LẬP TRÌNH ANDROID CƠ BẢN

BÀI 6 : SQLITE

NỘI DUNG

- Phần I: SQLite
  - Giới thiệu về SQLite
  - Các kiểu dữ liệu trong SQLite
- Phần II: Sử dụng SQLiteOpenHelper
  - SQLiteOpenHelper
  - Tạo Database và Table
- Phần III: Các thao tác trên SQLite
- Phần IV: Xây dựng lớp DAO

BÀI 6 : SQLITE

PHẦN I :
SQLITE

GIỚI THIỆU SQLITE

- SQLite là phần mềm quản lý cơ sở dữ liệu SQL nhưng không giống như hầu hết các cơ sở dữ liệu SQL khác, SQLite không có máy chủ riêng biệt để xử lý
- Đặc điểm: gọn nhẹ, đơn giản. Chương trình gồm 1 file duy nhất, không cần cài đặt, không cần cấu hình mà có thể sử dụng ngay
- Dữ liệu database được lưu vào một file duy nhất. Không có khái niệm user, password hay quyền hạn

TẠI SAO NÊN DÙNG SQLITE

- SQLite không yêu cầu một tiến trình Server riêng rẽ để hoạt động.
- SQLite không cần cấu hình, nghĩa là không cần thiết phải cài đặt.
- Một SQLite Database đầy đủ được lưu giữ trong một disk file đơn.SQLite rất nhỏ gọn, nhỏ hơn 400kB đã được cấu hình đầy đủ hoặc nhỏ hơn 250kB khi đã bỏ qua các tính năng tùy ý.
- SQLite là tự chứa, nghĩa là không có sự phụ thuộc ngoại vi.
- Các Transaction trong SQLite là tuân theo đầy đủ chuẩn ACID, đảm bảo truy cập an toàn từ nhiều tiến trình.
- SQLite hỗ trợ hầu hết các tính năng của một ngôn ngữ truy vấn trong chuẩn SQL92.

Lớp lưu trữ trong SQLite:

Các lớp lưu trữ (Storage Class) trong SQLite Database bao gồm:

Lớp lưu trữ	Miêu tả
NULL	Giá trị là một giá trị NULL
INTEGER	Giá trị là một số nguyên có dấu, được lưu giữ trong 1, 2, 3, 4, 6, hoặc 8 byte tùy thuộc vào độ lớn của giá trị
REAL	Giá trị số thực dấu chấm động, được lưu giữ như là một số thực dấu chấm động 8-byte IEEE
TEXT	Giá trị là một text string, được lưu trữ bởi sử dụng Encoding của cơ sở dữ liệu (UTF-8, UTF-16BE hoặc UTF-16LE)
BLOB	Giá trị là một blob của dữ liệu, nhập vào như thế nào thì lưu giữ chính xác như thế

KIỂU DỮ LIỆU SQLITE

✓ **Kiểu dữ liệu Boolean trong SQLite**

SQLite không hỗ trợ lớp lưu trữ Boolean riêng rẽ. Thay vào đó, các giá trị Boolean được lưu trữ dưới dạng các số nguyên: 0 cho false và 1 cho true.

✓ **Kiểu dữ liệu Date và Time trong SQLite**

SQLite không có một lớp lưu trữ riêng rẽ để lưu trữ date/time, nhưng SQLite có thể lưu giữ date/time dưới dạng các giá trị TEXT, REAL hoặc INTEGER.

Kiểu dữ liệu SQLite

Có thể chọn để lưu giữ date và time trong bất kỳ các kiểu định dạng này và tự do chuyển đổi giữa các định dạng bởi sử dụng các hàm xử lý date và time có sẵn.

Lớp lưu trữ	Định dạng Date
TEXT	Một date trong định dạng "YYYY-MM-DD HH:MM:SS.SSS"
REAL	Số ngày từ Greenwich November 24, 4714 B.C
INTEGER	Số giây từ 1970-01-01 00:00:00 UTC

BÀI 6 : SQLITE

PHẦN II :
SỬ DỤNG
SQLITEOPENHELPER

CLASS HỖ TRỢ CỦA SQLITE

Android cung cấp hai Class hỗ trợ cho việc tạo và quản lý Database:

- ✓ SQLiteDatabase: sử dụng đơn giản
- ✓ SQLiteOpenHelper: sử dụng chuyên nghiệp

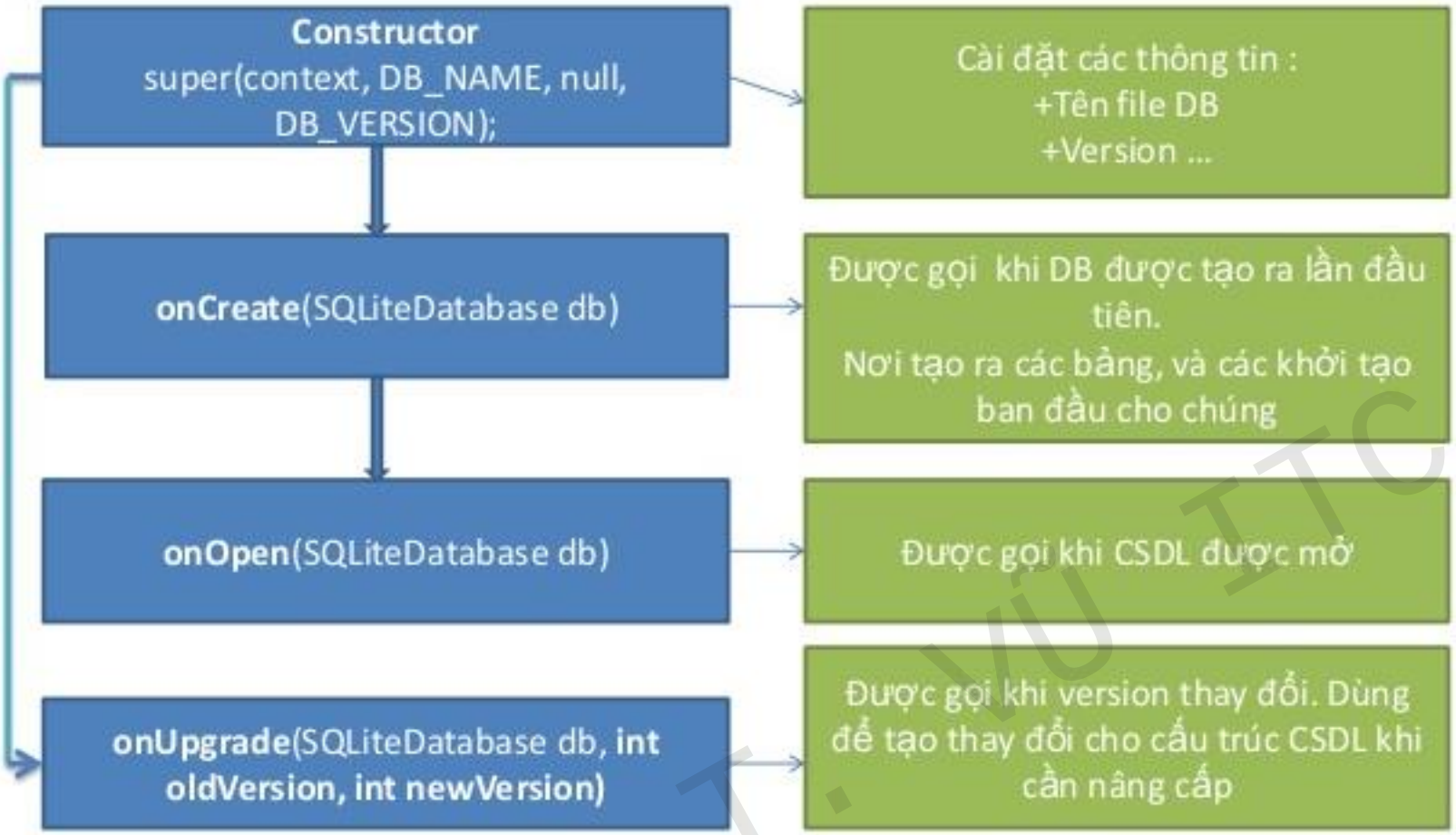
CLASS SQLITEOPENHELPER

Class SQLiteOpenHelper hỗ trợ quản lý Database và version SQLite.

3 phương thức của Class:

1. onCreate(SQLiteDatabase)
2. onUpgrade(SQLiteDatabase, int, int)
3. onOpen(SQLiteDatabase)

CLASS SQLITEOPENHELPER



SQLiteOpenHelper – Tạo Database & Table

Các bước thực hiện:

1. **Tạo các Class kế thừa SQLiteOpenHelper**

2. **Override phương thức onCreate:**
   - Tạo Table trong phương thức này

3. **Override phương thức onUprade:**
   - Phương thức này sẽ được gọi khi ta nâng version mới, cần xoá các table cũ và gọi lại onCreate

SQLiteOpenHelper – Tạo Database & Table

Ví dụ tạo database Demo6 và Table Nhanvien

```
package com.kietlpt.quanlynhanvien.SQLite;

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

public class DbHelper extends SQLiteOpenHelper {
    public static final String DB_NAME = "Demo6";
    public static final int DB_VERSION = 1;

    public DbHelper(Context context) { super(context, DB_NAME, null, DB_VERSION); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String createTableSql =
                "CREATE TABLE nhanvien (" +
                "id TEXT PRIMARY KEY,  " +
                "salary INTEGER NOT NULL, " +
                "name TEXT NOT NULL)";
        db.execSQL(createTableSql);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        String dropTableSql = "DROP TABLE IF EXISTS students";
        db.execSQL(dropTableSql);
        onCreate(db);
    }
}
```

SQLiteOpenHelper – Tạo Database & Table

Giải thích code:

- ✓ Hàm dựng **DbHelper**:
  Tạo database Demo6 với version 1

- ✓ Hàm **onCreate**:
  Thực hiện viết code tạo table Nhanvien

- ✓ Hàm **onUpgrade**:
  Thực hiện xoá và gọi lại onCreate nếu có version mới.

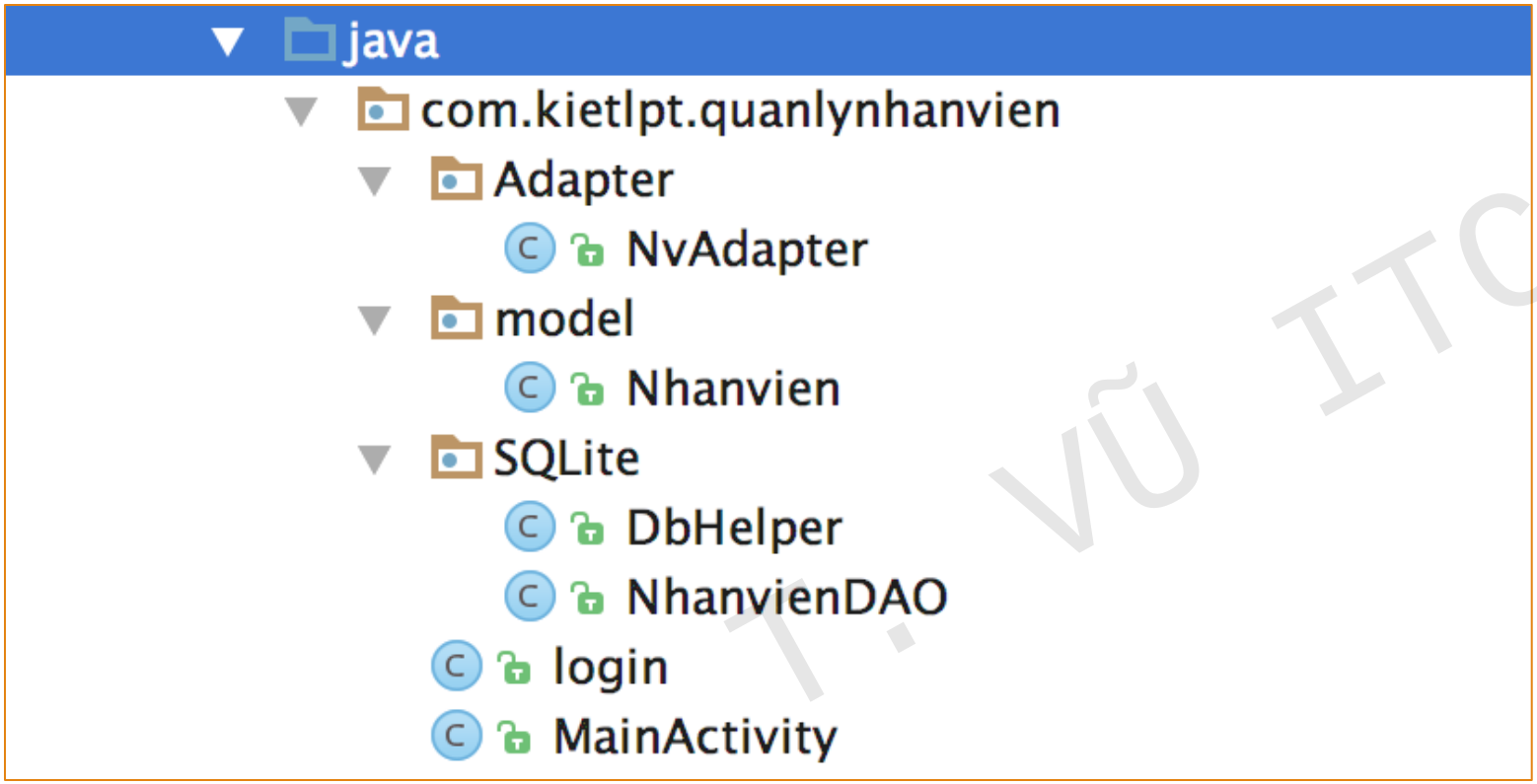
SQLiteOpenHelper – Tạo Database & Table

BÀI TẬP TẠI LỚP:

- ✓ Tạo Database QLDT:
- ✓ Tạo 2 table Lop và Sinhvien như sau:
  - Lop (IdLop, TenLop, Nganh)
  - Sinhvien (IdSv, TenSv, NoiSinh, NgaySinh, IdLop)

Tổ chức cấu trúc ứng dụng với SQLite

- Để dễ viết code và bảo trì chúng ta cần tổ chức cấu trúc ứng dụng theo nhiều package theo cấu trúc sau:



Tổ chức cấu trúc ứng dụng với SQLite

- Adapter: chứa các class dùng để Custom ListView
- Model: chứa các Class quản lý đối tượng
- SQLite: chứa các Class liên quan đến SQLite
- Ngoài ra các thể tạo thêm các package khác chứa các activity..

BÀI 6 : SQLITE

PHẦN III :
CÁC THAO TÁC TRÊN SQLITE

CÁC THAO TÁC TRÊN SQLITE

- Cũng giống như các database khác, các thao tác chính trên Sqlite bao gồm:
  - Insert
  - Update
  - Delete
  - Query

INSERT

- Dùng phương thức Insert() của lớp database để thêm dữ liệu vào table.
- Mỗi dòng dữ liệu được tạo bởi đối tượng ContentValues. Thứ tự giá trị phải tương ứng với thứ tự cột trong table

```
public long insert(Nhanvien s) {

    ContentValues values = new ContentValues();
    values.put("name", s.name);
    values.put("id", s.id);
    values.put("salary", s.salary);


    return db.insert("nhanvien", null, values);
}
```

UPDATE

- Dùng phương thức Update() của lớp database để sửa dữ liệu trong 1 bảng. Update() bao gồm 4 tham số:
  - Tên bảng
  - **ContentValues** chứa giá trị sửa
  - Sử dụng dấu **?** để thay cho tham số trong mệnh đề WHERE
  - Nếu có nhiều hơn 1 tham số thì 1 mảng các tham số sẽ được đưa vào

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

Phương thức này trả về số dòng được update thành công

UPDATE

```
public int update(Nhanvien s) {
    ContentValues values = new ContentValues();
    values.put("name", s.name  );
    values.put("salary", s.salary);


    return db.update("nhanvien", values, "id=?", new String[]{s.id});
}
```

- Khi cần sửa field nào thì **ContentValues** chỉ chứa field thay đổi đó, không nhất thiết phải chứa tất cả các field

DELETE

- Dùng phương thức delete() của lớp database để xóa 1 dòng dữ liệu trong bảng. delete() bao gồm 3 tham số:
  - Tên bảng
  - Sử dụng dấu **?** để thay cho tham số trong mệnh đề WHERE
  - Nếu có nhiều hơn 1 tham số thì 1 mảng các tham số sẽ được đưa vào

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

Phương thức này trả về số dòng được delete thành công

DELETE

- Xóa tất cả các dòng

```
database.delete("tbllop", null, null);
```

- Xóa dòng với id = “1”

```
public int delete(String id) {
    return db.delete("nhanvien", "id=?", new String[]{id});
}
```

TRUY VẤN (QUERYING)

- Để quản lý kết quả trả về từ các câu truy vấn dữ liệu, lớp database cung cấp đối tượng cursor.
- Cursor giống như Con trỏ (Pointer), nó cho phép chúng ta truy xuất dữ liệu tuần tự hoặc ngẫu nhiên

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

Trả về vị trí của con trỏ và là **vị trí trước vị trí đầu tiên** của dữ liệu được trả về

Cursor

1	Sinh vien 1
2	Sinh vien 2
3	Sinh vien 3

TRUY VẤN (QUERYING) – BẢNG THAM SỐ

table	Tên bảng
columns	Danh sách các cột được trả về. Nếu bỏ qua (null) thì tất cả các cột sẽ được trả về.
selection	Lọc dữ liệu trả về với mệnh đề WHERE. Nếu bỏ qua (null) sẽ trả về tất cả các dòng
selectionArgs	Mảng các điều kiện lọc. Các tham số điều hiểu theo kiểu String
groupBy	Nhóm dữ liệu. Để trống (null) là không nhóm
having	Điều kiện lọc sau khi nhóm. Để trống là không có điều kiện lọc
orderBy	Sắp xếp dữ liệu

TRUY VẤN (QUERYING) – VÍ DỤ

- Truy vấn không có các phép chiếu, chọn, …

```
Cursor c = db.query("NHANVIEN",null,null,null, null, null, null);
   while (c.moveToNext())
          {
          String data += c.getString(0)
           + c.getString(c.getColumnIndex("TENNV")));
          }
```

- Truy vấn với tham số **id=1**

```
Cursor c = db.query("NHANVIEN",null,"id=?",new String[]{"1"}, null, null, null);
   while (c.moveToNext())
          {
          String data += c.getString(0)
             + c.getString(c.getColumnIndex("TENNV")));
          }
```

TRUY VẤN (QUERYING)

- Ngoài ra, để đỡ phức tạp, chúng ta có thể sử dụng phương thức rawQuery(lenhsql) của lớp SqliteDatabase, với lenhsql là câu truy vấn đơn giản

```
public List<Nhanvien> getNV(String sql, String...selectionArgs) {

    List<Nhanvien> list = new ArrayList<~>();

    Cursor c = db.rawQuery(sql, selectionArgs);

    while (c.moveToNext()){

        Nhanvien nv = new Nhanvien();
        nv.id = c.getString(c.getColumnIndex("id"));
        nv.name = c.getString(c.getColumnIndex("name"));
        nv.salary = c.getInt(c.getColumnIndex("salary"));


        list.add(nv);
    }

    return list;
}
```

BÀI 6 : SQLITE

PHẦN IV :
XÂY DỰNG LỚP DAO

TẠO CLASS DATA BASE ACCESS OBJECT (DAO)

- Để chuẩn bị truy vấn và thao tác với CSDL, chúng ta cần tao các Class DAO để thực hiện các thao tác trên.
- Class DAO có hàm tạo phải thực hiện kết nối SQLite (gọi Class DbHelper)
- Tạo phương thức truy vấn (xem)
- Tạo phương thức thêm
- Tạo phương thức xoá
- Tạo phương thức sửa

TẠO CLASS DATA BASE ACCESS OBJECT (DAO)

- Hàm tạo:

```
public class NhanvienDAO {
    private SQLiteDatabase db;

    public NhanvienDAO(Context context) {
        DbHelper dbHelper = new DbHelper(context);
        db = dbHelper.getWritableDatabase();
    }
```

XỬ LÝ XEM, THÊM, SỬA, XOÁ TRONG ACTIVITY

- Show dữ liệu lên ListView đã custom:
  - Tạo đối tượng từ Class Dao
  - Gọi phương thức get data trong class Dao
  - Show lên ListView đã custom

XỬ LÝ XEM, THÊM, SỬA, XOÁ TRONG ACTIVITY

- Show dữ liệu lên ListView đã custom:

```
//Tạo đối tượng Class DAO
final NhanvienDAO nhanvienDAO = new NhanvienDAO(this);

//get all Nhan vien

nhanviens = nhanvienDAO.getNhanVienAll();

adapter = new NvAdapter(this,nhanviens);

lv.setAdapter(adapter);
```

XỬ LÝ XEM, THÊM, SỬA, XOÁ TRONG ACTIVITY

- Thêm dữ liệu :
  - Tạo đối tượng từ Class Dao
  - Gọi phương thức insert data trong class Dao
  - Update lại ListView (gọi lại show data)

```
Nhanvien nv = new Nhanvien();
nv.id = txtId.getText().toString();
nv.name = txtName.getText().toString();
nv.salary = Integer.parseInt(txtSalary.getText().toString());

nhanvienDAO.insert(nv);
```

XỬ LÝ XEM, THÊM, SỬA, XOÁ TRONG ACTIVITY

- Sửa dữ liệu :
  - Tạo đối tượng từ Class Dao
  - Gọi phương thức update data trong class Dao
  - Update lại ListView (gọi lại show data)

```
Nhanvien nv = new Nhanvien();
nv.id = txtId.getText().toString();
nv.name = txtName.getText().toString();
nv.salary = Integer.parseInt(txtSalary.getText().toString());

nhanvienDAO.update(nv);
```

XỬ LÝ XEM, THÊM, SỬA, XOÁ TRONG ACTIVITY

- Xoá dữ liệu :
  - Tạo đối tượng từ Class Dao
  - Gọi phương thức delete data trong class Dao
  - Update lại ListView (gọi lại show data)

```
Nhanvien nv = new Nhanvien();
nv.id = txtId.getText().toString();

nhanvienDAO.delete(nv.id);
```