

# CHƯƠNG 4

## LẬP TRÌNH LƯU TRỮ DỮ LIỆU TRÊN TBDĐ

Cơ sở dữ liệu SQLite

## SQLITE



**Database**



**SQLite**

EMEZETACOM

## KHÁI NIỆM CSDL TRÊN TBDD

- SQLite là phần mềm quản lý cơ sở dữ liệu (DBMS) tương tự như Mysql, PostgreSQL...
- Đặc điểm của SQLite là gọn, nhẹ, đơn giản, mã nguồn mở.
- Chương trình gồm 1 file duy nhất kích thước nhỏ hơn 500kB, không cần cài đặt, không cần cấu hình hay khởi động mà có thể sử dụng ngay.
- Dữ liệu database cũng được lưu ở một file duy nhất. Không có khái niệm user, password hay quyền hạn trong SQLite database

## LỊCH SỬ HÌNH THÀNH SQLITE.

- Năm 2000: D.Richard Hipp đã thiết kế SQLite dưới dạng thư viện bằng ngôn ngữ lập trình c với mục đích không cần quản trị để điều hành một chương trình.
- Tháng 8, SQLite 1.0 được công bố với GNU Database Manager
- Năm 2011: Hipp bổ sung UNQI Interface cho SQLite DB và để phát triển UNQLite

# 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 truy cập vào các tệp trong ứng dụng
- Cơ sở dữ liệu cho web.
  - SQLite là lựa chọn phổ biến làm cơ sở dữ liệu để quay lại các trang web vừa và nhỏ.
- Thử nghiệm hoặc demo ứng dụng trong doanh nghiệp.

## ƯU ĐIỂM CỦA SQLITE

Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng, đa nền tảng

Thao tác đơn giản nhanh hơn hệ thống tệp trực tiếp I/O

Kích thước chương trình gọn nhẹ( $\leq 400$  KB)



Tin cậy, không gây lỗi khi xảy ra sự cố phần cứng

Tuân theo chuẩn SQL92

Không cần cấu hình, có nghĩa là không cần thiết lập hoặc quản trị.

## NHƯỢC ĐIỂM CỦA SQLITE

- Do sử dụng cơ chế coarse-grained locking nên trong cùng một thời điểm SQLite có thể hỗ trợ nhiều người đọc dữ liệu, nhưng chỉ có 1 người có thể ghi dữ liệu.
- Khi sử dụng SQLite bạn sẽ phải viết những đoạn code lặp đi lặp lại vừa tốn thời gian vừa dễ mắc lỗi
- SQLite không phải là lựa chọn hoàn hảo để đáp ứng các nhu cầu xử lý trên một khối lượng **dữ liệu lớn**, phát sinh liên tục.
- Phù hợp phát triển và thử nghiệm các hệ thống vừa và nhỏ (không cần cấu hình khi cài đặt)



# CÁC ĐẶC ĐIỂM NỔI BẬT CỦA SQLITE

- AUTOINCREMENT?
  - Một cột được khai báo INTEGER PRIMARY KEY sẽ tự động tăng thêm
- Các kiểu dữ liệu: Integer, Real, Text, Blob, Null
- 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



# CÁC ĐẶC ĐIỂM NỔI BẬT CỦA SQLITE

- Khóa ngoài
  - SQLite hỗ trợ khóa ngoài nhưng theo mặc định thì khóa ngoài sẽ tắt.
- 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 \_ )

# CÚ PHÁP TRONG SQLITE

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

1	<code>CREATE TABLE &lt;dbname.table&gt; ();</code>	Tạo bảng
2	<code>DROP TABLE database_name.table_name;</code>	Xóa bảng
3	<code>INSERT INTO table_name [(column1, column2,...)] VALUES (value1, value2,...);</code>	Thêm dữ liệu vào bảng
4	<code>INSERT INTO table1 [(column...)] SELECT column FROM table2 [WHERE];</code>	Chèn dữ liệu vào bảng từ một bảng khác
5	<code>SELECT sql FROM table;</code>	Hiển thị thông tin bảng



## CÚ PHÁP TRONG SQLITE

6	<code>SELECT ( 12+8) AS ADDITION; #20</code>	Thực hiện biểu thức số học
7	<code>SELECT COUNT(*) AS "RECORDS" FROM table;</code>	đếm bản ghi trong bảng
8	<code>SELECT CURRENT_TIMESTAMP;</code>	Hiển thị thời gian hệ thống
9	<code>UPDATE table_name SET column1 = value,... WHERE ...;</code>	Update dữ liệu bảng
10	<code>DELETE FROM table_name WHERE ...; Xóa bản ghi</code>	<code>DELETE FROM table_name WHERE ...; Xóa bản ghi</code>

## CÚ PHÁP TRONG SQLITE

11	<code>SELECT ... FROM table1 CROSS JOIN table2 ...</code>	CROSS JOIN: kết nối mọi hàng của bảng đầu tiên với mỗi hàng của bảng thứ hai
12	<code>SELECT ... FROM table1 [INNER] JOIN table2 ON conditional_expression ...</code>	INNER JOIN
13	<code>SELECT ... FROM table1 LEFT OUTER JOIN table2 ON conditional_expression ...</code>	OUTER JOIN: chỉ hỗ trợ LEFT JOIN

## VỊ TRÍ LƯU TRỮ

- DATA/data/TÊN\_APP/databases/TÊN\_FILE.DB
  - **DATA**: Đường dẫn này được tạo nên từ kết quả trả về của hàm Environment.getDataDirectory()
  - **TÊN\_APP**: Tên ứng dụng.
  - **TÊN\_FILE**: Tên của CSDL

# ***CẤU TRÚC CỦA LỚP SQLITEOPENHELPER***

Cấu trúc	Mô tả
<code>SQLiteOpenHelper</code> (Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	tạo ra một đối tượng để tạo, mở và quản lý cơ sở dữ liệu.
<code>SQLiteOpenHelper</code> (Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	tạo ra một đối tượng để tạo, mở và quản lý cơ sở dữ liệu. Có chỉ định xử lý lỗi.



# ***PHƯƠNG THỨC CỦA LỚP SQLITEOPENHELPER***

Phương thức	Mô tả
public abstract void <b>onCreate</b> (SQLiteDatabase db)	Gọi chỉ một lần khi cơ sở dữ liệu được tạo ra lần đầu tiên.
public abstract void <b>onUpgrade</b> (SQLiteDatabase db, int oldVersion, int newVersion)	gọi khi cơ sở dữ liệu cần phải được nâng cấp.
public synchronized void <b>close</b> ()	đóng các đối tượng cơ sở dữ liệu.





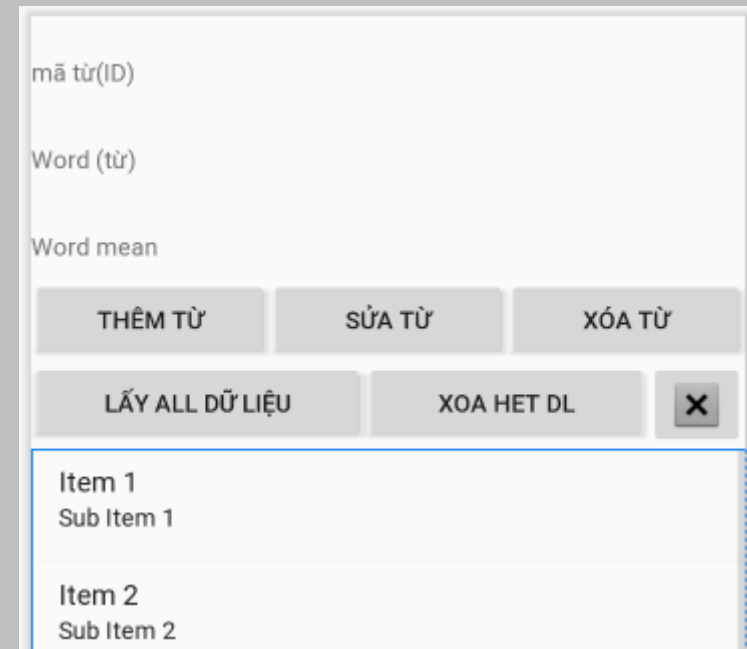
## ***PHƯƠNG THỨC CỦA LỚP SQLITEDATABASE***

Phương thức	Mô tả
public void <b>onDowngrade</b> (SQLiteDatabase db, int oldVersion, int newVersion)	gọi khi cơ sở dữ liệu cần phải được hạ cấp.
<b>void execSQL</b> (String sql)	thực hiện truy vấn sql query
<b>long Cursor cursor = db.rawQuery</b> (sql, null);	thực hiện truy vấn sql query lấy kết quả trả về



## DEMO SQLITE

- Giả sử 1 từ được biểu diễn thông qua từ và nghĩa của từ.
- Tạo bảng Word mô tả các từ: id tự tăng là khóa, từ (word) và mã của từ (mean)
- Tạo lập giao diện như hình vẽ thực hiện
  - Thêm từ
  - Sửa từ
  - Xóa từ
  - Lấy tất cả từ
  - Xóa hết từ trong bảng



The screenshot shows a web-based interface for managing a SQLite database. It features three input fields at the top: 'mã từ(ID)', 'Word (từ)', and 'Word mean'. Below these fields are three buttons: 'THÊM TỪ', 'SỬA TỪ', and 'XÓA TỪ'. Further down are two more buttons: 'LẤY ALL DỮ LIỆU' and 'XOA HET DL', along with a close button 'X'. At the bottom, there is a table displaying data with two rows: 'Item 1' with 'Sub Item 1' and 'Item 2' with 'Sub Item 2'.

mã từ(ID)	Word (từ)	Word mean

THÊM TỪ   SỬA TỪ   XÓA TỪ

LẤY ALL DỮ LIỆU   XOA HET DL   X

Item 1	Sub Item 1
Item 2	Sub Item 2

## DEMO SQLITE- TẠO CSDL

```
public class MyDatabaseHelper extends SQLiteOpenHelper {  
    private static final String TAG = "MyDatabaseHelper";  
    private static final String DATABASE_NAME = "DICTIONARY_DB";  
    private static final int DATABASE_VERSION = 1;  
    private static final String TABLE_WORD = "WORD";  
    private static final String ID_COLUMN = "id";  
    private static final String WORD_COLUMN = "word";  
    private static final String MEAN_COLUMN = "mean";  
    private static final String CREATE_WORD_TABLE_SQL =  
        "CREATE TABLE IF NOT EXISTS " + TABLE_WORD + " (" +  
            ID_COLUMN + " INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +  
            WORD_COLUMN + " TEXT NOT NULL," +  
            MEAN_COLUMN + " TEXT NOT NULL" +  
            ");";
```



## DEMO SQLITE- TẠO CSDL

```
private MyDatabaseHelper(Context context) {  
    super(context, DATABASE_NAME, null, DATABASE_VERSION);  
}  
@Override  
public void onCreate(SQLiteDatabase db) {  
    Log.e(TAG, "onCreate: ");  
    try{  
        // db.execSQL("DROP TABLE IF EXISTS " + TABLE_WORD);  
        db.execSQL(CREATE_WORD_TABLE_SQL);}  
    catch (Exception e){  
        Log.e(TAG,e.toString());  
    } }  
@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    Log.e(TAG, "onUpgrade: ");  
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_WORD);  
    onCreate(db);  
}
```



## SQLITE- INSERT

Phương thức	Mô tả
<code>long insert (</code> <code>String table,</code> <code>String nullColumnHack,</code> <code>ContentValues values)</code>	<p>chèn một bản ghi trên các cơ sở dữ liệu.</p> <p>Tham số 1: Bảng chỉ định tên bảng,</p> <p>Tham số 2: nullColumnHack không cho phép các giá trị null. Nếu số thứ hai là null, android sẽ lưu trữ các giá trị null nếu giá trị là rỗng.</p> <p>Tham số 3: xác định các giá trị được lưu trữ.</p>



## DEMO SQLITE- INSERT

```
public boolean insertWord(Word word) {  
    /**  
    Bảng chỉ định tên bảng,  
    * tham số thứ hai là null, android sẽ lưu trữ các giá trị null nếu giá trị  
    * là trống rỗng. Đối số thứ ba xác định các giá trị được lưu trữ.  
    */  
    SQLiteDatabase db = getWritableDatabase();  
    ContentValues values = new ContentValues();  
    values.put(WORD_COLUMN, word.getWord());  
    values.put(MEAN_COLUMN, word.getMean());  
    long rowId = db.insert(TABLE_WORD, null, values);  
    db.close();  
    if (rowId != -1)  
        return true;  
    return false;  
}
```

## SQLITE- UPDATE

Phương thức	Mô tả
<code>int update (String table, ContentValues values, String whereClause, String[] whereArgs)</code>	cập nhật một dòng.

- Đối số 1 là tên bảng
- Đối số 2 là đối tượng muốn chỉnh sửa (với giá trị mới)
- Đối số 3 là tập các điều kiện lọc (dùng dấu chấm hỏi ? để tạo điều kiện lọc)
- Đối số 4 là tập các giá trị của điều kiện lọc (lấy theo đúng thứ tự)
- Hàm trả về số dòng bị ảnh hưởng.



## DEMO SQLITE- UPDATE

```
public int updateWord(Word word) {  
    SQLiteDatabase db = getWritableDatabase();  
    ContentValues values = new ContentValues();  
    values.put(WORD_COLUMN, word.getWord());  
    values.put(MEAN_COLUMN, word.getMean());  
    int rowEffect = db.update(TABLE_WORD, values, ID_COLUMN + " = ?",  
        new String[]{String.valueOf(word.getId())});  
    db.close();  
    return rowEffect;  
}
```



## SQLITE- DELETE

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

- - Đối số 1 là tên bảng
- - Đối số 2 là tập điều kiện lọc (dùng ? để tạo)
- - Đối số 3 là tập các giá trị của điều kiện lọc
- - Hàm trả về số dòng bị ảnh hưởng.
- - Muốn xóa toàn bộ dữ liệu trong bảng thì ta truyền null vào 2 đối số cuối:
- **int rowEffect = db.delete(TABLE\_WORD, null,null);**

## DEMO SQLITE- DELETE

```
public int deleteAllWord() {
    SQLiteDatabase db = getReadableDatabase();
    int rowEffect = db.delete(TABLE_WORD, null, null);
    db.close();
    return rowEffect;
}

public int deleteWord(Word word) {
    SQLiteDatabase db = getReadableDatabase();
    int rowEffect = db.delete(TABLE_WORD, ID_COLUMN + " = ?",
                             new String[]{String.valueOf(word.getId())});
    db.close();
    return rowEffect;
}
```



## SQLITE- QUERY/RAWQUERY

Phương thức	Mô tả
<code>Cursor query</code> (String <code>table</code> , String[] <code>columns</code> , String <code>selection</code> , String[] <code>selectionArgs</code> , String <code>groupBy</code> , String <code>having</code> , String <code>orderBy</code> )	trả về một con trỏ trên resultset. <code>rawQuery()</code> : Chấp nhận lệnh truy vấn SQL trực tiếp. <code>query()</code> : cung cấp giao diện (interface) để viết lệnh truy vấn thuần với Java hơn.



## SQLITE- QUERY

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

columns	Ds các cột trả về. Nếu để trống sẽ lấy toàn bộ các cột trong bảng
selection	mệnh đề where
selectionArgs	Giá trị tương ứng với cột cần lấy giá trị cho mệnh đề where
groupBy	Mệnh đề group by
having	Mệnh đề having
orderBy	Mệnh đề order by.

## DEMO SQLITE- QUERY

```
public List<Word> getAllWord() {  
    SQLiteDatabase db = getReadableDatabase();  
    List<Word> words = new ArrayList<Word>();  
    String sql = "SELECT * FROM " + TABLE_WORD;  
    Cursor cursor = db.rawQuery(sql, null);  
    if (cursor != null && cursor.moveToFirst()) {  
        do {  
            words.add(new Word(cursor.getInt(cursor.getColumnIndex(ID_COLUMN)),  
                                cursor.getString(cursor.getColumnIndex(WORD_COLUMN)),  
                                cursor.getString(cursor.getColumnIndex(MEAN_COLUMN))));  
        } while (cursor.moveToNext());  
        cursor.close();  
    }  
    db.close();  
    return words;  
}
```



## TỔNG KẾT

Tài liệu tham khảo:

<https://developer.android.com/training>