

Android SQLite Tutorial

Introduction:

The objective of this tutorial is to demonstrate and explain how to manipulate an SQLite database in the android development environment. It involves the presentation of the database in the UI as well as the functions of CRUD (create, read, update and delete) records in the app demo. The audiences are expected to know how to manage an SQLite database and present the desired output in the android development after studying this tutorial. A simple birth weight database is used as the example.

Assumption:

The audience should know the following knowledge in prior to this tutorial:

- Java programming language,
- Xml language,
- SQL query, and
- Some basic knowledge in android development like layout, event handlers, manipulate the elements in UI, intent, context and so on.

The flow:

We will go through the following steps for this tutorial.

Step 1 – Create the layout of MainActivity with elements

Step 2 – Create a database subclass which inherits SQLiteOpenHelper class

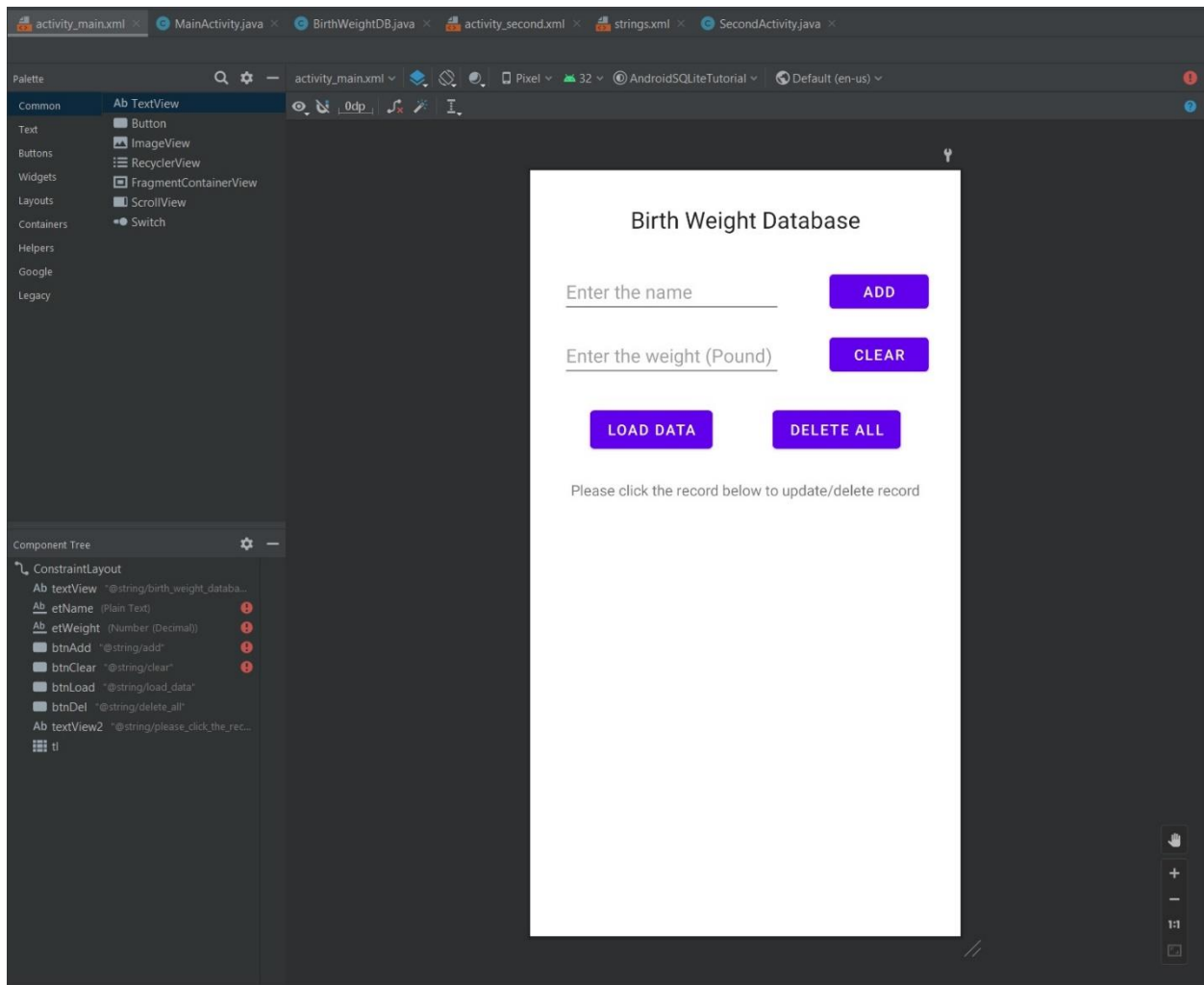
Step 3 – Declare and call the variables and methods in MainActivity.java for creating the database, adding records, and reading/displaying the records

Step 4 – Create the SecondActivity.java and its layout

Step 5 – Declare and call the variables and methods in SecondActivity.java for updating and deleting the records

Step 1 – Create the layout of MainActivity with elements

First of all, we design the layout of the MainActivity. There are several elements including 2 textviews for displaying messages, 2 edittexts for user input, 4 buttons for handling events and 1 tablelayout for displaying the records from the database. Here is the screenshot of the layout code and design.



```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×

1  <?xml version="1.0" encoding="utf-8"?>|
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:id="@+id/textView"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_marginTop="32dp"
14         android:text="@string/birth_weight_database"
15         android:textAppearance="@style/TextAppearance.AppCompat.Large"
16         app:layout_constraintEnd_toEndOf="parent"
17         app:layout_constraintStart_toStartOf="parent"
18         app:layout_constraintTop_toTopOf="parent" />
19
20      <EditText
21         android:id="@+id/etName"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:layout_marginTop="32dp"
25         android:layout_marginEnd="8dp"
26         android:ems="10"
27         android:hint="@string/enter_the_name"
28         android:inputType="textPersonName"
29         app:layout_constraintEnd_toStartOf="@+id/btnAdd"
30         app:layout_constraintStart_toStartOf="parent"
31         app:layout_constraintTop_toBottomOf="@+id/textView" />
32
33      <EditText
34         android:id="@+id/etWeight"
35         android:layout_width="wrap_content"
36         android:layout_height="wrap_content"
37         android:layout_marginTop="16dp"
38         android:layout_marginEnd="8dp"
39         android:ems="10"
40         android:hint="@string/enter_the_weight_pound"
41         android:inputType="numberDecimal"
42         app:layout_constraintEnd_toStartOf="@+id/btnClear"
```

```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
42         app:layout_constraintEnd_toStartOf="@+id/btnClear"
43         app:layout_constraintStart_toStartOf="parent"
44         app:layout_constraintTop_toBottomOf="@+id/etName" />
45
46     <Button
47         android:id="@+id/btnAdd"
48         android:layout_width="95dp"
49         android:layout_height="44dp"
50         android:layout_marginStart="8dp"
51         android:layout_marginTop="32dp"
52         android:onClick="addData"
53         android:text="@string/add"
54         app:layout_constraintEnd_toEndOf="parent"
55         app:layout_constraintStart_toEndOf="@+id/etName"
56         app:layout_constraintTop_toBottomOf="@+id/textView" />
57
58     <Button
59         android:id="@+id/btnClear"
60         android:layout_width="95dp"
61         android:layout_height="44dp"
62         android:layout_marginStart="8dp"
63         android:layout_marginTop="16dp"
64         android:onClick="clear"
65         android:text="@string/clear"
66         app:layout_constraintEnd_toEndOf="parent"
67         app:layout_constraintStart_toEndOf="@+id/etWeight"
68         app:layout_constraintTop_toBottomOf="@+id/btnAdd" />
69
70     <Button
71         android:id="@+id/btnLoad"
72         android:layout_width="wrap_content"
73         android:layout_height="wrap_content"
74         android:layout_marginTop="24dp"
75         android:onClick="loadData"
76         android:text="@string/load_data"
77         app:layout_constraintEnd_toStartOf="@+id/btnDel"
78         app:layout_constraintHorizontal_chainStyle="spread"
79         app:layout_constraintStart_toStartOf="parent"
80         app:layout_constraintTop_toBottomOf="@+id/etWeight" />
81
```

```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
78      app:layout_constraintHorizontal_chainStyle="spread"
79      app:layout_constraintStart_toStartOf="parent"
80      app:layout_constraintTop_toBottomOf="@+id/etWeight" />
81
82      <Button
83          android:id="@+id/btnDel"
84          android:layout_width="wrap_content"
85          android:layout_height="wrap_content"
86          android:layout_marginTop="24dp"
87          android:onClick="delDB"
88          android:text="@string/delete_all"
89          app:layout_constraintEnd_toEndOf="parent"
90          app:layout_constraintStart_toEndOf="@+id/btnLoad"
91          app:layout_constraintTop_toBottomOf="@+id/etWeight" />
92
93      <TextView
94          android:id="@+id/textView2"
95          android:layout_width="wrap_content"
96          android:layout_height="wrap_content"
97          android:layout_marginTop="24dp"
98          android:text="@string/please_click_the_record_below_to_update_delete_record"
99          app:layout_constraintEnd_toEndOf="parent"
100         app:layout_constraintStart_toStartOf="parent"
101         app:layout_constraintTop_toBottomOf="@+id/btnLoad" />
102
103     <TableLayout
104         android:id="@+id/tl"
105         android:layout_width="match_parent"
106         android:layout_height="wrap_content"
107         android:layout_marginStart="32dp"
108         android:layout_marginTop="16dp"
109         android:layout_marginEnd="32dp"
110         app:layout_constraintEnd_toEndOf="parent"
111         app:layout_constraintHorizontal_bias="1.0"
112         app:layout_constraintStart_toStartOf="parent"
113         app:layout_constraintTop_toBottomOf="@+id/textView2" />
114
115 </androidx.constraintlayout.widget.ConstraintLayout>
```

Step 2 – Create a database subclass which inherits SQLiteOpenHelper class

A database subclass of SQLiteOpenHelper class is required to allow the app to execute database-related methods from SQLiteOpenHelper class. We name “BirthWeightDB” as an example of creating the database subclass.

After creating the class, we add several variables which will be used in the methods. They include some constant strings which represent the database table and column names, context, an int for saving the number of records, and the database variable itself, SQLiteDatabase variable.

Next, we create a constructor to initiate the database object in the MainActivity. Super is used due to the inheritance from the super class. There are two override methods which are onCreate() and onUpgrade(). We use db.execSQL() with SQL “CREATE TABLE” query in onCreate() to create the table when the database object is initiated. onUpgrade() is used when the developer wants to upgrade the database. Here is the code.

```
BirthWeightDB.java | activity_main.xml | MainActivity.java | activity_second.xml | SecondActivity.java
1 package com.example.lab3sqllitetutorial_300345759;
2
3 import ...
12
13 public class BirthWeightDB extends SQLiteOpenHelper {
14
15     //Set the database name and table for repetitive use
16     static final private String DB_NAME = "Birth_Weight";
17     static final private String DB_TABLE = "Weight_with_names";
18
19     //Create instance variables
20     Context context;
21     SQLiteDatabase birthWeightDB;
22     public static int count = 0;
23
24     //Create the constructor
25     public BirthWeightDB (Context ct){
26         super(ct, DB_NAME, factory: null, version: 1);
27         context = ct;
28     }
29
30     //Create the table when a database object is initiated
31     @Override
32     public void onCreate(SQLiteDatabase db){
33         db.execSQL("CREATE TABLE " + DB_TABLE + " (_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, Infant_Name TEXT, Infant_Weight TEXT)");
34     }
35
36     //Upgrade the database when the version is different
37     @Override
38     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
39         db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
40         onCreate(db);
41     }
42 }
```

The last part of this class is creating corresponding methods to manage the database. There are five methods which are named as `insertData()`, `loadData()`, `delAll()`, `update()`, `delData()`. Besides `loadData()`, other methods use `getWritableDatabase()` which does not accept SQL query with "SELECT" allows us to create/update/delete the records in the database. For `loadData()`, `getReadableDatabase()`, which accepts SQL query with "SELECT", is used to read the database.

Similar to `onCreate()`, we use `BirthWeightDB.execSQL()` with SQL queries to execute the queries in the methods besides `loadData()`. There are some alternative methods which does not require SQL query for `update()`, `delAll()` and `delData()`. These methods, which are included in the code as comments, are safer for developers to prevent SQL syntax mistakes.

For `loadData()`, we use `ArrayList<String[]>` return method to return the database records into an `ArrayList`. We firstly create an empty `ArrayList` and make `BirthWeightDB` readable by using `getReadableDatabase()`. We next create a cursor, `cr`, onto a view/table with desired data by using `rawQuery()` with SQL "SELECT" query. The numeric value of number of records is assigned to the variable `count` by using `cr.getCount()`. Then we use while loop with `cr.moveToNext()` to screen the data from the selected view/table. By using `cr.getInt()` and `cr.getString()`, we can assign the values of each field from records to `String` variable and the `ArrayList`. At last but not least, we need to close the cursor and return the `ArrayList`. Here is the code of the 5 methods.

```

42
43 //Create a method to insert/add a record with the values from parameters
44 public void insertData(String name, String weight){
45     //Make the database writable
46     birthWeightDB = getWritableDatabase();
47     //Add the record into the database by executing the SQL INSERT query
48     birthWeightDB.execSQL("INSERT INTO " + DB_TABLE + " (Infant_Name, Infant_Weight) VALUES (" + name + ", " + weight + ");");
49 }
50
51 //Create a method to return an ArrayList<String[]> with the records from the database
52 public ArrayList<String[]> loadData(){
53
54     ArrayList<String[]> record = new ArrayList<>();
55
56     //Make the database readable
57     birthWeightDB = getReadableDatabase();
58
59     //Create a cursor to read the table created by the SQL SELECT query
60     Cursor cr = birthWeightDB.rawQuery( sql: "SELECT * FROM " + DB_TABLE, selectionArgs: null);
61     count = cr.getCount();
62
63     //Read every record and store the corresponding values to the arrayList
64     while (cr.moveToNext()){
65         int id = cr.getInt( columnIndex: 0);
66         String idxStr = String.valueOf(id);
67         String nameStr = cr.getString( columnIndex: 1);
68         String weightStr = cr.getString( columnIndex: 2);
69         record.add(new String[]{idxStr, nameStr, weightStr});
70     }
71
72     //Close the cursor
73     cr.close();
74
75     //Return the arrayList
76     return record;
77 }
78

```



```

75 //Return the arraylist
76 return record;
77 }
78
79 //Create a method to delete the table/database
80 public void delAll(){
81     //Make the database writable
82     birthWeightDB = getWritableDatabase();
83     //Delete the table by executing the SQL DELETE query
84     birthWeightDB.execSQL("DELETE FROM " + DB_TABLE);
85
86     //Here is another non-SQL way to delete the database (not a table)
87
88     //context.deleteDatabase(DB_NAME);
89 }
90
91 //Create a method to update a record in a database with the values of parameters
92 public void update(String newName, String newWeight, int id){
93     //Make the database writable
94     birthWeightDB = getWritableDatabase();
95     //Update the record to the database by executing the SQL UPDATE query with WHERE clause
96     birthWeightDB.execSQL("UPDATE " + DB_TABLE + " SET Infant_Name = '" + newName + "', Infant_Weight = '"
97         + newWeight + "' WHERE _id=" + id + ";"");
98
99     //Here is another non-SQL way to update an record
100
101     //ContentValues values = new ContentValues();
102     //values.put("Infant_Name", newName);
103     //values.put("Infant_Weight", newWeight);
104     //birthWeightDB.update(DB_TABLE, values, "_id=?", new String[]{String.valueOf(id)});
105 }
106
107 //Create a method to delete a record with id
108 public void delData(int id){
109     //Make the database writable
110     birthWeightDB = getWritableDatabase();
111     //Delete the record to the database by executing the SQL DELETE query with WHERE clause
112     birthWeightDB.execSQL("DELETE FROM " + DB_TABLE + " WHERE _id=" + id + ";"");
113
114     //birthWeightDB.delete(DB_TABLE, "_id=?", new String[]{String.valueOf(id)});
115 }
116 }

```

Step 3 – Declare and call the variables and methods in MainActivity.java for creating the database, adding records, and reading/displaying the records

Now we develop the methods in the MainActivity.java to create and manipulate a concrete database. We create a number of variables including two edittexts, one tablelayout, one public static BirthWeightDB (it is accessible in the SecondActivity), one ArrayList (for storing the records from the database), and a public static int id (the id from the first column of our database). We assign the values to the edittexts, the tablelayout and create the concrete database, bwDB, in onCreate(). Since we want to read and load the database directly after jumping back from SecondActivity, the if statement in onCreate() will automatically read and load the database if the intent is not empty. Here is the code.

```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
1 package com.example.lab3sqllitetutorial_300345759;
2
3 import ...
15
16 public class MainActivity extends AppCompatActivity {
17
18     //Create view variables
19     private EditText etName;
20     private EditText etWeight;
21     private TableLayout tl;
22
23     //Create an ArrayList<String[]> to store the data from the database and display the data in the views
24     private ArrayList<String[]> outputAL;
25
26     //Create the database variable and int id which will be used across two activities
27     public static BirthWeightDB bwDB;
28     public static int id;
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         //Connect the view variables with ids
36         etName = findViewById(R.id.etName);
37         etWeight = findViewById(R.id.etWeight);
38         tl = findViewById(R.id.tl);
39
40         //Create a concrete database object
41         bwDB = new BirthWeightDB(this);
42
43         //To load and display the database when launching the MainActivity from SecondActivity
44         if(getIntent().getExtras() != null){
45             loadDB(tl);
46             bwDB.close();
47         }
48     }
```

Next we create several methods for button event handlers. They include `addData()`, `clear()`, `loadDB()`, and `delDB()`. After getting the input from the edittexts, we use `if` statement to check if the input is empty. If not, we call the `insertData()` from `BirthWeightDB` class with input as parameters to insert the input into the database. We empty the edittext and popup a message to confirm the user that the record is added successfully. We close the database at the end. It is simple for `clear()` which just clear the edittext fields for better user experience. Here is the code

```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
48 }
49
50 //Create a button event method to add a record to the database from user input
51 public void addData(View view) {
52     //Store the input from user
53     String nameInput = etName.getText().toString();
54     String weightInput = etWeight.getText().toString();
55
56     //Check if the input is empty
57     if(nameInput.equals("") || weightInput.equals("")){
58         //If empty then ask the user to enter the values
59         Toast.makeText(context, this, "Please enter the name and the weight.", Toast.LENGTH_LONG).show();
60     } else {
61         //If not empty, insert the input into the database, show a notification to user, clear the input fields and close the database
62         bwDB.insertData(nameInput, weightInput);
63         Toast.makeText(context, this, "Data Saved Successfully", Toast.LENGTH_LONG).show();
64         etName.setText("");
65         etWeight.setText("");
66         bwDB.close();
67     }
68 }
69
70 //Create a button event method to clear the input fields
71 public void clear(View view) {
72     etName.setText("");
73     etWeight.setText("");
74 }
75 }
```

The `loadDB()` method involves the longest codes in the app. We firstly call the `loadData()` from the `BirthWeightDB` class to read the database and store the records to our `ArrayList`. This method serves for several purposes beside just reading the records from database. It not only checks if the database is empty (the first part `if` statement) but also dynamically to display the records in the `tablelayout`. We remove all views from the `tablelayout` to prevent overlapping display. Then we create a `tablerow` with two `textviews` to display the table header and add them to the `tablelayout`. We use `for-loop` to create number of `tablerows` and `textviews` to display each record. We also use `setId()`, `setClickable()`, and `setOnClickListener()` in the loop to allow user to click each `tablerow` for modifying/deleting the record in the `SecondActivity`. Do not forget to add each `tablerow` into the `tablelayout`. At last, we clear the edittexts.

To activate the on click event on each table row, an `onClickListener` variable is needed in the `setOnClickListener()` aforementioned. We create an intent which allows the user to jump to the `SecondActivity`. Then we use the `Id` from the selected `tablerow` to refer the record from the `ArrayList`. The fields of the selected record will be stored into the intent by using `putExtra()`. Finally, we start the intent to jump to `SecondActivity`.

The last method, `delDB()`, is simple. We call the `delAll()` from the `BirthWeightDB` class to delete the database. Then we remove all the views from the `tablelayout` and pop up the message about the database deletion. Close the database and this is done. Here is the code.

```

75
76 //Create a button event method to read the database and display the records to user
77 public void loadDB(View view) {
78     //Read the database and store the values to the ArrayList
79     outputAL = bwDB.loadData();
80
81     //Check if the database is empty
82     if(bwDB.count == 0) {
83         //Clear the tablelayout
84         tl.removeAllViews();
85
86         //Create a textview to display a message about no data
87         TextView txvNoData = new TextView( context: this);
88         txvNoData.setText("There are no data yet.");
89         txvNoData.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
90         txvNoData.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT));
91
92         //Insert the textview into the tablelayout
93         tl.addView(txvNoData);
94     } else {
95         //If the database is not empty
96         //Clear the tablelayout
97         tl.removeAllViews();
98
99         //Create a table row to show the headers of records
100         TableRow tr_head = new TableRow( context: this);
101
102         //Create two textviews for the two headers
103         TextView txvNameHeader = new TextView( context: this);
104         txvNameHeader.setText("Infant Name");
105         txvNameHeader.setTextSize(16);
106         txvNameHeader.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT, initWeight: 4f));
107
108         TextView txvWeightHeader = new TextView( context: this);
109         txvWeightHeader.setText("Weight (lbs)");
110         txvWeightHeader.setTextSize(16);
111         txvWeightHeader.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT, initWeight: 1f));
112
113         //Insert the two textviews into the table row
114         tr_head.addView(txvNameHeader);
115         tr_head.addView(txvWeightHeader);
116
117         //Insert the table row to the tablelayout
118         tl.addView(tr_head);

```

```

119
120 //Using for-loop to display every records from the database
121 for(int i = 0; i < bwDB.count; i++){
122     //Create a table row to show the data of each record
123     TableRow tr = new TableRow( context: this);
124
125     //Create two textviews for the fields of each record
126     TextView txv1 = new TextView( context: this);
127     //Read the first field from the ArrayList. ([0] is the id column)
128     String s1 = outputAL.get(i)[1];
129     txv1.setText(s1);
130     txv1.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT, initWeight: 4f));
131
132     TextView txv2 = new TextView( context: this);
133     //Read the second field from the ArrayList
134     String s2 = outputAL.get(i)[2];
135     txv2.setText(s2);
136     txv2.setLayoutParams(new TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT, TableRow.LayoutParams.WRAP_CONTENT, initWeight: 1f));
137
138     //Insert the two textviews into each table row
139     tr.addView(txv1);
140     tr.addView(txv2);
141
142     //Set Id and make each row to be clickable with events
143     tr.setId(i);
144     tr.setClickable(true);
145     tr.setOnClickListener(trOnClickListener);
146
147     //Insert each table row to the tablelayout
148     tl.addView(tr);
149 }
150
151 //Clear the input fields
152 etName.setText("");
153 etWeight.setText("");
154 }
155 }
156

```

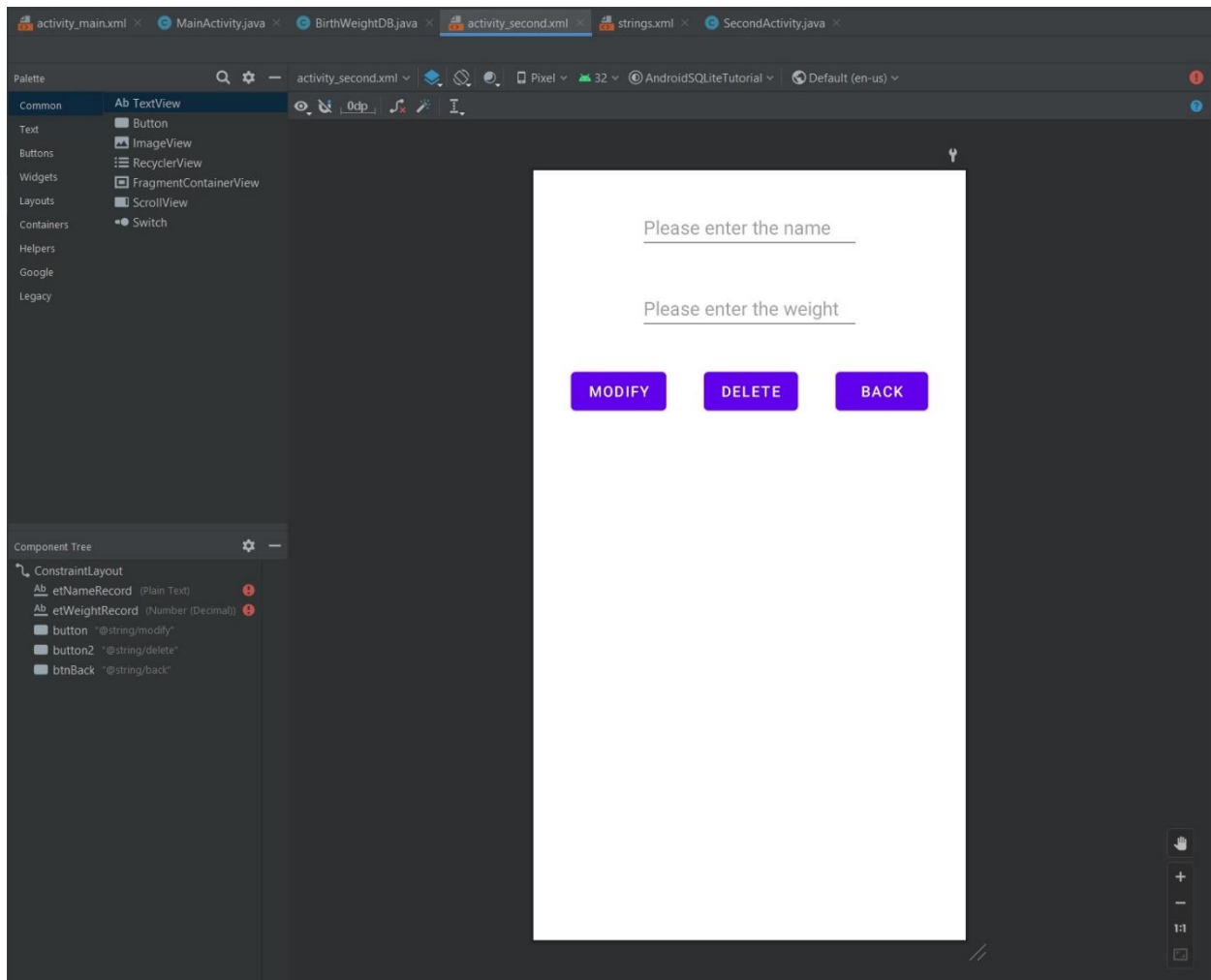
```

BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
156
157 //Create an onClickListener to execute the followings when the user clicks the table row
158 public View.OnClickListener trOnClickListener = new View.OnClickListener(){
159     @Override
160     public void onClick(View v) {
161
162         //Create an intent of SecondActivity
163         Intent intent1 = new Intent(v.getContext(), SecondActivity.class);
164
165         //Refer which table row is clicked
166         int idx = v.getId();
167         //Get the id from the ArrayList for modifying/deleting a record in the database
168         id = Integer.parseInt(outputAL.get(idx)[0]);
169
170         //Get the field values of the selected table row for user to recognize which row is selected
171         String s1 = outputAL.get(idx)[1];
172         String s2 = outputAL.get(idx)[2];
173
174         //Store the values into the intent
175         intent1.putExtra( name: "id", id);
176         intent1.putExtra( name: "name", s1);
177         intent1.putExtra( name: "weight", s2);
178
179         //Jump to SecondActivity
180         startActivity(intent1);
181     }
182 };
183
184 //Create a button event method to delete the table/database
185 public void delDB(View view) {
186     //Delete the table/database
187     bwDB.delAll();
188     //Clear the tablelayout
189     tl.removeAllViews();
190     //Notify the user that the table/database is deleted
191     Toast.makeText( context: this, text: "All data is erased.", Toast.LENGTH_LONG).show();
192     //Close the database
193     bwDB.close();
194 }
195 }

```

Step 4 – Create the SecondActivity.java and its layout

We create a new empty activity and called it SecondActivity for allowing user to modify/delete a certain record. The layout of the SecondActivity is much simpler. There are 2 edittexts and 3 buttons only. Here is the screenshot of the layout code and design.



BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".SecondActivity">
8
9     <EditText
10         android:id="@+id/etNameRecord"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_marginTop="32dp"
14         android:ems="10"
15         android:hint="@string/please_enter_the_name"
16         android:inputType="textPersonName"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent" />
20
21     <EditText
22         android:id="@+id/etWeightRecord"
23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:layout_marginTop="32dp"
26         android:ems="10"
27         android:hint="@string/please_enter_the_weight"
28         android:inputType="numberDecimal"
29         app:layout_constraintEnd_toEndOf="parent"
30         app:layout_constraintStart_toStartOf="parent"
31         app:layout_constraintTop_toBottomOf="@+id/etNameRecord" />
32
```



```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
33 <Button
34     android:id="@+id/button"
35     android:layout_width="wrap_content"
36     android:layout_height="wrap_content"
37     android:layout_marginTop="32dp"
38     android:text="@string/modify"
39     android:onClick="updateRecord"
40     app:layout_constraintEnd_toStartOf="@+id/button2"
41     app:layout_constraintHorizontal_bias="0.5"
42     app:layout_constraintHorizontal_chainStyle="spread"
43     app:layout_constraintStart_toStartOf="parent"
44     app:layout_constraintTop_toBottomOf="@+id/etWeightRecord" />
45
46 <Button
47     android:id="@+id/button2"
48     android:layout_width="wrap_content"
49     android:layout_height="wrap_content"
50     android:layout_marginTop="32dp"
51     android:text="@string/delete"
52     android:onClick="delRecord"
53     app:layout_constraintEnd_toStartOf="@+id/btnBack"
54     app:layout_constraintHorizontal_bias="0.5"
55     app:layout_constraintStart_toEndOf="@+id/button"
56     app:layout_constraintTop_toBottomOf="@+id/etWeightRecord" />
57
58 <Button
59     android:id="@+id/btnBack"
60     android:layout_width="wrap_content"
61     android:layout_height="wrap_content"
62     android:layout_marginTop="32dp"
63     android:text="@string/back"
64     android:onClick="back"
65     app:layout_constraintEnd_toEndOf="parent"
66     app:layout_constraintHorizontal_bias="0.5"
67     app:layout_constraintStart_toEndOf="@+id/button2"
68     app:layout_constraintTop_toBottomOf="@+id/etWeightRecord" />
69
70 </androidx.constraintlayout.widget.ConstraintLayout>
```

Step 5 – Declare and call the variables and methods in SecondActivity.java for updating and deleting the records

Firstly, we declare a String, two edittext variables and another intent to jump back MainActivity. We then assign the corresponding values to the intent and edittext variables and set the text from the MainActivity intent by using getIntent().getStringExtra() in onCreate(). The String flag is used to store some data to show that the intent is not empty. It will trigger the loadData() in the onCreate() of MainActivity to load and display the data automatically. Here is the code.

```
BirthWeightDB.java × activity_main.xml × MainActivity.java × activity_second.xml × SecondActivity.java ×
1 package com.example.lab3sqllitetutorial_300345759;
2
3 import ...
10
11 public class SecondActivity extends AppCompatActivity {
12
13     //Create view variables
14     private EditText etNameRecord;
15     private EditText etWeightRecord;
16     //Create an intent and a String variable for directing to MainActivity
17     private Intent intent2;
18     private String flag;
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_second);
24
25         //Connect the view variables with id
26         etNameRecord = findViewById(R.id.etNameRecord);
27         etWeightRecord = findViewById(R.id.etWeightRecord);
28
29         //Get the field values from MainActivity
30         etNameRecord.setText(getIntent().getStringExtra( name: "name"));
31         etWeightRecord.setText(getIntent().getStringExtra( name: "weight"));
32
33         //Create an intent object of MainActivity
34         intent2 = new Intent( packageContext: this, MainActivity.class);
35     }
36
```

Next we create 3 methods for 3 button event handlers which are updateRecord(), delRecord(), and back(). After getting the input from edittext and checking if the input is empty, we call the update() from BirthWeightDB class of the concrete database created in the MainActivity to update the record within the same database. We use the MainActivity id to indicate which record is needed to be updated. Set and assign the flag to the intent then start the intent to

jump back to MainActivity. The delRecord() is similar to updateRecord. It just calls another method, delData() from the BirthWeightDB(). The back() is just used to implement the intent to MainActivity without manipulating the database. Here is the code.

```
BirthWeightDB.java x activity_main.xml x MainActivity.java x activity_second.xml x SecondActivity.java x
37 public void updateRecord(View view) {
38     //Store the input from user
39     String nameInput = etNameRecord.getText().toString();
40     String weightInput = etWeightRecord.getText().toString();
41
42     //Check if the input is empty
43     if(nameInput.equals("") || weightInput.equals("")){
44         Toast.makeText(context: this, text: "Please enter the name and the weight.", Toast.LENGTH_LONG).show();
45     } else {
46         //If not, update the selected record of the database in the MainActivity by calling the update() with parameters
47         MainActivity.bwDB.update(nameInput, weightInput, MainActivity.id);
48         //Notification to user
49         Toast.makeText(context: this, text: "1 record is modified successfully", Toast.LENGTH_LONG).show();
50     }
51
52     //Store flag into the intent to trigger the code in the onCreate() "if" statement in the MainActivity
53     flag = "update";
54     intent2.putExtra(name: "flag", flag);
55     //Go back to MainActivity
56     startActivity(intent2);
57 }
58
59 public void delRecord(View view) {
60     //Delete the selected record from the database in the MainActivity by calling the delData() with the id parameter
61     MainActivity.bwDB.delData(MainActivity.id);
62     Toast.makeText(context: this, text: "1 record is deleted successfully.", Toast.LENGTH_LONG).show();
63
64     //Store flag into the intent to trigger the code in the onCreate() "if" statement in the MainActivity
65     flag = "delete";
66     intent2.putExtra(name: "flag", flag);
67     //Go back to MainActivity
68     startActivity(intent2);
69 }
70
71 public void back(View view) {
72     //Store flag into the intent to trigger the code in the onCreate() "if" statement in the MainActivity
73     flag = "back";
74     intent2.putExtra(name: "flag", flag);
75     //Go back to MainActivity
76     startActivity(intent2);
77 }
78 }
```

CopyRight © by Herbert Lam