Research and Application of SQLite Embedded Database Based on ARM-Linux

BI Chun-yue
School of Computer Science and Information Technology
Zhejiang Wanli University
Ningbo, Zhejiang
P.R.China
http://www.zwu.edu.cn

Abstract: - The embedded database SQLite is widely applied in the data management of embedded environment such as mobile devices, industrial control, information appliance etc., for its advantages of stability and reliability, fast and high efficiency, portability and so on. This paper analyses the features, architecture and development technology of SQLite, gives a detailed porting process from SQLite to ARM-Linux platform, and discusses concrete application of SQLite in embedded system through a development case about the home gateway based on ARM-Linux.

Key-Words: - Embedded Database; SQLite; Porting; ARM-linux; Home Gateway

1 Introduction
With the development of information technology, Embedded database technology has gone from research fields into a wide range of application fields, the close combination between a variety of embedded equipments and embedded database technology has been taken seriously. The users are eager to do more effective management for the embedded products data, the market based on embedded database application has also entered an accelerated development phase.

In recent years, a few database products appear on the market continuously, such as IBM DB2, Oracle, Sybase, SQL Server, and so on. However, all these databases are restricted by capacity and power consumption of embedded system, and can’t meet the needs of users, especially when the scale of the mobile users become very large. At present, embedded database based on the Linux open-source platform, such as Mysql, Berkeley DB, SQLite, and so on, which provide easy-to-use interface API, promote the development of open-source software, and facilitate to access data and manage data. In addition, the new research and application in different fields emerges endlessly, especially in some novel fields like mobile equipment, Automatic test System and Communication System, etc., therefore, with the rapid development of embedded technology and wireless communication networks, the research on embedded database will become even more important.

Embedded database technology has become a hot research field, the major database vendors have launched their own products, although these products are embedded field oriented, they have their own characteristics on the specific technical specifications. SQLite has attracted the majority of developers for its lightweight, easy to port, without copyright restrictions.

2 Features, Architecture of SQLite and Development Technology

2.1 Features of SQLite
SQLite is an open-source embedded database, which is written in C language by D. Richard Hipp. Compared with the usual databases such as SQL Server, Oracle etc., SQLite is a lightweight database and it realizes a complete and embeddable database engine without additional components. it is especially suitable for embedded applications, and has many advantages. So SQLite has been favored by the R & D personnel from the start. The main features are as follows: [1][2][3]

(1) SQLite is an open-source embedded database, the library is self-contained and implemented in less than 30,000 lines of ANSI C, which is free to use for any purpose. Furthermore, its database format is binary-compatible among machines with different byte orders and scales up to 2 terabytes (241 bytes) in size.

(2) SQLite doesn’t need to be installed and configured, and provides easy-to-use API. It accesses the database directly by calling the API function and
can support advanced languages to access, it is very suitable for embedded database.

(3) SQLite can read and write the database files on the hard disk directly and needn’t an additional service process.

(4) It supports ACID and offers many supports for SQL92, it can also support multiple tables and indexes, transactions, views, triggers and a vast array of client interfaces and drivers. Database files can be freely shared among machines with different byte orders.

(5) SQLite is fast, efficient and scalable, it doesn’t rely on the operating system, which can be used in a variety of operating systems, such as uCLinux, uC/OS-II, Windows CE, and so on.

(6) It provides many supports for the API, and supports the major programming languages including C/C++, PHP, Perl, etc.

(7) SQLite has good reliability and notes, and has more than 90% of test coverage.

2.2 The Architecture of SQLite

SQLite takes modular design. It can be divided into eight primary subsystems: Interface, Tokenizer, Parser, Code Generator, Virtual Machine, B-Tree, Pager, OS Interface [2]. They constitute three parts: Core, SQL Compiler, and Backend, in addition, Accessories are also included in the structure. SQLite internal architecture is shown in Fig. 1.

![Architecture Of SQLite](image)

The most top-level interface for SQLite is a C language library, even if the use of different languages API, which is still executed with C library in the bottom. SQL statements received from the interface, which go into the high-performance SQL compiler, and be decomposed into various identifiers by tokenizer. The parser, called lemon Analyzer, which can identify the identifiers, and then re-compose the identifiers and call code generator to generate the virtual machine code, the virtual machine carry out the code, and finally complete the specified task of SQL statement.

Virtual Machine is the heart of the internal structure of SQLite, can also be called the Virtual Data Engine (VDBE), which is an engine designed to deal with library files. The VDBE performs not only all operations related to data manipulation but also is the broker through which information is passed between client and storage. VDBE is designed specifically for data-processing virtual machine, The instructions of VDBE instruction set can complete a special database operations, such as insert a record, query, transaction processing. [3] The VDBE comes into play after the SQL is parsed. The code generator takes the parse tree and translates it into a mini-program, which is made up of a series of instructions expressed in the VDBE's virtual machine language. One by one, the VDBE executes each instruction, which ends by fulfilling whatever request was specified in the SQL statement.

Backend includes three part: B-tree, pager and OS interface [4]. The task of B-Tree is ordering, databases are stored in the discs in forms of B-Tree, by using adjustable Pager, the fast query and storage of data can be acquired. In order to facilitate the porting, SQLite uses an abstract layer interface (OS interface) to connect with different operating systems, Due to different operating systems use different methods to lock files, the task of OS interface is to provide an abstract layer to shield out these differences for the other components of SQLite.

2.3 Development Technology of SQLite

The API functions of SQLite is very easy to use for the basic functions of the database operations, such as the establishment of the table, query, modify, insert, delete, sort, and so on. Three of them are the core API function[5], that is, sqlite_open (), sqlite_exec (), sqlite_close () which are used to execute SQL and acquire data. In addition, it is extensible, allowing the programmers to define custom functions and pass on in the form of callback.

Using the C API requires only three steps. Firstly, you call sqlite_open() to open a database and establish the SQLite engine, in which you provide the filename and access mode. Secondly, you implement a callback function, which SQLite calls for each record it retrieves from the database. Next, you call sqlite_exec(), execute the SQL statement and deal with the result. Finally, sqlite_close() function is
responsible for the closure of embedded database file and the release of SQLite engine. The following are the composition of three API functions.

- Sqlite_open()
  Sqlite *sqlite_open (const char *dbname, int mode, char **errmsg);

- Sqlite_close()
  Void sqlite_close (sqlite *db);

- Sqlite_exec()
  int sqlite_exec (sqlite * db,char* sql,int(*Callback)(void*,int,char **,char**),void *parg,char **errmsg);

This function includes five parameters: Database structure pointer, SQL statement string, the first pointer point to Callback functions, the first parameter pointer of Callback, the pointer to error string.

### 3 Porting of SQLite in the ARM-linux platform

In order to do further application development, SQLite must be ported to the corresponding microprocessor. Due to good features and architecture of the SQLite, the porting of a database will be possible. For the porting of SQLite, we can do cross-compile to the source code according to the different platform. This paper takes ARM-Linux as the bottom operating system, porting will Concern three tools: arm-linux-gcc, arm-linux-ar and arm-linux-ranlib. The following is a specific course of porting [5][6]:

1. Using "echo PATH" to see whether the cross-compiler tool arm-linux-gcc has been included in the PATH.

2. Download the latest source code package sqlite-3.5.4.tar.gz (it can be downloaded at http://www.sqlite.org/), then decompress the package with tar command. After completing the Decompression, we can see the source code and some attached files under the sqlite-3.5.4 directory.

3. In order to run the SQLite correctly on ARM7-linux, the relevant documents need to be modified. Copy Makefile/Linux-gcc in Root directory and rename it Makefile, and then modify the files correspondingly with vim.

   ```
   TOP = ../sqlite amended to read: TOP = ;
   TCC = gcc- 06 amended to read:TCC = arm-linux-gcc-06 ;
   AR = ar cr amended to read: AR = arm-linux-ar cr ;
   RANLIB = ranlib amended to read:
   ```

   ```
   RANLIB = arm-linux-ranlib ;
   MKSHLIB = gcc-shared amended to read:MKSHLIB = arm-linux-gcc-shared
   ```

   Notes the following lines.

   ```
   #TCL_FLAGS = -
   /home/drh/tcltk/8.4linux
   #LIBTCL =
   /home/drh/tcltk/8.4linux/libtcl8.4g.a-lm –ldl
   ```

4. Modify the file main.mk and open the file Makefile, delete tclsqlite.o from this line select.o table.o tclsqlite.o tokenize.o trigger.o, when compiling, the tcl language binding of SQLite will not be compiled. Save these files have been modified, and then use the commands make & & make install to generate what we want, i.e. sqlite3.h, libsqlite3.a.

5. Run SQLite on the ARM board. Copy sqlite to ARM board, and then have a test, for example, we can take a test procedure to test SQLite, if the test result is correct, this shows that SQLite3 has been ported in the ARM board successfully, and then do some application developments based on SQLite.

### 4 A Case about SQLite Application in the Home Gateway

In the remote monitoring system of information appliances, a large number of real-time data need to be acquisition and processed, the diversity of data storage and management need a background database to support. Because of the limited hardware resources, it is very difficult to play a role for the traditional database. Therefore, it is of great importance to develop a home gateway, and the embedded database has been a part of it.

#### 4.1 The Whole Design

In this paper, we construct an embedded home gateway based on S3C44B0X + uCLinux + SQLite + Boa, as shown in Fig.2.

![Fig.2 Structure of Embedded Home Gateway](image-url)
In the home gateway applications, calling the API function from the SQLite embedded database allows you to complete the connection and operation of database, users can access some scattered data to integrate, query, analysis and effective storage, in addition, information appliances data need to be managed and operated [7].

AS is shown in fig.2, the hardware platform is the 32 bit ARM microprocessor S3C44B0X, it takes uCLinux as operating system and offers bottom software support for home gateway, in addition, embedded web server Boa and SQLite are ported respectively to the ARM. Information appliances register to the home gateway by using IAIDL--Information Appliance Interface Definition Language, SQLite storages registration information, parameters as well as the status information of appliances.

When necessary, we can realize the remote monitoring and information retrieval to appliances by means of visiting embedded database. At the same time, it preserves its users’ information table and gives different users to retrieve, modifies the corresponding permissions of appliances information. Through visiting the home gateway embedded WEB server Boa, remote monitoring browser can call the corresponding CGI application to execute the operation of the objectives.

4.2 Generation of SQLite Data Sheet
SQLite contains two types of data sheets, user information table and information appliances table. User information table store user’s name and password, information appliances table store the state data of appliance running.

By calling API function provided by SQLite, these information is stored in the related data table of SQLite eventually, and then generate the corresponding database table, such as equipment type table, equipment list, equipment properties table, equipment interface table, events channel table and equipment associated table, and so on. SQLite internal structure is shown in Fig.3.

![Fig.3 Generation of SQLite data sheet](image)

4.3 Application Procedure Design Based on SQLite
Application development based on SQLite is how to use CGI to establish procedures, access, and update the SQL database [8]. This process can be divided into three steps to complete, i.e.

1. Establish the backend database of home gateway;
2. Make a Web page and sheets with HTML documents;
3. Edit CGI procedure includes decoding, function and output.

This paper explain mainly how to complete the intercommunication with database by using API functions provided by SQLite in the function part of CGI. At first, you call sqlite_open() to connect to a database, in which you provide the filename and access mode. Next, call sqlite_exec(), providing a string containing the SQL you want to execute, in the end, call sqlite_exec() to close the database. If you want to get the result, you must implement a callback function for each record from the database.

The following procedure is how to write the client registration information into the user table of database file IA_database through the CGI procedure in the home gateway.

```c
#include<stdio.h>
#include<string.h>
#include<sqlite3.h>
#define MAXQRY 300

void main()
{
char *errmsg;
int ret;
char qry[MAXQRY];
sqlite3 *
db=sqlite_open("IA_database",0777,
&errmsg);
if(db==0)
{
fprint(stderr,"Cound not open
database:%s
",errmsg);
sqlite3_freemem(errmsg);
exit(1);
}
sprintf(qry,"INSERT INTO
USERS(name,password)
VALUES('%s','%s')",name,password);
ret=sqlite3_exec(db,qry,NULL,NULL,
&errmsg);
if(ret!=SQLITE3_OK)
{
fprint(stderr,"SQLerr:%s
%s
",errmsg,qry);
}
```

This procedure is how to write the client registration information into the user table of database file IA_database through the CGI procedure in the home gateway.
else
{
    printf("%s %s was inseted\n", name, password);
}
sqlite3_closed(db);

CGI application will regard the standard output as a method of WEB Server to pass data. Generally speaking, CGI application will put the results of the implementation out to the standard output, WEB server reads the information and returns them to the client. If you want to put SQL query results out to remote users, CGI applications can use printf() function and put the query results to the standard output in the form of HTML, then, Home Gateway server will return to the dynamic page to the client, and realize the interaction among users, WEB server and database SQLite.

In the design process of Gateway, embedded SQLite database is the core, it is not only the important repository to collaborate with among information appliances to achieve but also the key of whether gateway to realize monitor control functions of Information appliances, which in turn will affect the user to access Embedded database through the browser.

5 Conclusion
Embedded database has become the focus of study currently, SQLite has small core and can adapt to the needs of embedded system, it is very convenient to construct an embedded database system. At present, SQLite has become the mainstream database of embedded system rapidly with its unique advantages. In this paper, we have a detailed analysis to the features and the architecture of SQLite, and then SQLite has been ported to ARM-Linux platform successfully; using the API offered by SQLite, we have realized the application of SQLite in the home gateway. With the rapid development of embedded systems, SQLite will be more widely used in the embedded field, such as the remote control, information appliances control, home medical equipment, etc.

References: