SQLite Tutorial

1. Introduction

When we use Android for development, we inevitably use data, so we need to use a database to operate on the data. This time we will use the SQLite, which is built-in the Android system. SQLite is a lightweight relational database. It has fast computing speed, takes up few resources, and only takes up a few of storage space in the memory.

2. APIs and Data Types

Android SDK provides a series of classes and interfaces for operating databases. This time we are going to use the commonly used SQLiteOpenHelper class. This class is an abstract class used for database creation and database version updates.

Commonly Used Methods in SQLiteOpenHelper Class

onCreate(SQliteDatabase db)	Called when the database is first created
onUpgrade(SQliteDatabase db, int v, int v2)	Called when the database is upgraded
getWritableDatabase()	Open a read/write database
getReadableDatabase()	Open a readable database
query(), rawQuery()	Query database
insert()	Insert data
delete()	Delete data
close()	Close database

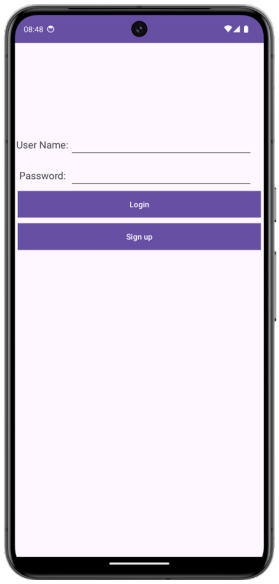
SQLite supports five main data types, including null, integer, real, text, and blob. The introduction of various data types is as follows.

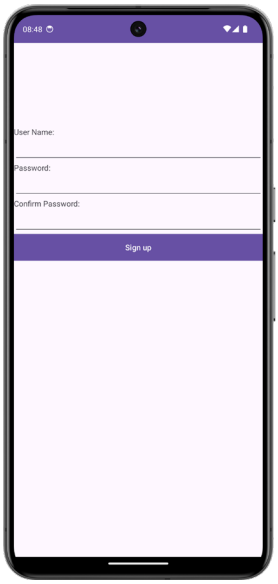
Datatypes of SQLite

null	null or missing value
integer	integer type
real	floating point type
text	text type
blob	binary type

3. Usage

Firstly, we should create a project. This time we are going to take login and register interface as an example. is as follows. Due to detailed understanding in lectures and space limitations, the code for the two interface XML files will no longer be provided here. The specific interface is shown in the following figure.





Create a class that inherits SQLiteOpenHelper

Inherit SQLiteOpenHelper, and implement the onCreate method and onUpgrade method to create and upgrade the database.

```
public class UserDatabase extends SQLiteOpenHelper {
    2 usages
    public UserDatabase(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    10 usages
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

Create database table

Determine the fields and types of the table, and define the user table statement.

```
public class UserDatabase extends SQLiteOpenHelper {
    no usages
    public static final String CREATE_USER = "create table users("
            + "id integer primary key autoincrement,"
            + "user_name text,"
            + "user_pwd text)";
```

So far, we have created a data table named users with user ID, name, and password fields. Don't forget to call SQLiteDatabase.execSQL(CREATE_USER) in the onCreate method, which contains the attributes we just defined. At the same time, we insert two initial usernames and passwords into the data table, i.e. user name "test01" with password "abc12345", and user name "test02" with password "2580".

```
@Override
public void onCreate(SQLiteDatabase db) {

    db.execSQL(CREATE_USER);

    ContentValues values = new ContentValues();
    values.put("user_name","test01");
    values.put("user_pwd", "abc12345");
    db.insert( table: "users", nullColumnHack: null, values);
    values.clear();

    values.put("user_name","test02");
    values.put("user_pwd", "2580");
    db.insert( table: "users", nullColumnHack: null, values);
    values.clear();
}
```

Add Database to Login Activity

Add the database to MainActivity (login module), add a database instance, and then initialize the database.

```
UserDatabase userDatabase;
2 usages
SQLiteDatabase db;

@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    name = this.findViewById(R.id.name);
    pwd = this.findViewById(R.id.pwd);
    login = this.findViewById(R.id.login);
    register = this.findViewById(R.id.register);

    userDatabase = new UserDatabase( context: this, name: "Userinfo", factory: null, version: 1);
    db = userDatabase.getReadableDatabase();
    login.setOnClickListener(this);
    register.setOnClickListener(this);
}
```

In the onClick method, we obtain the input username and password, and use a query statement to query in the users data table. If the returned result is not empty, it indicates that the account and password are entered by the user correctly, so that login is allowed. SQLiteDatabase.query can execute SQL query statements to the given table, and its return type is Cursor interface, which is used as the return value in database operations.

When using the SQLiteDatabase database, it is important to close it in a timely manner, otherwise a SQLiteException exception may be thrown.

```
@Override
public void onClick(View v) {

    if (v.getId() == R.id.login) {
        String username = name.getText().toString();
        String password = pwd.getText().toString();
        Cursor cursor = db.query( table: "users", new String[]{"user_name", "user_pwd"},
                selection: " user_name=? and user_pwd=?", new String[]{username, password}, groupBy: null, having: null, orderBy: null);

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

        if (flag != 0) {
            Toast.makeText( context: MainActivity.this, text: "Login successfully!", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText( context: MainActivity.this, text: "Wrong user or password!", Toast.LENGTH_LONG).show();
        }
    }
    if (v.getId() == R.id.register) {
        Intent intent = new Intent( packageContext: MainActivity.this, Register.class);
        startActivity(intent);
    }
}
```

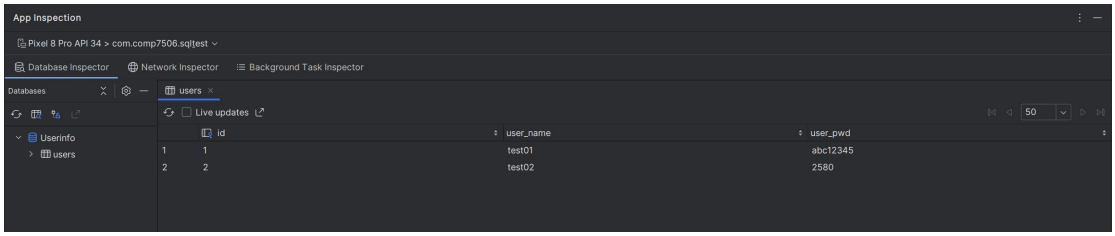
Add Database to Register Activity

Complete the registration module's functions. The specific implementation method is the same as the previous section. We will not provide a detailed explanation of whether the user input during the registration process is legal and whether the user already exists in the database due to space limitations. We believe that these codes are basic enough for everyone to easily understand, anyway.

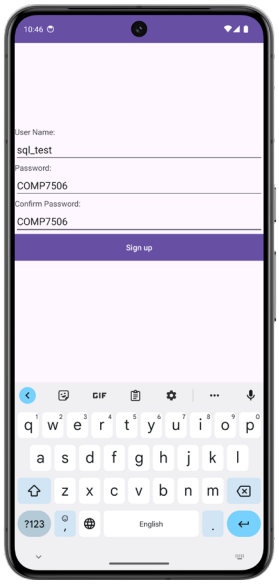
```
@Override
public void onClick(View v){
    boolean flag = true;
    String name = usr_name.getText().toString();
    String pwd1 = usr_pwd.getText().toString();
    String pwd2 = usr_pwd2.getText().toString();
    if(name.isEmpty() || pwd1.isEmpty() || pwd2.isEmpty()){
        Toast.makeText( context: Register.this, text: "Can not be null!", Toast.LENGTH_LONG).show();
    }
    else{
        Cursor cursor = db.query( table: "users",new String[]{"user_name"}, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
        while (cursor.moveToNext()){
            if(cursor.getString( columnIndex: 0).equals(name)){
                flag = false;
                break;
            }
        }
        cursor.close();
        if(flag){
            if (pwd1.equals(pwd2)) {
                ContentValues values = new ContentValues();
                values.put("user_name",name);
                values.put("user_pwd",pwd1);
                db.insert( table: "users", nullColumnHack: null,values);
                values.clear();

                Intent intent = new Intent();
                intent.setClass( packageContext: Register.this,MainActivity.class);
                startActivity(intent);
                db.close();
                Toast.makeText( context: Register.this, text: "Sign up successfully!", Toast.LENGTH_LONG).show();
            }
            else {
                Toast.makeText( context: Register.this, text: "Inconsistent password!", Toast.LENGTH_LONG).show();
            }
        }
        else{
            Toast.makeText( context: Register.this, text: "The user already exists!", Toast.LENGTH_LONG).show();
        }
    }
}
```

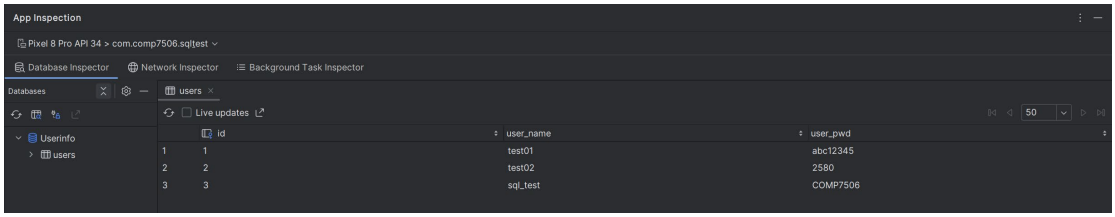
4. Demonstration

At this point, we have completed the writing of all the codes. Run the APP and in the Database Inspector, we can monitor the created data tables. As shown in the figure, the two records we pre-inserted have been correctly displayed.



Register a user with name “sql_test” and password “COMP7506” through the Register activity, and upon further observation on the Database Inspector, it can be seen that the data table has been updated as expected.





5. Summary

SQLiteOpenHelper is an auxiliary class used in Android systems to manage SQLite databases. Through the methods it provides, we can easily create and manage SQLite databases. It is simple to use, the code is easy to understand, and suitable for beginners.

You can obtain all the codes through https://github.com/MengqianCao/SQLite_test.