Android SQLite Database

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

```
1.  package example.javatpoint.com.sqlitetutorial;
2.
3.  public class Contact {
4.      int _id;
5.      String _name;
6.      String _phone_number;
7.      public Contact(){   }
8.      public Contact(int id, String name, String _phone_number){
9.          this._id = id;
10.         this._name = name;
11.         this._phone_number = _phone_number;
12.     }
13.
14.     public Contact(String name, String _phone_number){
15.         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.

```
1.  package example.javatpoint.com.sqlitetutorial;
2.
3.  import android.content.ContentValues;
4.  import android.content.Context;
5.  import android.database.Cursor;
6.  import android.database.sqlite.SQLiteDatabase;
7.  import android.database.sqlite.SQLiteOpenHelper;
8.  import java.util.ArrayList;
9.  import java.util.List;
10.
11.
12. public class DatabaseHandler extends SQLiteOpenHelper {
13.     private static final int DATABASE_VERSION = 1;
14.     private static final String DATABASE_NAME = "contactsManager";
15.     private static final String TABLE_CONTACTS = "contacts";
16.     private static final String KEY_ID = "id";
17.     private static final String KEY_NAME = "name";
18.     private static final String KEY_PH_NO = "phone_number";
```

```
19.
20.    public DatabaseHandler(Context context) {
21.        super(context, DATABASE_NAME, null, DATABASE_VERSION);
22.        //3rd argument to be passed is CursorFactory instance
23.    }
24.
25.    // Creating Tables
26.    @Override
27.    public void onCreate(SQLiteDatabase db) {
28.        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CON
     TACTS + "("
29.                + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
30.                + KEY_PH_NO + " TEXT" + ")";
31.        db.execSQL(CREATE_CONTACTS_TABLE);
32.    }
33.
34.    // Upgrading database
35.    @Override
36.    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
37.        // Drop older table if existed
38.        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);
39.
40.        // Create tables again
41.        onCreate(db);
42.    }
43.
44.    // code to add the new contact
45.    void addContact(Contact contact) {
46.        SQLiteDatabase db = this.getWritableDatabase();
47.
48.        ContentValues values = new ContentValues();
49.        values.put(KEY_NAME, contact.getName()); // Contact Name
50.        values.put(KEY_PH_NO, contact.getPhoneNumber()); // Contact Phone
51.
52.        // Inserting Row
53.        db.insert(TABLE_CONTACTS, null, values);
54.        //2nd argument is String containing nullColumnHack
55.        db.close(); // Closing database connection
56.    }
57.
58.    // code to get the single contact
59.    Contact getContact(int id) {
60.        SQLiteDatabase db = this.getReadableDatabase();
61.
62.        Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
63.                    KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
64.                new String[] { String.valueOf(id) }, null, null, null, null);
65.        if (cursor != null)
66.            cursor.moveToFirst();
67.
```

```
68.         Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
69.                 cursor.getString(1), cursor.getString(2));
70.         // return contact
71.         return contact;
72.     }
73.
74.     // code to get all contacts in a list view
75.     public List<Contact> getAllContacts() {
76.         List<Contact> contactList = new ArrayList<Contact>();
77.         // Select All Query
78.         String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;
79.
80.         SQLiteDatabase db = this.getWritableDatabase();
81.         Cursor cursor = db.rawQuery(selectQuery, null);
82.
83.         // looping through all rows and adding to list
84.         if (cursor.moveToFirst()) {
85.             do {
86.                 Contact contact = new Contact();
87.                 contact.setID(Integer.parseInt(cursor.getString(0)));
88.                 contact.setName(cursor.getString(1));
89.                 contact.setPhoneNumber(cursor.getString(2));
90.                 // Adding contact to list
91.                 contactList.add(contact);
92.             } while (cursor.moveToNext());
93.         }
94.
95.         // return contact list
96.         return contactList;
97.     }
98.
99.     // code to update the single contact
100.             public int updateContact(Contact contact) {
101.                 SQLiteDatabase db = this.getWritableDatabase();
102.
103.                 ContentValues values = new ContentValues();
104.                 values.put(KEY_NAME, contact.getName());
105.                 values.put(KEY_PH_NO, contact.getPhoneNumber());
106.
107.                 // updating row
108.                 return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
109.                         new String[] { String.valueOf(contact.getID()) });
110.             }
111.
112.             // Deleting single contact
113.             public void deleteContact(Contact contact) {
114.                 SQLiteDatabase db = this.getWritableDatabase();
115.                 db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
116.                         new String[] { String.valueOf(contact.getID()) });
117.                 db.close();
```

```
118.            }
119.
120.            // Getting contacts Count
121.            public int getContactsCount() {
122.                String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
123.                SQLiteDatabase db = this.getReadableDatabase();
124.                Cursor cursor = db.rawQuery(countQuery, null);
125.                cursor.close();
126.
127.                // return count
128.                return cursor.getCount();
129.            }
130.
131.        }
```

File: MainActivity.java

```
1.  package example.javatpoint.com.sqlitetutorial;
2.
3.  import android.support.v7.app.AppCompatActivity;
4.  import android.os.Bundle;
5.  import android.util.Log;
6.  import java.util.List;
7.
8.  public class MainActivity extends AppCompatActivity {
9.
10.    @Override
11.    protected void onCreate(Bundle savedInstanceState) {
12.        super.onCreate(savedInstanceState);
13.        setContentView(R.layout.activity_main);
14.        DatabaseHandler db = new DatabaseHandler(this);
15.
16.        // Inserting Contacts
17.        Log.d("Insert: ", "Inserting ..");
18.        db.addContact(new Contact("Ravi", "9100000000"));
19.        db.addContact(new Contact("Srinivas", "9199999999"));
20.        db.addContact(new Contact("Tommy", "9522222222"));
21.        db.addContact(new Contact("Karthik", "9533333333"));
22.
23.        // Reading all contacts
24.        Log.d("Reading: ", "Reading all contacts..");
25.        List<Contact> contacts = db.getAllContacts();
26.
27.        for (Contact cn : contacts) {
28.            String log = "Id: " + cn.getID() + " ,Name: " + cn.getName() + " ,Phone: " +
29.                    cn.getPhoneNumber();
30.            // Writing Contacts to log
31.            Log.d("Name: ", log);
32.        }
33.    }
```

```
34. }
```