Research and Design of Embedded Wireless Meal Ordering System Based on SQLite

Jihong Zhang, Xiaoquan Chen

Department of Computer Science, Transport Management Institute Ministry of Transport of the People's Republic of China, Beijing, China

Abstract

The paper describes features and internal architecture and developing method of SQLite. And then it gives a design and program of meal ordering system. The system realizes the information interaction among the users and embedded devices with SQLite as database system. The embedded database SQLite manages the data and achieves wireless communication by using Bluetooth. A system program based on Qt/Embedded and Linux drivers realizes the local management of environmental data.

Keywords: Embedded. SQLite. Database System.

1. Introduction

With the wide range of embedded system applications and user data processing and management needs continue to increase, a variety of intelligent devices and close integration of database technology has been paid much attention. In the near future will be ubiquitous embedded database. Looking at the current international and domestic application of the embedded database, embedded database applications currently on the market has entered a phase of accelerated development. Embedded database adopts Process driven, which by the program calls the corresponding API to implement data access. It is small, fast, efficient, reliable and portable which is very suitable for embedded data management environment.

2. SQLite Embedded Database

SQLite is an open source embeddable database engine written in C by D. Richard Hipp. It is entirely self-contained with no external dependencies.

Features of SQLite as follows:

© 2012 Published by Elsevier B.V. Selection and/or peer-review under responsibility of Garry Lee

Open access under CC BY-NC-ND license.

Keywords: Embedded. SQLite. Database System.
1) Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
2) Zero-configuration - no setup or administration needed.
3) A complete database is stored in a single cross-platform disk file.
4) Simple, easy to use API.
5) Small code footprint: less than 300KiB fully configured or less than 180KiB with optional features omitted.
6) Self-contained: no external dependencies.
7) Supports two terabyte-sized databases.
8) Sources are in the public domain. Use for any purpose.
9) Faster than popular client/server database engines for most common operations.

SQLite uses a modular design. SQLite mainly consists of three components: an SQL compiler, a core and a back end. Internal structure shows in figure 1.

SQLite uses a modular design. SQLite mainly consists of three components: an SQL compiler, a core and a back end. Internal structure shows in figure 1.

Fig. 1 Internal architecture of SQLite

Much of the public interface to the SQLite library is implemented by functions found in C library. SQL statement is effective through the interface into the SQL compiler. When a string containing SQL statements is to be executed, the interface passes that string to the tokenizer. The job of the tokenizer is to break the original string up into tokens and pass those tokens one by one to the parser. After the parser assembles tokens into complete SQL statements, it calls the code generator to produce virtual machine code that will do work that the SQL statements request. The program generated by the code generator which is executed by the virtual machine.

3. Developed Technology of SQLite Embedded Database

The SQLite version 3 API consists of approximately 80 functions. Only about eight functions, however, are needed to actually connect, process queries, and disconnect from a database. SQLite itself provides a C language API interface, making operation of the database is very simple, mainly for three API Function call.

We open a database with the sqlite3_open() functions, which have the following declaration:

```c
int sqlite3_open(
    const char *filename, /* Database filename (UTF-8) */
    sqlite3 **ppDb        /* OUT: SQLite db handle */
);
```
The sqlite3_exec() function provides a quick, easy way to execute SQL commands and is especially handy for commands that modify the database. This function has the following declaration:

```c
int sqlite3_exec(
    sqlite3*,                 /* An open database */
    const char *sql,           /* SQL to be executed */
    sqlite_callback,           /* Callback function */
    void *data,                /* 1st argument to callback function */
    char **errmsg            /* Error msg written here */
);      
```

You close a connection by using the sqlite3_close() function, which is declared as follows:

```c
int sqlite3_close(sqlite3*);  /* OUT: SQLite db handle */
```

This function closes the connection given by db. If the operation is successful, the function returns SQLITE_OK. If the db argument is not a valid connection pointer returned by sqlite3_open(), or if the connection pointer has been closed previously, sqlite3_close() will return SQLITE_ERROR. If there are prepared statements that have not been finalized, sqlite3_close() will return SQLITE_BUSY.

4. Mistakes Embedded Wireless Meal Ordering System

We develop a portable handheld wireless communication terminals based on an embedded processor. The system realizes wireless ordering capabilities with the corresponding software system. The completion of the project will greatly improve the efficiency of restaurant staff productivity, improved dining environment and bring huge economic benefits.

Embedded wireless meal ordering System is based on C/S architecture. Between front-end and back-end server make use of wireless communication, which combines wireless technology and handheld mobile communication technology in a computer terminal.

Front hand-held ordering devices may also be known as a client of the system for the restaurant's waiters. It can realize the guests to order now paperless operation. Background system platform for PC, also called the system's servers. Restaurant's administrators mainly use it. Administrators can add new food restaurants, modify the price of vegetables and query historical records to the system through system server. It is responsible for coordinating a variety of equipment and timely provides true, reliable data for the staff, managers.

Between the client and server communicate through the wireless devices. The system achieves wireless communication by using Bluetooth. We use the operating system Linux, the SQLite database on the realization of system. Meal ordering system architecture shows in figure 2.

![Fig. 2 Meal Ordering System Architecture](image-url)
In order to achieve data storage, query, and modification operations, Interface function is provided by embedded database SQLite which operates database of meal ordering system. Meal ordering system workflow chart shows in figure 3.

```cpp
int res = sqlite3_open(argv[1], &db);
if( res ){
    cout << "Can't open database: " << sqlite3_errmsg(db);
    sqlite3_close(db);
    return -1;
}

In Sqlite We create consume database. Under this database we create consumer account table according to the Consumer's name in program code. We develop the system with C++ API function. Creating consumer account carry out with defining crtAccount() function.

```cpp
//Create Consumer Account
int crtAccount()
{
    char** result;
    string sql;
    int nrows;
    int ncols;
    char* zErr;
    int res;
    cout << "Please enter the name of the account created: ";
    string sa;
    cin >> sa;
    sql = "create table " + sa + "(con_type text primary key,con_money int,con_time timespace);";
    res = sqlite3_exec(db, sql.c_str(),&result,&nrows,&ncols,&zErr);
    if(res!=SQLITE_OK)
    {
        cout << "Add " << sa << " Account Failed !" << endl;
        return -1;
    }
```
else
{
    cout<"Add "<<sa<" Account Success!"<<endl;
    for(int i=0;i<nrows;i++)
    {
        cout<i+1<".";
        for(int j=0;j<ncols;j++)
        {
            cout<result[(i+1)*ncols+j]<"t";
        }
        cout<endl;
    }
    cout<endl;
    sqlite3_free_table(result);
}

5. Conclusion and Outlook

The system completes the operation of selecting consumption records, inserting consumption records, deleting consumption records, updating consumption records, database file backup and recovery. After a lot of analysis and comparison, it is ideal for embedded database SQLite because of the characteristics of embedded system. The project was further development with SQLite. Proved, SQLite embedded system is well done to the database applications.

References