Android SQLite Database with Multiple Tables

61 Comments . By Ravi Tamada on 15th, Sep 2013 - 08:26 PM Tweet 21 g+1 43 Like 382

In my previous tutorial Android SQLite Database Tutorial I explained how to use SQLite database in your android application. But that covered the scenario, only when you have one table in the database. I am getting lot of queries about handling the sqlite database when it is having multiple tables.

I explained here how to handle the SQLite database when it is having multiple tables.

DOWNLOAD CODE

Use Case: Todo Application

To make it easier for you to understand, I am taking a real use case example of **TODO Application** database schema in this tutorial. This article doesn't covers how to design the application, but explains the database design, preparing database helper classes and models.

Database Design

I considered a basic Todo Application with minimal functionality like **creating a todo note** and **assigning it under a tag(s)** (category). So for this we just need three tables in the database.

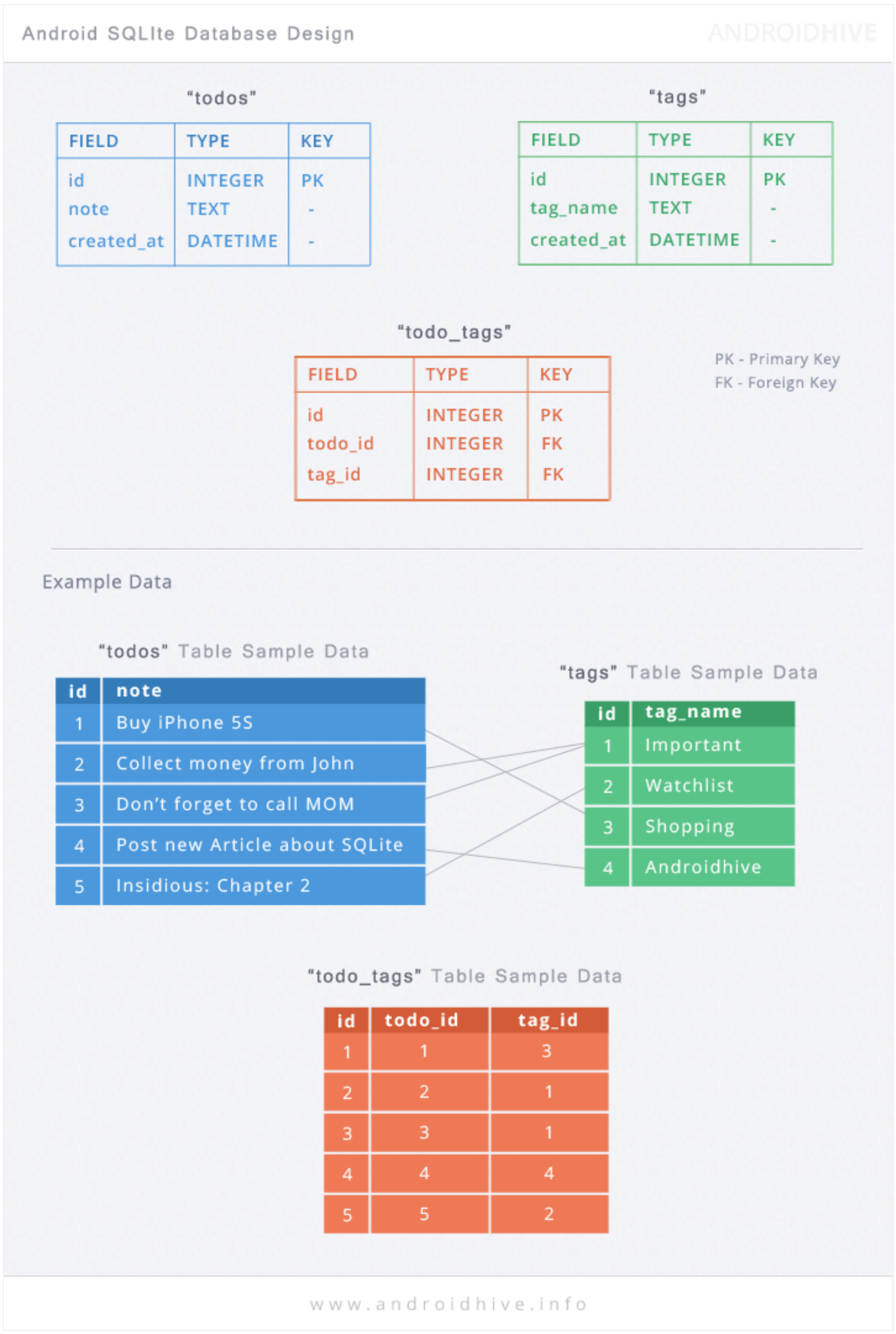
The three tables are

`todos` – to store all todo notes
`tags` – to store list of tags
`todo_tags` – to store the tags which are assigned to a todo

Check the below diagram that explains the table structure and the relationship between tables

Ravi Tamada
Hyderabad, INDIA

Tag Cloud


Android SQLIte Database Design

"todos"

FIELD	TYPE	KEY
id	INTEGER	PK
note	TEXT	-
created_at	DATETIME	-

"tags"

FIELD	TYPE	KEY
id	INTEGER	PK
tag_name	TEXT	-
created_at	DATETIME	-

"todo_tags"

FIELD	TYPE	KEY
id	INTEGER	PK
todo_id	INTEGER	FK
tag_id	INTEGER	FK

PK - Primary Key
FK - Foreign Key

Example Data

"todos" Table Sample Data

id	note
1	Buy iPhone 5S
2	Collect money from John
3	Don't forget to call MOM
4	Post new Article about SQLite
5	Insidious: Chapter 2

"tags" Table Sample Data

id	tag_name
1	Important
2	Watchlist
3	Shopping
4	Androidhive

"todo_tags" Table Sample Data

id	todo_id	tag_id
1	1	3
2	2	1
3	3	1
4	4	4
5	5	2

www.androidhive.info

Let's start a new Project

So let's start by creating a new project in Eclipse IDE

1. Create a new project in Eclipse from **File ⇒ New ⇒ Android ⇒ Application Project**. I named my package name as `info.androidhive.sqlite` and left the main activity name as `MainActivity.java`

2. We need two more packages to keep helpers and model classes. **Right Clicking on src ⇒ New ⇒ Package** and name them as `info.androidhive.sqlite.helper` and `info.androidhive.sqlite.model`

Creating Model Class for Tables

Action Bar Adapter Animation API
Apps Async Beginner Camera
Dashboard Database facebook
Fragments GCM Gestures Google
Google Plus GPS Grid HTTP
Intermediate json Libstreaming
List View Maps MySQL
Navigation Drawer PHP Pinch
Quick Tips REST sessions Slim
Speech Input Spinner sponsored
SQLite Swipe Tab View Twitter UI
Video Video Streaming View Pager
Volley xml

Ravi Tamada
google.com/+RaviTamada

5,439 followers

Most Popular

1 Android SQLite Database Tutorial - 805,01 views

2 How to connect Android with PHP, MySQL 658,976 views

3 Android Custom ListView with Image an Text - 642,710 views

4 Android JSON Parsing Tutorial - 642,42 views

5 Android Push Notifications using Googl Cloud Messaging (GCM), PHP and MySQL 538,522 views

6 Android Login and Registration with PHI MySQL and SQLite - 443,204 views

7 Android Tab Layout Tutorial - 401,641 views

8 Android GPS, Location Manager Tutorial 324,008 views

9 Android Login and Registration Scree Design - 317,570 views

10 Android XML Parsing Tutorial - 309,995 view

Next step is to create model classes for our database tables just to make single row as an object. We need only two models for **todos** and **tags**. For todo_tags we don't need a model class.

3. Create a new class file under `info.androidhive.sqlite.helper` package named `Todo.java` and type the code like below. This is the model class for **todos** table

```
Todo.java
package info.androidhive.sqlite.model;

public class Todo {

    int id;
    String note;
    int status;
    String created_at;

    // constructors
    public Todo() {
    }

    public Todo(String note, int status) {
        this.note = note;
        this.status = status;
    }

    public Todo(int id, String note, int status) {
        this.id = id;
        this.note = note;
        this.status = status;
    }

    // setters
    public void setId(int id) {
        this.id = id;
    }

    public void setNote(String note) {
        this.note = note;
    }

    public void setStatus(int status) {
        this.status = status;
    }

    public void setCreatedAt(String created_at){
        this.created_at = created_at;
    }

    // getters
    public long getId() {
        return this.id;
    }

    public String getNote() {
        return this.note;
    }

    public int getStatus() {
        return this.status;
    }
}
```

4. Create one more model class for **tags** table named `Tag.java` under the same package.

```
Tag.java
package info.androidhive.sqlite.model;

public class Tag {

    int id;
    String tag_name;

    // constructors
    public Tag() {

    }

    public Tag(String tag_name) {
        this.tag_name = tag_name;
    }

    public Tag(int id, String tag_name) {
        this.id = id;
        this.tag_name = tag_name;
    }

    // setter
    public void setId(int id) {
        this.id = id;
    }

    public void setTagName(String tag_name) {
        this.tag_name = tag_name;
    }

    // getter
    public int getId() {
        return this.id;
    }

    public String getTagName() {
        return this.tag_name;
    }
}
```

Database Helper Class

Database helper class contains all the methods to perform database operations like opening connection, closing connection, insert, update, read, delete and other things. As this class is helper class, place this under **helper** package.

5. So create another class named `DatabaseHelper.java` under `info.androidhive.sqlite.helper` package and extend the class from SQLiteOpenHelper

```
public class DatabaseHelper extends SQLiteOpenHelper {
```

6. Add required variables like database name, database version, column names. I also executed table create statements in **onCreate()** method. Type the following code in **DatabaseHelper.java** class

```
DatabaseHelper.java
public class DatabaseHelper extends SQLiteOpenHelper {

    // Logcat tag
    private static final String LOG = "DatabaseHelper";

    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "contactsManager";

    // Table Names
    private static final String TABLE_TODO = "todos";
    private static final String TABLE_TAG = "tags";
    private static final String TABLE_TODO_TAG = "todo_tags";

    // Common column names
    private static final String KEY_ID = "id";
    private static final String KEY_CREATED_AT = "created_at";

    // NOTES Table - column nmaes
    private static final String KEY_TODO = "todo";
    private static final String KEY_STATUS = "status";

    // TAGS Table - column names
    private static final String KEY_TAG_NAME = "tag_name";

    // NOTE_TAGS Table - column names
    private static final String KEY_TODO_ID = "todo_id";
    private static final String KEY_TAG_ID = "tag_id";

    // Table Create Statements
    // Todo table create statement
    private static final String CREATE_TABLE_TODO = "CREATE TABLE "
            + TABLE_TODO + "(" + KEY_ID + " INTEGER PRIMARY KEY," + KEY_TC
            + " TEXT," + KEY_STATUS + " INTEGER," + KEY_CREATED_AT
            + " DATETIME" + ")";

    // Tag table create statement
    private static final String CREATE_TABLE_TAG = "CREATE TABLE " + TABLE
            + "(" + KEY_ID + " INTEGER PRIMARY KEY," + KEY_TAG_NAME + " TE
            + KEY_CREATED_AT + " DATETIME" + ")";

    // todo_tag table create statement
    private static final String CREATE_TABLE_TODO_TAG = "CREATE TABLE "
            + TABLE_TODO_TAG + "(" + KEY_ID + " INTEGER PRIMARY KEY,"
            + KEY_TODO_ID + " INTEGER," + KEY_TAG_ID + " INTEGER,"
            + KEY_CREATED_AT + " DATETIME" + ")";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        // creating required tables
        db.execSQL(CREATE_TABLE_TODO);
        db.execSQL(CREATE_TABLE_TAG);
        db.execSQL(CREATE_TABLE_TODO_TAG);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersic
        // on upgrade drop older tables
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_TODO);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_TAG);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_TODO_TAG);

        // create new tables
        onCreate(db);
    }
```

CRUD (Create, Read, Update and Delete) Operations

From now on we are going to add one by one method into **DatabaseHelper.class**

1. Creating a Todo

The function will create a **todo** item in **todos** table. In this same function we are assigning the todo to a tag name which inserts a row in **todo_tags** table.

```
/*
 * Creating a todo
 */
public long createToDo(Todo todo, long[] tag_ids) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TODO, todo.getNote());
    values.put(KEY_STATUS, todo.getStatus());
    values.put(KEY_CREATED_AT, getDateTime());

    // insert row
    long todo_id = db.insert(TABLE_TODO, null, values);

    // assigning tags to todo
    for (long tag_id : tag_ids) {
        createTodoTag(todo_id, tag_id);
    }

    return todo_id;
}
```

2. Fetching a Todo

Following will fetch a todo from todos table.

```
SELECT * FROM todos WHERE id = 1;
```

```
/*
 * get single todo
 */
public Todo getTodo(long todo_id) {
    SQLiteDatabase db = this.getReadableDatabase();
    String selectQuery = "SELECT  * FROM " + TABLE_TODO + " WHERE "
            + KEY_ID + " = " + todo_id;

    Log.e(LOG, selectQuery);

    Cursor c = db.rawQuery(selectQuery, null);

    if (c != null)
        c.moveToFirst();

    Todo td = new Todo();
    td.setId(c.getInt(c.getColumnIndex(KEY_ID)));
    td.setNote((c.getString(c.getColumnIndex(KEY_TODO))));
    td.setCreatedAt(c.getString(c.getColumnIndex(KEY_CREATED_AT)));

    return td;
}
```

3. Fetching all Todos

Fetching all todos involves reading all todo rows and adding them to a list array.

```
SELECT * FROM todos;
```

```
/*
 * getting all todos
 * */
public List<Todo> getAllToDos() {
    List<Todo> todos = new ArrayList<Todo>();
    String selectQuery = "SELECT  * FROM " + TABLE_TODO;

    Log.e(LOG, selectQuery);

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor c = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (c.moveToFirst()) {
        do {
            Todo td = new Todo();
            td.setId(c.getInt((c.getColumnIndex(KEY_ID))));
            td.setNote((c.getString(c.getColumnIndex(KEY_TODO))));
            td.setCreatedAt(c.getString(c.getColumnIndex(KEY_CREATED_AT)))

            // adding to todo list
            todos.add(td);
        } while (c.moveToNext());
    }

    return todos;
}
```

4. Fetching all Todos under a Tag name

This is also same as reading all the rows but it filters the todos by tag name. Check the following select query which fetches the todos under Watchlist tag name.

```
SELECT * FROM todos td, tags tg, todo_tags tt WHERE tg.tag_name = 'Watchlist' AND tg.id = tt.tag_id AND
td.id = tt.todo_id;
```

```
/*
 * getting all todos under single tag
 * */
public List<Todo> getAllToDosByTag(String tag_name) {
    List<Todo> todos = new ArrayList<Todo>();

    String selectQuery = "SELECT  * FROM " + TABLE_TODO + " td, "
            + TABLE_TAG + " tg, " + TABLE_TODO_TAG + " tt WHERE tg."
            + KEY_TAG_NAME + " = '" + tag_name + "'" + " AND tg." + KEY_ID
            + " = " + "tt." + KEY_TAG_ID + " AND td." + KEY_ID + " = "
            + "tt." + KEY_TODO_ID;

    Log.e(LOG, selectQuery);
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor c = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (c.moveToFirst()) {
        do {
            Todo td = new Todo();
            td.setId(c.getInt((c.getColumnIndex(KEY_ID))));
            td.setNote((c.getString(c.getColumnIndex(KEY_TODO))));
            td.setCreatedAt(c.getString(c.getColumnIndex(KEY_CREATED_AT)))

            // adding to todo list
            todos.add(td);
        } while (c.moveToNext());
    }

    return todos;
}
```

5. Updating a Todo

Following function will update a todo. It will update Todo values only, not the tag name.

```
/*
 * Updating a todo
 */
public int updateToDo(Todo todo) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TODO, todo.getNote());
    values.put(KEY_STATUS, todo.getStatus());

    // updating row
    return db.update(TABLE_TODO, values, KEY_ID + " = ?",
            new String[] { String.valueOf(todo.getId()) });
}
```

6. Deleting a Todo

Pass todo id to the following function to delete the todo from db.

```
/*
 * Deleting a todo
 */
public void deleteToDo(long tado_id) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_TODO, KEY_ID + " = ?",
            new String[] { String.valueOf(tado_id) });
}
```

Until now we are done creating the CRUD methods onto **todos** table. Now we can start the methods required on **tags** table.

7. Creating Tag

Following method will insert a row into tags table.

```
/*
 * Creating tag
 */
public long createTag(Tag tag) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TAG_NAME, tag.getTagName());
    values.put(KEY_CREATED_AT, getDateTime());

    // insert row
    long tag_id = db.insert(TABLE_TAG, null, values);

    return tag_id;
}
```

8. Fetching all Tag names

Performing select all statement on tags table will give you list of tag names.

```
SELECT * FROM tags;
```

```
/**
 * getting all tags
 * */
public List<Tag> getAllTags() {
    List<Tag> tags = new ArrayList<Tag>();
    String selectQuery = "SELECT  * FROM " + TABLE_TAG;

    Log.e(LOG, selectQuery);

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor c = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (c.moveToFirst()) {
        do {
            Tag t = new Tag();
            t.setId(c.getInt((c.getColumnIndex(KEY_ID))));
            t.setTagName(c.getString(c.getColumnIndex(KEY_TAG_NAME)));

            // adding to tags list
            tags.add(t);
        } while (c.moveToNext());
    }
    return tags;
}
```

9. Updating Tags

Following method will update tag.

```
/*
 * Updating a tag
 */
public int updateTag(Tag tag) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TAG_NAME, tag.getTagName());

    // updating row
    return db.update(TABLE_TAG, values, KEY_ID + " = ?",
            new String[] { String.valueOf(tag.getId()) });
}
```

10. Deleting Tag and Todos under the Tag name

Following method will delete a tag from db. This also will delete all the todos under the tag name, but this is optional.

`should_delete_all_tag_todos` = Passing **true** will delete all the todos under the tag name

```
/*
 * Deleting a tag
 */
public void deleteTag(Tag tag, boolean should_delete_all_tag_todos) {
    SQLiteDatabase db = this.getWritableDatabase();

    // before deleting tag
    // check if todos under this tag should also be deleted
    if (should_delete_all_tag_todos) {
        // get all todos under this tag
        List<Todo> allTagToDos = getAllToDosByTag(tag.getTagName());

        // delete all todos
        for (Todo todo : allTagToDos) {
            // delete todo
            deleteToDo(todo.getId());
        }
    }

    // now delete the tag
    db.delete(TABLE_TAG, KEY_ID + " = ?",
            new String[] { String.valueOf(tag.getId()) });
}
```

Below are the methods to access the rows from **todo_tags** table

11. Assigning a Tag to Todo

Following method will assign a todo under a tag name. You can also assign multiple tags to a todo by calling this function multiple times.

```
/*
     * Creating todo_tag
     */
    public long createTodoTag(long todo_id, long tag_id) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_TODO_ID, todo_id);
        values.put(KEY_TAG_ID, tag_id);
        values.put(KEY_CREATED_AT, getDateTime());

        long id = db.insert(TABLE_TODO_TAG, null, values);

        return id;
    }
```

12. Removing Tag of Todo

Following method will remove the tag assigned to a todo

```
/*
 * Updating a todo tag
 */
public int updateNoteTag(long id, long tag_id) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TAG_ID, tag_id);

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

13. Changing the tag of todo

Following simply replaces the tag name of a todo

```
/*
 * Updating a todo tag
 */
public int updateNoteTag(long id, long tag_id) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TAG_ID, tag_id);
    // updating row
    return db.update(TABLE_TODO, values, KEY_ID + " = ?",
            new String[] { String.valueOf(id) });
}
```

14. Closing Database Connection

Importantly don't forget to close the database connection once you done using it. Call following method when you don't need access to db anymore.

```
// closing database
    public void closeDB() {
        SQLiteDatabase db = this.getReadableDatabase();
        if (db != null && db.isOpen())
            db.close();
    }
```

How to Use / Testing

As this tutorial already seems lengthy I am not considering giving an example with a sample application. In upcoming tutorial I will give you a simple todo application which will give you complete picture of using multiple SQLite tables in your android apps.

For now we will test the class just by printing the data to **Logcat**.

Open your main activity class and type the following. In the below I just created sample tags and todo data and performed the all the operations by calling the methods which we prepared in DatabaseHelper class.

```
MainActivity.java
package info.androidhive.sqlite;

import info.androidhive.sqlite.helper.DatabaseHelper;
import info.androidhive.sqlite.model.Tag;
import info.androidhive.sqlite.model.Todo;

import java.util.List;

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

public class MainActivity extends Activity {

    // Database Helper
    DatabaseHelper db;

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

        db = new DatabaseHelper(getApplicationContext());

        // Creating tags
        Tag tag1 = new Tag("Shopping");
        Tag tag2 = new Tag("Important");
        Tag tag3 = new Tag("Watchlist");
        Tag tag4 = new Tag("Androidhive");

        // Inserting tags in db
        long tag1_id = db.createTag(tag1);
        long tag2_id = db.createTag(tag2);
        long tag3_id = db.createTag(tag3);
        long tag4_id = db.createTag(tag4);

        Log.d("Tag Count", "Tag Count: " + db.getAllTags().size());

        // Creating ToDos
        Todo todo1 = new Todo("iPhone 5S", 0);
        Todo todo2 = new Todo("Galaxy Note II", 0);
        Todo todo3 = new Todo("Whiteboard", 0);
```

```java
        Todo todo4 = new Todo("Riddick", 0);
        Todo todo5 = new Todo("Prisoners", 0);
        Todo todo6 = new Todo("The Croods", 0);
        Todo todo7 = new Todo("Insidious: Chapter 2", 0);

        Todo todo8 = new Todo("Don't forget to call MOM", 0);
        Todo todo9 = new Todo("Collect money from John", 0);

        Todo todo10 = new Todo("Post new Article", 0);
        Todo todo11 = new Todo("Take database backup", 0);

        // Inserting todos in db
        // Inserting todos under "Shopping" Tag
        long todo1_id = db.createToDo(todo1, new long[] { tag1_id });
        long todo2_id = db.createToDo(todo2, new long[] { tag1_id });
        long todo3_id = db.createToDo(todo3, new long[] { tag1_id });

        // Inserting todos under "Watchlist" Tag
        long todo4_id = db.createToDo(todo4, new long[] { tag3_id });
        long todo5_id = db.createToDo(todo5, new long[] { tag3_id });
        long todo6_id = db.createToDo(todo6, new long[] { tag3_id });
        long todo7_id = db.createToDo(todo7, new long[] { tag3_id });

        // Inserting todos under "Important" Tag
        long todo8_id = db.createToDo(todo8, new long[] { tag2_id });
        long todo9_id = db.createToDo(todo9, new long[] { tag2_id });

        // Inserting todos under "Androidhive" Tag
        long todo10_id = db.createToDo(todo10, new long[] { tag4_id });
        long todo11_id = db.createToDo(todo11, new long[] { tag4_id });

        Log.e("Todo Count", "Todo count: " + db.getToDoCount());

        // "Post new Article" - assigning this under "Important" Tag
        // Now this will have - "Androidhive" and "Important" Tags
        db.createTodoTag(todo10_id, tag2_id);

        // Getting all tag names
        Log.d("Get Tags", "Getting All Tags");

        List<Tag> allTags = db.getAllTags();
        for (Tag tag : allTags) {
            Log.d("Tag Name", tag.getTagName());
        }

        // Getting all Todos
        Log.d("Get Todos", "Getting All ToDos");

        List<Todo> allToDos = db.getAllToDos();
        for (Todo todo : allToDos) {
            Log.d("ToDo", todo.getNote());
        }

        // Getting todos under "Watchlist" tag name
        Log.d("ToDo", "Get todos under single Tag name");

        List<Todo> tagsWatchList = db.getAllToDosByTag(tag3.getTagName());
        for (Todo todo : tagsWatchList) {
            Log.d("ToDo Watchlist", todo.getNote());
        }

        // Deleting a ToDo
        Log.d("Delete ToDo", "Deleting a Todo");
        Log.d("Tag Count", "Tag Count Before Deleting: " + db.getToDoCount

        db.deleteToDo(todo8_id);

        Log.d("Tag Count", "Tag Count After Deleting: " + db.getToDoCount(

        // Deleting all Todos under "Shopping" tag
        Log.d("Tag Count",
                "Tag Count Before Deleting 'Shopping' Todos: "
                        + db.getToDoCount());

        db.deleteTag(tag1, true);

        Log.d("Tag Count",
                "Tag Count After Deleting 'Shopping' Todos: "
                        + db.getToDoCount());

        // Updating tag name
        tag3.setTagName("Movies to watch");
        db.updateTag(tag3);

        // Don't forget to close database connection
        db.closeDB();

    }
}
```

Run the application and the **check the Logcat**.

Complete Code of DatabaseHelper.java Class

DatabaseHelper.java

```java
package info.androidhive.sqlite.helper;

import info.androidhive.sqlite.model.Tag;
import info.androidhive.sqlite.model.Todo;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;

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

public class DatabaseHelper extends SQLiteOpenHelper {

    // Logcat tag
    private static final String LOG = DatabaseHelper.class.getName();

    // Database Version
    private static final int DATABASE_VERSION = 1;

    // Database Name
    private static final String DATABASE_NAME = "contactsManager";

    // Table Names
    private static final String TABLE_TODO = "todos";
    private static final String TABLE_TAG = "tags";
    private static final String TABLE_TODO_TAG = "todo_tags";

    // Common column names
    private static final String KEY_ID = "id";
    private static final String KEY_CREATED_AT = "created_at";

    // NOTES Table - column nmaes
    private static final String KEY_TODO = "todo";
    private static final String KEY_STATUS = "status";

    // TAGS Table - column names
    private static final String KEY_TAG_NAME = "tag_name";

    // NOTE_TAGS Table - column names
    private static final String KEY_TODO_ID = "todo_id";
    private static final String KEY_TAG_ID = "tag_id";

    // Table Create Statements
    // Todo table create statement
    private static final String CREATE_TABLE_TODO = "CREATE TABLE "
            + TABLE_TODO + "(" + KEY_ID + " INTEGER PRIMARY KEY," + KEY_TC
            + " TEXT," + KEY_STATUS + " INTEGER," + KEY_CREATED_AT
            + " DATETIME" + ")";

    // Tag table create statement
    private static final String CREATE_TABLE_TAG = "CREATE TABLE " + TABLE
            + "(" + KEY_ID + " INTEGER PRIMARY KEY," + KEY_TAG_NAME + " TE
            + KEY_CREATED_AT + " DATETIME" + ")";

    // todo_tag table create statement
    private static final String CREATE_TABLE_TODO_TAG = "CREATE TABLE "
            + TABLE_TODO_TAG + "(" + KEY_ID + " INTEGER PRIMARY KEY,"
            + KEY_TODO_ID + " INTEGER," + KEY_TAG_ID + " INTEGER,"
            + KEY_CREATED_AT + " DATETIME" + ")";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        // creating required tables
        db.execSQL(CREATE_TABLE_TODO);
        db.execSQL(CREATE_TABLE_TAG);
        db.execSQL(CREATE_TABLE_TODO_TAG);
    }
```

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersic
    // on upgrade drop older tables
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TODO);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TAG);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_TODO_TAG);

    // create new tables
    onCreate(db);
}

// ------------------------ "todos" table methods ----------------//

 /**
 * Creating a todo
 */
public long createToDo(Todo todo, long[] tag_ids) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_TODO, todo.getNote());
    values.put(KEY_STATUS, todo.getStatus());
    values.put(KEY_CREATED_AT, getDateTime());

    // insert row
    long todo_id = db.insert(TABLE_TODO, null, values);

    // insert tag_ids
    for (long tag_id : tag_ids) {
        createTodoTag(todo_id, tag_id);
    }

    return todo_id;
}

/**
 * get single todo
 */
public Todo getTodo(long todo_id) {
    SQLiteDatabase db = this.getReadableDatabase();

    String selectQuery = "SELECT  * FROM " + TABLE_TODO + " WHERE "
            + KEY_ID + " = " + todo_id;

    Log.e(LOG, selectQuery);

    Cursor c = db.rawQuery(selectQuery, null);

    if (c != null)
        c.moveToFirst();

    Todo td = new Todo();
    td.setId(c.getInt(c.getColumnIndex(KEY_ID)));
    td.setNote((c.getString(c.getColumnIndex(KEY_TODO))));
    td.setCreatedAt(c.getString(c.getColumnIndex(KEY_CREATED_AT)));

    return td;
}

/**
 * getting all todos
 * */
public List<Todo> getAllToDos() {
    List<Todo> todos = new ArrayList<Todo>();
    String selectQuery = "SELECT  * FROM " + TABLE_TODO;

    Log.e(LOG, selectQuery);

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor c = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (c.moveToFirst()) {
        do {
            Todo td = new Todo();
            td.setId(c.getInt((c.getColumnIndex(KEY_ID))));
            td.setNote((c.getString(c.getColumnIndex(KEY_TODO))));
            td.setCreatedAt(c.getString(c.getColumnIndex(KEY_CREATED_A

            // adding to todo list
            todos.add(td);
        } while (c.moveToNext());
    }

    return todos;
}

/**
 * getting all todos under single tag
 * */
public List<Todo> getAllToDosByTag(String tag_name) {
    List<Todo> todos = new ArrayList<Todo>();
```

```
        String selectQuery = "SELECT  * FROM " + TABLE_TODO + " td, "
                + TABLE_TAG + " tg, " + TABLE_TODO_TAG + " tt WHERE tg."
                + KEY_TAG_NAME + " = '" + tag_name + "'" + " AND tg." + KE
                + " = " + "tt." + KEY_TAG_ID + " AND td." + KEY_ID + " = "
                + "tt." + KEY_TODO_ID;

        Log.e(LOG, selectQuery);

        SQLiteDatabase db = this.getReadableDatabase();
        Cursor c = db.rawQuery(selectQuery, null);

        // looping through all rows and adding to list
        if (c.moveToFirst()) {
            do {
                Todo td = new Todo();
                td.setId(c.getInt((c.getColumnIndex(KEY_ID))));
                td.setNote((c.getString(c.getColumnIndex(KEY_TODO))));
                td.setCreatedAt(c.getString(c.getColumnIndex(KEY_CREATED_A

                // adding to todo list
                todos.add(td);
            } while (c.moveToNext());
        }

        return todos;
    }

    /**
     * getting todo count
     */
    public int getToDoCount() {
        String countQuery = "SELECT  * FROM " + TABLE_TODO;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);

        int count = cursor.getCount();
        cursor.close();

        // return count
        return count;
    }

    /**
     * Updating a todo
     */
    public int updateToDo(Todo todo) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_TODO, todo.getNote());
        values.put(KEY_STATUS, todo.getStatus());

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

    /**
     * Deleting a todo
     */
    public void deleteToDo(long tado_id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_TODO, KEY_ID + " = ?",
                new String[] { String.valueOf(tado_id) });
    }

    // ------------------------ "tags" table methods ----------------//

    /**
     * Creating tag
     */
    public long createTag(Tag tag) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_TAG_NAME, tag.getTagName());
        values.put(KEY_CREATED_AT, getDateTime());

        // insert row
        long tag_id = db.insert(TABLE_TAG, null, values);

        return tag_id;
    }

    /**
     * getting all tags
     * */
    public List<Tag> getAllTags() {
        List<Tag> tags = new ArrayList<Tag>();
        String selectQuery = "SELECT  * FROM " + TABLE_TAG;
        Log.e(LOG, selectQuery);

        SQLiteDatabase db = this.getReadableDatabase();
        Cursor c = db.rawQuery(selectQuery, null);

        // looping through all rows and adding to list
        if (c.moveToFirst()) {
            do {
                Tag t = new Tag();
                t.setId(c.getInt((c.getColumnIndex(KEY_ID))));
                t.setTagName(c.getString(c.getColumnIndex(KEY_TAG_NAME)));

                // adding to tags list
                tags.add(t);
            } while (c.moveToNext());
        }
        return tags;
    }

    /**
     * Updating a tag
     */
    public int updateTag(Tag tag) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_TAG_NAME, tag.getTagName());

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

    /**
     * Deleting a tag
     */
    public void deleteTag(Tag tag, boolean should_delete_all_tag_todos) {
        SQLiteDatabase db = this.getWritableDatabase();

        // before deleting tag
        // check if todos under this tag should also be deleted
        if (should_delete_all_tag_todos) {
            // get all todos under this tag
            List<Todo> allTagToDos = getAllToDosByTag(tag.getTagName());

            // delete all todos
            for (Todo todo : allTagToDos) {
                // delete todo
                deleteToDo(todo.getId());
            }
        }

        // now delete the tag
        db.delete(TABLE_TAG, KEY_ID + " = ?",
                new String[] { String.valueOf(tag.getId()) });
    }

    // ------------------------ "todo_tags" table methods ----------------

    /**
     * Creating todo_tag
     */
    public long createTodoTag(long todo_id, long tag_id) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_TODO_ID, todo_id);
        values.put(KEY_TAG_ID, tag_id);
        values.put(KEY_CREATED_AT, getDateTime());

        long id = db.insert(TABLE_TODO_TAG, null, values);

        return id;
    }

    /**
     * Updating a todo tag
     */
    public int updateNoteTag(long id, long tag_id) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_TAG_ID, tag_id);

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

    /**
     * Deleting a todo tag
     */
    public void deleteToDoTag(long id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_TODO, KEY_ID + " = ?",
                new String[] { String.valueOf(id) });
    }

    // closing database
    public void closeDB() {
        SQLiteDatabase db = this.getReadableDatabase();
        if (db != null && db.isOpen())
            db.close();
    }

    /**
     * get datetime
     * */
    private String getDateTime() {
        SimpleDateFormat dateFormat = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss", Locale.getDefault());
        Date date = new Date();
        return dateFormat.format(date);
    }
}
```

What's Next ?

An example of Todo application is coming soon … stay tuned …

You May Also Like

Android RSS Reader Application using SQLite Part 1

Android SQLite Database Tutorial

Android RSS Reader Application using SQLite Part 2

Android Login and Registration with PHP, MySQL and SQLite