

ĐỀ CƯƠNG BÀI GIẢNG

BÀI 10: THỰC HÀNH TƯƠNG TÁC VỚI CƠ SỞ DỮ LIỆU SQLite

Nội dung bài học trước khi lên lớp (trang 1 đến trang 4):

- Thực hành sử dụng SQLite, thực hiện các thao tác thêm bớt, sửa, xóa trên CSDL, hiển thị dữ liệu bằng listView: phiếu thực hành Bài 10.1

Nội dung bài học thực hiện lên lớp (trang 4 đến hết):

- Thực hành xây dựng ứng dụng quản lý sách với SQLite: phiếu thực hành bài 10.2

Nội dung bài học sau khi lên lớp: Làm bài tập 10.3, bài 10.4, bài 10.5

NỘI DUNG BÀI HỌC

1. Tổng quan về SQLite	1
1.1. Các lớp SQLite.....	2
1.2. Một số hạn chế của SQLite	4
2. Thực hành tạo ứng dụng quản lý sinh viên đơn giản.....	5
2.1. Cơ sở dữ liệu	5
2.2. Tạo lớp mô hình dữ liệu (Data model class).....	5
2.3. Tạo lớp xử lý dữ liệu (Data Handler Class).....	7

1. Tổng quan về SQLite

SQLite là hệ quản trị cơ sở dữ liệu quan hệ giống các hệ khác như SQL Server, MySQL, Oracle, v.v. Các hệ quản trị như SQL Server, MySQL, Oracle, v.v. là các hệ xử lý độc lập và ứng dụng sẽ kết nối đến khi cần truy cập dữ liệu. Tuy nhiên, khác với các hệ trên, SQLite là một hệ nhúng với hình thức là một thư viện được liên kết đến ứng dụng.

Cơ sở dữ liệu SQLite có thể được truy cập bằng cách dùng ngôn ngữ truy vấn SQL (Structured Query Language).

1.1. Các lớp SQLite

Android SDK cung cấp một tập các lớp hỗ trợ ứng dụng truy cập đến cơ sở dữ liệu SQLite. Có 4 lớp cơ bản cùng với các phương thức được mô tả như bảng sau đây.

Lớp	Phương thức
Cursor: dùng để truy cập đến tập kết quả từ truy vấn dữ liệu thông qua các thể hiện (instance).	<ul style="list-style-type: none">• close(): giải phóng tài nguyên và đóng thể hiện lớp Cursor• getCount(): trả về số hàng từ tập kết quả• moveToFirst(): di chuyển đến hàng đầu tiên từ tập kết quả• moveToLast(): di chuyển đến hàng cuối cùng từ tập kết quả• moveToNext(): di chuyển đến hàng kế tiếp từ tập kết quả• move(): di chuyển từ vị trí hiện tại đến vị trí mới theo một khoảng cách cho trước.• get<type>(): trả về giá trị kiểu <type> tại cột xác định của hàng tại vị trí cursor hiện tại. Các phương thức phổ biến gồm getString(), getInt(), getFloat(), getDouble(), getShort().
SQLiteDatabase: cung cấp giao diện giữa ứng dụng và cơ sở dữ liệu SQLite cho phép tạo, xóa và thực hiện các truy vấn SQL.	<ul style="list-style-type: none">• insert(): chèn một hàng mới vào bảng• delete(): xóa hàng từ bảng• query(): thực hiện truy vấn và trả về kết quả phù hợp thông qua một đối tượng Cursor• execSQL(): thực hiện một lệnh truy vấn SQL đơn và không trả về kết quả• rawQuery(): thực hiện truy vấn và trả về kết quả phù hợp thông qua một đối tượng Cursor
SQLiteOpenHelper: lớp này được tạo giúp cho việc tạo và cập nhật cơ sở dữ liệu được dễ dàng hơn. Lớp này phải có lớp con thực thi các phương thức onCreate() và onUpgrade() .	<ul style="list-style-type: none">• onCreate(): được gọi khi cơ sở dữ liệu được tạo lần đầu• onUpgrade(): được gọi khi ứng dụng chứa nhiều phiên bản cơ sở dữ liệu• getWritableDatabase(): mở hay tạo một cơ sở dữ liệu cho việc đọc và ghi. Trả về một tham chiếu

	<p>đến một cơ sở dữ liệu trong hình thức một đối tượng <i>SQLiteDatabase</i></p> <ul style="list-style-type: none">• <i>getReadableDatabase()</i>: mở hay tạo một cơ sở dữ liệu chỉ cho việc đọc. Trả về một tham chiếu đến một cơ sở dữ liệu trong hình thức một đối tượng <i>SQLiteDatabase</i>• <i>close()</i>: đóng cơ sở dữ liệu
<i>ContentValues</i> : cho phép khai báo các cột và giá trị bên trong nó dưới dạng các cặp khóa/giá trị (<i>key/value</i>). Hữu ích khi chèn hay cập nhật dữ liệu đến bảng.	<ul style="list-style-type: none">• <i>put()</i>: thêm một giá trị đến tập dữ liệu
Lớp	Phương thức
<i>Cursor</i> : dùng để truy cập đến tập kết quả từ truy vấn dữ liệu thông qua các thể hiện (<i>instance</i>).	<ul style="list-style-type: none">• <i>close()</i>: giải phóng tài nguyên và đóng thể hiện lớp <i>Cursor</i>• <i>getCount()</i>: trả về số hàng từ tập kết quả• <i>moveToFirst()</i>: di chuyển đến hàng đầu tiên từ tập kết quả• <i>moveToLast()</i>: di chuyển đến hàng cuối cùng từ tập kết quả• <i>moveToNext()</i>: di chuyển đến hàng kế tiếp từ tập kết quả• <i>move()</i>: di chuyển từ vị trí hiện tại đến vị trí mới theo một khoảng cách cho trước.• <i>get<type>()</i>: trả về giá trị kiểu <i><type></i> tại cột xác định của hàng tại vị trí <i>cursor</i> hiện tại. Các phương thức phổ biến gồm <i>getString()</i>, <i>getInt()</i>, <i>getFloat()</i>, <i>getDouble()</i>, <i>getShort()</i>.
<i>SQLiteDatabase</i> : cung cấp giao diện giữa ứng dụng và cơ sở dữ liệu <i>SQLite</i> cho phép tạo, xóa và thực hiện các truy vấn <i>SQL</i> .	<ul style="list-style-type: none">• <i>insert()</i>: chèn một hàng mới vào bảng• <i>delete()</i>: xóa hàng từ bảng• <i>query()</i>: thực hiện truy vấn và trả về kết quả phù hợp thông qua một đối tượng <i>Cursor</i>• <i>execSQL()</i>: thực hiện một lệnh truy vấn <i>SQL</i> đơn và không trả về kết quả• <i>rawQuery()</i>: thực hiện truy vấn và trả

	về kết quả phù hợp thông qua một đối tượng Cursor
SQLiteOpenHelper: lớp này được tạo giúp cho việc tạo và cập nhật cơ sở dữ liệu được dễ dàng hơn. Lớp này phải có lớp con thực thi các phương thức <code>onCreate()</code> và <code>onUpgrade()</code> .	<ul style="list-style-type: none">• <code>onCreate()</code>: được gọi khi cơ sở dữ liệu được tạo lần đầu• <code>onUpgrade()</code>: được gọi khi ứng dụng chứa nhiều phiên bản cơ sở dữ liệu• <code>getWritableDatabase()</code>: mở hay tạo một cơ sở dữ liệu cho việc đọc và ghi. Trả về một tham chiếu đến một cơ sở dữ liệu trong hình thức một đối tượng <code>SQLiteDatabase</code>• <code>getReadableDatabase()</code>: mở hay tạo một cơ sở dữ liệu chỉ cho việc đọc. Trả về một tham chiếu đến một cơ sở dữ liệu trong hình thức một đối tượng <code>SQLiteDatabase</code>• <code>close()</code>: đóng cơ sở dữ liệu
ContentValues: cho phép khai báo các cột và giá trị bên trong nó dưới dạng các cặp khóa/giá trị (<code>key/value</code>). Hữu ích khi chèn hay cập nhật dữ liệu đến bảng.	<ul style="list-style-type: none">• <code>put()</code>: thêm một giá trị đến tập dữ liệu

1.2. Một số hạn chế của SQLite

SQLite do tính năng nhỏ gọn tích hợp cùng ứng dụng nên cũng có một số hạn chế :

STT	Đặc điểm	Mô tả
1	RIGHT OUTER JOIN	Chỉ có LEFT OUTER JOIN được thực hiện.
2	FULL OUTER JOIN	Chỉ có LEFT OUTER JOIN được thực hiện.
3	ALTER TABLE	Các biến thể RENAME TABLE và ADD COLUMN của lệnh ALTER TABLE được hỗ trợ. DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT không được hỗ trợ.
4	Trigger support	Trigger FOR EACH ROW được hỗ trợ nhưng không hỗ trợ FOR EACH STATEMENT.
5	VIEWS	VIEWS trong SQLite là chỉ đọc. Bạn không thể thực thi câu lệnh DELETE, INSERT hoặc UPDATE trên một view.
6	GRANT và REVOKE	Các quyền truy cập duy nhất có thể được áp dụng là các quyền truy cập file thông thường (normal file) của hệ điều hành.

2. Thực hành tạo ứng dụng quản lý sinh viên đơn giản

2.1. Cơ sở dữ liệu

Chúng ta sẽ tạo một ứng dụng Android tên SQLiteDemoApplication (tên activity là SQLiteDemoApplicationActivity và tên layout tương ứng là activity_sqlite_demo_application) tương tác với cơ sở dữ liệu tên StudentsDB.db gồm một bảng Students có lược đồ như sau:

Column	DataType
StudentID	Integer/PK
StudentName	Text

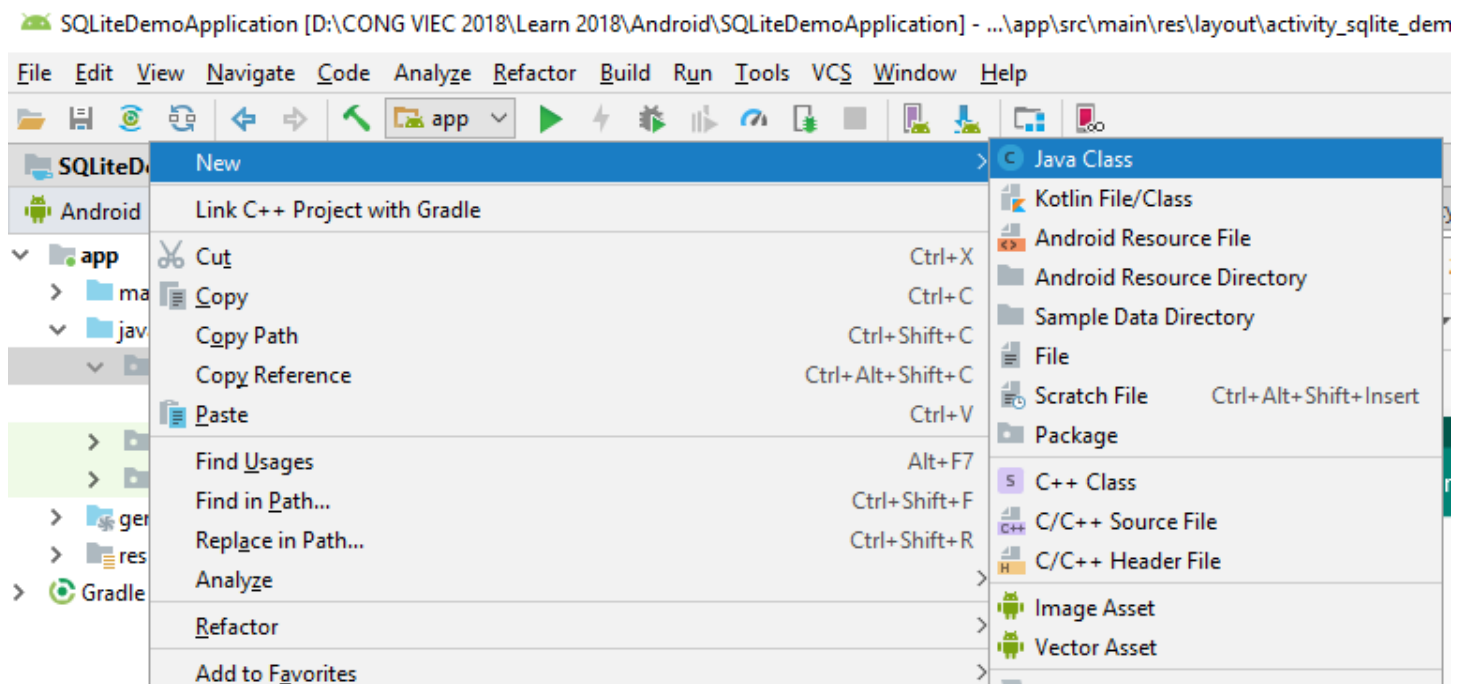
Lúc này chúng ta có một lớp Activity (SQLiteDemoApplicationActivity) và một cơ sở dữ liệu (StudentsDB.db) như hình minh họa:



2.2. Tạo lớp mô hình dữ liệu (Data model class)

Để có thể thể tương tác với bảng Students bằng mã Java trong ứng dụng, chúng ta cần chuyển bảng cơ sở dữ liệu thành đối tượng bằng cách tạo lớp Student như sau:

Tạo lớp tên Student bằng cách nhấn chuột phải vào tên gói chọn New > Java Class :



Lớp Student (trong tập tin Student.java) có nội dung như sau:

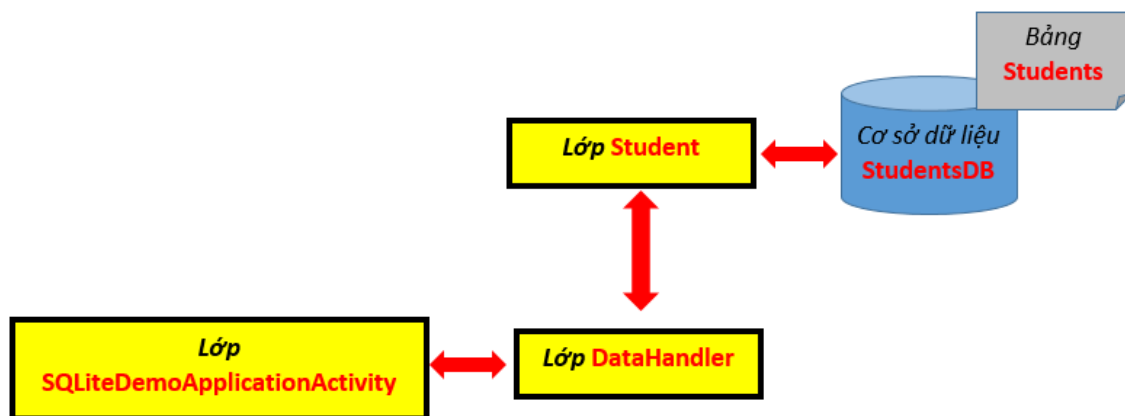
```
package com.ngocminhtran.sqlitedemoapplication;
```

```
public class Student {  
    //các biến tương ứng với các cột của bảng Students  
    private int _studentid;  
    private String _studentname;  
  
    //Các phương thức khởi tạo (constructors)  
    //Phương thức khởi tạo mặc định  
    public Student(){  
  
    }  
    //Phương thức khởi tạo có tham số  
    public Student(int id, String name){  
        this._studentid = id;  
        this._studentname = name;  
    }  
  
    //các phương thức truy cập các biến thành viên  
  
    public int getStudentID(){  
        return this._studentid;  
    }  
    public void setStudentID(int id){  
        this._studentid = id;  
    }  
}
```

```
public String getStudentName(){  
    return this._studentname;  
}  
public void setStudentName(String name){  
    this._studentname = name;  
}  
}
```

2.3. Tạo lớp xử lý dữ liệu (Data Handler Class)

Chúng ta đã tạo ra lớp Student, là lớp mô hình dữ liệu từ bảng Students của cơ sở dữ liệu StudentsDB.db. Bây giờ, chúng ta sẽ tạo lớp xử lý các truy vấn dữ liệu. Lớp xử lý dữ liệu, tên DataHandler, kế thừa từ lớp SQLiteOpenHelper, thực thi các phương thức onCreate(), onUpgrade() và các phương thức truy vấn dữ liệu như hiển thị dữ liệu, thêm dữ liệu, xóa dữ liệu hay cập nhật dữ liệu. Để dễ hình dung, mô hình ứng dụng của chúng ta lúc này gồm các lớp sau:



Như vậy, lớp Activity không tương tác một cách trực tiếp đến cơ sở dữ liệu mà thông qua các lớp trung gian (Student và DataHandler) làm cho việc xử lý được dễ dàng hơn.

Lớp DataHandler được tạo trong gói com.ngocminhtran.sqlitedemoapplication với nội dung như sau:

```
package com.ngocminhtran.sqlitedemoapplication;  
  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
  
public class DataHandler extends SQLiteOpenHelper {  
  
    @Override
```



```
public void onCreate(SQLiteDatabase db) {  
  
}  
  
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion,  
    int newVersion) {  
  
}  
}
```

Lúc này chúng ta thấy lớp sẽ bị lỗi vì chúng ta chưa thêm phương thức khởi tạo cho lớp DataHandler. Thêm các biến mô tả cơ sở dữ liệu và phương thức khởi tạo như sau:

```
package com.ngocminhtran.sqlitedemoapplication;  
  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
  
public class DataHandler extends SQLiteOpenHelper {  
  
    // các biến mô tả cơ sở dữ liệu  
    private static final int DATABASE_VERSION = 1;  
    private static final String DATABASE_NAME = "StudentsDB.db";  
    public static final String TABLE_NAME = "Students";  
    public static final String COLUMN_ID = "StudentID";  
    public static final String COLUMN_NAME = "StudentName";  
  
    //phương thức khởi tạo  
    public DataHandler(Context context, String name,  
        SQLiteDatabase.CursorFactory factory, int version) {  
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion,  
        int newVersion) {  
  
    }  
}
```



```
}
```

Phương thức khởi tạo sẽ khởi tạo cơ sở dữ liệu StudentsDB.db. Để tạo bảng Students chúng ta dùng phương thức `execSQL` của lớp `SQLiteDatabase` để thực hiện lệnh truy vấn tại bảng. Câu lệnh SQL tạo bảng như sau:

Tạo bảng Students:

```
CREATE TABLE Students (  
    COLUMN_ID INTEGER PRIMARY KEY,  
    COLUMN_NAME TEXT);
```

Lệnh SQL tạo bảng Students sẽ được chuyển thành dạng chuỗi để chuyển đến phương thức `execSQL()` thực thi trong phương thức `onCreate()` của lớp `DataHandler` như sau:

```
@Override  
public void onCreate(SQLiteDatabase db) {  
  
    //chuỗi lệnh truy vấn tạo bảng Students  
    String CREATE_STUDENTS_TABLE = "CREATE TABLE " +  
        TABLE_NAME + "("  
        + COLUMN_ID + " INTEGER PRIMARY KEY," +  
        COLUMN_NAME + " TEXT )";  
  
    //thực thi truy vấn  
    db.execSQL(CREATE_STUDENTS_TABLE);  
}
```

Phương thức `onUpgrade()` được gọi khi cần nâng cấp đến phiên bản cơ sở dữ liệu cao hơn.

```
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion,  
    int newVersion) {  
    //Xóa bảng nếu tồn tại  
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);  
    //Tạo bảng mới  
    onCreate(db);  
}
```

Chúng ta đã tạo cơ sở dữ liệu StudentsDB.db và bảng Students, bây giờ chúng ta sẽ thêm hai phương thức truy vấn dữ liệu là hiển thị dữ liệu (phương thức `loadDataHandler`) và thêm dữ liệu (phương thức `addDataHandler`). Các phương thức khác sẽ được bổ sung sau.

Hiển thị dữ liệu từ một bảng, cụ thể là bảng Students, chúng ta dùng lệnh truy vấn `SELECT` như sau:

```
SELECT *  
FROM Students;
```

Lệnh SQL trên sẽ được chuyển thành chuỗi và thực thi bằng phương thức `rawQuery()`. Kết quả trả về được lưu trong đối tượng `Cursor`. Hiển thị dữ liệu từ đối tượng `Cursor` dùng phương thức `moveToNext()`. Phương thức `loadDataHandler()` có nội dung như sau:

```
//hiển thị dữ liệu từ bảng Students  
public String loadDataHandler() {  
    String result = "";  
    //chuỗi truy vấn SELECT  
    String query = "SELECT* FROM " + TABLE_NAME;  
    //sẵn sàng thực thi các truy vấn  
    SQLiteDatabase db = this.getWritableDatabase();  
    //thực thi truy vấn bằng phương thức rawQuery()  
    //kết quả trả về lưu trong đối tượng Cursor  
    Cursor cursor = db.rawQuery(query, null);  
    //duyệt qua dữ liệu từ đối tượng Cursor  
    while (cursor.moveToNext()) {  
        //nhận giá trị cột thứ nhất (StudentID)  
        int result_0 = cursor.getInt(0);  
        //nhận giá trị cột thứ hai (StudentName)  
        String result_1 = cursor.getString(1);  
        //hiển thị mỗi hàng trong một chuỗi  
        result += String.valueOf(result_0) + " " + result_1 +  
            System.getProperty("line.separator");  
    }  
    //đóng đối tượng Cursor  
    cursor.close();  
    //đóng đối tượng SQLiteDatabase  
    db.close();  
    return result;  
}
```

Lưu ý rằng, phương thức `loadDataHandler` trả về chuỗi các hàng dữ liệu trong bảng `Students`.

Để thêm dữ liệu đến bảng `Students` chúng ta dùng đối tượng `ContentValues` và phương thức `put()` để lưu trữ dữ liệu trong đối tượng này. Dữ liệu từ đối tượng `ContentValues` được thêm vào bảng `Students` bằng phương thức `insert()` của đối tượng `SQLiteDatabase`. Nội dung phương thức `addDataHandler` như sau:

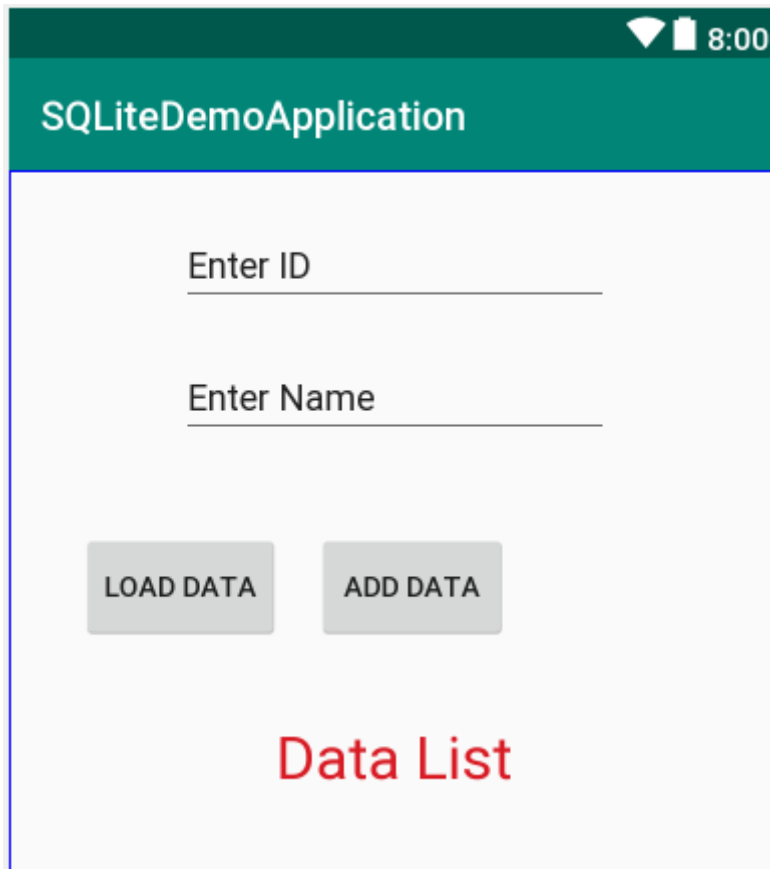
```
//thêm dữ liệu đến bảng Students  
public void addDataHandler(Student student) {  
    //tạo đối tượng ContentValues
```

```
ContentValues values = new ContentValues();  
//thêm giá trị các cột đến đối tượng ContentValues  
values.put(COLUMN_ID, student.getStudentID());  
values.put(COLUMN_NAME, student.getStudentName());  
SQLiteDatabase db = this.getWritableDatabase();  
//chèn dữ liệu đến bảng  
db.insert(TABLE_NAME, null, values);  
db.close();  
}
```

Lớp xử lý dữ liệu bây giờ đã sẵn sàng. Chúng ta trở lại Activity của ứng dụng và tạo giao diện gồm các views sau:

View	ID	Text	textSize	textColor
<i>PlainText</i>	<i>studentid</i>	<i>Enter ID</i>		
<i>PlainText</i>	<i>studentname</i>	<i>Enter Name</i>		
<i>Button</i>	<i>btnLoad</i>	<i>Load Data</i>		
<i>Button</i>	<i>btnAdd</i>	<i>Add</i>		
<i>TextView</i>	<i>txtData</i>	<i>Data List</i>	<i>30sp</i>	<i>@color/colorAccen</i>

Sau đó chúng ta xây dựng được giao diện ứng dụng như sau :



Trong lớp SQLiteDemoApplicationActivity tạo các biến tham chiếu đến các views trên giao diện như sau:

```
package com.ngocminhtran.sqlitedemoapplication;
```

```
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.TextView;
```

```
public class SQLiteDemoApplicationActivity extends  
AppCompatActivity {
```

```
    TextView dataList;  
    EditText studentid;  
    EditText studentname;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_sqlite_demo_application);
```

```
    dataList = (TextView) findViewById(R.id.txtData);  
    studentid = (EditText) findViewById(R.id.studentid);  
    studentname = (EditText) findViewById(R.id.studentname);
```

```
}
```

```
}
```

Thêm phương thức addStudent() đến lớp SQLiteDemoApplicationActivity

```
public void addStudent(View view) {  
    //khởi tạo đối tượng xử lý dữ liệu  
    DataHandler dbHandler = new DataHandler(this, null, null, 1);  
    //nhận id  
    int id = Integer.parseInt(studentid.getText().toString());  
    //nhận name  
    String name = studentname.getText().toString();  
    //gán id và name đến đối tượng Student  
    Student student = new Student(id, name);  
    //thêm đối tượng Student đến bảng dữ liệu  
    dbHandler.addDataHandler(student);  
    //xóa sạch các PlainText  
    studentid.setText("");  
    studentname.setText("");
```

}

Thêm phương thức loadStudents() đến lớp SQLiteDemoApplicationActivity

```
public void loadStudents(View view) {  
    //khởi tạo đối tượng xử lý dữ liệu  
    DataHandler dbHandler = new DataHandler(this, null, null, 1);  
    //hiển thị dữ liệu  
    datalist.setText(dbHandler.loadDataHandler());  
    //xóa sạch các PlainText  
    studentid.setText("");  
    studentname.setText("");  
}
```

Mở tập tin activity_sqlite_demo_application.xml trong chế độ Text và tìm đến button Load Data để thêm phương thức loadStudents() đến thuộc tính sự kiện onClick:

<Button

```
    android:id="@+id/btnLoad"  
    android:text="Load Data"  
    android:onClick="loadStudents"
```

... />

Tìm đến button Add để thêm phương thức addStudent() đến thuộc tính sự kiện onClick

<Button

```
    android:id="@+id/btnAdd"  
    android:onClick="addStudent"  
    android:text="Add "
```

.../>

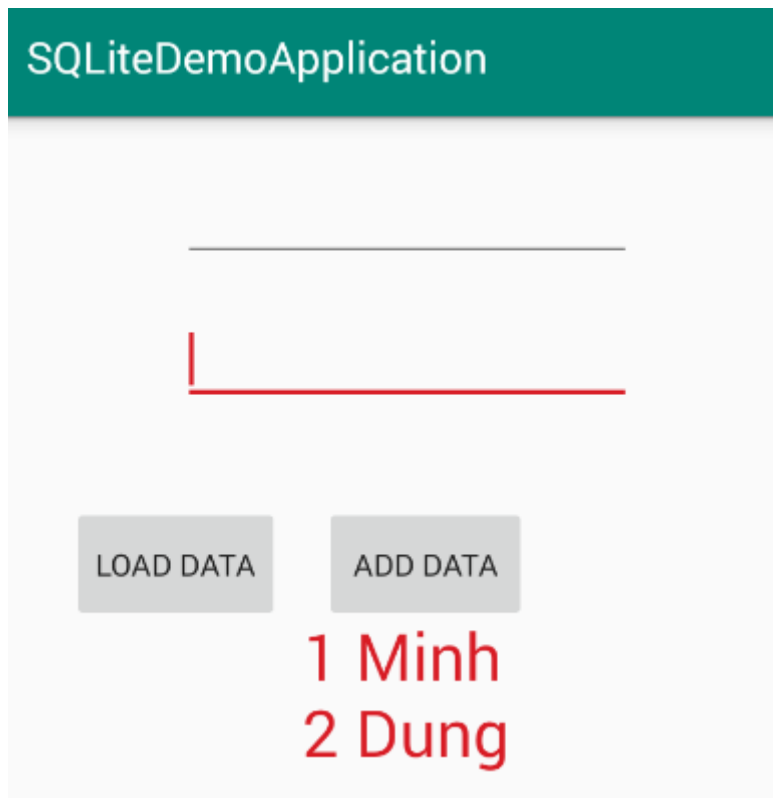
Chạy ứng dụng theo các bước:

Nhập ID là 1, Name là Minh và nhấn nút ADD DATA

Nhập ID là 2, Name là Dung và nhấn nút ADD DATA

Nhấn nút LOAD DATA

Kết quả:



Chúng ta đã có thể hiển thị và thêm dữ liệu đến bảng Students. Bây giờ chúng ta sẽ thêm các truy vấn xóa dữ liệu, cập nhật dữ liệu và tìm kiếm dữ liệu từ bảng.

Để xóa dữ liệu từ bảng Students, cụ thể là xóa một Student theo StudentID, chúng ta dùng lệnh SQL sau:

```
SELECT *  
FROM Students  
WHERE StudentID của Student cần xóa;
```

Thêm phương thức deleteDataHandler() đến lớp DataHandler dùng để thực hiện truy vấn xóa dữ liệu như sau :

```
public boolean deleteDataHandler(int ID) {  
    boolean result = false;  
    String query = "Select * FROM "  
        + TABLE_NAME + " WHERE "  
        + COLUMN_ID + " = '"  
        + String.valueOf(ID) + "'";  
    SQLiteDatabase db = this.getWritableDatabase();  
    Cursor cursor = db.rawQuery(query, null);  
    Student student = new Student();  
    if (cursor.moveToFirst()) {  
        student.setStudentID(Integer.parseInt(cursor.getString(0))  
    );  
    db.delete(TABLE_NAME, COLUMN_ID + "=?",  
        new String[] {  
            String.valueOf(student.getStudentID())  
        }  
    );  
}
```

```
    });  
    cursor.close();  
    result = true;  
}  
db.close();  
return result;  
}
```

Truy vấn được thực thi bằng phương thức *rawQuery()* của đối tượng *SQLiteDatabase* và kết quả trả về được lưu trong đối tượng *Cursor*. Xóa hàng đầu tiên trong đối tượng *Cursor* dùng phương thức *delete()* của đối tượng *SQLiteDatabase*.

Thêm phương thức *deleteStudent()* đến lớp *SQLiteDemoApplicationActivity* để thực thi phương thức xử lý dữ liệu *deleteDataHandler()*:

```
public void deleteStudent(View view) {  
    DataHandler dbHandler = new DataHandler(this, null, null, 1);  
    boolean result =  
dbHandler.deleteDataHandler(Integer.parseInt(  
    studentid.getText().toString()));  
    if (result) {  
        studentid.setText("");  
        studentname.setText("");  
        datalist.setText("Student Deleted");  
    } else  
        studentid.setText("No Match Found");  
}
```

Để cập nhật dữ liệu đến bảng *Students* chúng ta sử dụng đối tượng *ContentValues* và phương thức *update()* của đối tượng *SQLiteDatabase*. Thêm phương thức *updateDataHandler()* đến lớp *DataHandler* dùng để thực hiện truy vấn cập nhật dữ liệu đến bảng *Students* theo *StudentID* như sau:

```
public boolean updateDataHandler(int ID, String name) {  
    SQLiteDatabase db = this.getWritableDatabase();  
    ContentValues args = new ContentValues();  
    args.put(COLUMN_ID, ID);  
    args.put(COLUMN_NAME, name);  
    return db.update(TABLE_NAME, args, COLUMN_ID + " = " + ID,  
null) > 0;  
}
```

Thêm phương thức *updateStudent()* đến lớp *SQLiteDemoApplicationActivity* để thực thi phương thức xử lý dữ liệu *updateDataHandler()*:


```
public void updateStudent(View view) {
    DataHandler dbHandler = new DataHandler(this, null, null, 1);
    boolean result = dbHandler.updateDataHandler(Integer.parseInt(
        studentid.getText().toString()),
        studentname.getText().toString());
    if (result) {
        studentid.setText("");
        studentname.setText("");
        datalist.setText("Student Updated");
    } else
        studentid.setText("No Match Found");
}
```

Để tìm kiếm thông tin về Student từ bảng Students theo StudentName, chúng ta dùng lệnh SQL sau:

*SELECT * FROM Students*

WHERE StudentName của Student cần tìm;

Thêm phương thức findFirstDataHandler() đến lớp DataHandler dùng để thực hiện truy vấn tìm kiếm Student từ bảng Students theo StudentName. Kết quả trả về (lưu trong đối tượng Cursor) của phương thức là hàng đầu tiên trong tập kết quả (nếu Student tồn tại trong bảng):

```
//tìm kiếm Student theo StudentName
//kết quả trả về là Student đầu tiên trong danh sách kết quả
public Student findFisrtDataHandler(String studentname) {

    //chuỗi truy vấn tìm kiếm Student theo StudentName
    String query = "Select * FROM " + TABLE_NAME
        + " WHERE " + COLUMN_NAME + " = "
        + "'" + studentname + "'";
    SQLiteDatabase db = this.getWritableDatabase();
    // Thực thi truy vấn và gán kết quả đến đối tượng Cursor
    Cursor cursor = db.rawQuery(query, null);
    Student student = new Student();
    //trả về hàng đầu tiên trong kết quả
    if (cursor.moveToFirst()) {
        cursor.moveToFirst();
        student.setStudentID(Integer.parseInt(cursor.getString(0)))
    };
    student.setStudentName(cursor.getString(1));
    cursor.close();
} else {
    student = null;
}
```

```
}
db.close();
//trả về sinh viên đầu tiên tìm được
return student;
}
```

Thêm phương thức findFirstStudent() đến lớp SQLiteDemoApplicationActivity để thực thi phương thức xử lý dữ liệu findFirstDataHandler():

```
public void findFirstStudent(View view) {
    DataHandler dbHandler = new DataHandler(this, null, null, 1);
    Student student =
        dbHandler.findFisrtDataHandler
            (studentname.getText().toString());
    if (student != null) {
        datalist.setText(String.valueOf(student.getStudentID())
            + " " + student.getStudentName()
            + System.getProperty("line.separator"));
        studentid.setText("");
        studentname.setText("");
    } else {
        datalist.setText("No Match Found");
        studentid.setText("");
        studentname.setText("");
    }
}
```

Dữ liệu trả về từ việc tìm kiếm thông tin có thể chứa nhiều Student vì các StudentName có thể trùng nhau. Thêm phương thức findAllDataHandler() đến lớp DataHandler dùng để thực hiện truy vấn tìm kiếm Student từ bảng Students theo StudentName. Kết quả trả về của phương thức là tất cả các hàng (Student) trong tập kết quả (nếu các Student tồn tại trong bảng)

```
//tìm kiếm Student theo StudentName
//kết quả trả về là tất cả Student trong danh sách kết quả
public List<Student> findAllDataHandler(String studentname) {
    //chuỗi truy vấn tìm kiếm Student theo StudentName
    String query = "Select * FROM " + TABLE_NAME
        + " WHERE " + COLUMN_NAME + " = "
        + "'" + studentname + "'";
    SQLiteDatabase db = this.getWritableDatabase();
    //danh sách chứa tất cả các Student tìm được
    List<Student> lst = new ArrayList<Student>();
    // Thực thi truy vấn và gán kết quả đến đối tượng Cursor
    Cursor cursor = db.rawQuery(query, null);
    //duyet qua tất cả các hàng từ hàng đầu tiên
```

```
if(cursor.moveToFirst()) {
    do {
        Student student = new Student();
        student.setStudentID
            (Integer.parseInt(cursor.getString(0)));
        student.setStudentName(cursor.getString(1));
        lst.add(student);
    }while (cursor.moveToNext());
}
//đóng các đối tượng
cursor.close();
db.close();
//trả về danh sách sinh viên tìm được
return lst;
}
```

Các hàng dữ liệu trong đối tượng Cursor sẽ lần lượt được gán đến một ArrayList kiểu Student. Kết quả trả về của phương thức là một ArrayList chứa tất cả các Student có cùng StudentName. Thêm phương thức findAllStudent() đến lớp SQLiteDemoApplicationActivity để thực thi phương thức xử lý dữ liệu

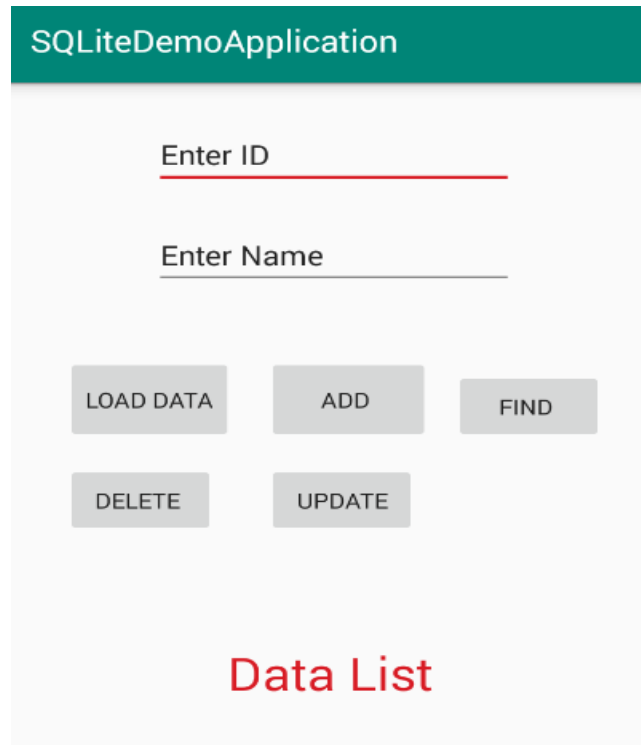
findAllDataHandler():

```
public void findAllStudent(View view) {
    DataHandler dbHandler = new DataHandler(this, null, null, 1);
    List<Student> lst =
        dbHandler.findAllDataHandler
            (studentname.getText().toString());
    String studentsList = "";
    if (!lst.isEmpty()) {
        for(Student st:lst)
        {
            studentsList += String.valueOf(st.getStudentID())
                + " " + st.getStudentName()
                + System.getProperty("line.separator");
            studentid.setText("");
            studentname.setText("");
        }
        datalist.setText(studentsList);
    } else {
        datalist.setText("No Match Found");
        studentid.setText("");
        studentname.setText("");
    }
}
```

Thêm các button đến giao diện ứng dụng:

View	ID	Text	onClick
Button	<i>btnDelete</i>	<i>Delete</i>	<i>deleteStudent</i>
Button	<i>btnUpdate</i>	<i>Update</i>	<i>updateStudent</i>
Button	<i>btnFind</i>	<i>Find</i>	<i>findAllStudent</i> (hay <i>findFirstStudent</i>)

Giao diện:



Thực thi ứng dụng và kiểm tra kết quả.

Link mã nguồn có thể xem tại đây : <https://github.com/TranNgocMinh/Kotlin-and-Android/tree/master/CodeList/CodeList11>