

SQLite3 C APIs

核心对象和接口

数据库引擎的核心任务就是执行SQL语句，为完成该目的，站在开发者的立场上，必须首先理解两个对象

- 数据库连接对象(The database connection object): `sqlite3`
- 预编译的语句对象(The prepared statement object): `sqlite3_stmt`

由于`sqlite3_exec`或`sqlite3_get_table`接口的方便的封装，预编译的语句对象已经不在是必须要求的。但是，理解预编译的语句对完整的理解SQLite还是有帮助的

数据库连接和预编译语句对象由如下接口控制：

- `sqlite3_open()`
- `sqlite3_prepare_v2()`
- `sqlite3_step()`
- `sqlite3_column()`
- `sqlite3_finalize()`
- `sqlite3_close()`

- `sqlite3_open()`

创建数据库连接对象的方法

- `sqlite3_prepare_v2()`

该函数将**SQL**语句转成预编译的语句对象

- `sqlite3_step()`

执行预编译的语句，每次处理一行，不需要返回值的语句（如**INSERT**、**UPDATE**、**DELETE**）只需要执行该函数执行即可

- `sqlite3_column()`

运行该函数每次返回 `sqlite3_step()` 执行结果中的一列，该函数在这里只是占位，实际使用中根据不同的数据类型，使用如下相应的函数

- `sqlite3_column_blob()`
- `sqlite3_column_bytes()`

- `sqlite3_column_bytes16()`
 - `sqlite3_column_count()`
 - `sqlite3_column_double()`
 - `sqlite3_column_int()`
 - `sqlite3_column_int64()`
 - `sqlite3_column_text()`
 - `sqlite3_column_text16()`
 - `sqlite3_column_type()`
 - `sqlite3_column_value()`
- `sqlite3_finalize()`

该函数销毁由 `sqlite3_prepare_v2()` 函数创建的预编译 语句对象

- `sqlite_close()`

该函数关闭数据库连接(即销毁数据库连接对象)

执行SQL语句，应用程序需要遵守如下几步：

1. 通过 `sqlite3_prepare()` 创建预编译语句
2. 调用 `sqlite3_step()` 一次或多次来运行预编译的语句
3. 对于查询操作，在两次 `sqlite3_step()` 之间调用 `sqlite3_column()` 来提取结果
4. 调用 `sqlite3_finalize()` 来销毁预编译语句

便利的封装

`sqlite3_exec()` 和 `sqlite3_get_table()` 这两个接口是对以上4步的方便的封装，不同的是 `sqlite3_exec()` 是通过传入的回调函数来处理每一行的结果，而 `sqlite3_get_table()` 没有回调，并且将查询的结果存储在对内存上

绑定参数和重用预编译语句

- `sqlite3_reset()` // 重用预编译语句
- `sqlite3_bind()` // 绑定参数

在SQLite中，参数的使用可以按如下任一格式

```
?  
?NNN  
:AAA  
$AAA  
@AAA
```

在上面的例子中，*NNN*是一个整数值，*AAA*是一个标示符

附： SQLite命令行Shell [SQLite命令行Shell](#)