Birzeit University
Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Advance Computer Systems Engineering Lab ENCS515

EXP. No. 4. SQLite Database

1. Objectives

- How to create your own database using SQLite.
- How to create custom Views from scratch to suit a specific need.
- How to create Confirmation Dialogs.

2. Introduction

In this lab, you will be learning how to save data to a database for repeating or structured data, such as contact information. This experiment assumes that you are familiar with SQL databases in general and helps you get started with SQLite databases on Android.

2.1. Define a schema and contract

One of the main principles of SQL databases is the schema: a formal declaration of how the database is organized. The schema is reflected in the SQL statements that you use to create your database. You may find it helpful to create a companion class, known as a contract class, which explicitly specifies the layout of your schema in a systematic and self-documenting way.

A contract class is a container for constants that define names for URIs, tables, and columns. The contract class allows you to use the same constants across all the other classes in the same package. This lets you change a column name in one place and have it propagate throughout your code.

2.2. Create a database using an SQL helper

Once you have defined how your database looks, you should implement methods that create and maintain the database and tables. Just like files that you save on the device's internal storage, Android stores your database in your app's private folder. Your data is secure, because by default this area is not accessible to other apps or the user. The SQLiteOpenHelper class contains a useful set of APIs for managing your database. When you use this class to obtain references to your database, the system performs the potentially long-running operations of creating and updating the database only when needed and not during app startup. All you need to do is call getWritableDatabase() or getReadableDatabase().

2.3. Put information into a database

Insert data into the database by passing a ContentValues object to the insert(). The first argument for insert() is simply the table name. The second argument tells the framework what to do if the ContentValues is empty. The insert() methods returns the ID for the newly created row, or it will return -1 if there was an error inserting the data. This can happen if you have a conflict with pre-existing data in the database.

2.4. Read information from a database

To read from a database use the query() method, passing it your selection criteria and desired columns. The method combines elements of insert() and update(), except the column list defines the data you want to fetch (the "projection"), rather than the data to insert. The results of the query are returned to you in a Cursor object.

2.5. onCreate and onResume methods in Activities

- onCreate()

You must implement this callback, which fires when the system first creates the activity. On activity creation, the activity enters the Created state.

- onResume()

When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user.

3. Procedure

You are going to upgrade the second experiment to save the customers in your app's private folder using SQLiteOpenHelper API, rather than saving it in array List as you did in the second experiment. You will not change the layout of the second experiment so start by opening the second experiment project.

3.1. Creating DataBaseHelper class that extends SQLiteOpenHelper Class

Create a new class and call it DataBaseHelper by right-Cliking on the package name to add a new class, fill the name to "DataBaseHelper" and the Superclass to "android.database.sqlite.SQLiteOpenHelper" and then click ok as shown in Figure 4.1.

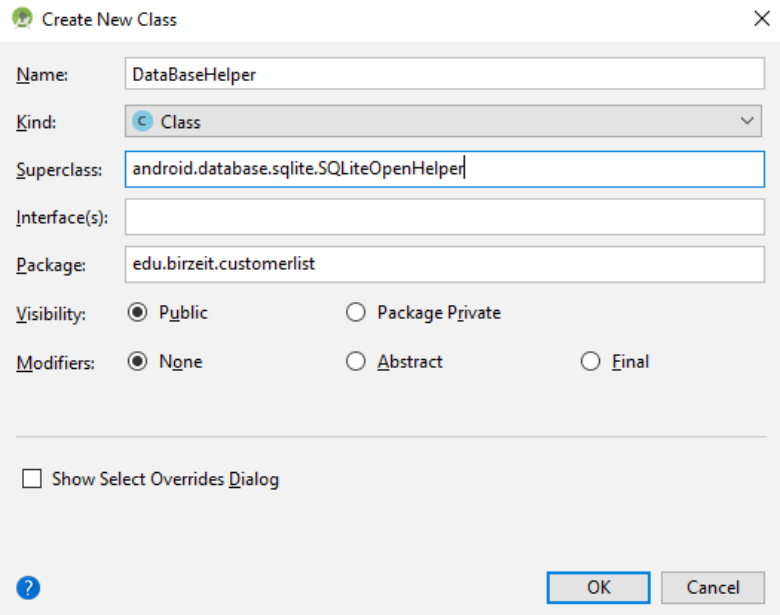

Figure 4.1 Adding DataBaseHelper Class

- Add a constructor to the DataBaseHelper class, this constructor will call the super constructor which will create the data base, this constructor will take four values (Context, name of the database, factory, and the version of the data base). As shown in the code below.

```
public DataBaseHelper(Context context, String name,
SQLiteDatabase.CursorFactory factory, int version) {
    super(context, name, factory, version);
}
```

- override the onCreate and OnUpgrade methods

As shown in Figure 4.2 override the onCreate and onUpgrade methods. onCreate will create the table and onUpgrade will update the table if needed in the future, In this lab you will implement onCreate method.

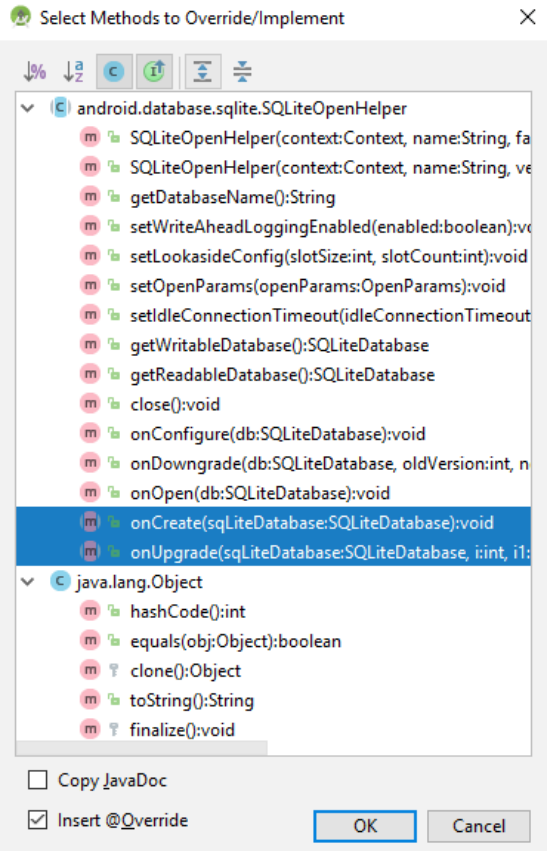

Figure 4.2 Overriding OnCreate and OnUpgrade methods

- implementing onCreate method

You will execute the query for creating a customer table, this table has ID column with prototype Long and it's the primary key, a NAME column with prototype Text, a PHONE column with prototype Text and a GENDER entry with prototype Text as shown in the code below.

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    sqLiteDatabase.execSQL("CREATE TABLE CUSTOMER(ID LONG PRIMARY
KEY,NAME TEXT, PHONE TEXT,GENDER TEXT)")
);
}
```

- implement a method to add a customer

Before inserting you will create an object from SQLiteDataBase which will call the getWritableDatabase() method, this will give access to write to the database. To add a Customer to the data base you will use the insert method from the SQLiteDataBase class. This method takes three values, the fist is the table name to insert the value, the second will be set to null (If you specify null, the framework does not insert a row when there are no values) and a ContentValues which the entry that will be used to be added to the DataBase. The code below shows the method to insert an entry to the Customer Table.

```
public void insertCustomer(Customer customer) {
    SQLiteDatabase sqLiteDatabase = getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("ID", customer.getmCustomerId());
    contentValues.put("NAME", customer.getmName());
    contentValues.put("PHONE", customer.getmPhone());
    contentValues.put("GENDER", customer.getmGender());
    sqLiteDatabase.insert("CUSTOMER", null, contentValues);
}
```

- implementing a method to get all Customers from Customer table

To get entry from the database tables, you will create an object from SQLiteDataBase which will call the getReadableDatabase () method, this will give access to read from the database. To get all customers you will execute a raw query which will select from the customer table all the entries. The returned value will be as Cursor. (read about Cursor to know how to deal with them Link.). The code below shows how to get all customers from the Customer table in the database.

```
public Cursor getAllCustomers() {
    SQLiteDatabase sqLiteDatabase = getReadableDatabase();
    return sqLiteDatabase.rawQuery("SELECT * FROM CUSTOMER", null);
}
```

3.2. Removing the ArrayList and saving the Values to the Customer Table

- Go to the Customer Class, remove the customersArrayList (since you will not need for it)
- In the AddCustomerActivity you will replace the Customer.customersArrayList.add(newCustomer) with a new code to add to the Customer table in the data base.

  - First create an object from the DataBaseHelper class. And pass the context (AddCustomerActivity.this), name of the database "EXP4", the factory to "null" and the version to 1 as shown in the code below. Then call the insertCustomer method by passing the customer object to it.

```
DataBaseHelper dataBaseHelper =new
DataBaseHelper(AddCustomerActivity.this,"DB_NAME_EXP4",null,1);
dataBaseHelper.insertCustomer(newCustomer);
```

3.3. Displaying all Customers that were added to the Customer Table.

- In the MainActivity make the declaration of the secondLinearLayout global. (only the declaration. Do not create new object, the creation must be in the onCreate method)
- Remove the code which display the old Customer.customersArrayList.
- Override the OnResume method.
- Get All customers from the Customer table form the database

  - First create an object from the DataBaseHelper class. And pass the context (MainActivity.this), name of the database "EXP4", the factory to "null" and the version to 1 as shown in the code below.
  - Create a Cursor and call the getAllCustomers method the returned value will be saved in the created Cursor.

- Display All Customers returned from the database

  - Remove all old views from the secondLinearLayout by calling removeAllViews Method.
  - Display all customers returned from the Cursor in textViews
  - Add the textviews to the second Linerlayout.

- The code below shows how the implementation of onResume method

```
protected void onResume() {
    super.onResume();
    DataBaseHelper dataBaseHelper =new
DataBaseHelper(MainActivity.this,"EXP4", null,1);
    Cursor allCustomersCursor = dataBaseHelper.getAllCustomers();
    secondLinearLayout.removeAllViews();
    while (allCustomersCursor.moveToNext()){
        TextView textView =new TextView(MainActivity.this);
        textView.setText(
                "Id= "+allCustomersCursor.getString(0)
                +"\nName= "+allCustomersCursor.getString(1)
                +"\nPhone= "+allCustomersCursor.getString(2)
                +"\nGender= "+allCustomersCursor.getString(3)
                +"\n\n"
        );
        secondLinearLayout.addView(textView);
    }

}
```

- The output is shown in Figure 4.3

Figure 4.3 Experiment 4 Output

4. Todo

This part will be given to you by the teacher assistant in the lab time.