Android SQLite Tutorial

SQLite is an **open-source relational database** i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.

It is embedded in android bydefault. So, there is no need to perform any database setup or administration task.

Here, we are going to see the example of sqlite to store and fetch the data. Data is displayed in the logcat. For displaying data on the spinner or listview, move to the next page.

SQLiteOpenHelper class provides the functionality to use the SQLite database.

SQLiteOpenHelper class

The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management. For performing any database operation, you have to provide the implementation of **onCreate()** and **onUpgrade()** methods of SQLiteOpenHelper class.

Constructors of SQLiteOpenHelper class

There are two constructors of SQLiteOpenHelper class.

Constructor	Description
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	creates an object for creating, opening and managing the database.

⇧

SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	creates an object for creating, opening and managing the database. It specifies the error handler.

Methods of SQLiteOpenHelper class

There are many methods in SQLiteOpenHelper class. Some of them are as follows:

Method	Description
public abstract void onCreate(SQLiteDatabase db)	called only once when database is created for the first time.
public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be upgraded.
public synchronized void close ()	closes the database object.
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be downgraded.

SQLiteDatabase class

It contains methods to be performed on sqlite database such as create, update, delete, select etc.

Methods of SQLiteDatabase class

There are many methods in SQLiteDatabase class. Some of them are as follows:

Method	Description
void execSQL(String sql)	executes the sql query not select query.
long insert(String table, String nullColumnHack, ContentValues values)	inserts a record on the database. The table specifies the table name, nullColumnHack doesn't allow completely null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	updates a row.
Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	returns a cursor over the resultset.

Example of android SQLite database

Let's see the simple example of android sqlite database.

File: Contact.java

```
package com.example.sqlite;
public class Contact {
  int _id;
```

```
String _name;
String _phone_number;
 public Contact(){   }
public Contact(int id, String name, String _phone_number){
    this._id = id;
    this._name = name;
    this._phone_number = _phone_number;
}

public Contact(String name, String _phone_number){
    this._name = name;
    this._phone_number = _phone_number;
}
public int getID(){
    return this._id;
}

public void setID(int id){
    this._id = id;
}

public String getName(){
    return this._name;
}

public void setName(String name){
    this._name = name;
```

⇧

```
    }

    public String getPhoneNumber(){
        return this._phone_number;
    }

    public void setPhoneNumber(String phone_number){
        this._phone_number = phone_number;
    }
}
```

File: DatabaseHandler.java

Now, let's create the database handler class that extends SQLiteOpenHelper class and provides the implementation of its methods.

```
package com.example.sqlite;
import java.util.ArrayList;
import java.util.List;

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

public class DatabaseHandler extends SQLiteOpenHelper {
   private static final int DATABASE_VERSION = 1;
   private static final String DATABASE_NAME = "contactsManager";
private static final String TABLE_CONTACTS = "contacts";
 private static final String KEY_ID = "id";
private static final String KEY_NAME = "name";
private static final String KEY_PH_NO = "phone_number";

public DatabaseHandler(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
    //3rd argument to be passed is CursorFactory instance
}

// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
            + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
            + KEY_PH_NO + " TEXT" + ")";
    db.execSQL(CREATE_CONTACTS_TABLE);
}

// Upgrading database
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Drop older table if existed
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

    // Create tables again
    onCreate(db);
```

```
}

 // code to add the new contact
 void addContact(Contact contact) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName()); // Contact Name
    values.put(KEY_PH_NO, contact.getPhoneNumber()); // Contact Phone

    // Inserting Row
    db.insert(TABLE_CONTACTS, null, values);
    //2nd argument is String containing nullColumnHack
    db.close(); // Closing database connection
}

// code to get the single contact
Contact getContact(int id) {
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
            KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
            new String[] { String.valueOf(id) }, null, null, null, null);
    if (cursor != null)
        cursor.moveToFirst();

    Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
```

⇧

```
            cursor.getString(1), cursor.getString(2));
        // return contact
        return contact;
    }

    // code to get all contacts in a list view
    public List<Contact> getAllContacts() {
        List<Contact> contactList = new ArrayList<Contact>();
        // Select All Query
        String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                Contact contact = new Contact();
                contact.setID(Integer.parseInt(cursor.getString(0)));
                contact.setName(cursor.getString(1));
                contact.setPhoneNumber(cursor.getString(2));
                // Adding contact to list
                contactList.add(contact);
            } while (cursor.moveToNext());
        }

        // return contact list
        return contactList;
    }

    // code to update the single contact
    public int updateContact(Contact contact) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_NAME, contact.getName());
        values.put(KEY_PH_NO, contact.getPhoneNumber());

        // updating row
        return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
                new String[] { String.valueOf(contact.getID()) });
    }

    // Deleting single contact
    public void deleteContact(Contact contact) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
                new String[] { String.valueOf(contact.getID()) });
        db.close();
    }

    // Getting contacts Count
    public int getContactsCount() {
        String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();

        // return count
        return cursor.getCount();
    }

}
```

File: MainActivity.java

```
package com.example.sqlite;

import java.util.List;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.Menu;

public class MainActivity extends Activity {

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

```

⇧

```
    DatabaseHandler db = new DatabaseHandler(this);

     // Inserting Contacts
    Log.d("Insert: ", "Inserting ..");
    db.addContact(new Contact("Ravi", "9100000000"));
    db.addContact(new Contact("Srinivas", "9199999999"));
    db.addContact(new Contact("Tommy", "9522222222"));
    db.addContact(new Contact("Karthik", "9533333333"));

    // Reading all contacts
    Log.d("Reading: ", "Reading all contacts..");
    List<Contact> contacts = db.getAllContacts();

    for (Contact cn : contacts) {
     String log = "Id: "+cn.getID()+" ,Name: " + cn.getName() + " ,Phone: " +
       cn.getPhoneNumber();
    // Writing Contacts to log
    Log.d("Name: ", log);
}
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.activity_main, menu);
    return true;
}
```

```
}
```

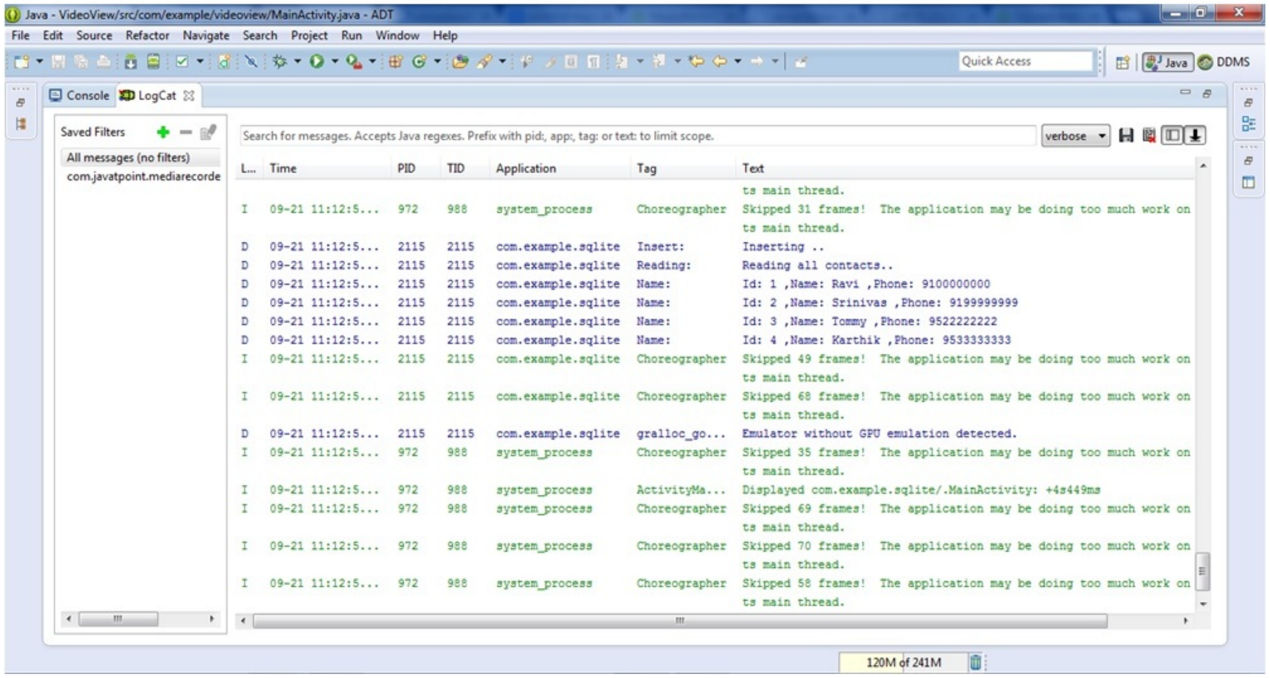
download this sqlite example

Output:

Open Logcat and see the output. It is the basic example of android sqlite without any GUI.

For GUI application with android SQLite, visit next page.

Output:


Java - VideoView/src/com/example/videoview/MainActivity.java - ADT
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
DDMS
Console
LogCat
Saved Filters
All messages (no filters)
com.javatpoint.mediarecorde
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose
120M of 241M

```
L...  Time               PID   TID   Application         Tag             Text
                                                                         ts main thread.
I     09-21 11:12:5...   972   988   system_process      Choreographer   Skipped 31 frames!  The application may be doing too much work on
                                                                         ts main thread.
D     09-21 11:12:5...   2115  2115  com.example.sqlite  Insert:         Inserting ..
D     09-21 11:12:5...   2115  2115  com.example.sqlite  Reading:        Reading all contacts..
D     09-21 11:12:5...   2115  2115  com.example.sqlite  Name:           Id: 1 ,Name: Ravi ,Phone: 9100000000
D     09-21 11:12:5...   2115  2115  com.example.sqlite  Name:           Id: 2 ,Name: Srinivas ,Phone: 9199999999
D     09-21 11:12:5...   2115  2115  com.example.sqlite  Name:           Id: 3 ,Name: Tommy ,Phone: 9522222222
D     09-21 11:12:5...   2115  2115  com.example.sqlite  Name:           Id: 4 ,Name: Karthik ,Phone: 9533333333
I     09-21 11:12:5...   2115  2115  com.example.sqlite  Choreographer   Skipped 49 frames!  The application may be doing too much work on
                                                                         ts main thread.
I     09-21 11:12:5...   2115  2115  com.example.sqlite  Choreographer   Skipped 68 frames!  The application may be doing too much work on
                                                                         ts main thread.
D     09-21 11:12:5...   2115  2115  com.example.sqlite  gralloc_go...   Emulator without GPU emulation detected.
I     09-21 11:12:5...   972   988   system_process      Choreographer   Skipped 35 frames!  The application may be doing too much work on
                                                                         ts main thread.
I     09-21 11:12:5...   972   988   system_process      ActivityMa...   Displayed com.example.sqlite/.MainActivity: +4s449ms
I     09-21 11:12:5...   972   988   system_process      Choreographer   Skipped 69 frames!  The application may be doing too much work on
                                                                         ts main thread.
I     09-21 11:12:5...   972   988   system_process      Choreographer   Skipped 70 frames!  The application may be doing too much work on
                                                                         ts main thread.
I     09-21 11:12:5...   972   988   system_process      Choreographer   Skipped 58 frames!  The application may be doing too much work on
                                                                         ts main thread.
```

⇧

<<prev next>>

Please Share

Learn Latest Tutorials

Kotlin

OS

.Net

JOGL

Django

⇧