ĐỀ CƯƠNG BÀI GIẢNG

BÀI 9: TƯƠNG TÁC VỚI CSLD SQLite

Nội dung bài học trước khi lên lớp (trang 2 đến 5)

- SQLite là gì? Ưu điểm và nhược điểm của SQLite
- SQLite lưu trữ ở đâu?
- Một số câu lệnh đơn giản thường dùng trong SQLite

Nội dung bài học thực hiện lên lớp (Trang 6 đến hết):

- Cách khai báo sử dụng SQLite
- Demo và giải thích

Nội dung bài học sau khi lên lớp: Phiếu bài tập bài 1 đến bài 6

NỘI DUNG BÀI HỌC

1. Tổng quan về SQLite

- SQLite là hệ quả trị cơ sở dữ liệu (DBMS) quan hệ tương tự như Mysql, MS SQL... Đặc điểm nổi bật của SQLite so với các DBMS khác là gọn, nhẹ, đơn giản, đặt biệt không cần mô hình server-client, không cần cài đặt, cấu hình hay khởi động nên không có khái niệm user, password hay quyền hạn trong SQLite Database. Dữ liệu cũng được lưu ở một file duy nhất.
- SQLite thường không được sử dụng với các hệ thống lớn nhưng với những hệ thống ở quy mô vùa và nhỏ thì SQLite không thua các DBMS khác về chức năng hay tốc độ. Vì không cần cài đặt hay cấu hình nên SQLite được sử dụng nhiều trong việc phát triển, thử nghiệm.

1.1. Tính năng của SQLite

- Giao dịch trong SQLite tuân thủ theo nguyên tắc (ACID) ngay cả sau khi hệ thống treo và mất điện.
- Không cần cấu hình
- Không cần thiết lập hoặc quản trị
- SQLite hỗ trợ với đầy đủ tính năng với các khả năng nâng cao như các chỉ mục 1 phần, các chỉ mục về các biểu thức, JSON và các biểu thức bảng chung.
- Một sở dữ liệu hoàn chỉnh được lưu trữ trong một tệp đa nền tảng duy nhất. Phù hợp với sử dụng dưới dạng định dạng tệp ứng dụng
- Hỗ trợ các cơ sở dữ liệu có kích thước terabyte và các chuỗi có kích thước gigabyte.
- Hỗ trợ nhiều API
- Đơn giản dễ sử dụng
- Nhanh: Trong một số trường hợp, SQLite nhanh hơn hệ thống tệp tin trực tiếp I/O.

- Được viết bằng ANSI-C.
- Bindings cho hàng chục ngôn ngữ khác có sẵn 1 cách riêng biệt.
- Mã nguồn đầy, nguồn mở đủ có thể kiểm tra nhánh 100%.
- Là công nghệ đa nền tảng
- SQLite là có sẵn trên Android, *BSD, iOS, Linux, Mac, Solaris, Windows,.. Dễ dàng dịch chuyển sang các hệ thống khác.

1.2. Ứng dụng của SQLite

- Cơ sở dữ liệu cho Internet Of Things.
- SQLite là lựa chọn phổ biến cho các công cụ cơ sở dữ liệu trong điện thoại di động, PDA, máy nghe nhạc mp3, hộp set-top, và các tiện ích điện tử khác.
- Định dạng tệp ứng dụng.
- Thay vì sử dụng fopen() để viết XML, JSON, CSV hoặc một số định dạng động quyền vào các tệp đĩa được ứng dụng của bạn sử dụng, hãy sử dụng SQLite.
- Cơ sở dữ liệu cho web.
- Bởi vì SQLite không yêu cầu cấu hình và lưu trữ thông tin trong các tệp đĩa thông thường nên SQLite là lựa chọn phổ biến làm cơ sở dữ liệu để quay lại các trang web vừa và nhỏ.
- Stand-in cho một doanh nghiệp RDBMS:
- SQLite được sử dụng như một RDBMS doanh nghiệp cho mục đích trình diễn hoặc để thử nghiệm vì SQLite nhanh và không yêu cầu thiết lập.

1.3. Ưu điểm của SQLite

- AUTOINCREMENT?
  - Một cột được khai báo INTEGER PRIMARY KEY sẽ tự động tăng thêm.
  - Khi chèn NULL vào cột này thì nó sẽ được tự động chuyển thành 1 số nguyên lớn nhất cột.
- Các kiểu dữ liệu

- SQLite sử dụng một hệ thống kiểu động. Trong SQLite, kiểu dữ liệu là một giá trị được liên kết với chính giá trị đó, không liên kết với Container.
- Các kiểu dữ liệu Integer, Real, Text, Blob, Null
- SQLite không thực thi ràng buộc dữ liệu.
  - Dữ liệu loại nào cũng có có thể chèn vào bất kì cột. Nhưng Integer primary key chỉ có thể chèn được số nguyên 64bit.
  - SQLite sử dụng khai báo dữ liệu của một cột làm gợi ý định dạng dữ liệu.

- VD: Với một cột được khai báo integer bạn chèn kiểu chuỗi vào thì nó sẽ cố chuyển chuỗi thành số nguyên nếu có thể sẽ chèn số nguyên đó thay cho chuỗi. => Loại mối quan hệ.

- Xóa cột trong bảng
  - SQLite có hỗ trợ ALTER TABLE nhưng rất hạn chế chỉ có thể thêm cột và thay đổi tên.
  - Nếu muốn xóa cột thì chúng ta thực hiện các bước sau:
    - Tạo bảng mới có các cột cần thiết
    - Sao chép dữ liệu từ bảng cũ vào
    - Xóa bảng cũ
    - Tạo lại bảng với tên bảng cũ
    - Sao chép dữ liệu từ bảng tạm vào

```
TRANSACTION;
CREATE TEMPRARY TABLE new_backup (a,b)
INSERT INTO new_backup SELECT a,b FORM old;
DROP TABLE old;
CREATE TABLE old(a,b);
INSERT INTO SELECT a,b FORM new_backup;
DROP TABLE new_backup;
COMMIT;
```

- Khóa ngoài
  - SQLite hỗ trợ khóa ngoài nhưng theo mặc định thì khóa ngoài sẽ tắt.
  - Để cho phép thực thi khóa ngoài ta sử dụng câu lệnh

```
PRAGMA foreign keys = ON
```

```
# hoặc biên dịch với
DSQLITE DEFAULT FOREIGN KEYS = 1
```

- Dấu nháy

SQLite phần biệt cách sử dụng dấu nháy

- Nháy đôi cho tên bảng, cột
- Đơn cho giá trị

- Mệnh đề GLOB

- So khớp giá trị với các giá trị tương tự bởi sử dụng các toán tử wildcard.
- Không giống LIKE, GLOB phân biệt kiểu chữ và nó theo cú pháp của UNIX để xác định các toán tử Wildcard sau:
  - '*' : số 0,1 hoặc nhiều số hoặc kí tự (tương tự như %)
  - '?' : 1 số hoặc 1 kí tự đơn (tương tự như _)

```
SELECT FROM table_name
WHERE column GLOB 'XXXX*'
```

1.4. Nhược điểm của SQLite

- Một số tính năng của SQL92 không được hỗ trợ trong SQLite như ALTER DROP COLUMN, ADD CONSTRAINT không được hỗ trợ; RIGHT JOIN; TRIGGER; phân quyền GRANT và REVOKE.
- Vì SQLite không cần cấu hình, cài đặt, không hỗ trợ GRANT và REVOKE nên việc phân quyền truy cập cơ sở dữ liệu chỉ có thể là quyền truy cập file của hệ thống.
- SQLite sử dụng cơ chế coarse-gained locking nên trong cùng một thời điểm có thể hỗ trợ nhiều người đọc dữ liệu, nhưng chỉ có 1 người có thể ghi dữ liệu.
- SQLite không phù hợp với các hệ thống có nhu cầu xử lý trên một khối lượng dữ liệu lớn, phát sinh liên tục.

1.5. Một số lệnh cơ bản trong SQLite

SQLite hỗ trợ gần như đầy đủ các cú pháp trong chuẩn SQL92.

STT	Cú pháp	Ý nghĩa

1	sqlite3 <name.db>	Tạo database
2	ATTACH DATABASE '<databasename>' As '<alias-name>';	Sử dụng database, có thể đặt alias cho database và sử dụng như tên của database, mỗi một lần gọi lệnh sử dụng thì ta có thể sử dụng tên alias khác nhau
3	DETACH DATABASE '<name-name>';	Xóa cơ sở dữ liệu sử dụng với tên alias
4	CREATE TABLE <databasename.tablename>();	Tạo bảng
5	DROP TABLE database_name.table_name;	Xóa bảng
6	INSERT INTO table_name [(column1, column2,..)] VALUES (value1, value2,..);	Thêm dữ liệu vào bảng
7	INSERT INTO table1 [(column…)] SELECT column FROM table2 [WHERE];	Chèn dữ liệu vào bảng từ một bảng khác
8	SELECT sql FROM table;	Hiển thị thông tin bảng
9	SELECT (12+8) AS ADDITION; #20	Thực hiện biểu thức số học
10	SELECT COUNT(*) AS "RECORDS" FROM table;	đếm bảng ghi trong bảng
11	SELECT CURRENT_TIMESTAMP;	Hiển thị thời gian hệ thống
12	UPDATE table_name SET column1 = value,... WHERE ..;	Update dữ liệu bảng
13	DELETE FROM table_name WHERE …;	Xóa bản ghi
14	PRAGMA pragma_name;	Điều khiển các biến môi trường và các flag trạng thái đa dạng
15	PRAGMA pragma_name = value;	Thiết lập giá trị

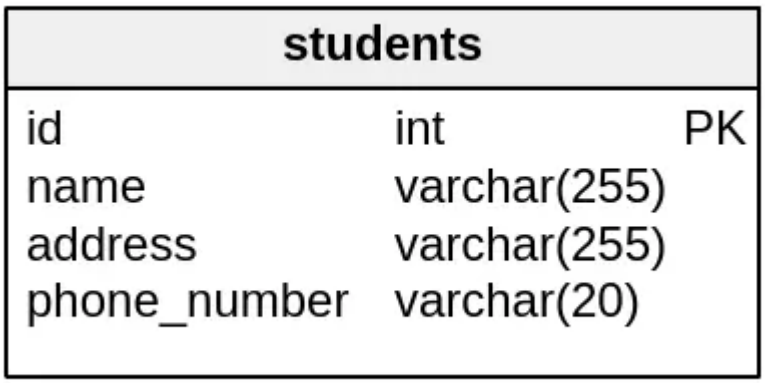
16	SELECT ... FROM table1 CROSS JOIN table2 ...	CROSS JOIN: kết nối mọi hàng của bảng đầu tiên với mỗi hàng của bảng thứ hai
17	SELECT ... FROM table1 [INNER] JOIN table2 ON conditional_expression ...	INNER JOIN
18	SELECT ... FROM table1 LEFT OUTER JOIN table2 ON conditional_expression ...	OUTER JOIN: chỉ hỗ trợ LEFT JOIN

2. Minh họa sử dụng SQLite trong Android

Dưới đây sẽ minh họa ứng dụng lưu trữ thông tin của các sinh viên vào trong CSDL SQLite.

2.1. Thiết kế bảng Students

Thiết kế bảng Student gồm 4 trường Id, name, address, phone_number như hình dưới trong đó id là khóa chính.



2.2. Tạo class Student

Trước hết, chúng ta cần tạo Student class với các phương thức khởi tạo, getter, setter:

```
# Student.java
```

```
package com.framgia.androidsqlite;

public class Student {
	int id;
	String name;
	String address;
	String phone_number;

	public Student(int id, String name, String address, String
phone_number) {
		super();
		this.id = id;
		this.name = name;
		this.address = address;
		this.phone_number = phone_number;
	}

	public Student(String name, String address, String phone_number) {
		super();
		this.name = name;
		this.address = address;
		this.phone_number = phone_number;
	}

	public int getId() {
		return id;
	}

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

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public String getPhone_number() {
		return phone_number;
	}

	public void setPhone_number(String phone_number) {
```

```
            this.phone_number = phone_number;
      }
}
```

2.3 Tạo SQLite Database Handler class

Tiếp theo, ta sẽ tạo class để xử lý các thao tác CRUD(create, read, update, delete) đối với database.

Đầu tiên, tạo một Android project (File => New Android Project)

Tạo một class DatabaseHandler.java (src/package => New => Class)

Class DatabaseHandler sẽ kế thừa class SQLiteOpenHelper (Đây là một class mà Android cho phép bạn xử lý các thao tác đối với database của SQLite, vì vậy bạn có thể tạo một class khác thừa kế nó và tùy chỉnh việc điều khiển database theo ý mình):

Sau khi kế thừa từ class SQLiteOpenHelper, việc chúng ta cần làm tiếp theo đó là override lại 2 phương thức onCreate() và onUpgrade

onCreate(): Đây là nơi để chúng ta viết những câu lệnh tạo bảng. Nó được gọi khi database đã được tạo.

onUpgrade(): Nó được gọi khi database được nâng cấp, ví dụ như chỉnh sửa cấu trúc các bảng, thêm những thay đổi cho database,..

```
# DatabaseHandler.java
package com.framgia.androidsqlite;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHandler extends SQLiteOpenHelper{
      private static final String DATABASE_NAME = "schoolManager";
      private static final int DATABASE_VERSION = 1;
      private static final String TABLE_NAME = "students";

      private static final String KEY_ID = "id";
      private static final String KEY_NAME = "name";
      private static final String KEY_ADDRESS = "address";
      private static final String KEY_PHONE_NUMBER = "phone_number";

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

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        String create_students_table = String.format("CREATE TABLE %s(%s
INTEGER PRIMARY KEY, %s TEXT, %s TEXT, %s TEXT)", TABLE_NAME, KEY_ID,
KEY_NAME, KEY_ADDRESS, KEY_PHONE_NUMBER);
        db.execSQL(create_students_table);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        String drop_students_table = String.format("DROP TABLE IF EXISTS
%s", TABLE_NAME);
        db.execSQL(drop_students_table);

        onCreate(db);
    }
}
```

Tiếp theo, chúng ta sẽ viết những phương thức để xử lý việc đọc, ghi đối với database

2.4. Thêm một record mới

Chúng ta sẽ tạo một phương thức là addStudent nhận một object Student như là một tham số. ContentValues được sử dụng để lưu các giá trị tương ứng với các trường trong bảng. SQLiteDatabase có chứa các phương thức tạo, xóa, thực thi các lệnh SQL, nó sẽ được sử dụng để insert các giá trị từ object Student vào các trường trong bảng students.

```
public void addStudent(Student student) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, student.getName());
    values.put(KEY_ADDRESS, student.getAddress());
    values.put(KEY_PHONE_NUMBER, student.getPhone_number());

    db.insert(TABLE_NAME, null, values);
    db.close();
    }
```

2.5. Truy vấn dữ liệu trong bảng

Ta sẽ dùng Cursor để lưu giá trị trả về của các hàm sau đây:

```
    public Cursor query(String table, String[] columns, String selection,
String[] selectionArgs, String groupBy, String having, String orderBy)
    public Cursor rawQuery(String sql, String[] selectionArgs)
```

a. Phương thức getStudent() sẽ đọc một record student trong bảng, nhận student id là tham số.

```
public Student getStudent(int studentId) {
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(TABLE_NAME, null, KEY_ID + " = ?", new String[]
{ String.valueOf(studentId) },null, null, null);
    if(cursor != null)
        cursor.moveToFirst();
    Student student = new Student(cursor.getInt(0), cursor.getString(1),
cursor.getString(2), cursor.getString(3));
    return student;
    }
```

- db.query sẽ trả về một Cursor, lúc này Cursor đầu đọc chưa trỏ tới dòng dữ liệu nào cả, do đó, ta phải gọi lệnh .moveToFirst() để Cursor có thể trỏ tới dòng đầu tiên.

b. Phương thức getAllStudents() sẽ trả về tất cả student có trong bảng dưới dạng một array list của Student.

```
public List<Student> getAllStudents() {
    List<Student>  studentList = new ArrayList<>();
    String query = "SELECT * FROM" + TABLE_NAME;

    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.rawQuery(query, null);
    cursor.moveToFirst();

    while(cursor.isAfterLast() == false) {
        Student student = new Student(cursor.getInt(0), cursor.getString(1),
cursor.getString(2), cursor.getString(3));
        studentList.add(student);
        cursor.moveToNext();
    }
    return studentList;
    }
```

2.6. Cập nhật dữ liệu trong bảng

Chúng ta sử dụng hàm update của SQLiteDatabase để cập nhật dữ liệu trong bảng theo một điều kiện bất kỳ nào đó.

```
public int update(String table, ContentValues values, String
whereClause, String[] whereArgs)
```

Trong đó:

Tham số 1: tên bảng

Tham số 2: đối tượng muốn chỉnh sửa (với giá trị mới)

Tham số 3: tập các điều kiện lọc (dùng dấu ? để tạo điều kiện lọc)

Tham số 4: tập các giá trị của điều kiện lọc (lấy theo đúng thứ tự)

```
public void updateStudent(Student student) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_NAME, student.getName());
    values.put(KEY_ADDRESS, student.getAddress());
    values.put(KEY_PHONE_NUMBER, student.getPhone_number());

    db.update(TABLE_NAME, values, KEY_ID + " = ?", new String[] {
String.valueOf(student.getId()) });
    db.close();
     }
```

2.7. Cập nhật dữ liệu trong bảng

Chúng ta sử dụng hàm update của SQLiteDatabase để cập nhật dữ liệu trong bảng theo một điều kiện bất kỳ nào đó.

```
public int update(String table, ContentValues values, String
whereClause, String[] whereArgs)
```

Trong đó:

Tham số 1: tên bảng

Tham số 2: đối tượng muốn chỉnh sửa (với giá trị mới)

Tham số 3: tập các điều kiện lọc (dùng dấu ? để tạo điều kiện lọc)

Tham số 4: tập các giá trị của điều kiện lọc (lấy theo đúng thứ tự)

```
public void updateStudent(Student student) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_NAME, student.getName());
    values.put(KEY_ADDRESS, student.getAddress());
    values.put(KEY_PHONE_NUMBER, student.getPhone_number());

    db.update(TABLE_NAME, values, KEY_ID + " = ?", new String[] {
String.valueOf(student.getId()) });
    db.close();
     }
```

2.8. Xóa một bản ghi (record)

Chúng ta sử dụng hàm delete của SQLiteDatabase để xóa dữ liệu của một hoặc một số record trong bảng theo một điều kiện bất kỳ nào đó

```
public int delete(String table, String whereClause, String[] whereArgs)
```

Trong đó :

Tham số 1: tên bảng

Tham số 2: tập điều kiện lọc

Tham số 3: tập các giá trị của điều kiện lọc

```
public void deleteStudent(int studentId) {
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_NAME, KEY_ID + " = ?", new String[] {
String.valueOf(studentId) });
    db.close();
     }
```

3. Một số link tham khảo

1. https://www.youtube.com/watch?v=iERUSMe6n4s
2. https://developer.android.com/training/data-storage/sqlite?hl=vi
3. https://www.youtube.com/watch?v=rg_TrAKwTbY
4. https://www.youtube.com/watch?v=9qvs2qP3jAE
5. https://www.youtube.com/watch?v=WTRt1Xpkios