.db";
    private static final int DBVer = 2;

    public DBHelper(Context context) {
        super(context, DBName, null, DBVer);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(
            "CREATE TABLE contacts ( _id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, tel TEXT);");
    }

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

예제 코드 – MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    editName = (EditText)findViewById(R.id.name);
    editTel = (EditText)findViewById(R.id.tel);

    // SQLiteOpenHelper 클래스의 subclass인 DBHelper 클래스  객체  생성
       helper = new DBHelper(this);
    // DBHelper 객체를  이용하여 DB 생성
       try {
        db = helper.getWritableDatabase();
    } catch (SQLiteException e) {
        db = helper.getReadableDatabase();
    }
}
```

예제 코드 – MainActivity.java

```
// 추가 버튼을 눌렀을 때 호출되는 메소드
public void insert(View v) {
    // EditText에 입력된 이름과 전화번호를 String 데이터로 추출
    String name = editName.getText().toString();
    String tel = editTel.getText().toString();

    // 이름과 전화번호를 가지고 INSERT 문을 만들어 실행
    db.execSQL("INSERT INTO contacts VALUES (null, '" + name + "', '" + tel + "');");
    Toast.makeText(getApplicationContext(), "성공적으로 추가되었음", Toast.LENGTH_SHORT).show();

    // EditText 초기화
    editName.setText("");
    editTel.setText("");
}
```

예제 코드 – MainActivity.java

```
// 탐색 버튼을 눌렀을 때 호출되는 메소드
public void search(View v) {
    String name = editName.getText().toString();
    Cursor cursor;
    // EditText에 입력된 이름을 가지고 쿼리문을 만들어 실행
    cursor = db.rawQuery("SELECT name, tel FROM contacts WHERE name='" + name + "';", null);

    // 반환된 커서에 ResultSets의 행의 개수가 0개일 경우
    if(cursor.getCount() == 0) {
        Toast.makeText(getApplicationContext(), "해당 이름이 없습니다", Toast.LENGTH_SHORT).show();
        return;
    }
    // 반환된 커서를 가지고 전화번호 얻고 EditText에 표시
    while(cursor.moveToNext()) {
        String tel = cursor.getString(1);
        editTel.setText(tel);
    }
}
```

Cursor 이용 – Cursor 클래스에 정의된 메소드

- Cursor 위치 변경
  - Cursor는 기본적으로 행(row) 단위로 참조
  - boolean moveToFirst() / moveToLast(): 첫번째/마지막 행으로 이동
    - cursor가 비어 있으면 false
  - boolean moveToNext() / moveToPrevious(): 다음 행/이전 행으로 이동
    - Cursor가 이미 마지막/첫번째 행이면 false 반환
  - boolean moveToPosition (int position): 해당 위치로 이동
- Cursor로 값 가져오기
  - getFloat(int columnIndex), getInt(int columnIndex), getLong(int columnIndex), getString(int columnIndex), …
    - 인덱스 값에 해당하는 열(column)의 값을 해당 타입으로 반환
    - DB 테이블 생성 시 사용한 데이터 타입과 일치하는 메소드를 이용해야 함

Cursor 이용 – Cursor 클래스에 정의된 메소드

_id	name	tel	email
1	김철수	01012345678	chulsoo@ggmail.com
2	이영희	01022229999	yhlee@ggmail.com
3	홍길동	01011118821	hgd@ggmail.com
4	이철수	01033337766	cslee@ggmail.com

⬅ 현재 Cursor 위치라면

cursor.getInt(0) → 1
cursor.getString(1) → 김철수
cursor.getString(3) → chulsoo@ggmail.com

- int getColumnIndex(String columnName)
  - 해당 이름에 대한 열의 인덱스 값 반환 / cursor.getColumnIndex(“email) → 3
- String getColumnName(int columnIndex)
  - 해당 인덱스 값에 대한 열의 이름 반환 / cursor.getColumnName(2) → tel
- int getCount()
  - 현재 cursor가 가리키는 테이블의 행의 개수 반환 / cursor.getCount() → 4
- String[] getColumnNames() : 열의 이름을 String 배열로 반환
- int getColumnCount() : 열의 개수를 반환

데이터베이스 질의 결과 표시하기

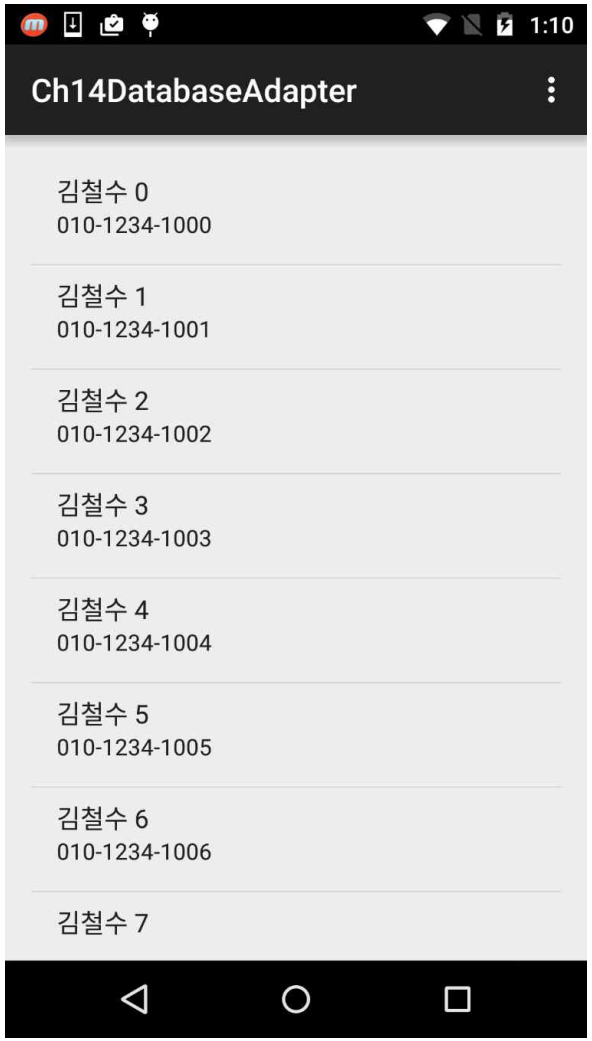
- 데이터베이스 질의 결과의 항목이 몇 개가 될지 미리 알 수 없음
  - 동적으로 가변적인 결과를 화면에 표시할 수 있어야 함
- SimpleCursorAdapter 이용
  - 데이터베이스에서 데이터를 가져와서 AdapterView에 공급해주는 어댑터 (06 리스트뷰 내용 참고)

- SimpleCursorAdapter(Context context, int layout, Cursor c, String[] from, int[] to)
  - layout: 데이터를 화면에 표시할 레이아웃 리소스 id
  - c: 커서 객체
  - from: 화면에 표시하고 싶은 컬럼의 이름
  - to: from 안의 각 컬럼이 표시되는 뷰의 리스트. 모두 텍스트 뷰여야 함

예제 – 연락처 목록 보기

- 예제 프로젝트 이름
  - 13_Database\Ch14DatabaseAdapter

- Java 소스
  - MainActivity.java
    - DBHelper 객체를 이용해 DB 생성 (앞의 예제와 동일)
    - DB에 질의문 실행하여 데이터 레코드를 액세스 할 수 있는 커서 얻음
    - SimpleCursorAdapter를 이용하여 리스트 뷰에 데이터 표시
  - DBHelper.java
    - onCreate() 메소드에서 DB 테이블 생성 및 초기화
      - 테스트용 데이터를 임의 생성 후 입력



예제 코드 snippet

```
// contacts 테이블에서 모든 레코드를 retrieve
c = db.rawQuery("SELECT * FROM contacts", null);

String[] from = {"name", "tel"};
int[] to = {android.R.id.text1, android.R.id.text2};
// SimpleCursorAdapter 객체 생성
// 하나의 리스트 아이템에 2개의 텍스트뷰를 표시하는 레이아웃
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_2, c, from, to);

// 레이아웃에 정의된 리스트뷰에 대한 참조 객체 얻음
ListView list = (ListView)findViewById(R.id.list);
// 리스트뷰 객체에 어댑터 설정
list.setAdapter(adapter);
```

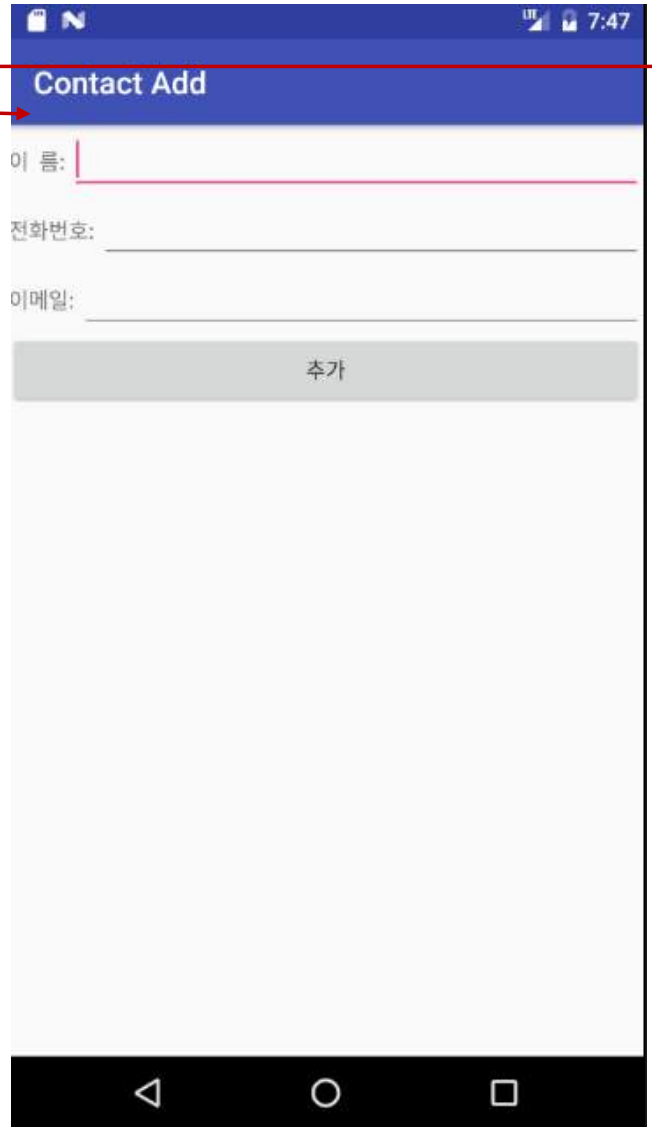
실습

- Github에 업로드된 프로젝트를 이용하여 연락처 앱 만들기
  - 예제 프로젝트: 13_Database\Ch14DBContactPractice
  - 앞의 두 예제를 활용하여 만든다

- 3개의 Activity로 구성
  - 연락처 목록 보기 화면
  - 연락처 추가 화면
    - 이름, 전화번호, 이메일 주소
  - 연락처 검색 화면
- 구현 기능
  - 연락처 추가, 연락처 검색 기능을 구현한다

연락처 보기 화면

연락처 추가 화면



연락처 검색 화면