CONTENTS

Android SQLite

// TUTORIAL //

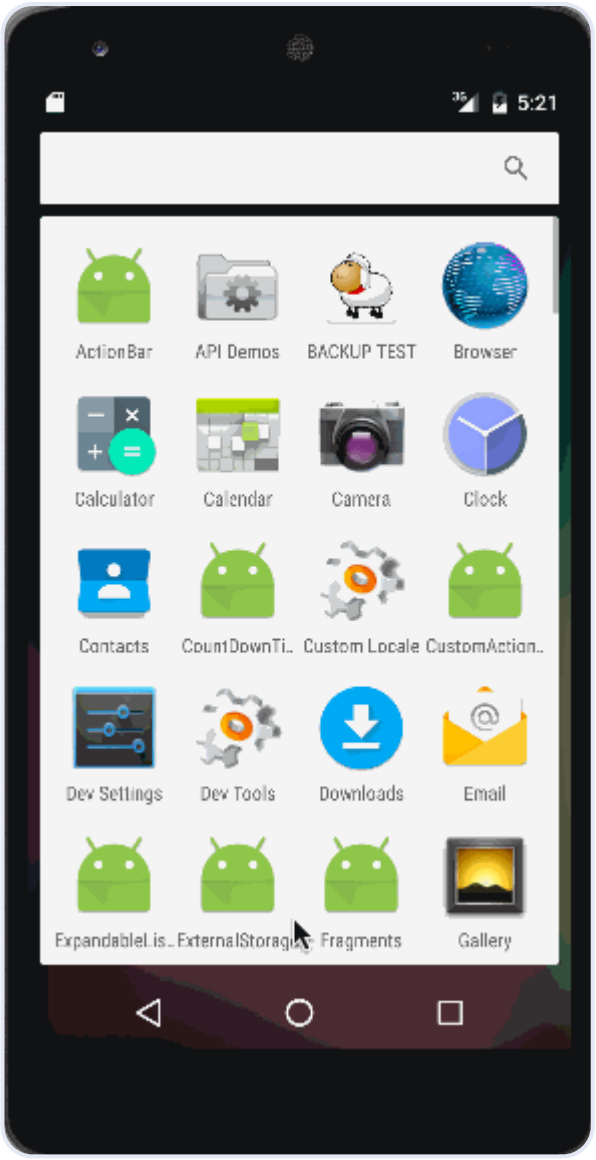
Android SQLite Database Example Tutorial

Pu n August 4, 2022

Anupam Chugh

Welcome to Android SQLite Example Tutorial. Android SQLite is the mostly preferred way to store data for android applications. For many applications, SQLite is the apps backbone whether it’s used directly or via some third-party wrapper. Below is the final

app we will create today using Android SQLite database.



Android SQLite

Android SQLite is a very lightweight database which comes with Android OS. Android SQLite combines a clean SQL interface with a very small memory footprint and decent speed. For Android, SQLite is “baked into” the Android runtime, so every Android application can create its own SQLite databases. Android SQLite native API is not JDBC, as JDBC might be too much overhead for a memory-limited smartphone. Once a database is created successfully its located in **data/data//databases/** accessible from Android Device Monitor. SQLite is a typical **relational database**, containing tables (which consists of rows and columns), indexes etc. We can create our own tables to hold the data accordingly. This structure is referred to as a **schema**.

Android SQLite SQLiteOpenHelper

Android has features available to handle changing database schemas, which mostly depend on using the `SQLiteOpenHelper` class. **SQLiteOpenHelper** is designed to get rid of

two very common problems.

1. When the application runs the first time - At this point, we do not yet have a database. So we will have to create the tables, indexes, starter data, and so on.
2. When the application is upgraded to a newer schema - Our database will still be on the old schema from the older edition of the app. We will have option to alter the database schema to match the needs of the rest of the app.

`SQLiteOpenHelper` wraps up these logic to create and upgrade a database as per our specifications. For that we'll need to create a custom subclass of `SQLiteOpenHelper` implementing at least the following three methods.

1. **Constructor** : This takes the Context (e.g., an Activity), the name of the database, an optional cursor factory (we'll discuss this later), and an integer representing the version of the database schema you are using (typically starting from 1 and increment later).

```
public DatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }
```

2. **onCreate(SQLiteDatabase db)** : It's called when there is no database and the app needs one. It passes us a `SQLiteDatabase` object, pointing to a newly-created database, that we can populate with tables and initial data.
3. **onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)** : It's called when the schema version we need does not match the schema version of the database, It passes us a **SQLiteDatabase** object and the old and new version numbers. Hence we can figure out the best way to convert the database from the old schema to the new one.

We define a `DBManager` class to perform all database CRUD(Create, Read, Update and Delete) operations.

Opening and Closing Android SQLite Database Connection

Before performing any database operations like insert, update, delete records in a table, first open the database connection by calling **getWritableDatabase()** method as shown below:

```
public DBManager open() throws SQLException {
        dbHelper = new DatabaseHelper(context);
        database = dbHelper.getWritableDatabase();
```

```
        return this;
    }
```

The **dbHelper** is an instance of the subclass of `SQLiteOpenHelper`. To close a database connection the following method is invoked.

```
public void close() {
        dbHelper.close();
    }
```

Inserting new Record into Android SQLite database table

The following code snippet shows how to insert a new record in the android SQLite database.

```
public void insert(String name, String desc) {
        ContentValues contentValue = new ContentValues();
        contentValue.put(DatabaseHelper.SUBJECT, name);
        contentValue.put(DatabaseHelper.DESC, desc);
        database.insert(DatabaseHelper.TABLE_NAME, null, contentValue);
    }
```

Content Values creates an empty set of values using the given initial size. We'll discuss the other instance values when we jump into the coding part.

Updating Record in Android SQLite database table

The following snippet shows how to update a single record.

```
public int update(long _id, String name, String desc) {
        ContentValues contentValues = new ContentValues();
        contentValues.put(DatabaseHelper.SUBJECT, name);
        contentValues.put(DatabaseHelper.DESC, desc);
        int i = database.update(DatabaseHelper.TABLE_NAME, contentValues, DatabaseHel
        return i;
    }
```

Android SQLite - Deleting a Record

We just need to pass the id of the record to be deleted as shown below.

```
public void delete(long _id) {
        database.delete(DatabaseHelper.TABLE_NAME, DatabaseHelper._ID + "=" + _id, nu
    }
```

Android SQLite Cursor

A Cursor represents the entire result set of the query. Once the query is fetched a call to **cursor.moveToFirst()** is made. Calling moveToFirst() does two things:

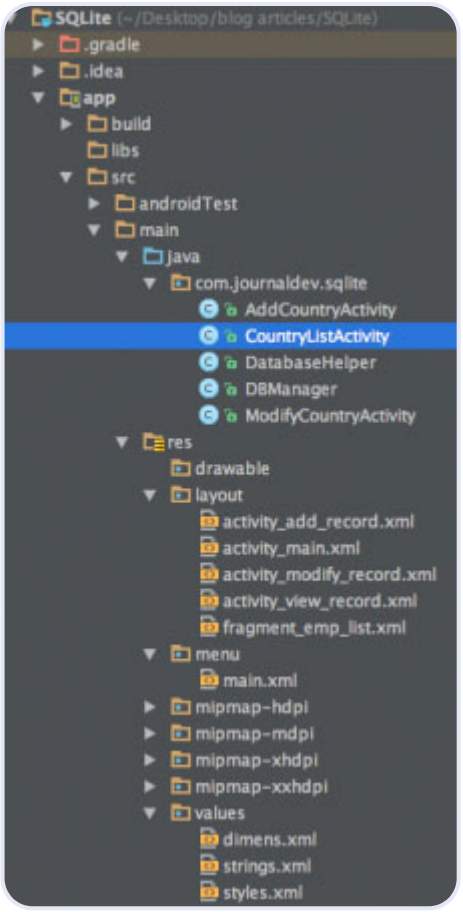
- It allows us to test whether the query returned an empty set (by testing the return value)
- It moves the cursor to the first result (when the set is not empty)

The following code is used to fetch all records:

```
public Cursor fetch() {
        String[] columns = new String[] { DatabaseHelper._ID, DatabaseHelper.SUBJECT,
        Cursor cursor = database.query(DatabaseHelper.TABLE_NAME, columns, null, null
        if (cursor != null) {
            cursor.moveToFirst();
        }
        return cursor;
    }
```

Another way to use a Cursor is to wrap it in a `CursorAdapter`. Just as `ArrayAdapter` adapts arrays, `CursorAdapter` adapts Cursor objects, making their data available to an `AdapterView` like a `ListView`. Let's jump to our project that uses SQLite to store some meaningful data.

Android SQLite Example Project Structure



In this application we wish to create records that store Country names and their respective currencies in the form of a **ListView**. We cover all the features discusses above.

Android SQLite Project Code

The application consists of 5 classes. We begin with defining with **DatabaseHelper**, which is a subclass of SQLiteOpenHelper as follows: `DatabaseHelper.java`

```
package com.journaldev.sqlite;

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

public class DatabaseHelper extends SQLiteOpenHelper {

    // Table Name
    public static final String TABLE_NAME = "COUNTRIES";

    // Table columns
    public static final String _ID = "_id";
    public static final String SUBJECT = "subject";
    public static final String DESC = "description";

    // Database Information
    static final String DB_NAME = "JOURNALDEV_COUNTRIES.DB";
```

```
    // database version
    static final int DB_VERSION = 1;

    // Creating table query
    private static final String CREATE_TABLE = "create table " + TABLE_NAME + "(" + _
            + " INTEGER PRIMARY KEY AUTOINCREMENT, " + SUBJECT + " TEXT NOT NULL, " +

    public DatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

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

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

As discussed above we have overridden the `onCreate()` and `onUpgrade()` methods besides the constructor. We've assigned the names to the database and the table as JOURNALDEV_COUNTRIES.DB and COUNTRIES respectively. The index column is auto incremented whenever a new row is inserted. The column names for country and currency are "subject" and "description". The DBManager classes is where the DatabaseHelper is initialized and the CRUD Operations are defined. Below is the code for this class: `DBManager.java`

```
package com.journaldev.sqlite;


import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class DBManager {

    private DatabaseHelper dbHelper;

    private Context context;

    private SQLiteDatabase database;
```

```
    public DBManager(Context c) {
        context = c;
    }

    public DBManager open() throws SQLException {
        dbHelper = new DatabaseHelper(context);
        database = dbHelper.getWritableDatabase();
        return this;
    }

    public void close() {
        dbHelper.close();
    }

    public void insert(String name, String desc) {
        ContentValues contentValue = new ContentValues();
        contentValue.put(DatabaseHelper.SUBJECT, name);
        contentValue.put(DatabaseHelper.DESC, desc);
        database.insert(DatabaseHelper.TABLE_NAME, null, contentValue);
    }

    public Cursor fetch() {
        String[] columns = new String[] { DatabaseHelper._ID, DatabaseHelper.SUBJECT,
        Cursor cursor = database.query(DatabaseHelper.TABLE_NAME, columns, null, null
        if (cursor != null) {
            cursor.moveToFirst();
        }
        return cursor;
    }

    public int update(long _id, String name, String desc) {
        ContentValues contentValues = new ContentValues();
        contentValues.put(DatabaseHelper.SUBJECT, name);
        contentValues.put(DatabaseHelper.DESC, desc);
        int i = database.update(DatabaseHelper.TABLE_NAME, contentValues, DatabaseHel
        return i;
    }

    public void delete(long _id) {
        database.delete(DatabaseHelper.TABLE_NAME, DatabaseHelper._ID + "=" + _id, nu
    }

}
```

The `CountryListActivity.java` class is the activity which is launched when the application starts. Below is layout defined for it: `fragment_emp_list.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="https://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent" >

    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="1dp"
        android:padding="10dp" >
    </ListView>

    <TextView
        android:id="@+id/empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="@string/empty_list_text" />

</RelativeLayout>
```

Here a ListView component is defined to included the records stored in the database. Initially the ListView would be empty hence a TextView is used to display the same.

`CountryListActivity.java`

```
package com.journaldev.sqlite;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.support.v4.widget.SimpleCursorAdapter;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;

public class CountryListActivity extends ActionBarActivity {

    private DBManager dbManager;

    private ListView listView;

    private SimpleCursorAdapter adapter;

    final String[] from = new String[] { DatabaseHelper._ID,
            DatabaseHelper.SUBJECT, DatabaseHelper.DESC };

    final int[] to = new int[] { R.id.id, R.id.title, R.id.desc };
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.fragment_emp_list);

        dbManager = new DBManager(this);
        dbManager.open();
        Cursor cursor = dbManager.fetch();

        listView = (ListView) findViewById(R.id.list_view);
        listView.setEmptyView(findViewById(R.id.empty));

        adapter = new SimpleCursorAdapter(this, R.layout.activity_view_record, cursor
        adapter.notifyDataSetChanged();

        listView.setAdapter(adapter);

        // OnCLickListiner For List Items
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, l
                TextView idTextView = (TextView) view.findViewById(R.id.id);
                TextView titleTextView = (TextView) view.findViewById(R.id.title);
                TextView descTextView = (TextView) view.findViewById(R.id.desc);

                String id = idTextView.getText().toString();
                String title = titleTextView.getText().toString();
                String desc = descTextView.getText().toString();

                Intent modify_intent = new Intent(getApplicationContext(), ModifyCour
                modify_intent.putExtra("title", title);
                modify_intent.putExtra("desc", desc);
                modify_intent.putExtra("id", id);

                startActivity(modify_intent);
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();
        if (id == R.id.add_record) {

            Intent add_mem = new Intent(this, AddCountryActivity.class);
```

```
            startActivity(add_mem);

        }
        return super.onOptionsItemSelected(item);
    }

}
```

In this activity the DBManager object is invoked to perform the CRUD Operations. A SimpleCursorAdapter is defined to add elements to the list from the query results that are returned in an Cursor Object. On list item click an intent is performed to open the ModifyCountryActivity class. The menu contains an item to add a new record from the ActionBar. Here again an intent is performed to open the AddCountryActivity class. Below is `menu.xml` code. `menu.xml`

```
<menu xmlns:android="https://schemas.android.com/apk/res/android"
    xmlns:app="https://schemas.android.com/apk/res-auto"
    xmlns:tools="https://schemas.android.com/tools"
    tools:context="com.example.sqlitesample.MainActivity" >

    <item
        android:id="@+id/add_record"
        android:icon="@android:drawable/ic_menu_add"
        android:orderInCategory="100"
        android:title="@string/add_record"
        app:showAsAction="always"/>

</menu>
```

The xml layout and code of `AddCountryActivity.java` file are defined below:
`activity_add_record.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="https://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20dp" >

    <EditText
        android:id="@+id/subject_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/enter_title" >

        <requestFocus />
```

```
    </EditText>

    <EditText
        android:id="@+id/description_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/enter_desc"
        android:inputType="textMultiLine"
        android:minLines="5" >
    </EditText>

    <Button
        android:id="@+id/add_record"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/add_record" />

</LinearLayout>
```

Two EditText components that take the inputs for country and currency along with a button to add the values to the database and display it in the ListView are defined.

`AddCountryActivity.java`

```
package com.journaldev.sqlite;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class AddCountryActivity extends Activity implements OnClickListener {

    private Button addTodoBtn;
    private EditText subjectEditText;
    private EditText descEditText;

    private DBManager dbManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setTitle("Add Record");

        setContentView(R.layout.activity_add_record);
```

```
        subjectEditText = (EditText) findViewById(R.id.subject_edittext);
        descEditText = (EditText) findViewById(R.id.description_edittext);

        addTodoBtn = (Button) findViewById(R.id.add_record);

        dbManager = new DBManager(this);
        dbManager.open();
        addTodoBtn.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.add_record:

                final String name = subjectEditText.getText().toString();
                final String desc = descEditText.getText().toString();

                dbManager.insert(name, desc);

                Intent main = new Intent(AddCountryActivity.this, CountryListActivity
                        .setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

                startActivity(main);
                break;
        }
    }

}
```

The CRUD operation performed here is adding a new record to the database. The xml layout and code of ModifyCountryActivity.java file are defined below:

`activity_modify_record.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="https://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp" >

    <EditText
        android:id="@+id/subject_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:ems="10"
        android:hint="@string/enter_title" />
```

```
    <EditText
        android:id="@+id/description_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/enter_desc"
        android:inputType="textMultiLine"
        android:minLines="5" >
    </EditText>


    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:weightSum="2"
        android:gravity="center_horizontal"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btn_update"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/btn_update" />

        <Button
            android:id="@+id/btn_delete"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/btn_delete" />
    </LinearLayout>

</LinearLayout>
```

It's similar to the previous layout except that modify and delete buttons are added.

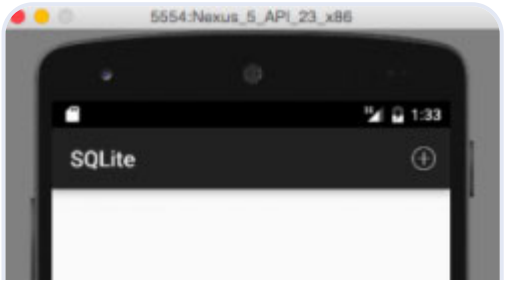
`ModifyCountryActivity.java`

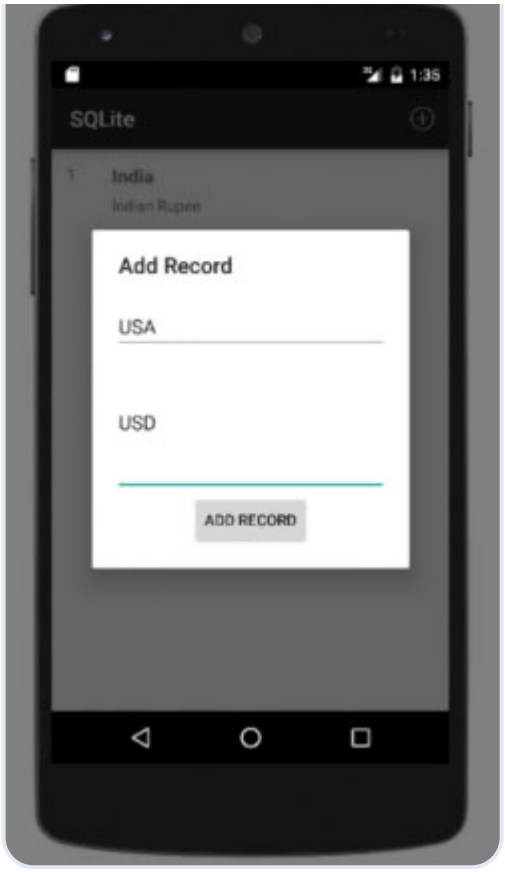
```
package com.journaldev.sqlite;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class ModifyCountryActivity extends Activity implements OnClickListener {

    private EditText titleText;
```

```
    private Button updateBtn, deleteBtn;
    private EditText descText;

    private long _id;

    private DBManager dbManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setTitle("Modify Record");

        setContentView(R.layout.activity_modify_record);

        dbManager = new DBManager(this);
        dbManager.open();

        titleText = (EditText) findViewById(R.id.subject_edittext);
        descText = (EditText) findViewById(R.id.description_edittext);

        updateBtn = (Button) findViewById(R.id.btn_update);
        deleteBtn = (Button) findViewById(R.id.btn_delete);

        Intent intent = getIntent();
        String id = intent.getStringExtra("id");
        String name = intent.getStringExtra("title");
        String desc = intent.getStringExtra("desc");

        _id = Long.parseLong(id);

        titleText.setText(name);
        descText.setText(desc);

        updateBtn.setOnClickListener(this);
        deleteBtn.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_update:
                String title = titleText.getText().toString();
                String desc = descText.getText().toString();

                dbManager.update(_id, title, desc);
                this.returnHome();
                break;

            case R.id.btn_delete:
                dbManager.delete(_id);
                this.returnHome();
                break;
```

```
        }
    }

    public void returnHome() {
        Intent home_intent = new Intent(getApplicationContext(), CountryListActivity.
                .setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(home_intent);
    }
}
```

The CRUD operations performed here are updating and deleting a record. The below images are the screenshots of the final output of our project. The first image is the output seen when the application is launched for the first time.



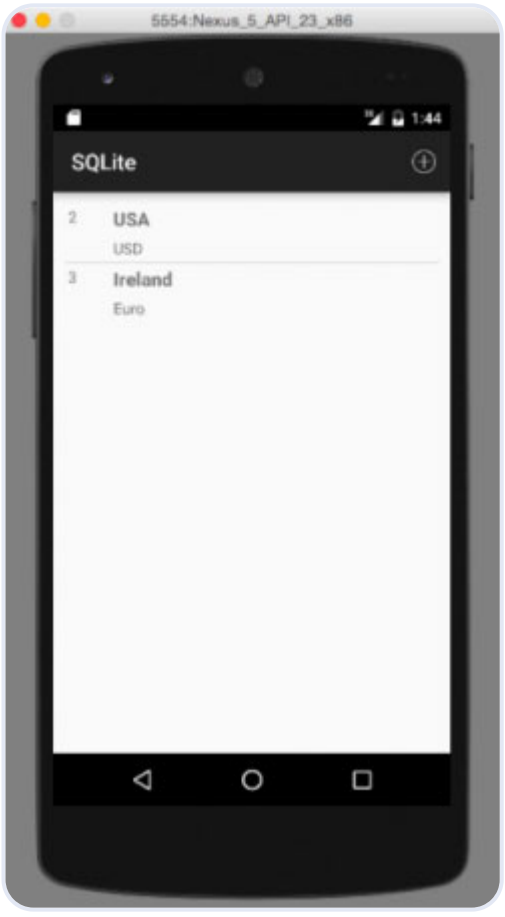


The third image shows an output when 3 records are added :

The fourth image shows the output when any list item is clicked to modify or delete a

record :

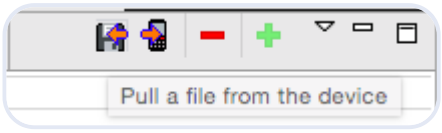
The final image is the output when a record is deleted. In this example we delete the first record :



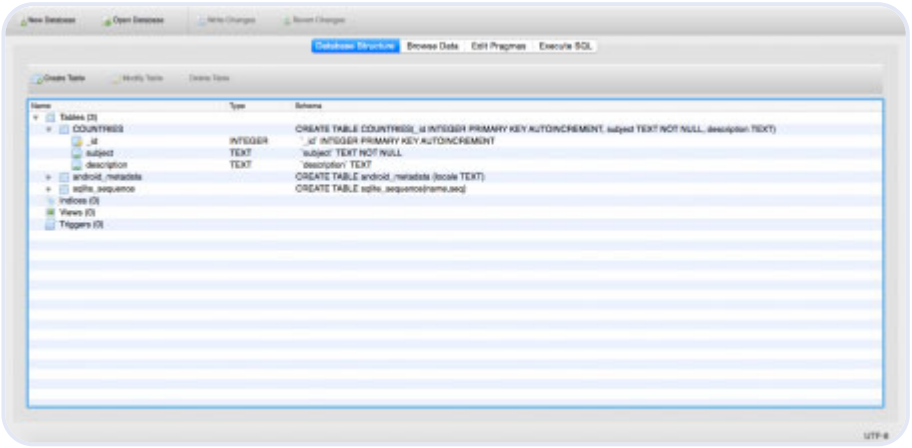
Opening the Android SQLite Database file

As we've discussed earlier in this tutorial, the database file is stored in the internal storage that is accessible from the Android Device Monitor as visible in the pic below.

To view this database we need to pull this file from the device to our desktop. This is done by clicking the menu option in the top right as seen in the image below :



To open this file download the SQLiteBrowser from this link. The snippets below show the schema and tables in the browser.



To view the table go to the Browse Data tab on top. The following image is seen:

	_id	subject	description
	Filter	Filter	Filter
1	2	USA	USD
2	3	Ireland	Euro

This brings an end to Android SQLite tutorial. The final **Android SQLite Project** is downloadable from the below link.

Download Android SQLite Example Project

About the author(s)

See author profile

Category: Tutorial

Tags: Android

While we believe that this content benefits our community, we have not yet thoroughly reviewed it. If you have any suggestions for improvements, please let us know by clicking the “report an issue“ button at the bottom of the tutorial.

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

JournalDev • January 19, 2016

Thanks for this. It's simple and nicely explained.

- Ion

Show replies

JournalDev • January 30, 2016

Hello there. Where is the "activity_view_record" XML file? Please email me back!

- Rara

Show replies

JournalDev • February 23, 2016

Thanks for this nice tutorial. Can you please help me how to get the image from sqlite and display in listVeiw

- Mohammed Illyas M A

JournalDev • March 29, 2016

Awesome tutorial. I've tried adding extra fields and it works fine apart from when I go to the update/delete fragment. Only the data fields in the listview layout are populated in the update/delete fragment. It looks like the listview is not passing the field data it is not using to the update/delete fragment. How can I fix that? Thanks!

- Marcus

JournalDev • June 25, 2016

Good tutorial But update and delete does not work.When i want to update and delete it does not change anything

- Abir khan

JournalDev • July 5, 2016

Hi! Thank you for a pretty great tutorial. I'm trying to implement that CRUD in a fragment (cause I'm using navigation drawer =sandwich menu) and I cannot get to see the actionbar. Could you please help me? Also, is there a way to make a button that will do all of the add country. I've been straggling with this for weeks. Please help. Thanks

- Patrick

Show replies

JournalDev • July 14, 2016

Hi, in CountryListActivity.java, at this line : final int[] to = new int[] { R.id.id, R.id.title, R.id.desc }; where can i find R.id.id, R.id.title and R.id.desc ? Thanks.

- Dian

Show replies

JournalDev • October 21, 2016

This is the simple and nice tutorial

- Abdul Khaliq

JournalDev • October 25, 2016

very good tutorial for all

- tushar sharma

JournalDev • December 5, 2016

First, thanks for sharing this... and thanks to this also I learn how getMenuInflater works, likes how you open the view with that Intent in the same view and thet activity_view_record Thanks from Sinaloa, México

- sunco

Load more comments

Popular Topics

AI/ML

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →

Congratulations on unlocking the whale ambience easter egg!

Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

Reset easter egg to be discovered again

Permanently dismiss and hide easter egg

Thank you to the Glacier Bay National Park & Preserve and Merrick079 for the sounds behind this easter egg.

Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the Whale and Dolphin Conservation.

Become a contributor for community

Get paid to write technical tutorials and select a tech-focused charity to receive a matching donation.

Sign Up →

→

Resources for startups and SMBs

The Wave has everything you need to know about building a business, from raising funding to marketing your product.

Learn more →

Get our newsletter

Stay up to date by signing up for DigitalOcean's Infrastructure as a Newsletter.

Email address Submit

New accounts only. By submitting your email you agree to our Privacy Policy

*This promotional offer applies to new accounts only.