جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأنظمة الطبية الذكية
Intelligent Medical Systems Department

Subject: SQLite database management system

Class: 3rd

Lecturer: Asst.Lect Mustafa Ameer Awadh

Lecture: (9)

Study Year: 2024-2025

Introduction

Most the techniques you have seen are useful for saving simple sets of data. To manage persistent data. For saving relational data, we can store the data in a file or in a database which is much more efficient. For example, if you want to store the test results of all the students in a school, it is much more efficient to use a database to represent them because you can use database querying to retrieve the results of specific students. Using databases enables you to enforce data integrity by specifying the relationships between different sets of data.

Android SQLite Database

SQLite is a popular choice for embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages. In Android, SQLite is one way of storing application data. It is a very lightweight database that comes with Android OS. SQLite is a relational database management system (RDBMS).

SQLite

SQLite is a lightweight and powerful, SQLite differs from many conventional database engines by using a loosely typed approach to column definitions. Rather than requiring column values to conform to a single type, the values in each row for each column are individually typed. As a result, there's no strict type checking when assigning or extracting values from each column within a row.

Database Package & Classes

- The main package is android.database.sqlite that contains the classes to manage your own databases
- The main class can be used to manage your own databases:

Class	Description
SQLiteOpenHelper	Extend this abstract class to manage a database and its version. We must override the onCreate and onUpgrade methods.
SQLiteDatabase	Includes methods to execute SQL statements
Cursor	Encapsulates a table returned by a select SQL query.

Database Packets & Classes

- SQLiteOpenHelper class: A helper class to manage database creation and version management.
- You create a subclass implementing onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and optionally onOpen(SQLiteDatabase), and this class takes care of opening the database if it exists, creating it if it does not, and upgrading it as necessary.
- SQLiteDatabase class: SQLiteDatabase has methods to create, delete, execute SQL commands, and perform other common database management tasks.

- Cursor and Content Values: ContentValues objects are used to insert new rows into database tables (and Content Providers).

- Each Content Values object represents a single row, as a map of column names to values.
- Queries in Android are returned as Cursor objects.
- Rather than extracting and returning a copy of the result values, Cursors act as pointers to a subset of the underlying data.
- Cursors are a managed way of controlling your position (row) in the result set of a database query.
- The Cursor class includes several functions to navigate query results including, but not limited to, the following:
  1. moveToFirst: Moves the cursor to the first row in the query result.
  2. moveToNext: Moves the cursor to the next row.
  3. moveToPrevious: Moves the cursor to the previous row.
  4. getCount: Returns the number of rows in the result set.
  5. getColumnIndexOrThrow: Returns an index for the column with the specified name (throwing an exception if no column exists with that name).
  6. getColumnName: Returns the name of the specified column index.
  7. getColumnNames: Returns a String array of all the column names in the current cursor.
  8. moveToPosition: Moves the cursor to the specified row.
  9. getPosition: Returns the current cursor position.

Opening and Creating Databases Using the SQLiteOpenHelper

SQLiteOpenHelper is an abstract class that wraps up the best practice pattern for creating, opening, and upgrading databases. By implementing and using a SQLiteOpenHelper, you hide the logic used to decide if a database needs to be created or upgraded before it's opened. You need to extend the SQLiteOpenHelper class by overriding the constructor, and to add onCreate, and onUpgrade methods to handle. the creation of a new database and upgrading to a new version, respectively.To use an implementation of the helper class, create a new instance, passing in the context, database name, current version, and a CursorFactory (if you're using one).

Call getReadableDatabase or getWriteableDatabase to open and return a readable/writable instance of the database. Behind the scenes, if the database doesn't exist, the helper executes its onCreate handler. If the database version has changed, the onUpgrade handler will fire. In both cases, the get<read/write>ableDatabase call will return the existing, newly created, or upgraded database as appropriate.

Querying Your Database

To execute a query on a database, use the query method on the database object, passing in:

1. An optional Boolean that specifies if the result set should contain only unique values.
2. The name of the table to query.
3. A projection, as an array of Strings, that lists the columns to include in the result s step-by-step guide et.
4. A "where" clause that defines the rows to be returned. You can include? wildcards that will be replaced by the values stored in the selection argument parameter.
5. An array of selection argument strings that will replace the ?'s in the "where" clause.
6. A "group by" clause that defines how the resulting rows will be grouped.
7. A "having" filter that defines which row groups to include if you specified a "group by" clause.
8. A String that describes the order of the returned rows.
9. An optional String that defines a limit to the returned rows.

step-by-step guide

Step 1– Create new Android project. Provide Activity name as SQLiteApp as shown below:

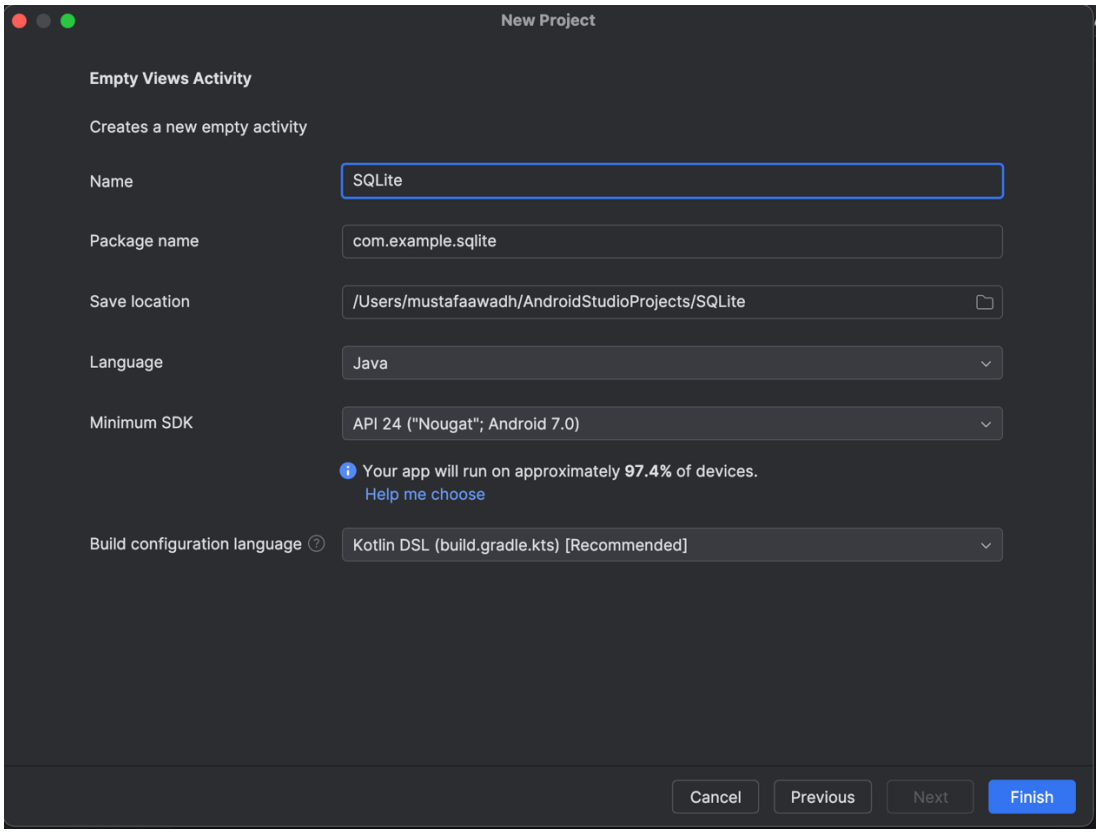

Fig 1. create a new project.

Step 2 – Add components in the main activity as shown in the picture.

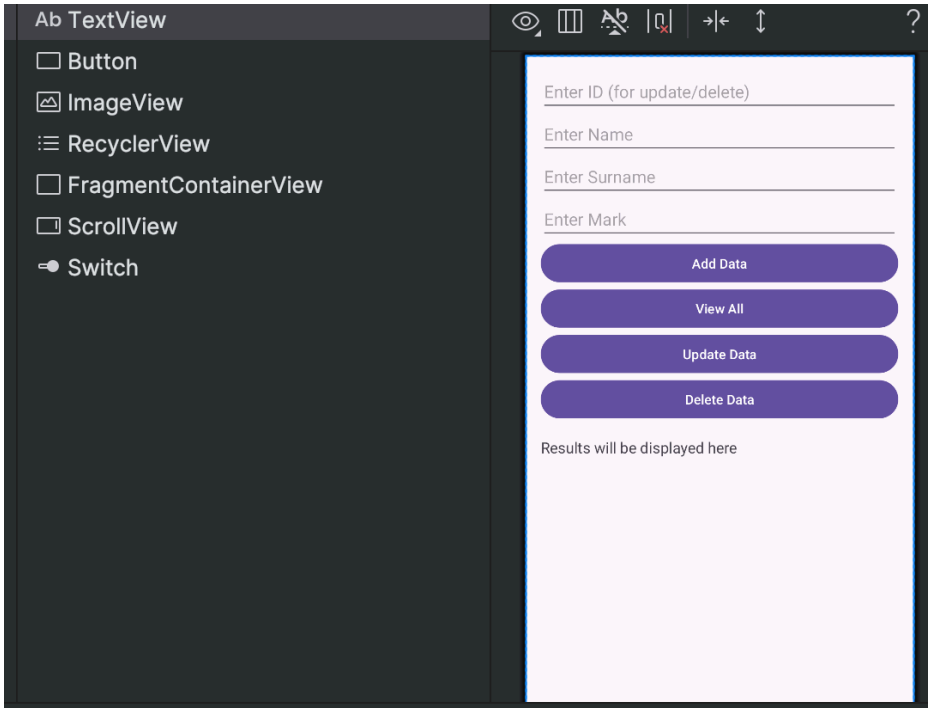

Fig 2. Adding components to main activity.

Step 3 – Now create a new Java class called DatabaseHelper.java.

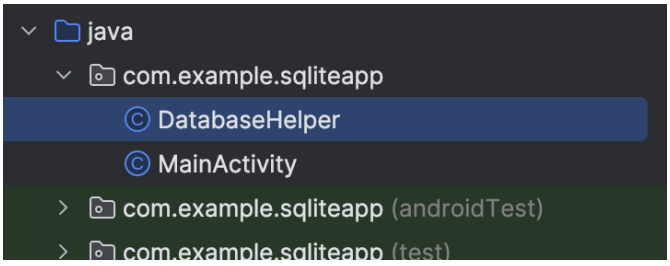

Fig 3. Create a java class (DatabaseHelper).

- Extend this class from SQLiteOpenHelper
- Import the class library
- import android.database.sqlite.SQLiteOpenHelper;
- Impalement its abstract methods onCreate(), onUpgrade().
- Create constructor matching super (first one).
- Declare following variables in the DatebaseHelper class.

- public static final String DATABASE_NAME = "Student.db";
- public static final String TABLE_NAME = "student_table";
- public static final String COL_1 = "ID";
- public static final String COL_2 = "NAME";
- public static final String COL_3 = "SURNAME";
- public static final String COL_4 = "MARKS";

step 4 - write the constructor as following:

```
public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
SQLiteDatabase db = this.getWritableDatabase();
    }
```

Add the following code to onCreate and onUpgrate methods

```
@Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE_NAME +" (ID INTEGER
PRIMARY KEY AUTOINCREMENT,NAME TEXT,SURNAME
TEXT,MARKS INTEGER)");
 }
    @Override
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);
        onCreate(db);
```

Step 5 - In the main Activity java class, define an instance from your newly created class.

DatabaseHelper myDb = **new** DatabaseHelper(**this**);

Step 6 - Run the application.