**Tìm hiểu SQL lite**

SQLite là hệ thống cơ sở dữ liệu quan hệ nhỏ gọn, hoàn chỉnh, có thể cài đặt bên trong các trình ứng dụng khác. SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C. Trang chủ <http://www.sqlite.org/>

**Nét đặc trưng của SQL lite.**

* Transactions là nguyên tử, nhất quán, cô lập và bền vững (ACID) ngay cả sau khi sự cố hệ thống và mất điện.
* Không cần cài đặt trước khi sử dụng
* Tuân theo chuẩn SQL92(chỉ môt vài đặc điểm là không hỗ trợ). Những tính năng không được hỗ trợ được liệt kê bên dưới:

|  |  |
| --- | --- |
| **RIGHT and FULL OUTER JOIN** | LEFT OUTER JOIN is implemented, but not RIGHT OUTER JOIN or FULL OUTER JOIN. |
| **Complete ALTER TABLE support** | Only the RENAME TABLE and ADD COLUMN variants of the ALTER TABLE command are supported. Other kinds of ALTER TABLE operations such as DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT, and so forth are omitted. |
| **Complete trigger support** | FOR EACH ROW triggers are supported but not FOR EACH STATEMENT triggers. |
| **Writing to VIEWs** | VIEWs in SQLite are read-only. You may not execute a DELETE, INSERT, or UPDATE statement on a view. But you can create a trigger that fires on an attempt to DELETE, INSERT, or UPDATE a view and do what you need in the body of the trigger. |
| **GRANT and REVOKE** | Since SQLite reads and writes an ordinary disk file, the only access permissions that can be applied are the normal file access permissions of the underlying operating system. The GRANT and REVOKE commands commonly found on client/server RDBMSes are not implemented because they would be meaningless for an embedded database engine. |

* Toàn bộ CSDL được lưu trên một file duy nhất. Định dạng file của SQLite là cross-platform. Ta có thể sao chép , sử dụng dữ liệu ở bất cứ hệ điều hành nào.
* Hỗ trợ CSDL cỡ terabyte, những xâu cỡ gigabyte.
* Kích thước ngọn nhẹ.
* Thực hiện hầu hết các thao tác nhanh hơn các hệ thống cơ sở dữ liệu khách/chủ
* Đơn giản, dễ dàng sử dụng các API
* Được viêt bằng ANSI-C. SQLite được phát triển các binding (gói kèm) các ngôn ngữ thông dụng: C++, PHP, Perl, Java, Python, Ruby, Tcl...
* Có 1 file mã nguồn ANSI-C, dễ dàng cho vào các project khác
* Khép kín: không phụ thuộc vào bên ngoài
* Cross-platform: Unix (Linux and Mac OS X), OS/2, và Windows (Win32 và WinCE) đều hỗ trợ
* Mã nguồn mở
* Đi kèm với một giao diện dòng (CLI)ệnh độc lập, để quản lý CSDL.

**Đề nghị sử dụng SQLite**

* *Application File Format*: Thay vì sử dụng fopen() để ghi định dạng XML hay một định dạng độc quyền nào khác vào đĩa, hãy sử dụng SQLite để thay thế. Sẽ dễ dàng để thao tác với dữ liệu
* *Database For Gadgets*: SQLite là database engine phổ biến trong cellphones, PDA, MP3 player…
* *Stand-in For An Enterprise RDBMS*: SQLite thường được sử dụng như à sự thay thế cho một Enterprise RDBMS cho mục đích trình diễn hay thử nghiệm

**SQLite được sử dụng bởi:** Adobe, Airbus, Apple(Mac OS X, Safari, Iphone..), Mozilla( firefox, thunderbird), Google(Android, Chrome..), PHP, …

**C-language interface to SQLite.**

**1.Core Object and interfaces**

Có nhiều đối tượng nhưng 3 đối tượng quan trọng là : đối tượng database connection *sqlite3,* đối tượng prepared statement *sqlite\_stmt,*  và kiểu dữ liệu số nguyên 64 bit *sqlite\_int64.*

Nhiệm vụ chủ yếu của một SQL databse engine là thực hiệ các câu lệnh SQL. Với mục đích này thì t chỉ cần biết về 2 đối tượng:

* + Đối tượng database connection: sqlite3
  + Đối tượng prepared statement: sqlite3\_stmt

Đối tượng database connection và prepared statement được điều khiển bởi một danh sách C/C++ interface:

* sqlite3\_open()
* sqlite3\_prepare()
* sqlite3\_step()
* sqlite3\_column()
* sqlite3\_finalize()
* sqlite3\_close()

Với 6 hàm và 2 đối tượng ở trên là các chức năng cốt lõi của SQLite. Hiểu chúng ta có một nền tảng tốt để sử dụng SQLite.

Mô tả core interfaces:

|  |  |
| --- | --- |
| sqlite3\_open() | Là thủ tục mở một file SQLite database và trả về một đối tượng *database connection* . Nó thường là SQLite API được ứng dụng gọi đầu tiên that và là điều kiện tiên quyết đối với hầu hết SQLite APIs. Nhiều SQLite interfaces yêu cầu một con trỏ tới một đối tượng *database connection* như là tham số đầu tiên and và có thể được dùng như là một phương thức trên đối tượng *database connection*. Thủ tục này là constructor cho đối tượng *database connection*. |
| sqlite3\_prepare() | Là thủ tục chuyển văn bản SQL vào một đố tượng [prepared statement](http://www.sqlite.org/c3ref/stmt.html) và trả về con trỏ tới đối tượng đó. Nó yêu cầu một con trỏ *database connection* được tạo ra bởi lời gọi *sqlite3\_open()* trước đó và câu lệnh SQL. Nó không đánh giá (evaluate) câu lệnh SQL. Nó chỉ đơn thuần chuẩn bị câu lệnh SQL cho sự đánh giá. |
| sqlite3\_step() | Thủ tục được dùng để đánh giá (evaluate) một *prepared statement*. Câu lệnh được đánh giá tới hàng đầu tiên của kết quả. Để tiến tới hàng thứ 2 của kết quả gọi *sqlite\_step()* một lần nữa. Tiếp tục cho tới khi câu lệnh đc hoàn tất. Câu lệnh không trả về kết quả( vd câu lệnh INSERT, UPDATE hoặc DELETE) hoàn tất trong một lần gọi *sqlite\_step().* |
| sqlite3\_column() | Thủ tục trả về một cột đơn từ hàng hiện tại của kết tập quả(). Mỗi lần *sqlite3\_step()* dừng với một tập kết quả mới, thủ tục này có thể gọi nhiều lần để tìm giá trị của các cột trong hàng. Với mỗi kiểu dữ liệu ta có cá hàm khác nhau. Chúng đều trả về kích thước của kết quả (nếu nó là string hoặc BLOB) và số cột trong tập kết quả.   * *sqlite3\_column\_blob()* * *sqlite3\_column\_bytes()* * *sqlite3\_column\_bytes16()* * *sqlite3\_column\_count()* * *sqlite3\_column\_double()* * *sqlite3\_column\_int()* * [*sqlite3\_column\_int64()*](http://www.sqlite.org/c3ref/column_blob.html) * *sqlite3\_column\_text()* * *sqlite3\_column\_text16()* * *sqlite3\_column\_type()* * *sqlite3\_column\_value()* |
| sqlite3\_finalize() | Là thủ tục hủy một *prepared statement* được tạo bởi lời gọi *sqlite3\_prepare()* trước đó*.* Mỗi *prepared statement* phải được hủy bằng cách gọi thủ tục này để tránh memory leaks. |
| sqlite3\_close() | Thủ tục đóng một *database connection* được mở bởi lời gọi *sqlite3\_open()* trước đó. Tất cả *prepared statement* liên kết với kết nối phải được kết thúc(*finalized*) trước khi đóng kết nối. |

**2.Cách sử dụng Core Object and interfaces**

Để sử dụng SQLite cần các bước:

* *sqlite3\_open():* Để tạo một database connection
* Để dùng câu lệnh SQL cần thực hiện 4 bước sau:
  + Sử dụng *sqlite3\_prepare():*Tạo một *prepare stament*
  + Gọi *sqlite3\_step():*để đánh giá *prepare stament* một hoặc nhiều lần.
  + Đối với mỗi truy vấn ta lấy kết quả bằng *sqlite3\_column()* giữa hai lần gọi *sqlite3\_step()*
  + Hủy *prepare stament* bằng cách sử dụng hàm *sqlite3\_finalize().*
* *splite3\_close():* Hủy database connection

**Các Wrapper tiện lợi quanh các thủ tục lõi.**

*sqlite3\_exec()* là một Wrapper tiện lợi, nó thực hiện cả 4 bước ở trên chỉ với một lời gọi hàm. Một callback function được truyền vào *sqlite3\_exec()* để xử lý các hàng của tập kết quả

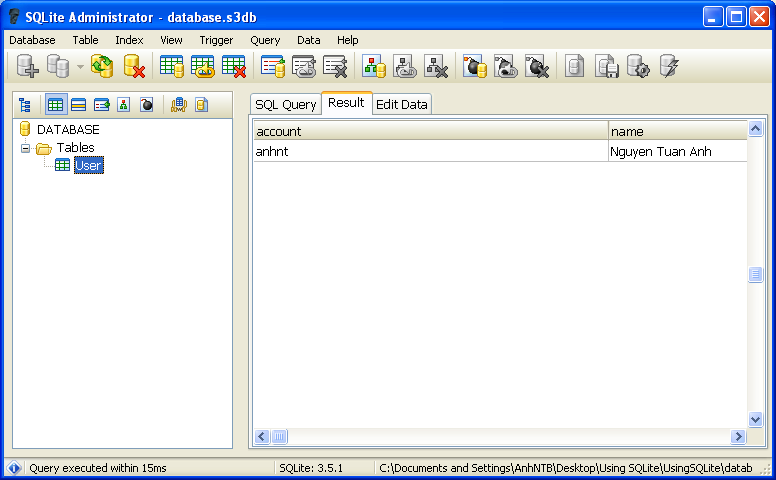
*sqlite3\_get\_table()* cũng là một Wrapper tiện lợi khác, nó thực hiện cả 4 bước ở trên. *sqlite3\_get\_table()* khác *sqlite3\_exec()* ở chỗ nó lưu kết quả của truy vấn ở heap memory thay vì gọi callback fucntion.

**Kiểu dữ liệu trong SQLite v3**

|  |  |  |
| --- | --- | --- |
| **Example Typenames From The CREATE TABLE Statement or CAST Expression** | **Resulting Affinity** | **Rule Used To Determine Affinity** |
| INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8 | INTEGER | 1 |
| CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB | TEXT | 2 |
| BLOB *no datatype specified* | NONE | 3 |
| REAL DOUBLE DOUBLE PRECISION FLOAT | REAL | 4 |
| NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME | NUMERIC | 5 |

**Bảo mật CSDL dùng SQLite**

**Chương trình đồ họa thao tác CSDL :sqliteadmin**

****

**Câu lệnh SQL**

|  |  |
| --- | --- |
| **Câu lệnh SQL** | **Câu lệnh thay thế** |
| SELECT TOP 2 \* FROM table; | SELECT \* FROM table LIMIT 2; |
| DELETE \* FROM table; | DELETE FROM table; |
|  |  |
|  |  |