Sqlite3 tutorial

https://www.sqlite.org/

The tutorial example is from

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

Sqlite3 tutorial and sample programs

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

SQLite

About Sitemap Documentation Download License News Support

Welcome

QLite is a software library that implements a self-contained, serverless, zero-onfiguration, transactional SQL database engine. SQLite is the most widely deployed atabase engine in the world. The source code for SQLite is in the public domain. More...

Sponsors

ngoing development and maintenance of SQLite is sponsored in part by SQLite onsortium members, including:

Sqlite3 tutorial

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

To run the sample codes from the tutorial, you need to do first:

Download sqlite3 (which is included here for linux or cs1.utdallas.edu

** If you use Mac, you need to download the Mac version of sqlite3

Create a test database test.db (by Unix/Linux command: touch test.d

Read the tutorials and try these programs from test0.c to test5.c

To compile the sample program (e.g., test0.c), for example, with c++ c

```
g++  test0.c –o test0 –l sqlite3
```

To have programs (test0.c to test5.c) and

sqlite3 files from www.sqlite.org: sqlite3 and shell.c

Sqlite3 – sample programs

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

test0.c – to connect & open sqlite3 database, and to close.

test1.c – to create a table

test2.c – to insert records

test3.c – to select records and print

test4.c – to update a record

test5.c – to delete a record

The name of the database file used here is: test.db

- You may create the file by touch command: touch test.db
- If you need to start all over again, delete test.db and then create it.

test0.c – to open and close DB

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
/ g++ test0.c -l sqlite3
#include <stdio.h>
#include <sqlite3.h>
int main(int argc, char* argv[])
   sqlite3 *db;
  char *zErrMsg = 0;
  int rc;
 rc = sqlite3_open(“test.db", &db);            // you may create test.db with touch
 if( rc ){      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
 }else{      fprintf(stderr, "Opened database successfully\n");
 }
 sqlite3_close(db);
```

test1.c – to create a table

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
#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;
```

test1.c – to create a table

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
nt main(int argc, char* argv[])

 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");
 }
```

test1.c – to create a table

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
/* 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 );" ;
```

test1.c – to create a table

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
* Execute SQL statement */
 rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
 if( rc != SQLITE_OK ){     // not OK
        fprintf(stderr, "SQL error: %s\n", zErrMsg);
        sqlite3_free(zErrMsg);
 }else{
        fprintf(stdout, "Table created successfully\n");
 }
 sqlite3_close(db);
 return 0;
```

test2.c – insert records

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
#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;
```

test2.c – insert records

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
nt main(int argc, char* argv[])

 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(stderr, "Opened database successfully\n");
 }
```

test2.c – insert records

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
* Create SQL statement */
 sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "  \
                    "VALUES (1, 'Paul', 32, 'California', 20000.00 ); " \
        "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "  \
                    "VALUES (2, 'Allen', 25, 'Texas', 15000.00 ); "       \
        "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)"   \
                    "VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );"   \
        "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)"   \
                    "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );";
```

test2.c – insert records

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
/* 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, "Records created successfully\n");
}
sqlite3_close(db);
return 0;
```

test3.c – select records & print

http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
#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;
```

test3.c – select records & print http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
nt main(int argc, char* argv[])

 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(stderr, "Opened database successfully\n");
 }
```

test3.c – select records & print http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
/* 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;
```

test4.c – update a record

ttp://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
include <stdio.h>
include <stdlib.h>
include <sqlite3.h>
tatic 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;
```

test4.c – update a record http://www.tutorialspoint.com/qlite/sqlite_c_cpp.htm

```
nt main(int argc, char* argv[])

 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(stderr, "Opened database successfully\n");
 }
```

test4.c – update a record http://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
/* Create merged SQL statement */
 sql = "UPDATE COMPANY set SALARY = 25000.00 where ID=1; " \
        "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;
```

test5.c – delete a record

ttp://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
include <stdio.h>
include <stdlib.h>
include <sqlite3.h>
tatic 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;
```

test5.c – delete a record

ttp://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
nt main(int argc, char* argv[])

 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(stderr, "Opened database successfully\n");
 }
```

test5.c – delete a record

ttp://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

```
/* Create merged SQL statement */
 sql = "DELETE from COMPANY where ID=2;  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;
```

Download sqlite3 and sample programs & Try yourself

cslinux1.utdallas.edu - PuTTY

```
{cslinux1:~/sqlite/cs3376} ls
shell.c                            test1    test3.c                    test5.c
sqlite3                            test1.c  test3 - select & printf.c  test.db
sqlite-shell-linux-x86-3081101.zip test2    test4
test0                              test2.c  test4.c
test0.c                            test3    test5
{cslinux1:~/sqlite/cs3376} ls -l
total 1220
-rw------- 1 rkm010300 cs 151981 Sep 24 20:10 shell.c
-rw------- 1 rkm010300 cs 623632 Sep 24 20:10 sqlite3
-rw------- 1 rkm010300 cs 354348 Sep 24 20:10 sqlite-shell-linux-x86-3081101.zip
-rw------- 1 rkm010300 cs   7775 Sep 26 16:42 test0
-rw------- 1 rkm010300 cs    393 Oct 13 11:12 test0.c
-rw------- 1 rkm010300 cs   9542 Sep 24 20:10 test1
-rw------- 1 rkm010300 cs   1237 Sep 24 20:10 test1.c
-rw------- 1 rkm010300 cs   9542 Sep 24 20:10 test2
-rw------- 1 rkm010300 cs   1494 Sep 24 20:10 test2.c
-rw------- 1 rkm010300 cs   9542 Sep 24 20:17 test3
-rw------- 1 rkm010300 cs   1128 Sep 24 20:15 test3.c
-rw------- 1 rkm010300 cs   1128 Sep 24 20:15 test3 - select & printf.c
-rw------- 1 rkm010300 cs   9542 Sep 24 20:19 test4
-rw------- 1 rkm010300 cs   1199 Sep 24 20:18 test4.c
-rw------- 1 rkm010300 cs   9542 Sep 24 20:20 test5
-rw------- 1 rkm010300 cs   1182 Sep 24 20:20 test5.c
-rw------- 1 rkm010300 cs      0 Sep 26 16:44 test.db
{cslinux1:~/sqlite/cs3376}
```

You may delete test.db and create a new one

```
m test.db
        rm: remove regular empty file `test.db'? y
ouch test.db
```

Compile the sample programs

```
cslinux1:~/sqlite/cs3376} c++ test1.c -o test1 -l sqlite3
est1.c: In function ‘int main(int, char**)’:
est1.c:36: warning: deprecated conversion from string constant to ‘char*’
cslinux1:~/sqlite/cs3376} g++ test1.c -o test1 -l sqlite3
est1.c: In function ‘int main(int, char**)’:
est1.c:36: warning: deprecated conversion from string constant to ‘char*’
cslinux1:~/sqlite/cs3376} c++ test2.c -o test2 -l sqlite3
est2.c: In function ‘int main(int, char**)’:
est2.c:38: warning: deprecated conversion from string constant to ‘char*’
```

test1-5 programs to run

/test0

Opened database successfully

/test1

Opened database successfully

Table created successfully

/test2

Opened database successfully

Records created successfully

test3 – select records and print

```
/test3
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
```

test4 – update a record

/test4

Opened database successfully

Callback function called: ID = 1

NAME = Paul

AGE = 32

ADDRESS = California

SALARY = 25000.0

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

test5 – delete record #2

/test5

```
Opened database successfully
Callback function called: ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 25000.0

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
```

Sqlite3 shell (from console)

```
> sqlite3 test.db
QLite version 3.6.20
nter ".help" for instructions
nter SQL statements terminated with a ";"

qlite> .help
backup ?DB? FILE      Backup DB (default "main") to FILE
bail ON|OFF           Stop after hitting an error.  Default OFF
databases             List names and files of attached databases
dump ?TABLE? ...      Dump the database in an SQL text format
                      If TABLE specified, only dump tables matching
```

Sqlite3 – a few important commands

```
qlite> .help
databases            List names and files of attached databases
exit                 Exit this program
help                 Show this message
output FILENAME      Send output to FILENAME
quit                 Exit this program
schema ?TABLE?       Show the CREATE statements
                     If TABLE specified, only show tables matching
                     LIKE pattern TABLE.
tables ?TABLE?       List names of tables
                     If TABLE specified, only list tables matching
                     LIKE pattern TABLE.
```

Sqlite3 examples

```
sqlite> .schema
CREATE TABLE COMPANY(ID INT PRIMARY KEY     NOT NULL,NAME
TEXT    NOT NULL,AGE            INT     NOT NULL,ADDRESS
CHAR(50),SALARY         REAL );
sqlite> .tables
COMPANY
sqlite> select * from company;
1|Paul|32|California|25000.0
3|Teddy|23|Norway|20000.0
4|Mark|25|Rich-Mond |65000.0
sqlite> .exit
```