SQLite 接口

1. C/C++

1.1 安装

SQLite 安装章节了解安装过程。

1.2 C/C++ 接口 API

以下是重要的 C&C++ / SQLite 接口程序，可以满足您在 C/C++ 程序中使用 SQLite 数据库的需求。如果您需要了解更多细节，请查看 SQLite 官方文档。

序号	API & 描述
1	**sqlite3_open(const char *filename, sqlite3 **ppDb)** 该例程打开一个指向 SQLite 数据库文件的连接，返回一个用于其他 SQLite 程序的数据库连接对象。 如果 *filename* 参数是 NULL 或 ':memory:'，那么 sqlite3_open() 将会在 RAM 中创建一个内存数据库，这只会在 session 的有效时间内持续。 如果文件名 filename 不为 NULL，那么 sqlite3_open() 将使用这个参数值尝试打开数据库文件。如果该名称的文件不存在，sqlite3_open() 将创建一个新的命名为该名称的数据库文件并打开。
2	**sqlite3_exec(sqlite3*, const char *sql, sqlite_callback, void *data, char **errmsg)** 该例程提供了一个执行 SQL 命令的快捷方式，SQL 命令由 sql 参数提供，可以由多个 SQL 命令组成。 在这里，第一个参数 *sqlite3* 是打开的数据库对象，*sqlite_callback* 是一个回调，*data* 作为其第一个参数，errmsg 将被返回用来获取程序生成的任何错误。 sqlite3_exec() 程序解析并执行由 **sql** 参数所给的每个命令，直到字符串结束或者遇到错误为止。
3	**sqlite3_close(sqlite3*)** 该例程关闭之前调用 sqlite3_open() 打开的数据库连接。所有与连接相关的语句都应在连接关闭之前完成。 如果还有查询没有完成，sqlite3_close() 将返回 SQLITE_BUSY 禁止关闭的错误消息。

1.3 连接数据库

下面的 C 代码段显示了如何连接到一个现有的数据库。如果数据库不存在，那么它就会被创建，最后将返回一个数据库对象。

```
1 #include <stdio.h>
2 #include <sqlite3.h>
```

```
3
4  int main(int argc, char* argv[])
5  {
6      sqlite3 *db;
7      char *zErrMsg = 0;
8      int rc;
9
10     rc = sqlite3_open("test.db", &db);
11
12     if( rc ){
13       fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
14       exit(0);
15     }else{
16       fprintf(stderr, "Opened database successfully\n");
17     }
18     sqlite3_close(db);
19 }
```

现在，让我们来编译和运行上面的程序，在当前目录中创建我们的数据库 **test.db**。您可以根据需要改变路径。

```
1 $gcc test.c -l sqlite3
2 $./a.out
3 Opened database successfully
```

如果要使用 C++ 源代码，可以按照下列所示编译代码：

```
1 $g++ test.c -l sqlite3
```

在这里，把我们的程序链接上 sqlite3 库，以便向 C 程序提供必要的函数。这将在您的目录下创建一个数据库文件 test.db，您将得到如下结果：

```
1 -rwxr-xr-x. 1 root root 7383 May  8 02:06 a.out
2 -rw-r--r--. 1 root root  323 May  8 02:05 test.c
3 -rw-r--r--. 1 root root    0 May  8 02:06 test.db
```

1.4 创建表

下面的 C 代码段将用于在先前创建的数据库中创建一个表：

```c
#include <stdio.h>
#include <stdlib.h>
#include <sqlite3.h>

static int callback(void *NotUsed, int argc, char **argv, char **azColName){
    int i;
    for(i=0; i<argc; i++){
      printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
    }
    printf("\n");
    return 0;
}

int main(int argc, char* argc[])
{
    sqlite3 *db;
    char *zErrMsg = 0;
    int rc;
    char *sql;

    // Open database
    rc = sqlite3_open("test.db", &db);
    if( rc ){
      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
      exit(0);
    }else{
      fprintf(stdout, "Opened database successfully\n");
    }

    // Create SQL statement
    sql = "CREATE TABLE COMPANY("  \
         "ID INT PRIMARY KEY     NOT NULL," \
         "NAME           TEXT    NOT NULL," \
         "AGE            INT     NOT NULL," \
         "ADDRESS        CHAR(50)," \
         "SALARY         REAL );";

    // Execute SQL statement
    rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
    if( rc != SQLITE_OK ){
    fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
    }else{
      fprintf(stdout, "Table created successfully\n");
    }
    sqlite3_close(db);
    return 0;
```

```
48 }
```

上述程序编译和执行时，它会在 test.db 文件中创建 COMPANY 表，最终文件列表如下所示：

```
1 -rwxr-xr-x. 1 root root 9567 May  8 02:31 a.out
2 -rw-r--r--. 1 root root 1207 May  8 02:31 test.c
3 -rw-r--r--. 1 root root 3072 May  8 02:31 test.db
```

1.5 INSERT 操作

1.6 SELECT 操作

在我们开始讲解获取记录的实例之前，让我们先了解下回调函数的一些细节，这将在我们的实例使用到。这个回调提供了一个从 SELECT 语句获得结果的方式。它声明如下：

```
1 typedef int (*sqlite3_callback)(
2 void*,    /* Data provided in the 4th argument of sqlite3_exec() */
3 int,      /* The number of columns in row */
4 char**,   /* An array of strings representing fields in the row */
5 char**    /* An array of strings representing column names */
6 );
7
```

如果上面的回调在 sqlite_exec() 程序中作为第三个参数，那么 SQLite 将为 SQL 参数内执行的每个 SELECT 语句中处理的每个记录调用这个回调函数。

下面的 C 代码段显示了如何从前面创建的 COMPANY 表中获取并显示记录：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sqlite3.h>
4
5  static int callback(void *data, int argc, char **argv, char **azColName){
6     int i;
7     fprintf(stderr, "%s: ", (const char*)data);
8     for(i=0; i<argc; i++){
9        printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
10    }
11    printf("\n");
```

```c
   return 0;
}

int main(int argc, char* argv[])
{
   sqlite3 *db;
   char *zErrMsg = 0;
   int rc;
   char *sql;
   const char* data = "Callback function called";

   /* Open database */
   rc = sqlite3_open("test.db", &db);
   if( rc ){
      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
      exit(0);
   }else{
      fprintf(stderr, "Opened database successfully\n");
   }

   /* Create SQL statement */
   sql = "SELECT * from COMPANY";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
   if( rc != SQLITE_OK ){
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   }else{
      fprintf(stdout, "Operation done successfully\n");
   }
   sqlite3_close(db);
   return 0;
}
```

上述程序编译和执行时，它会产生以下结果：

```
Opened database successfully
Callback function called: ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 20000.0

Callback function called: ID = 2
```

```
NAME = Allen
AGE = 25
ADDRESS = Texas
SALARY = 15000.0

Callback function called: ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0

Callback function called: ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0

Operation done successfully
```

1.7 UPDATE 操作

1.8 DELETE 操作

1.9 注意

因为 SQLite 经常被用于嵌入式，在嵌入式环境中，本身数据量可能不会很大、查询效率比较高，业务逻辑使用上偏向于同步接口。

所以建议加上同步接口的说明和示例。

```c
SQLITE_API int sqlite3_get_table(
  sqlite3 *db,          /* An open database */
  const char *zSql,     /* SQL to be evaluated */
  char ***pazResult,    /* Results of the query */
  int *pnRow,           /* Number of result rows written here */
  int *pnColumn,        /* Number of result columns written here */
  char **pzErrmsg       /* Error msg written here */
);
SQLITE_API void sqlite3_free_table(char **result);
```

2. Java

2.1 安装

在 Java 程序中使用 SQLite 之前，我们需要确保机器上已经有 SQLite JDBC Driver 驱动程序和 Java。可以查看 Java 教程了解如何在计算机上安装 Java。现在，我们来看看如何在机器上安装 SQLite JDBC 驱动程序。

- 本站提供 sqlite-jdbc 3.7.2 版本下载，最新 *sqlite-jdbc-(VERSION).jar* 版本可以访问 https://github.com/xerial/sqlite-jdbc/releases 下载。
- 在您的 class 路径中添加下载的 jar 文件 *sqlite-jdbc-(VERSION).jar*，或者在 -classpath 选项中使用它，这将在后面的实例中进行讲解。

在学习下面部分的知识之前，您必须对 Java JDBC 概念有初步了解。如果您还未了解相关知识，那么建议您可以先花半个小时学习下 JDBC 教程相关知识，这将有助于您学习接下来讲解的知识。

2.2 连接数据库

下面的 Java 程序显示了如何连接到一个现有的数据库。如果数据库不存在，那么它就会被创建，最后将返回一个数据库对象。

```
import java.sql.*;

public class SQLiteJDBC
{
  public static void main( String args[] )
  {
    Connection c = null;
    try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
    } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);
    }
    System.out.println("Opened database successfully");
  }
}
```

现在，让我们来编译和运行上面的程序，在当前目录中创建我们的数据库 **test.db**。您可以根据需要改变路径。我们假设当前路径下可用的 JDBC 驱动程序的版本是 *sqlite-jdbc-3.7.2.jar*。

```
$javac SQLiteJDBC.java
$java -classpath ".:sqlite-jdbc-3.7.2.jar" SQLiteJDBC
Open database successfully
```

2.3 创建表

下面的 Java 程序将用于在先前创建的数据库中创建一个表：

```
import java.sql.*;

public class SQLiteJDBC
{
  public static void main( String args[] )
  {
    Connection c = null;
    Statement stmt = null;
    try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
      System.out.println("Opened database successfully");

      stmt = c.createStatement();
      String sql = "CREATE TABLE COMPANY " +
                   "(ID INT PRIMARY KEY     NOT NULL," +
                   " NAME           TEXT    NOT NULL, " +
                   " AGE            INT     NOT NULL, " +
                   " ADDRESS        CHAR(50), " +
                   " SALARY         REAL)";
      stmt.executeUpdate(sql);
      stmt.close();
      c.close();
    } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);
    }
    System.out.println("Table created successfully");
  }
}
```

上述程序编译和执行时，它会在 **test.db** 中创建 COMPANY 表，最终文件列表如下所示：

```
-rw-r--r--. 1 root root 3201128 Jan 22 19:04 sqlite-jdbc-3.7.2.jar
```

```
-rw-r--r--. 1 root root    1506 May  8 05:43 SQLiteJDBC.class
-rw-r--r--. 1 root root     832 May  8 05:42 SQLiteJDBC.java
-rw-r--r--. 1 root root    3072 May  8 05:43 test.db
```

2.4 INSERT 操作

2.5 SELECT 操作

2.6 UPDATE 操作

2.7 DELETE 操作