

## Chương 4:

# LƯU TRỮ DỮ LIỆU TRONG ANDROID

Trong Android có các cách lưu dữ liệu nào?

1. Lưu dữ liệu đơn giản với đối tượng SharedPreferences
2. Lưu dữ liệu với SQLite.
3. Lưu dữ liệu với Content Provider.
4. Lưu dữ liệu bằng tập tin trong bộ nhớ trong và thẻ nhớ.
5. Lưu dữ liệu với dịch vụ đám mây
6. Lưu trữ bằng tập tin XML hoặc JSON

Các kỹ thuật được thảo luận trong chương này có thể ứng dụng để **tạo ra và truy cập dữ liệu cá nhân, chia sẻ dữ liệu giữa các ứng dụng.**



# 1. Lưu dữ liệu với đối tượng SharedPreferences

- **SharePreferences** cho phép lưu trữ, điều chỉnh và phục hồi các **thiết lập** của người dùng trong một ứng dụng.
- **Thiết lập (setting)** là gì?  
các tính năng/hoạt động của ứng dụng được mặc định bởi nhà sản xuất thiết bị/người phát triển phần mềm/người dùng.
- **Preference** là gì?  
Là thiết lập tùy chỉnh (custom setting) bởi người dùng.
- Mỗi đối tượng **Preference** được dùng để xây dựng một thiết lập:  
**ListPreference** → danh sách các lựa chọn, **CheckBoxPreference** → chọn lựa các thiết lập, **EdittextPreference** → nhập vào một văn bản và lưu giá trị của nó như là một String.
- Mỗi Preference tương ứng với một cặp **key-value** mà hệ thống dùng để lưu một thiết lập trong một **tập tin SharePreferences** mặc định. Các value được lưu trong Sharereferences cho mỗi thiết lập có thể lấy một trong các giá trị sau: *Boolean, Float, Int, Long, String, String set*.
- **Đối tượng Preference** là kế thừa của **PreferenceFragment** hoặc **PreferenceActivity**.



# Lưu dữ liệu bằng cách sử dụng đối tượng SharedPreferences (2)

## ❖ Định nghĩa một Preference bằng XML

- Các thiết lập này được xây dựng bằng các lớp con của lớp **Preference** và chúng được khai báo trong một **tập tin XML**.
- Nên định nghĩa danh sách các thiết lập trong XML và có thể thay đổi bộ thiết lập này trong thời gian chạy bằng Java code.
- Mỗi lớp con của Preference được khai báo với một phần tử XML trùng với tên lớp, ví dụ như **<CheckBoxPreference>**.
- Chúng ta phải lưu tập tin XML vào thư mục **res/xml/**, tên truyền thống của nó là **preference.xml**.
- **Nút gốc của tập tin XML** phải là một phần tử **<PreferenceScreen>**. Bên trong ta thêm vào các đối tượng Preference.
- Nếu muốn tạo ra một layout nhiều cửa sổ, thì ta phải khai báo các tập tin XML khác nhau cho mỗi fragment.
- Nếu ta trình bày một danh sách từ 10 thiết lập trở lên, chia một số hoặc tất cả các thiết lập thành các nhóm. → hai cách sau: *dùng các tiêu đề* hoặc *dùng các màn hình con*. → Xem ví dụ



# Lưu dữ liệu bằng cách sử dụng đối tượng SharedPreferences (3)

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
xmlns:android=
"http://schemas.android.com/apk/res/android">
  <PreferenceCategory android:title="Category 1">
    <CheckBoxPreference
      android:title="Checkbox"
      android:defaultValue="false"
      android:summary="True or False"
      android:key="checkboxPref" />
  </PreferenceCategory>

  <PreferenceCategory android:title="Category 2">
    <EditTextPreference
      android:summary="Enter a string"
      android:defaultValue="[Enter a string here]"
      android:title="Edit Text"
      android:key="editTextPref" />
  </PreferenceCategory>
</PreferenceScreen>
```

```
<RingtonePreference
  android:summary="Select a ringtone"
  android:title="Ringtones"
  android:key="ringtonePref" />
<PreferenceScreen
  android:title="Second Preference Screen"
  android:summary="Click here to go to the
second Preference Screen"
  android:key="secondPrefScreenPref" >
  <EditTextPreference
    android:summary="Enter a string"
    android:title="Edit Text (second Screen)"
    android:key="secondEditTextPref" />
</PreferenceScreen>

</PreferenceCategory>

</PreferenceScreen>
```

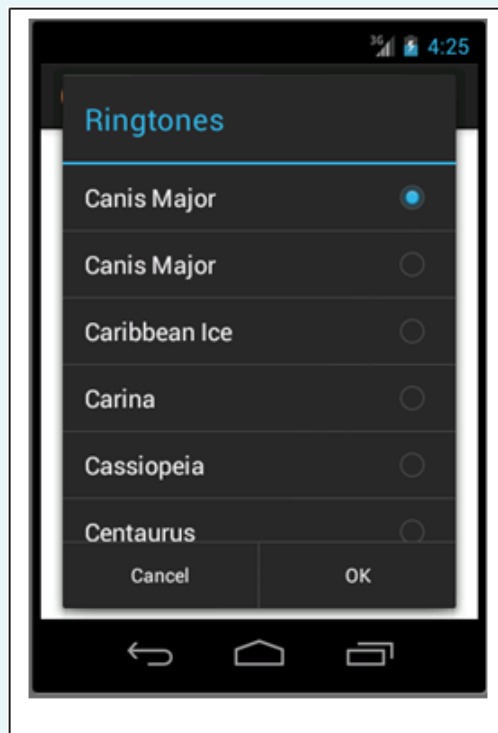
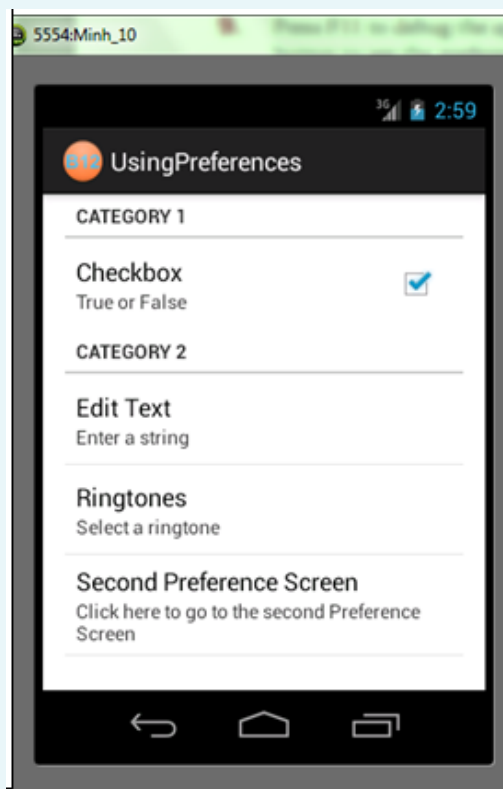


CANTHO UNIVERSITY

# Lưu dữ liệu bằng cách sử dụng đối tượng SharedPreferences (4)

**Dùng tiêu đề:** đặt các nhóm đối tượng [Preference](#) vào bên trong [PreferenceCategory](#).

**Dùng màn hình con:** đặt các nhóm đối tượng [Preference](#) vào bên trong [PreferenceScreen](#).



Hình 5.4: Chọn ringtones



Hình 5.5: Màn hình thứ 2



# Lưu dữ liệu bằng cách sử dụng đối tượng SharedPreferences (5)

## Dùng intents

- Trong vài trường hợp, ta muốn có một mục preference để mở một activity khác thay vì một màn hình thiết lập, chẳng hạn như mở một trình duyệt web để xem một trang web.
- Để gọi một intent khi người dùng chọn một mục preference, ta thêm vào một phần tử <intent> như là con của phần tử <Preference> tương ứng.

**Ví dụ**, dùng một mục preference để mở một trang web:

```
<Preference android:title="@string/prefs_web_page" >  
  <intent android:action="android.intent.action.VIEW"  
    android:data="http://www.example.com" />  
</Preference>
```



# Lưu dữ liệu bằng cách sử dụng đối tượng SharedPreferences (6)

- **Tạo ra một Preference Activity**
  - Dùng lớp [PreferenceActivity](#).
  - Dùng **PreferenceFragments**
- **Thiết lập giá trị mặc định**
  - android:defaultValue
  - Người dùng thay đổi, VD:  
`PreferenceManager.setDefaultValues(this, R.xml.My_preferences, false);`
- **Dùng Preference Headers**
  - **Tạo ra một tập tin header**
  - **Hiển thị header**
  - **Hỗ trợ preference headers cho phiên bản cũ**
- **Đọc Preferences:** Dùng hàm [getDefaultSharedPreferences\(\)](#)
- **Lắng nghe những thay đổi của preference:** dùng hàm `OnSharedPreferenceChangeListener`
- **Quản lý việc sử dụng mạng:** cho phép người dùng thấy được có bao nhiêu dữ liệu mạng mà ứng dụng đang dùng trong tiến trình foreground và background → tùy chỉnh mức độ truy cập
- **Xây dựng Preference tùy chọn**
  - Đặc tả giao diện người dùng
  - Lưu giá trị của các thiết lập
  - Khởi tạo giá trị hiện tại
  - Cung cấp một giá trị mặc định
  - Lưu và khôi phục trạng thái của Preference





# Lưu dữ liệu với SQLite – Giới thiệu

- Hệ cơ sở dữ liệu SQLite Database là hệ thống mã nguồn mở được sử dụng rộng rãi trong các ứng dụng, nhiều cty sử dụng (Adobe, Apple, Google, Sun, Symbian,...)
- SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C. Chương trình gồm một tập tin duy nhất chưa đến 300kB, không cần cài đặt, không cần cấu hình chỉ khởi động là có thể sử dụng ngay. Do có dung lượng nhỏ nên việc truy xuất dữ liệu được nhanh chóng, không chiếm dụng quá nhiều tài nguyên hệ thống.
- Dữ liệu “database” cũng được lưu ở một tập tin duy nhất. Không có khái niệm user, password hay quyền hạn trong SQLite database.
- Cung cấp cho người dùng gần như đầy đủ các chức năng mà một hệ CSDL cần có như tạo database; tạo bảng; thêm, xóa, sửa dữ liệu.
- Trong Android , cơ sở dữ liệu mà bạn tạo cho 1 ứng dụng thì chỉ ứng dụng đó có quyền truy cập và sử dụng, các ứng dụng khác thì không.
- Khi đã được tạo, cơ sở dữ liệu SQLite được chứa trong thư mục `/data/data/<package_name>/databases` .





# Lưu dữ liệu với SQLite - Giới thiệu

- SQLite hướng đến tiêu chí gọn nhẹ → không thể có đầy đủ các chức năng so với các hệ CSDL khác.
- Sử dụng dưới dạng thư viện nhúng, không chạy ở theo kiểu server độc lập.
- Tuy SQLite hỗ trợ trigger nhưng không thể viết trigger cho view. Hoặc SQLite không hỗ trợ lệnh ALTER TABLE, do đó không thể thực hiện chỉnh sửa hoặc xóa cột trong bảng.
- SQLite không hỗ trợ ràng buộc khóa ngoại, các transactions lồng nhau, phép kết RIGHT OUTER JOINT, FULL OUTER JOINT.
- SQLite sử dụng kiểu dữ liệu khác biệt so với hệ quản trị CSDL tương ứng. Có thể insert dữ liệu kiểu string vào cột kiểu integer mà không gặp phải bất kỳ lỗi nào.
- Vài tiến trình hoặc luồng có thể truy cập tới cùng một CSDL. Việc đọc dữ liệu có thể chạy song song, còn việc ghi dữ liệu thì không được phép chạy đồng thời.



# Lưu dữ liệu với SQLite.

## NỘI DUNG

- Cách tạo/xóa một cơ sở dữ liệu SQLite trong Android.
- Cách tạo/xóa bảng trong SQLite
- Cách thêm/sửa/xóa dữ liệu trong bảng
- Cách truy vấn dữ liệu trong bảng.
- Tạo 1 ứng dụng Android đơn giản sử dụng SQLite.



# Lưu dữ liệu với SQLite.

- Để thao tác với cơ sở dữ liệu lưu trữ SQLite trên Android ta sử dụng 2 đối tượng:
  - **SQLiteOpenHelper**: Đối tượng dùng để tạo, nâng cấp, đóng mở kết nối trên một cơ sở dữ liệu.
  - **SQLiteDatabase**: Đối tượng dùng để thực thi các câu lệnh SQL trên một cơ sở dữ liệu.
- Mỗi cơ sở dữ liệu của một ứng dụng sẽ có một đối tượng SQLiteOpenHelper.
- Thông qua đối tượng SQLiteOpenHelper ta lấy về một đối tượng kiểu SQLiteDatabase, đây chính là thể hiện của cơ sở dữ liệu ta cần thao tác.
- SQLiteOpenHelper hỗ trợ hai phương thức để lấy về một đối tượng SQLiteDatabase là:
  - `getReadableDatabase()`: Lấy về một đối tượng SQLiteDatabase ở dạng “chỉ đọc”.
  - `getWritableDatabase()`: Lấy về một đối tượng SQLiteDatabase ở dạng “đọc và ghi”.



# Lưu dữ liệu với SQLite.

- Để tạo mới một cơ sở dữ liệu ta kế thừa lớp SQLiteOpenHelper. SQLiteOpenHelper hỗ trợ cho chúng ta 3 phương thức chính:
  - **Phương thức khởi tạo** (constructor): Phương thức này cung cấp các tham số cần thiết để SQLiteOpenHelper có thể làm việc với cơ sở dữ liệu trên Android.
  - **Phương thức onCreate():** tạo **tập tin** lưu trữ cơ sở dữ liệu, tạo các **bảng** có trong cơ sở dữ liệu cũng như **nhập** các dữ liệu ban đầu.
  - **Phương thức onUpgrade():** được dùng để giúp bạn **cập nhật** lại các bảng (table) trong cơ sở dữ liệu.
- Mỗi cơ sở dữ liệu trên một ứng dụng Android có nhiều phiên bản. Các phiên bản này có thể tương ứng với các phiên bản của ứng dụng.  
**Ví dụ:** Ứng dụng quản lý buổi thực hành của sinh viên
  - Phiên bản 01: Gồm các chức năng: học kỳ, môn học, phòng thực hành.
  - Phiên bản 02: Thêm chức năng quản lý SV nên có thêm bảng sau: sinh viên.
  - Tương ứng với mỗi phiên bản ứng dụng kể trên sẽ có hai phiên bản cơ sở dữ liệu khác nhau. Mỗi khi nâng cấp **phiên bản ứng dụng** ta cũng phải tiến hành nâng **cập phiên bản cơ sở dữ liệu** với phương thức **onUpgrade()**.
- SQLite cũng sử dụng các câu truy vấn như các phần mềm quản lý cơ sở dữ liệu khác. Hai câu lệnh truy vấn phổ biến là **rawQuery()** và **query()**.



# Lưu dữ liệu với SQLite.

❖ **Tạo mới CSDL** - Chúng ta có thể dùng 3 cách sau đây:

**Cách 1:** Kế thừa lớp **SQLiteOpenHelper** với 3 phương thức chính:

- Phương thức khởi tạo (constructor): cung cấp các tham số cần thiết để SQLiteOpenHelper có thể thực thi CSDL trên Android.
- Phương thức onCreate(): tạo tập tin lưu trữ CSDL, tạo các bảng có trong CSDL cũng như nhập các dữ liệu ban đầu.
- Phương thức onUpgrade(): cập nhật lại các bảng (table) trong trong CSDL.

**Cách 2:** Dùng hàm **OpenOrCreateDatabase**. Ví dụ, chúng ta cần tạo CSDL có tên là "qlhs", cú pháp của hàm này như sau:

```
database=openOrCreateDatabase("qlhs.db", SQLiteDatabase.CREATE_IF_NECESSARY, null);
```

**Cách 3:** Dùng hàm **openDatabase** để tạo CSDL, câu lệnh sẽ có dạng như:  
`openDatabase("qlhs.db", null, SQLiteDatabase.CREATE_IF_NECESSARY);`



# Lưu dữ liệu với SQLite.

- Sau khi tạo xong CSDL, nó sẽ được lưu ở đường dẫn:  
`//data/data/<Your-Application-Package-Name>/databases/<your-database-name>`
- Trong trường hợp muốn lưu trữ CSDL vào trong thẻ nhớ, ta cần thêm lệnh permission sau đây vào tập tin ***AndroidManifest.xml***  
`<uses-permission  
Android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`
- Lúc này ở tham số thứ nhất của hàm tạo CSDL, chúng ta chỉ cần chỉ rõ đường dẫn của thư mục trong thẻ nhớ mà bạn muốn lưu trữ CSDL vào đó.



# Lưu dữ liệu với SQLite.

## ❖ Xóa CSDL

- Dùng hàm **deleteDatabase(<tên CSDL>)** để thực hiện việc xóa CSDL, nếu thành công trả về **true**, thất bại trả về **false**.
- Đoạn lệnh sau minh họa việc xóa CSDL, và thông báo thành công hoặc thất bại khi xóa CSDL:

```
String msg="";  
if(deleteDatabase("qlhs")==true)  
msg="Xóa thành công CSDL";  
else  
msg="Việc xóa thất bại. Vui lòng thử lại!";  
Toast.makeText(this,msg, Toast.LENGTH_LONG).show();
```





# Lưu dữ liệu với SQLite.

## ❖ Tạo bảng/Xóa bảng

- Chúng ta có thể dùng lệnh SQL kết hợp với phương thức **execSQL** để thực thi việc tạo các bảng

- **Bước 1:** Gán chuỗi gồm các câu lệnh SQL. Ví dụ:

//Lệnh tạo bảng

```
database=openOrCreateDatabase("qlhs.db",  
    SQLiteDatabase.CREATE_IF_NECESSARY, null);
```

```
String CreatetbSchool="create table schools ("+"id integer primary key  
autoincrement,"+"nameschool text, "+"address text)";
```

//Lệnh xóa bảng:

```
String DeletetbSchool="drop table schools";
```

- **Bước 2:** Thực thi các câu lệnh

```
database.execSQL(CreatetbSchool);
```

```
database.execSQL(DeletetbSchool);
```



# Lưu dữ liệu với SQLite.

## ❖ Thêm/Sửa/Xóa dữ liệu trong bảng

### ➤ Thêm/Sửa dữ liệu:

- **Bước 1:** Dùng đối tượng **ContentValues** để đưa dữ liệu vào bảng. Đối tượng này có các phương thức **put (tên cột , dữ liệu)**.

Cú pháp: `put(<tên cột>,<dữ liệu muốn thêm vào cho cột>`

- **Bước 2:** Gọi phương thức **insert** để đưa thêm dữ liệu cho bảng hoặc phương thức **update** để cập nhật giá trị trong bảng.

Cú pháp: `insert(<tên bảng>,<Cột cho phép null>,<dữ liệu muốn thêm vào>`

Cú pháp: `update(<tên bảng>,<DL cần cập nhật >,<Cột lọc điều kiện>,<Giá trị của các điều kiện>`

### ➤ Xóa dữ liệu:

- Dùng hàm **delete** để xóa dữ liệu trong bảng, cú pháp:  
`public int delete (String<Tên bảng>, String <Điều kiện lọc để xóa>,String[] <Giá trị của điều kiện>).`



# Lưu dữ liệu với SQLite

## ❖ Truy vấn dữ liệu trong bảng

➤ Dùng hàm query để truy vấn DL và biến Cursor để lưu trữ giá trị trả về:

```
public Cursor query(String table, String[] columns, String selection, String[] s  
electionArgs, String groupBy, String having, String orderBy)
```

<b>table</b>	Tên bảng được chọn để truy vấn
<b>columns</b>	Danh sách các cột trả về, nếu tham số này null thì tất cả các cột trong bảng sẽ được trả về
<b>selection</b>	Điều kiện lọc (danh sách các cột trong điều kiện lọc), nếu null thì mặc nhiên lấy hết các dữ liệu trong bảng
<b>selectionArgs</b>	Tập hợp các giá trị của điều kiện lọc
<b>groupBy</b>	Nhóm các dữ liệu, nếu tham số này null thì sẽ không nhóm dữ liệu
<b>Having</b>	Điều kiện trên hàm kết tập (khi nhóm dữ liệu)
<b>orderBy</b>	Sắp xếp dữ liệu trả về theo các cột được liệt kê



# Lưu dữ liệu với SQLite.

## Ví dụ: Các bước thực hiện với CSDL

- Tạo 1 CSDL ( thông thường chỉ cần làm 1 lần )
- Mở CSDL đó
- Thêm giá trị vào trong table
- Truy vấn
- Đóng CSDL

Ta sẽ tạo chương trình gồm hai button để thêm và hiển thị thông tin về quyền sách, thông tin sẽ được hiển thị trên một Toast.

Trước tiên tạo Android Project (tên **TestSQLite**).

### Tạo cơ sở dữ liệu.

Trước tiên ta tạo class **DBAdapter** để xử lý tất cả các thao tác liên quan đến CSDL. → Khai báo các biến



# Lưu dữ liệu với SQLite. Ví dụ

```
public static final String KEY_ID = "_id";  
public static final String KEY_NAME = "name";  
private DatabaseHelper mDbHelper;  
private SQLiteDatabase mDB;  
private static final String DATABASE_CREATE = "create table users  
(_id integer primary key autoincrement, " + "name text not null);";  
private static final String DATABASE_NAME = "Database_Demo";  
private static final String DATABASE_TABLE = "users";  
private static final int DATABASE_VERSION = 2;  
private final Context mContext;  
private DatabaseHelper DBHelper;  
private SQLiteDatabase db;
```



# Lưu dữ liệu với SQLite.

## Các bước thực hiện với CSDL

- Bên trong DBAdapter ta tạo 1 lớp **DatabaseHelper** kế thừa lớp **SQLiteOpenHelper**,
- override 2 phương thức **onCreate()** và **onUpgrade()** để quản lý việc tạo CSDL và version của CSDL đó.

```
private static class DatabaseHelper extends SQLiteOpenHelper{ public
DatabaseHelper(Context context, String name, CursorFactory factory, int version) {
super(context, name, factory, version);
}
@Override public
void onCreate(SQLiteDatabase db) {
db.execSQL(DATABASE_CREATE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
Log.i(TAG, "Upgrading DB");
db.execSQL("DROP TABLE IF EXISTS users");
onCreate(db); }
}
```



# Lưu dữ liệu với SQLite. Các bước thực hiện với CSDL

## Mở CSDL :

```
public DBAdapter open() {  
    mDbHelper = new DatabaseHelper(mContext,  
        DATABASE_NAME, null, DATABASE_VERSION);  
    mDB = mDbHelper.getWritableDatabase();  
    return this;  
}
```

## Thêm giá trị vào CSDL

```
public long createUser(String name) {  
    ContentValues inititalValues = new ContentValues();  
    inititalValues.put(KEY_NAME, name);  
    return mDB.insert(DATABASE_TABLE, null, inititalValues);  
}
```





# Lưu dữ liệu với SQLite.

## Các bước thực hiện với CSDL

- **Truy vấn:** có thể get toàn bộ data hoặc có thể get data theo ID (tiện cho việc chỉnh sửa hay cập nhật thông tin của từng bản ghi).  

```
public Cursor getAllUsers() {  
    return mDB.query(DATABASE_TABLE, new String[] {KEY_ID,  
        KEY_NAME}, null, null, null, null, null);  
}
```
- Còn rất nhiều các thao tác như sửa, xóa, update.... bản ghi, các bạn có thể tự phát triển. Tất cả các chức năng đó đều được cung cấp bởi lớp SQLiteDatabase, các bạn chỉ cần cụ thể hóa bằng các câu truy vấn là được.
- **Đóng CSDL**  

```
public void close() {  
    mDbHelper.close();  
}
```



# Các bước thực hiện với CSDL

- **Sử dụng CSDL:** Để test CSDL mà bạn vừa tạo, các bạn có thể thêm 1 vài dòng code để thêm 1 user và hiển thị CSDL lên màn hình thông qua lớp Activity → tạo ra 1 user thông qua câu lệnh  
`mDB.createUser("Username");`  
tiếp theo là:  

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    mDB = new DBAdapter(this);  
    mDB.open();  
    mDB.createUser("Tran Van B");  
    getData();  
}
```



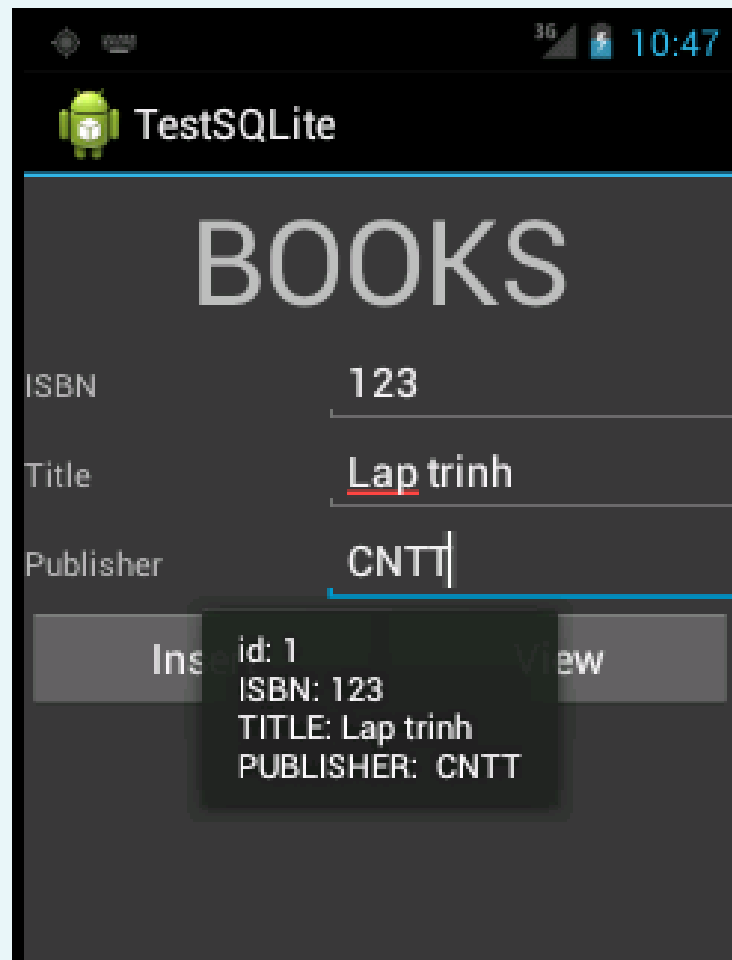
## Lưu dữ liệu với SQLite. Các bước thực hiện với CSDL

```
private void getData() {  
    mCursor = mDB.getAllUsers();  
    startManagingCursor(mCursor);  
    String[] from = new String[]{DBAdapter.KEY_NAME};  
    int[] to = new int[] {R.id.text1};  
    SimpleCursorAdapter users = new SimpleCursorAdapter(this,  
        R.layout.users_row, mCursor, from, to);  
    setListAdapter(users);  
}
```

- Ứng dụng gồm: lớp [DBAdapter](#); [Layout](#); [Activity](#)



# Lưu dữ liệu với SQLite. Kết quả





### 3. Lưu dữ liệu với Content Provider

- Để chia sẻ dữ liệu giữa các ứng dụng ta sẽ dùng thành phần Content Provider.
- Content Provider có thể coi như là một cơ chế cho phép truy xuất/thêm/sửa/xóa dữ liệu của một ứng dụng từ một ứng dụng khác.
- Trong Android có nhiều Content Provider hữu dụng như :
  - Browser - Lưu trữ các dữ liệu như browser bookmarks, browser history, ...
  - CallLog - Lưu trữ các dữ liệu như cuộc gọi nhỡ, chi tiết cuộc gọi, ...
  - Contacts - Lưu trữ các liên lạc.
  - MediaStore - Lưu trữ các tập tin đa phương tiện như audio, video, và hình ảnh.
  - Settings - Lưu trữ các thiết lập của thiết bị và các tùy chọn của người dùng.
  - ...



### 3. Lưu dữ liệu với Content Provider

- Bên cạnh các Content Provider xây dựng sẵn, ta cũng có thể tạo ra các Content Provider.
- Để truy vấn một Content Provider, chúng ta khai báo một chuỗi truy vấn giống như một URI. Cú pháp chung như sau:

`<standard_prefix>://<authority>/<data_path>/<id>`

Trong đó:

- **standard prefix** - cho các nội dung luôn luôn là **content://**
- **authority** - xác định tên của Content Provider. Ví dụ **contacts** cho Content Provider Contacts được dựng sẵn. Với Content Provider của ta tạo ra có thể là một tên đầy đủ, chẳng hạn như *com.bai5.provider*.
- **data\_path** - xác định loại dữ liệu được yêu cầu. Ví dụ, nếu chúng ta lấy tất cả các contacts từ một Contacts content provider, thì đường dẫn dữ liệu sẽ là *people*, và URI sẽ như sau: *content://contacts/people*.
- **id** chỉ định một record được yêu cầu. Ví dụ, nếu ta tìm contact số 2 trong Contacts Content Provider, thì URI sẽ là: *content://contacts/people/2*.



### 3. Lưu dữ liệu với Content Provider

Ví dụ:

Query string	Diễn giải
<b>content://media/internal/images</b>	Trả về một danh sách của tất cả các hình ảnh trong thiết bị.
<b>content://media/external/images</b>	Trả về một danh sách của tất cả các hình ảnh được lưu trữ trong bộ nhớ ngoài (ví dụ như thẻ SD) trên thiết bị.
<b>content://call_log/calls</b>	Trả về một danh sách của tất cả các cuộc gọi đã đăng ký trong Call Log
<b>content://browser/bookmarks</b>	Trả về một danh sách các bookmark được lưu trữ trong trình duyệt.





### 3. Lưu dữ liệu với Content Provider

- Để lấy các kết quả trả về ta cũng có thể dùng con trỏ để quản lý, có 2 cách sử dụng hàm lấy kết quả ở đây:

#### Cách 1:

```
CursorLoader loader=new CursorLoader(context, uri, null, null, null,null);  
Cursor c=loader.loadInBackground();
```

#### Cách 2:

```
Cursor c = getContentResolver().query(uri, null, null, null, null);
```

CursorLoader(Context context) – Tạo ra 1 CursorLoader rỗng không xác định  
CursorLoader(Context context, Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) Creates a fully-specified CursorLoader

Các tham của hàm query(Uri,projection,selection,selectionArgs,sortOrder) khớp với 1 câu lệnh SQL SELECT:



### 3. Lưu dữ liệu với Content Provider

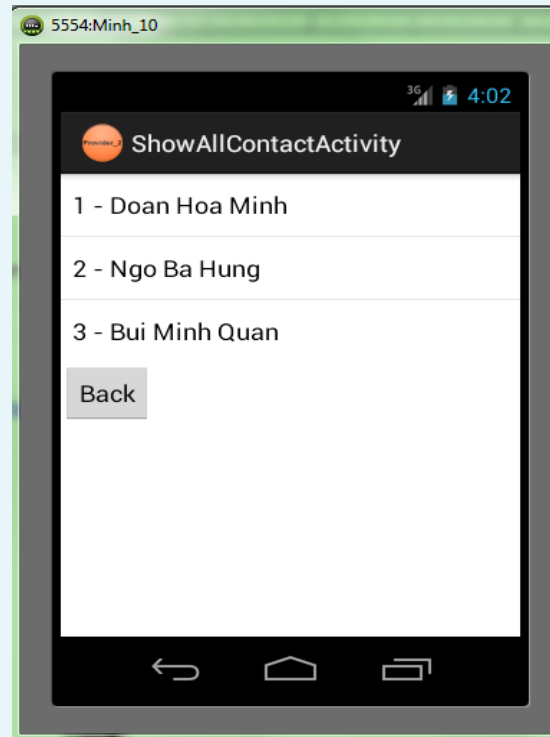
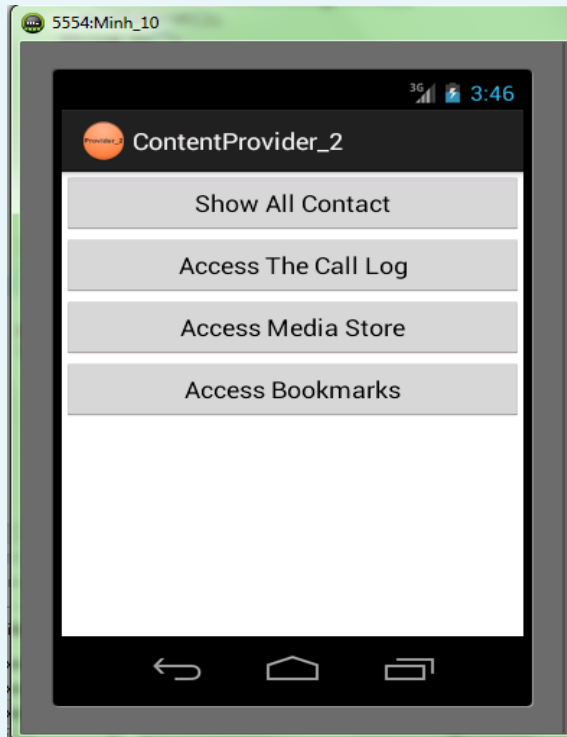
#### Parameters

uri	URI để truy vấn. Đây sẽ là URI đầy đủ do client gửi; Nếu client yêu cầu một bản ghi cụ thể, URI sẽ kết thúc một con số record mà việc thực thi cần phân tích cú pháp và thêm vào một mệnh đề WHERE hoặc HAVING, chỉ rõ giá trị _id đó.
projection	String: Danh sách các cột để đưa vào con trỏ. Nếu null tất cả các cột được bao gồm.
selection	String: Một tiêu chí lựa chọn để áp dụng khi lọc hàng. Nếu null thì tất cả các hàng được bao gồm.
selectionArgs	String: Có thể bao gồm ?s trong lựa chọn, sẽ được thay thế bằng các giá trị từ selectionArgs, để chúng xuất hiện trong vùng lựa chọn. Các giá trị sẽ bị ràng buộc như Strings.
sortOrder	String: Cách để các hàng trong cursor cần được sắp xếp. Nếu null thì provider có thể tự do định nghĩa thứ tự sắp xếp.



### 3. Lưu dữ liệu với Content Provider

**Ví dụ:** xây dựng 1 ứng dụng hiển thị contact, call log, media và bookmarks trên một điện thoại di động. Giao diện gồm 4 button như hình sau, khi người dùng muốn xem nội dung nào thì chọn button đó



xây dựng 2 Activity:  
[MainActivity.java](#)  
và  
**ShowAllContactActivity.java**, tương ứng có 2 tập tin layout là **activity\_main.xml** và **activity\_show\_all\_contact.xml**.




### 3. Lưu dữ liệu với Content Provider

Chạy thử với Andoid emulator

1. Thêm 1 folder vào sdCard:

DDMS → File Explorer → mnt/sdcard → click vào biểu tượng plus (+) ở góc trên phải → cửa sổ xuất hiện → đặt tên cho folder mới → OK.

2. Thêm ringtone cho emulator:

DDMS → Storage → sdcard → Ringtone (nếu chưa có thì tạo ra folder mới này) → vào biểu tượng “push a file onto the device”  → cửa sổ push a file onto the device hiện ra, tìm tập tin được lưu trong thư mục nào đó của PC → Chọn Open.

3. Thêm nhạc cho emulator:

Tương tự như mục 2, nhưng thêm tập tin nhạc vào thư mục Music.

4. Thêm hình cho emulator:

Tương tự như mục 2, nhưng thêm tập tin hình vào thư mục Picture.

5. Thêm contact cho emulator: trực tiếp trên emulator.



## 4. LƯU TRỮ DỮ LIỆU BẰNG TẬP TIN

### ❖ Lưu trữ dữ liệu trên bộ nhớ trong

- Để lưu văn bản vào tập tin ta đã dùng lớp `FileOutputStream`, dùng phương thức `openFileOutput()` mở tập tin

```
FileOutputStream fOut = openFileOutput("textfile.txt", MODE_WORLD_READABLE);
```

- `MODE_WORLD_READABLE` để chỉ định rằng tập tin có thể đọc bởi ứng dụng khác. Nếu ta dùng hằng `MODE_PRIVATE` thì tập tin chỉ có thể đọc bởi ứng dụng đã tạo ra nó, và nếu dùng `MODE_WORLD_WRITEABLE` thì tập tin có thể được truy xuất để đọc và chỉnh sửa với mọi ứng dụng.
- Để chuyển đổi một dòng ký tự thành một dòng byte, ta sử dụng một thể hiện của lớp `OutputStreamWriter`:

```
OutputStreamWriter osw = new OutputStreamWriter(fOut);
```

- Sau đó, ta sử dụng phương thức **`write ()`** để viết các chuỗi ký tự vào tập tin. Để đảm bảo rằng tất cả các byte được ghi vào tập tin, ta dùng phương thức **`flush()`**. Cuối cùng, dùng phương thức **`close()`** để đóng tập tin.



## 4. LƯU TRỮ DỮ LIỆU BẰNG TẬP TIN

- Để đọc tập tin ta dùng lớp **FileInputStream** kết hợp với lớp **InputStreamReader**. Vì không biết trước kích thước của tập tin sẽ đọc, nội dung được đọc từng khối 100 ký tự đưa vào bộ đệm (character array). Sau đó các ký tự đã đọc được chép vào một đối tượng String.
- Hàm `read()` kiểm tra số ký tự đã đọc và trả về giá trị -1 khi kết thúc tập tin.
- Thử ứng dụng với thiết bị giả lập, ta dùng DDMS để kiểm tra

```
FileInputStream fln = openFileInput("textfile.txt");
InputStreamReader isr = new
    InputStreamReader(flن);
char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";
int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
    //---convert the chars to a String---
    String readString =
        String.copyValueOf(inputBuffer, 0, charRead);
    s += readString;
    inputBuffer = new char[READ_BLOCK_SIZE];
}
```

[Xem code](#)



## 4. LƯU TRỮ DỮ LIỆU BẰNG TẬP TIN

- ❖ Lưu dữ liệu vào bộ nhớ ngoài (chẳng hạn như SD card)
- Dùng phương thức **getExternalStorageDirectory()** để trả về đường dẫn đầy đủ tới bộ nhớ ngoài. Cụ thể là đường dẫn “/sdcard” cho thiết bị thật và “/mnt/sdcard” cho thiết bị giả lập.
- Tuy nhiên, ta không nên “mã cứng” đường dẫn đến SD card, khi nhà sản xuất thiết bị chỉ định tên đường dẫn khác tới SD card. Vì vậy, cần phải dùng phương thức **getExternalStorageDirectory()** để trả về đường dẫn đầy đủ đến SD card.

```
File sdCard =  
Environment.getExternalStorageDirectory();  
File directory = new File  
(sdCard.getAbsolutePath() + “/MyFiles”);  
File file = new File(directory, “textfile.txt”);  
FileInputStream fln = new FileInputStream(file);  
InputStreamReader isr = new  
InputStreamReader(fln);
```

Lưu ý, để có thể ghi lên bộ nhớ ngoài, ta cần thêm permission **WRITE\_EXTERNAL\_STORAGE** vào tập tin **AndroidManifest.xml**, như sau;

```
<uses-permission  
android:name=”android.permission.WRITE_  
EXTERNAL_STORAGE” />
```

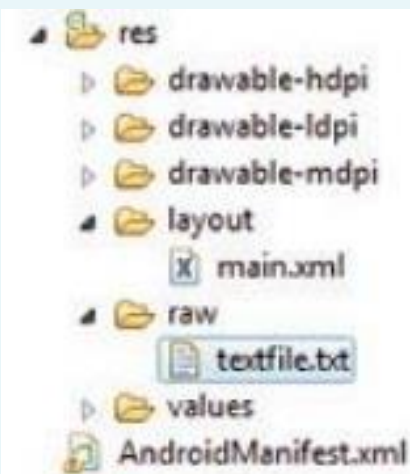




## 5. SỬ DỤNG TÀI NGUYÊN TĨNH

### ❖ Tài nguyên tĩnh là gì?

- Bên cạnh việc sinh ra tập tin trong thời gian chạy, ta cũng cần tạo ra các tập tin trong thời gian phát triển ứng dụng, sau đó sẽ được dùng trong thời gian chạy. Ví dụ, ta muốn tạo ra tập tin trợ giúp (Help) cho ứng dụng.
- Trong trường hợp này, ta thêm tập tin help vào thư mục res/raw. Hình 5.14 trình bày thư mục res/raw chứa tập tin có tên là textfile.txt đã được tạo ra khi lập trình.



Để xây dựng một tài nguyên tĩnh như trên, trong Activity, ta sử dụng hàm **getResources()** để trả về một đối tượng **resources** và dùng hàm **openRawResource()** để mở tập tin chứa trong thư mục res/raw.

Lưu ý, ID của tài nguyên được lưu trong thư mục res/raw được đặt tên không có phần mở rộng sau tên tập tin. VD, với tên tập tin là textfile.txt, thì resource ID là R.raw.textfile.

### Mã nguồn



## 6. ĐA NGÔN NGỮ TRONG ANDROID

- Android mặc định tiếng Anh là ngôn ngữ chính, tạo sẵn và nạp chuỗi từ tập tin strings trong thư mục values (đường dẫn: res/values/strings.xml). Khi ta muốn bổ sung thêm những ngôn ngữ khác, ta cần phải tạo thư mục mới với tên được đặt theo quy tắc là value-mã ngôn ngữ quốc gia. Ví dụ nếu bạn muốn thêm tiếng Việt, ta phải tạo một thư mục với tên values-vi, sao chép tập tin strings.xml vào trong thư mục này và dịch sang tiếng Việt.
- Việc thay đổi chuỗi sang ngôn ngữ khác nhau được gọi là bản địa hóa (localization). Với ứng dụng được hỗ trợ nhiều ngôn ngữ, người dùng có thể chọn ngôn ngữ như sau:
  1. Trên điện thoại (kể cả máy ảo), chọn Settings hoặc Custom Locale ⇒ check chọn ngôn ngữ mà ta muốn.
  2. Nếu ứng dụng hỗ trợ lựa chọn ngôn ngữ, android tìm kiếm các nguồn tài nguyên phù hợp với ngôn ngữ được chọn từ values-(mã ngôn ngữ)
  3. Nếu các chuỗi nội địa hoá bị thiếu, android sẽ nạp các chuỗi đó từ tập tin strings.xml (res ⇒ values ⇒ strings.xml).



CANTHO UNIVERSITY

## 4. LƯU TRỮ DỮ LIỆU BẰNG TẬP TIN

