

ANDROID LOCAL DB SQLITE

Efthimios Alepis

The background features a gradient from light yellow at the top to orange at the bottom. On the right side, there are several parallel white lines of varying lengths and positions, creating a sense of motion or a stylized graphic element.

- ▶ Android provides full support for SQLite databases
- ▶ Any databases you create will be accessible by name to any class in the application, but not outside the application
- ▶ The recommended method to create a new SQLite database is to create a subclass of SQLiteOpenHelper and override the onCreate() method, in which you can execute a SQLite command to create tables in the database
- ▶ You can then get an instance of your SQLiteOpenHelper implementation using the constructor you've defined
- ▶ To write to and read from the database, call getWritableDatabase() and getReadableDatabase(), respectively

USING SQLITE IN ANROID

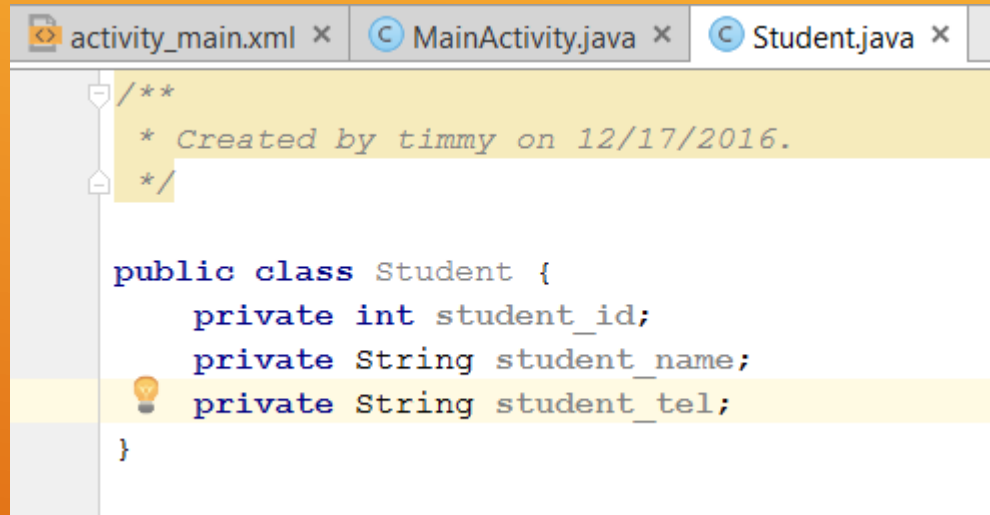


- ▶ You can execute SQLite queries using the SQLiteDatabase query() methods
- ▶ These methods return a Cursor that points to all the rows found by the query
- ▶ The Cursor is always the mechanism with which you can navigate results from a database query and read rows and columns

CURSOR INTERFACE

Android API	SQLite Version
API 24	3.9
API 21	3.8
API 11	3.7
API 8	3.6
API 3	3.5
API 1	3.4

SQLITE VERSIONS

A screenshot of an IDE window with three tabs: 'activity_main.xml', 'MainActivity.java', and 'Student.java'. The 'Student.java' tab is active, showing a Java class definition. The code includes a multi-line comment at the top, followed by the class declaration and three private member variables. The third variable, 'student_tel', is highlighted in yellow and has a lightbulb icon next to it, indicating a suggestion or warning.

```
/**
 * Created by timmy on 12/17/2016.
 */

public class Student {
    private int student_id;
    private String student_name;
    private String student_tel;
}
```

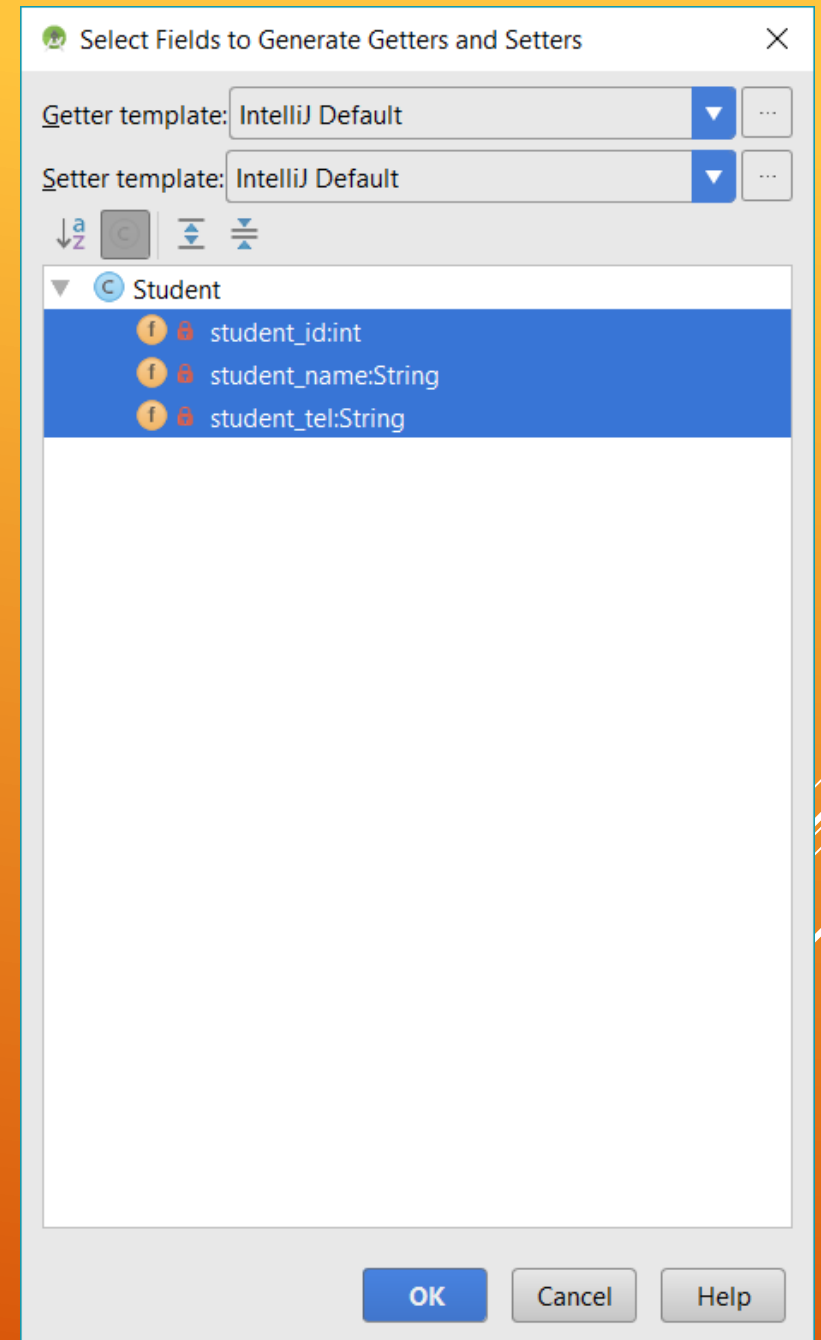
GENERATE ANDROID CLASS

```
activity_main.xml x MainActivity.java x Student.java x
/**
 * Created by timmy on 12/17/2016.
 */

public class Student {
    private int student_id;
    private String student_name;
    private String student_tel;
}
```

Generate

- Constructor
- Getter
- Setter
- Getter and Setter
- equals() and hashCode()
- toString()
- Override Methods... Ctrl+O
- Delegate Methods...
- Copyright



INSERT GETTERS AND SETTERS
(ALT+INSERT SHORTCUT)

```
activity_main.xml x MainActivity.java x Student.java x
/**
 * Created by timmy on 12/17/2016.
 */

public class Student {
    private int student_id;
    private String student_name;
    private String student_tel;

    public int getStudent_id() {
        return student_id;
    }

    public void setStudent_id(int student_id) {
        this.student_id = student_id;
    }

    public String getStudent_name() {
        return student_name;
    }

    public void setStudent_name(String student_name) {
        this.student_name = student_name;
    }

    public String getStudent_tel() {
        return student_tel;
    }

    public void setStudent_tel(String student_tel) {
        this.student_tel = student_tel;
    }
}
```

```
public Student(int student_id, String student_name, String student_tel) {  
    this.student_id = student_id;  
    this.student_name = student_name;  
    this.student_tel = student_tel;  
}  
  
public Student(String student_name, String student_tel) {  
    this.student_name = student_name;  
    this.student_tel = student_tel;  
}
```

INSERT THE APPROPRIATE CLASS
CONSTRUCTORS

- ▶ A contract class is a container for constants that define names for URIs, tables, and columns
- ▶ A good way to organize a contract class is to put definitions that are global to your whole database in the root level of the class
- ▶ Then create an inner class for each table that enumerates its columns
- ▶ Tip: A good advice for your inner class is to implement the “BaseColumns” interface!

CREATE CONTRACT CLASS

```
activity_main.xml x MainActivity.java x Student.java x UnipiDbContract.java x UnipiDbHelper.java x
package com.unipi.talepis.localdb1;

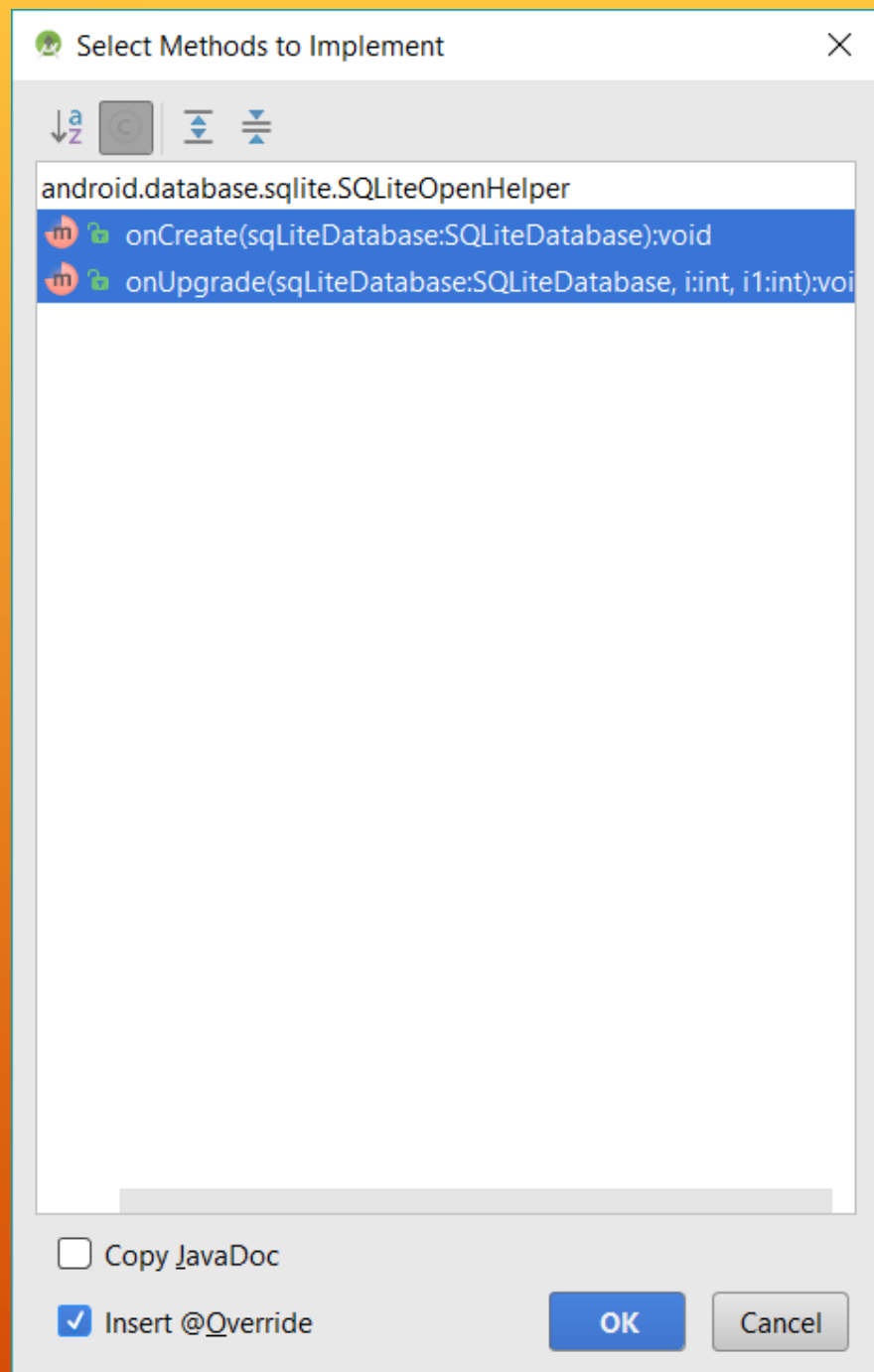
import android.provider.BaseColumns;

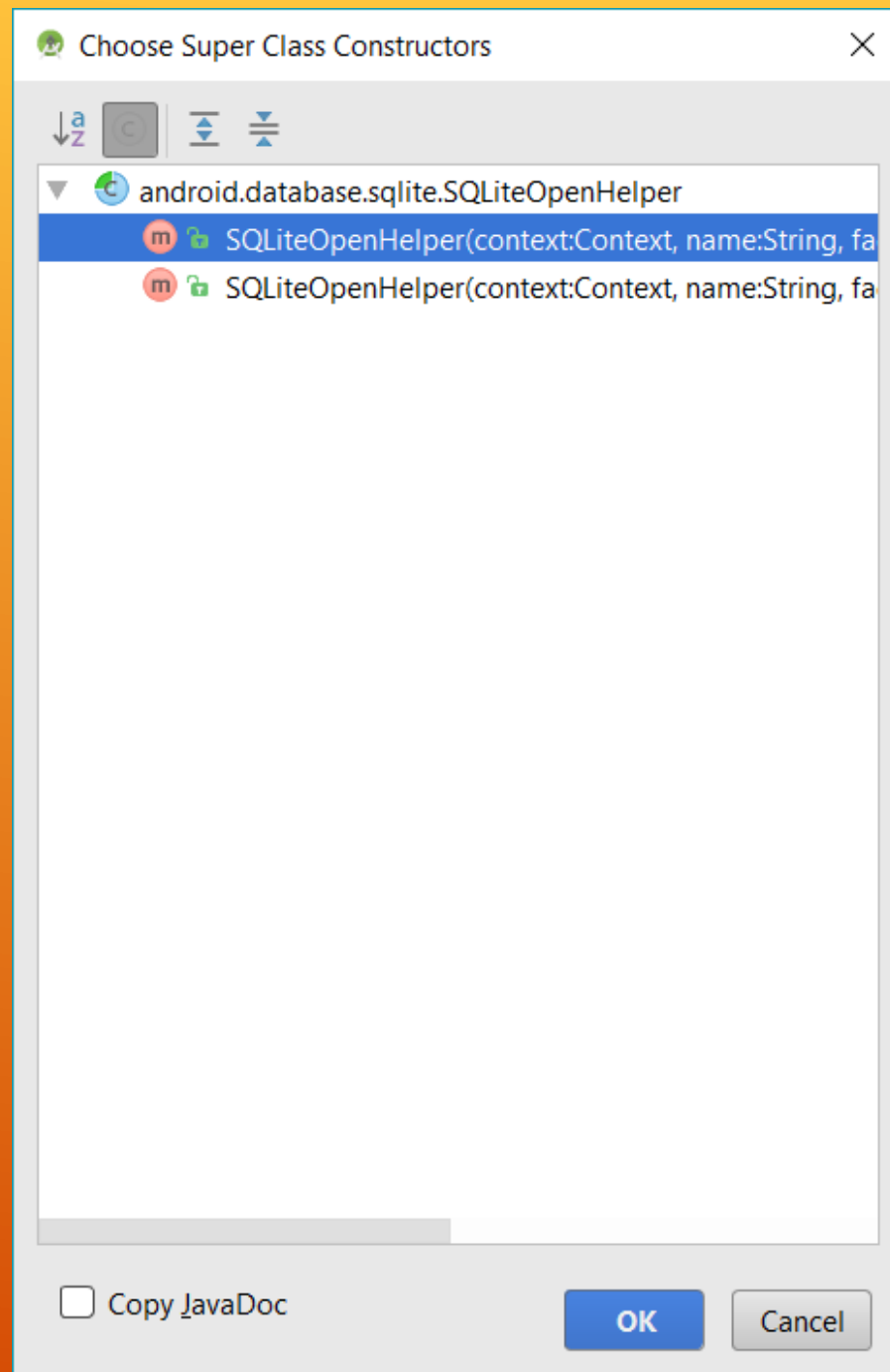
/**
 * Created by timmy on 12/17/2016.
 */

public class UnipiDbContract {
    // To prevent someone from accidentally instantiating the contract class,
    // make the constructor private.
    private UnipiDbContract() {}
    // Inner class that defines the table contents
    public static class StudentEntry implements BaseColumns{
        public static final String TABLE_NAME = "student";
        public static final String COLUMN_NAME_STUDENT_NAME = "student_name";
        public static final String COLUMN_NAME_STUDENT_TEL = "student_tel";
    }
}
```

- ▶ When you use this class to obtain references to your database, the system performs the potentially long-running operations of creating and updating the database only when needed and not during app startup
- ▶ Implement methods that create and maintain the database and tables
- ▶ Read and Write through calls to `getWritableDatabase()` and `getReadableDatabase()`
- ▶ Tip: For long running operations call `getWritableDatabase()` or `getReadableDatabase()` in a background thread

CREATE A DB HELPER CLASS USING SQLITEOPENHELPER CLASS





```
activity_main.xml x MainActivity.java x Student.java x UnipiDbContract.java x UnipiDbHelper.java x
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by timmy on 12/17/2016.
 */

public class UnipiDbHelper extends SQLiteOpenHelper {
    public static final String DATABASE_NAME = "UnipiDB.db";
    public static final int DATABASE_VERSION = 1;
    private static final String TEXT_TYPE = " TEXT";
    private static final String COMMA_SEP = ",";
    private static final String SQL_CREATE_STUDENT_TABLE =
        "CREATE TABLE " + UnipiDbContract.Student.TABLE_NAME + " (" +
        UnipiDbContract.Student._ID + " INTEGER PRIMARY KEY," +
        UnipiDbContract.Student.COLUMN_NAME_STUDENT_NAME + TEXT_TYPE + COMMA_SEP +
        UnipiDbContract.Student.COLUMN_NAME_STUDENT_TEL + TEXT_TYPE + " )";
    private static final String SQL_DELETE_ENTRIES =
        "DROP TABLE IF EXISTS " + UnipiDbContract.Student.TABLE_NAME;

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

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL(SQL_CREATE_STUDENT_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int il) {
        sqLiteDatabase.execSQL(SQL_DELETE_ENTRIES);
        onCreate(sqLiteDatabase);
    }
}
```


```
activity_main.xml x MainActivity.java x Student.java x UnipiDbContract.java x
package com.unipi.talepis.localdb1;

+import ...

public class MainActivity extends AppCompatActivity {
    UnipiDbHelper mDbHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mDbHelper = new UnipiDbHelper(getApplicationContext());
    }
}
```

TO ACCESS YOUR DATABASE,
INSTANTIATE YOUR HELPER CLASS

```
 // CRUD OPERATIONS
// Adding new Student
public void addStudent(Student Student) {}

// Getting single Student
public Student getStudent(int id) {}

// Getting All Students
public List<Student> getAllStudents() {}

// Getting Students Count
public int getStudentsCount() {}

// Updating single Student
public int updateStudent(Student Student) {}

// Deleting single Student
public void deleteStudent(Student Student) {}
}
```

ADD SOME BASIC CRUD OPERATIONS


```
// CRUD OPERATIONS
// Adding new Student
public void addStudent(Student student) {
    SQLiteDatabase db = mDbHelper.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_NAME, student.getStudent_name());
    values.put(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_TEL, student.getStudent_tel());

    // Inserting Row
    db.insert(UnipiDbContract.StudentEntry.TABLE_NAME, null, values);
    db.close(); // Closing database connection
}
```

INSERT NEW RECORD

insert

```
long insert (String table,  
            String nullColumnHack,  
            ContentValues values)
```

Convenience method for inserting a row into the database.

```

// Getting single Student
public Student getStudent(int id) {
    SQLiteDatabase db = mDbHelper.getReadableDatabase();
    // Define a projection that specifies which columns from the database
    // you will actually use after this query.
    String[] projection = {
        UnipiDbContract.StudentEntry._ID,
        UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_NAME,
        UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_TEL
    };
    // Filter results WHERE _ID = id
    String selection = UnipiDbContract.StudentEntry._ID + " = ?";
    String[] selectionArgs = { String.valueOf(id) };

    Cursor c = db.query(
        UnipiDbContract.StudentEntry.TABLE_NAME, // The table to query
        projection, // The columns to return
        selection, // The columns for the WHERE clause
        selectionArgs, // The values for the WHERE clause
        null, // don't group the rows
        null, // don't filter by row groups
        null // don't sort
    );
    if (c.getCount() > 0) {
        c.moveToFirst();
        Student student = new Student(c.getInt(c.getColumnIndex(UnipiDbContract.StudentEntry._ID)),
            c.getString(c.getColumnIndex(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_NAME)),
            c.getString(c.getColumnIndex(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_TEL)));
        // return student
        db.close();
        return student;
    } else {
        db.close();
        return null;
    }
}

```

READ FROM DATABASE WITH CRITERIA

query

```
Cursor query (String table,  
             String[] columns,  
             String selection,  
             String[] selectionArgs,  
             String groupBy,  
             String having,  
             String orderBy)
```

Query the given table, returning a [Cursor](#) over the result set.

```

// Getting All Students
public List<Student> getAllStudents() {
    SQLiteDatabase db = mDbHelper.getReadableDatabase();
    List<Student> studentList = new ArrayList<>();
    String[] projection = {
        UnipiDbContract.StudentEntry._ID,
        UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_NAME,
        UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_TEL
    };
    Cursor c = db.query(
        UnipiDbContract.StudentEntry.TABLE_NAME, // The table to query
        projection, // The columns to return
        null, // null columns means all
        null, // null values for the WHERE clause
        null, // don't group the rows
        null, // don't filter by row groups
        null // don't sort
    );
    while (c.moveToNext()) {
        Student student = new Student(c.getInt(c.getColumnIndex(UnipiDbContract.StudentEntry._ID)),
            c.getString(c.getColumnIndex(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_NAME)),
            c.getString(c.getColumnIndex(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_TEL)));
        studentList.add(student);
    }
    db.close();
    return studentList;
}

```

READ ALL FROM DATABASE

```
// Updating single Student
public int updateStudent(Student student) {
    SQLiteDatabase db = mDbHelper.getReadableDatabase();
    // New value for two columns
    ContentValues values = new ContentValues();
    values.put(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_NAME, student.getStudent_name());
    values.put(UnipiDbContract.StudentEntry.COLUMN_NAME_STUDENT_TEL, student.getStudent_tel());
    // Which row to update, based on the ID
    String selection = UnipiDbContract.StudentEntry._ID + " =?";
    String[] selectionArgs = { String.valueOf(student.getStudent_id()) };
    int count = db.update(
        UnipiDbContract.StudentEntry.TABLE_NAME,
        values,
        selection,
        selectionArgs);
    db.close();
    return count;
}
```

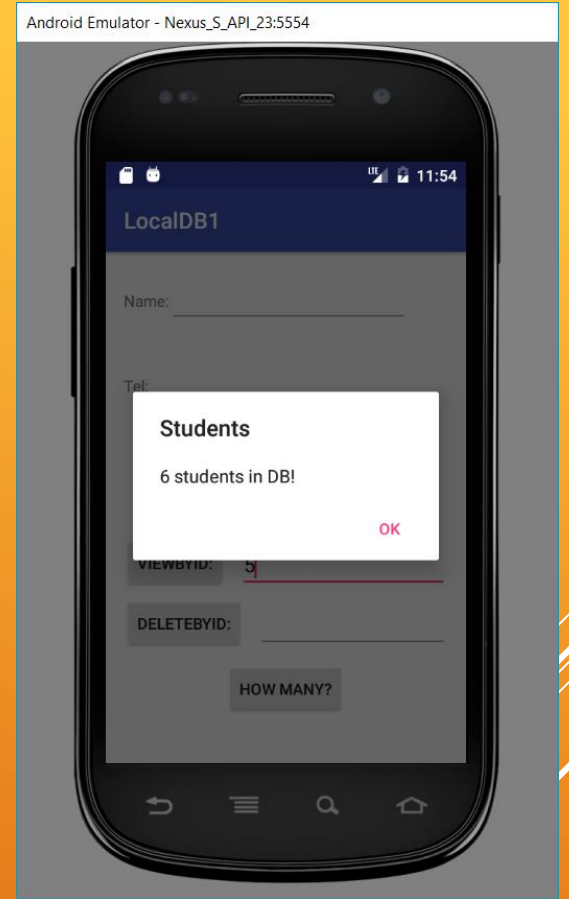
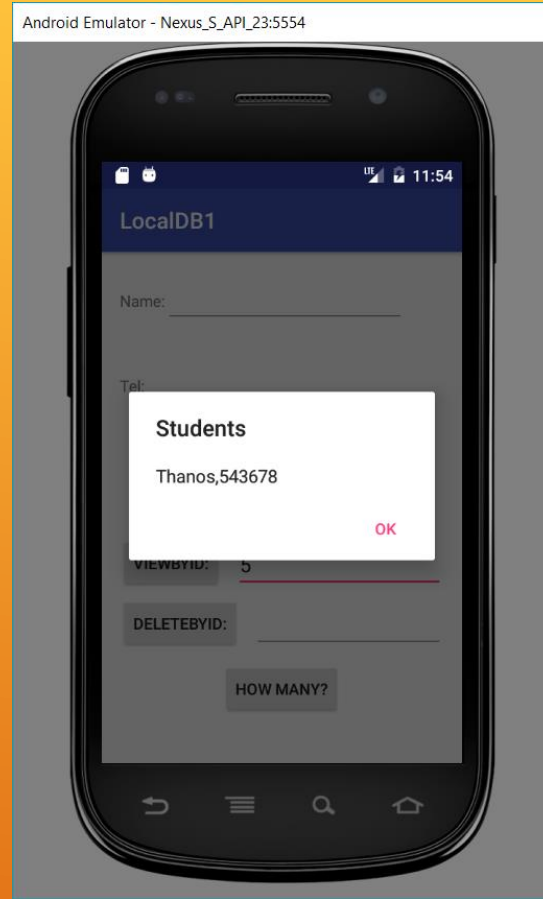
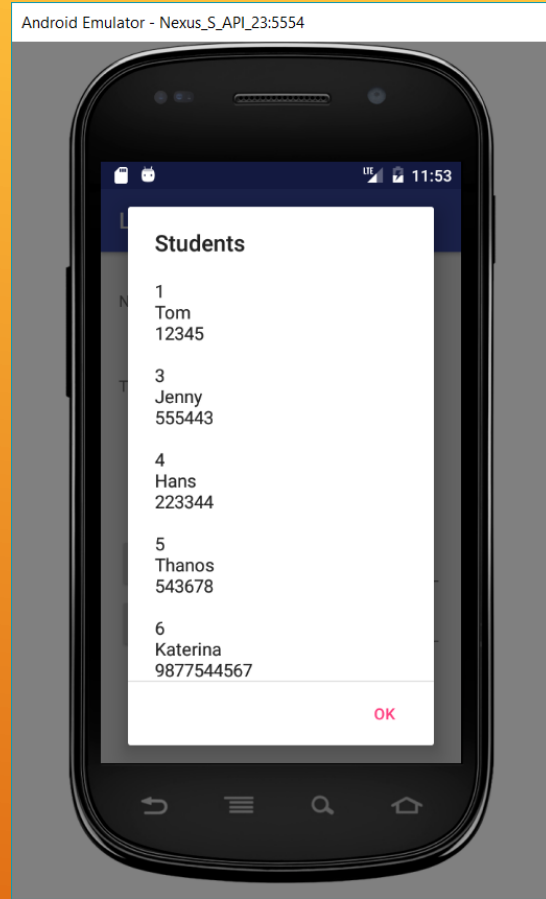
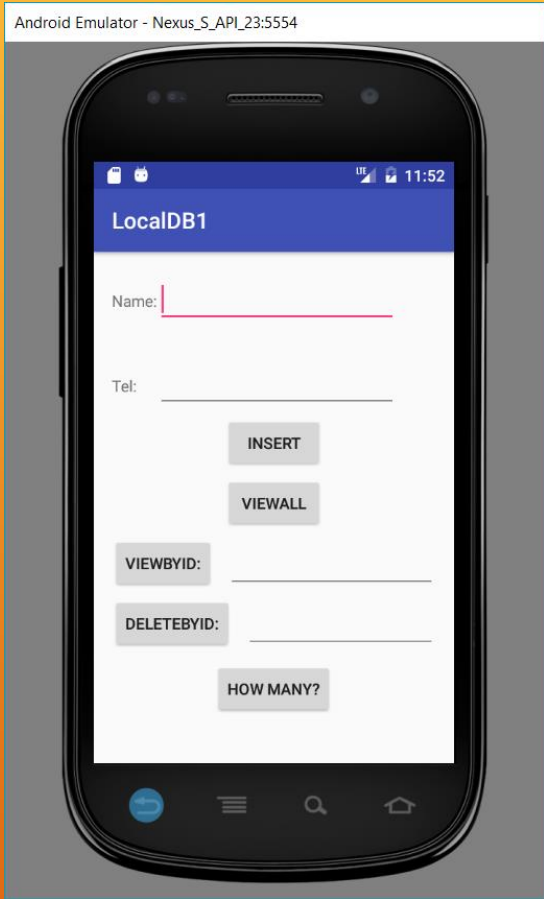
UPDATE DATABASE

```
// Deleting single Student
public int deleteStudent(Student student) {
    SQLiteDatabase db = mDbHelper.getReadableDatabase();
    // Which row to delete, based on the ID
    String selection = UnipiDbContract.StudentEntry._ID + "=?";
    String[] selectionArgs = { String.valueOf(student.getStudent_id()) };
    int count = db.delete(
        UnipiDbContract.StudentEntry.TABLE_NAME,
        selection,
        selectionArgs);
    db.close();
    return count;
}
```

DELETE FROM DATABASE

```
// Getting Students Count
public int getStudentsCount() {
    String countQuery = "SELECT * FROM " + UnipiDbContract.StudentEntry.TABLE_NAME;
    SQLiteDatabase db = mDbHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery(countQuery, null);
    int count = cursor.getCount();
    db.close();
    // return count
    return count;
}
```

COUNT RECORDS IN DATABASE
(THE OLD WAY)



BUILD THE APP