# Lab 14: Reading and Storing App data in to SQLite Databases

**Upon successful completion of this lab the students will be able to:**

- Create and Manage SQLite Databases.
- Query, insert, Update and delete records in SQLite Database.

## 14.1 Tools

- Android Studio (Version: 3.3 or higher)

## 14.2 Background

### 14.2.1 Database in Android

Android provides structured data persistence through a combination of SQLite databases and Content Providers. SQLite databases can be used to store application data using a managed, structured approach. Android offers a full SQLite relational database library. Every application can create its own databases over which it has complete control.

Having created your underlying data store, Content Providers offer a generic, well-defined interface for using and sharing data that provides a consistent abstraction from the underlying data source.

### 14.2.2 SQLite Databases

SQLite is an open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. To access this database, you don't need to establish any kind of connections for it like JDBC, ODBC etc.

SQLite is a well-regarded relational database management system (RDBMS). It is:

- Open-source
- Standards-compliant
- Lightweight
- Single-tier

It has been implemented as a compact C library that's included as part of the Android software stack. By being implemented as a library, rather than running as a separate ongoing process, each SQLite database is an integrated part of the application that created it. This reduces external dependencies, minimizes latency, and simplifi es transaction locking and synchronization.

SQLite has a reputation for being extremely reliable and is the database system of choice for many consumer electronic devices, including many MP3 players and smartphones.

Lightweight and powerful, SQLite differs from many conventional database engines by loosely

typing each column, meaning that column values are not required to conform to a single type; instead, each value is typed individually in each row. As a result, type checking isn't necessary when assigning or extracting values from each column within a row.

### 14.2.3 Content Values and Cursors

Android fragments have their own life cycle very similar to an android activity. This section briefs different stages of its life cycle.

Content Values are used to insert new rows into tables. Each ContentValues object represents a single table row as a map of column names to values.

Database queries are returned as Cursor objects. Rather than extracting and returning a copy of the result values, Cursors are pointers to the result set within the underlying data. Cursors provide a managed way of controlling your position (row) in the result set of a database query.

The Cursor class includes a number of navigation functions, including, but not limited to, the following:

- moveToFirst — Moves the cursor to the fi rst row in the query result
- moveToNext — Moves the cursor to the next row
- moveToPrevious — Moves the cursor to the previous row
- getCount — Returns the number of rows in the result set
- getColumnIndexOrThrow — Returns the zero-based index for the column with the specified name (throwing an exception if no column exists with that name)
- getColumnName — Returns the name of the specified column index
- getColumnNames — Returns a string array of all the column names in the current Cursor
- moveToPosition — Moves the cursor to the specified row
- getPosition — Returns the current cursor position

### 14.2.4 SQLiteOpenHelper

SQLiteOpenHelper is an abstract class used to implement the best practice pattern for creating, opening, and upgrading databases.

By implementing an SQLite Open Helper, you hide the logic used to decide if a database needs to be created or upgraded before it's opened, as well as ensure that each operation is completed efficiently.

Below code shows how to extend the SQLiteOpenHelper class by overriding the constructor, onCreate, and onUpgrade methods to handle the creation of a new database and upgrading to a new version, respectively.

```
private static class HoardDBOpenHelper extends SQLiteOpenHelper {

   private static final String DATABASE_NAME = "myDatabase.db";
   private static final String DATABASE_TABLE = "GoldHoards";
   private static final int DATABASE_VERSION = 1;

   // SQL Statement to create a new database.
   private static final String DATABASE_CREATE = "create table " +
   DATABASE_TABLE + " (" + KEY_ID +
   " integer primary key autoincrement, " +
   KEY_GOLD_HOARD_NAME_COLUMN + " text not null, " +
   KEY_GOLD_HOARDED_COLUMN + " float, " +
   KEY_GOLD_HOARD_ACCESSIBLE_COLUMN + " integer);";

   public HoardDBOpenHelper(Context context, String name,
   CursorFactory factory, int version) {
      super(context, name, factory, version);
   }

   // Called when no database exists in disk and the helper class needs
   // to create a new one.

   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL(DATABASE_CREATE);
   }

   // Called when there is a database version mismatch meaning that
   // the version of the database on disk needs to be upgraded to
   // the current version.

   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion,
   int newVersion) {
      // Log the version upgrade.
      Log.w("TaskDBAdapter", "Upgrading from version " +
      oldVersion + " to " +
      newVersion + ", which will destroy all old data");
      // Upgrade the existing database to conform to the new
      // version. Multiple previous versions can be handled by
      // comparing oldVersion and newVersion values.
      // The simplest case is to drop the old table and create a new one.
      db.execSQL("DROP TABLE IF IT EXISTS " + DATABASE_TABLE);
      // Create a new one.
      onCreate(db);
   }
}
```

To access a database using the SQLite Open Helper, call getWritableDatabase or getReadableDatabase to open and obtain a writable or read-only instance of the underlying database, respectively. Behind the scenes, if the database doesn't exist, the helper executes its onCreate handler.

### 14.2.5 Querying a Database

Each database query is returned as a Cursor. This lets Android manage resources more efficiently by retrieving and releasing row and column values on demand.

To execute a query on a Database object, use the query method, passing in the following:

- An optional Boolean that specifi es if the result set should contain only unique values.

- The name of the table to query.
- A projection, as an array of strings, that lists the columns to include in the result set.
- A where clause that defines the rows to be returned. You can include ? wildcards that will be replaced by the values passed in through the selection argument parameter.
- An array of selection argument strings that will replace the ? wildcards in the where clause.
- A group by clause that defines how the resulting rows will be grouped.
- A having clause that defines which row groups to include if you specified a group by clause.
- A string that describes the order of the returned rows.
- A string that defines the maximum number of rows in the result set.

```
// Specify the result column projection. Return the minimum set
// of columns required to satisfy your requirements.

String[] result_columns = new String[] {
KEY_ID, KEY_GOLD_HOARD_ACCESSIBLE_COLUMN, KEY_GOLD_HOARDED_COLUMN };
// Specify the where clause that will limit our results.
String where = KEY_GOLD_HOARD_ACCESSIBLE_COLUMN + "=" + 1;
// Replace these with valid SQL statements as necessary.

String whereArgs[] = null;
String groupBy = null;
String having = null;
String order = null;

SQLiteDatabase db = hoardDBOpenHelper.getWritableDatabase();
Cursor cursor = db.query(HoardDBOpenHelper.DATABASE_TABLE,
result_columns, where,
whereArgs, groupBy, having, order);
```

### 14.2.6 Adding, Updating, and Removing Rows

The SQLiteDatabase class exposes insert, delete, and update methods that encapsulate the SQL statements required to perform these actions. Additionally, the execSQL method lets you execute any valid SQL statement on your database tables, should you want to execute these (or any other) operations manually.

Any time you modify the underlying database values, you should update your Cursors by running a new query.

#### 14.2.6.1 Inserting Rows

```
// Create a new row of values to insert.
ContentValues newValues = new ContentValues();

// Assign values for each row.
newValues.put(KEY_GOLD_HOARD_NAME_COLUMN, hoardName);
newValues.put(KEY_GOLD_HOARDED_COLUMN, hoardValue);
newValues.put(KEY_GOLD_HOARD_ACCESSIBLE_COLUMN, hoardAccessible);

// [ ... Repeat for each column / value pair ... ]
// Insert the row into your table
SQLiteDatabase db = hoardDBOpenHelper.getWritableDatabase();
db.insert(HoardDBOpenHelper.DATABASE_TABLE, null, newValues);
```

#### 14.2.6.2 Updating Rows

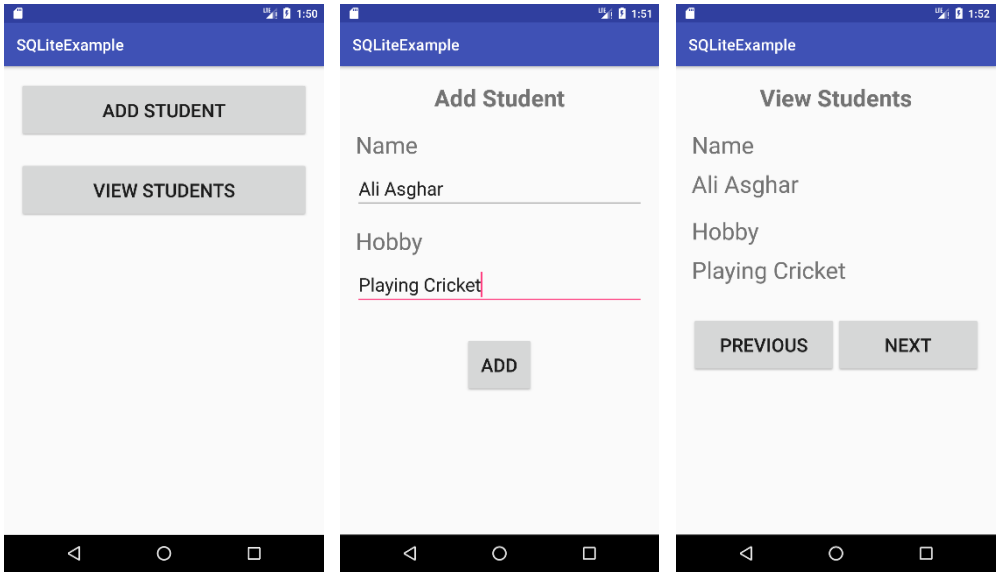
```
// Create the updated row Content Values.
ContentValues updatedValues = new ContentValues();

// Assign values for each row.
updatedValues.put(KEY_GOLD_HOARDED_COLUMN, newHoardValue);
// [ ... Repeat for each column to update ... ]
// Specify a where clause the defines which rows should be
// updated. Specify where arguments as necessary.
String where = KEY_ID + "=" + hoardId;
String whereArgs[] = null;
// Update the row with the specified index with the new values.
SQLiteDatabase db = hoardDBOpenHelper.getWritableDatabase();
db.update(HoardDBOpenHelper.DATABASE_TABLE, updatedValues,
where, whereArgs);
```

#### 14.2.6.3 Deleting Rows

```
// Specify a where clause that determines which row(s) to delete.
// Specify where arguments as necessary.
String where = KEY_GOLD_HOARDED_COLUMN + "=" + 0;
String whereArgs[] = null;
// Delete the rows that match the where clause.
SQLiteDatabase db = hoardDBOpenHelper.getWritableDatabase();
db.delete(HoardDBOpenHelper.DATABASE_TABLE, where, whereArgs);
```

## 14.3 Procedure

Example: Creating Database App to add and view students records.




Step 01: Create new android project.

Step 02: Write the Application name "SQLiteExample" for new Android Studio Project.

Step 03: Choose Target Android Device "Phone and Tablet".

Step 04: Choose Minimum SDK "IceCreamSandwich" and press Next.

Step 05: Select Empty Activity and press Next.

Step 06: Set the activity name "MainActivity" and press Finish.

Step 07: Create a new class DBHelper and extend it from SQLiteOpenHelper.

Step 08: Write the following code in to DBHelper.java file:

```
package com.example.ali.sqliteexample;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by Ali on 01-Oct-17.
 */

public class DBHelper extends SQLiteOpenHelper{

    private static String DBNAME = "StudentDB.db";
    private static int DB_VERSION = 1;

    public DBHelper(Context context){
        super(context, DBNAME,null,DB_VERSION);
    }

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

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

Step 09: Write the following code in to activity_main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context="com.example.ali.sqliteexample.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:text="Add Student"
        android:onClick="onAddStudent"
        android:textSize="24sp" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:padding="20dp"
        android:text="VIew Students"
        android:text="VIew Students"
        android:onClick="onViewStudents"
        android:textSize="24sp" />
```

Step 10: Write the following code in to MainActivity.java file:

```
package com.example.ali.sqliteexample;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onAddStudent(View v){
        Intent i = new Intent(this,AddStudentActivity.class);
        startActivity(i);
    }

    public void onViewStudents(View v) {
        Intent i = new Intent(this, ViewStudentsActivity.class);
        startActivity(i);
    }
}
```

Step 11: Add an Empty activity as AddStudentActivity.

Step 12: Write the following code in to activity_add_student.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context="com.example.ali.sqliteexample.AddStudentActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Add Student"
        android:textAlignment="center"
        android:textSize="30sp"
        android:textStyle="bold" />
```

```
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Name"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/etName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:ems="10"
        android:hint="Your name here"
        android:inputType="textPersonName"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Hobby"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/etHobby"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:ems="10"
        android:hint="Your hobby here"
        android:inputType="textPersonName"
        android:textSize="24sp" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="40dp"
        android:padding="20dp"
        android:text="Add"
        android:onClick="onAddStudent"
        android:textAlignment="center"
        android:textSize="24sp" />

</LinearLayout>
```

Step 13: Write the following code in to AddStudentActivity.java file:

```
package com.example.ali.sqliteexample;

import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
```

```
public class AddStudentActivity extends AppCompatActivity {

    DBHelper helper;
    SQLiteDatabase db;

    EditText etName;
    EditText etHobby;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_student);

        helper = new DBHelper(this);
        etName = (EditText)findViewById(R.id.etName);
        etHobby = (EditText)findViewById(R.id.etHobby);

    }

    public void onAddStudent(View v){

        db = helper.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put("name",etName.getText().toString());
        values.put("hobby",etHobby.getText().toString());

        db.insert("Student",null,values);
    }
}
```

Step 14: Add an Empty activity as ViewStudentsActivity.

Step 15: Write the following code in to activity_view_students.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context="com.example.ali.sqliteexample.ViewStudentsActivity">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="View Students"
        android:textAlignment="center"
        android:textSize="30sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textView5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Name"
        android:textSize="30sp" />
```

```
    <TextView
        android:id="@+id/tvName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:textSize="30sp" />

    <TextView
        android:id="@+id/textView7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Hobby"
        android:textSize="30sp" />

    <TextView
        android:id="@+id/tvHobby"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:textSize="30sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="40dp"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button5"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="onPrevioous"
            android:padding="20dp"
            android:text="Previous"
            android:textSize="24sp" />

        <Button
            android:id="@+id/button4"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="onNext"
            android:padding="20dp"
            android:text="Next"
            android:textSize="24sp" />
    </LinearLayout>

</LinearLayout>
```

Step 16: Write the following code in to ViewStudentsActivity.java file:

```
package com.example.ali.sqliteexample;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class ViewStudentsActivity extends AppCompatActivity {

    DBHelper helper;
    SQLiteDatabase db;
    Cursor c;

    TextView tvName;
    TextView tvHobby;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_students);

        helper = new DBHelper(this);
        tvName = (TextView)findViewById(R.id.tvName);
        tvHobby = (TextView)findViewById(R.id.tvHobby);

        db = helper.getReadableDatabase();
        c = db.rawQuery("SELECT name, hobby FROM Student",null);
        c.moveToFirst();

        tvName.setText(c.getString(0));
        tvHobby.setText(c.getString(1));
    }

    public void onNext(View v){
        if(c.moveToNext()){
            tvName.setText(c.getString(0));
            tvHobby.setText(c.getString(1));
        }
    }

    public void onPrevioous(View v){
        if(c.moveToPrevious()){
            tvName.setText(c.getString(0));
            tvHobby.setText(c.getString(1));
        }
    }
}
```

# EXERCISE

Exercise 14.1: Implement the following Android App activities using SQLite database to store and read records of students.: