Small. Fast. Reliable.
Choose any three.

Here is what you do to start experimenting with SQLite without having to do a lot of tedious reading and configuration:

Download The Code

- Get a copy of the prebuilt binaries for your machine, or get a copy of the sources and compile them yourself. Visit the download page for more information.

Create A New Database

- At a shell or DOS prompt, enter: "**sqlite3 test.db**". This will create a new database named "test.db". (You can use a different name if you like.)
- Enter SQL commands at the prompt to create and populate the new database.
- Additional documentation is available here.

Write Programs That Use SQLite

- Below is a simple TCL program that demonstrates how to use the TCL interface to SQLite. The program executes the SQL statements given as the second argument on the database defined by the first argument. The commands to watch for are the **sqlite3** command on line 7 which opens an SQLite database and creates a new object named "**db**" to access that database, the use of the eval method on the **db** object on line 8 to run SQL commands against the database, and the closing of the database connection on the last line of the script.

```
01  #!/usr/bin/tclsh
02  if {$argc!=2} {
03    puts stderr "Usage: %s DATABASE SQL-STATEMENT"
04    exit 1
05  }
06  package require sqlite3
07  sqlite3 db [lindex $argv 0]
08  db eval [lindex $argv 1] x {
09    foreach v $x(*) {
10      puts "$v = $x($v)"
11    }
12    puts ""
13  }
14  db close
```

- Below is a simple C program that demonstrates how to use the C/C++ interface to SQLite. The name of a database is given by the first argument and the second argument is one or more SQL statements to execute against the database. The function calls to pay attention to here are the call to sqlite3_open() on line 22 which opens the database, sqlite3_exec() on line 28 that executes SQL commands against the database, and sqlite3_close() on line 33 that closes the database connection.

  See also the Introduction To The SQLite C/C++ Interface for an introductory overview and roadmap to the dozens of SQLite interface functions.

```
01  #include <stdio.h>
02  #include <sqlite3.h>
03
04  static int callback(void *NotUsed, int argc, char **argv, char **azColName){
05    int i;
06    for(i=0; i<argc; i++){
07      printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
08    }
09    printf("\n");
10    return 0;
11  }
12
13  int main(int argc, char **argv){
14    sqlite3 *db;
15    char *zErrMsg = 0;
16    int rc;
17
18    if( argc!=3 ){
```

```
19      fprintf(stderr, "Usage: %s DATABASE SQL-STATEMENT\n", argv[0]);
20      return(1);
21    }
22    rc = sqlite3_open(argv[1], &db);
23    if( rc ){
24      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
25      sqlite3_close(db);
26      return(1);
27    }
28    rc = sqlite3_exec(db, argv[2], callback, 0, &zErrMsg);
29    if( rc!=SQLITE_OK ){
30      fprintf(stderr, "SQL error: %s\n", zErrMsg);
31      sqlite3_free(zErrMsg);
32    }
33    sqlite3_close(db);
34    return 0;
35  }
```

See the How To Compile SQLite document for instructions and hints on how to compile the program shown above.