DATABASE

SQLite

- Giới thiệu
- Khai báo và khởi tạo
- Truy xuất dữ liệu
- Các đặc điểm khác
- Leak memory

GIỚI THIỆU

- Tương tự như MySQL hay SQLserver, SQLite cũng là một hệ thống quản lý dữ liệu nhưng gọn, nhẹ và đơn giản hơn
- Không có khái niệm user, password hay quyền hạn
- Công cụ quản lý trực quan có SQLite hoặc SQLite Administrator (window) và SQLiteBrowser (Ubuntu)
- Phiên bản được sử dụng trong Android là SQLite 3.x

GIỚI THIỆU

- Cơ sở dữ liệu dạng quan hệ (RDBMS), hỗ trợ chuẩn SQL-92
- Sử dụng dưới dạng thư viện nhúng, không chạy ở theo kiểu server độc lập
- Hỗ trợ các ngôn ngữ phổ biến: C, C++, C#, Basic, Perl, Ruby, Python, PHP, Java ...
- Không cần chỉ định kiểu dữ liệu (SQLite is typeless)
- Hỗ trợ mã UTF8
- Hỗ trợ command line
- Hỗ trợ transaction, view
- Hỗ trợ C extensions
- ...

KHỞI TẠO DATABASE

- SQLiteHelper : lớp cho phép tạo và quản lý phiên bản của database
- Để khởi tạo một database cần có một lớp kế thừa từ SQLiteHelper. Trong đó tao override lại phương thức onCreate()

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
                "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
                KEY_WORD + " TEXT, " +
                KEY_DEFINITION + " TEXT);";
    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
}
```

KHỞI TẠO DATABASE

- Để sử dụng database, trước tiên cần tạo một lớp điều khiển đóng mở và thao tác dữ liệu với database.

```
Public class MyDatabase{
        DictionaryOpenHelper mDbHelper ;
        SQLiteDatabase mDB;
        …
}
```

- Mở một database và cho phép có thể ghi dữ liệu

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

TRUY VẤN DỮ LIỆU

- Trong Android, lớp SQLiteDatabase cung cấp sẵn cho ta các phương thức insert, update, delete, query ... Ngoài ra còn có thể thực thi thông qua rawQuery(String sql, String[] args) hoặc execSQL(String sql)

```
public long createUser(String name){
        ContentValues inititalValues = new ContentValues();
        inititalValues.put("columnName", name);
        return mDB.insert("tableName", null, inititalValues);
}
…
public Cursor getAllUsers(){
        return mDB.query("tableName", new String[] {"columnId", "columnName"}, null, null,
null, null, null);
}
```

TRUY VẤN DỮ LIỆU

- Chúng ta cũng có thể thực hiện với rawQuery hay execSQL như sau

```
public Cursor getAllUsers(){
        String sqlCommand = "Select * from tableUser";
        return mDB.rawQuery(sqlCommand, null);
}
…
public Cursor getAllUsers(){
        String sqlCommand = "Select * from tableUser";
        return mDB.execSQL(sqlCommand);
}
```

- *Khuyến cáo tránh sử dụng rawQuery hay execSQL trừ trường hợp các truy vấn quá phức tạp*

LEAK MEMORY

- Sau khi kết thúc phiên làm việc với database, nhất thiết phải đóng database lại

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

- Nếu database không được đóng, sẽ gây ra hiện tượng leak memory và làm chậm chương trình

CONTEN PROVIDER

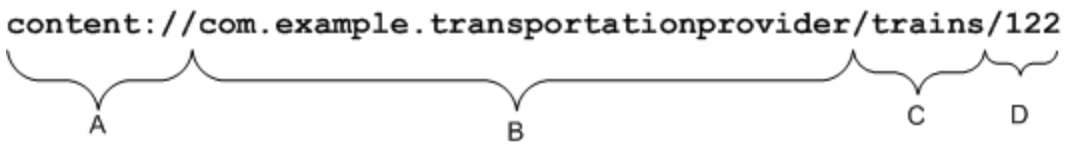
- Giới thiệu
- Uri
- Cách khai báo và khởi tạo
- Cách truy xuất dữ liệu

GIỚI THIỆU

- Một Content Provider cung cấp một tập chi tiết dữ liệu ứng dụng đến các ứng dụng khác. Thường được sử dụng khi chúng ta muốn tạo cơ sở dữ liệu dưới dạng public (các ứng dụng khác có thể truy xuất).
  Dữ liệu thường được lưu trữ ở file hệ thống, hoặc trong một SQLite database.
- Ví dụ về Content Providers : danh bạ, tin nhắn, thông số hệ thống, lịch sử cuộc gọi …

URI

- Uri



- A : phần bắt buộc
- B : phần xác thực provider được khai báo trong manifest
- C : đường dẫn định danh các kiểu dữ liệu được request
- D : id của một record

KHỞI TẠO

- Khai báo trong manifest

  <provider android:name="org.android.testlib.MyContentProvider"

  android:authorities="org.android.testlib.contentprovider"/>

- Ở đây ta thấy trong package org.android.testlib sẽ có một file MyContentProvider.java. Khi ứng dụng được thực thi, hệ thống sẽ tự động thực thi file này để tạo ra một content provider
- Lớp MyContentProvider phải được kế thừa từ lớp ContentProvider, trong đó implement các phương thức tương tác với dữ liệu

  Public class MyContentProvider extends ContentProvider{

  …

  public boolean onCreate(){}

  }

TRUY VẤN DỮ LIỆU

- Lưu trữ dữ liệu trong ContentProvider có nhiều cách, ở đây ta dùng SQLite. Việc tạo một database cũng tương tự như phần SQLite.
- Tạo một lớp DatabaseHelper kế thừa từ lớp SQLiteHelper
- Trong hàm onCreate() của MyContentProvider ta thực thi

```
Context context = getContext();
DatabaseHelper dbHelper = new DatabaseHelper(context);
SQLiteDatabase sqlDB = dbHelper.getWritableDatabase();
return (sqlDB == null) ? false : true;
```

- Sử dụng Cursor để thao tác trên tập dữ liệu : query (), update(), insert(), delete()….. Có thể gọi phương thức ContentResolver.notifyChange() để biếtkhi nào dữ liệu được cập nhật.

TRUY VẤN DỮ LIỆU

```
@Override
    public Uri insert(Uri uri, ContentValues values) {
        long rowID = sqlDB .insert(DATABASE_TABLE, "", values);
        if(rowID > 0)
        {
            Uri mUri = ContentUris.withAppendedId(CONTENT_URI, rowID);
            getContext().getContentResolver().notifyChange(mUri, null);
            return mUri;

        }
        throw new SQLException("Failed to insert new row into " + uri);
    }
```

TRUY VẤN DỮ LIỆU

- Truy vấn từ một ứng dụng khác

Uri uriGetListTitles = Uri.parse(CONTENT_URI);

Cursor c = managedQuery(uriGetListTitles, projection, selection, selectionArgs, orderBy);