Research and Application of SQLite Embedded Database Technology

CHUNYUE BI
School of Computer Science and Information Technology
Zhejiang Wanli University
No.8, South Qian Hu Road 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., it has become the focus of the development of related areas. For its advantages of stability and reliability, fast and high efficiency, portability and so on, which occupies the unique advantages among many of the main embedded databases. This paper first describes the definition, basic characteristics, structure and the key technologies of embedded database, analyses the features, architecture and the main interface functions 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 operating system opens up a new development space for database technology according to the requirements of mobile database system. With the popularity of Intelligent Network Devices, Mobile Terminals, Information Appliances, 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. Because of the popularity of mobile terminals, the users improve continuously the real-time processing and management of mobile data, they 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 great many database products appear on the domestic 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 scales of the mobile users become very large. The traditional database is much slowly and it ties up too much computer memory in some real-time systems, while the embedded database shows excellent performance in related fields. At present, a large number of embedded databases have appeared in the international and domestic, they provide a lot of conveniences for the development of embedded devices. Domestic products include Xiao-Jin-Ling embedded database system KingbaseLite, OpenBASE Mini developed by Neusoft Group, etc.; international products include Sybase's Sybase iAnywhere, IBM's DB2 Satellite and DB2 Everywhere, Oracle's OracleLite, etc., in addition, embedded database based on the Linux open-source platform has also appeared in the market, 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 emerge 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.

Nowadays, 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 strongly the majority of developers for its advantages such as lightweight, easy to port, without copyright restrictions and so on.

2 The Definition and Characteristics, Structure and the Key Technologies of Embedded Database

2.1 The Definition and Characteristics of Embedded Database

In general, embedded database system can be defined from the architecture [1]. Embedded database system is a Database Management System that supports mobile computing or a particular computing mode, which is usually integrated with the operating system and specific application, and running in the intelligent embedded devices or mobile devices. Because the embedded database always combines with mobile computing, it is usually also known as embedded mobile database. Embedded database technology concerns several subject areas such as the database, real-time systems, distributed computing and mobile communications technology and so on, which has become a new direction of research and application about the database technology development. The structure of Embedded System gives the importance of embedded database, as is shown in Fig.1.

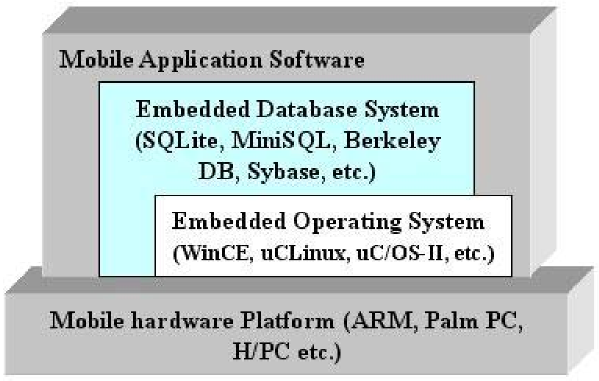
Mobile Application Software
Embedded Database System (SQLite, MiniSQL, Berkeley DB, Sybase, etc.)
Embedded Operating System (WinCE, uCLinux, uC/OS-II, etc.)
Mobile hardware Platform (ARM, Palm PC, H/PC etc.)

Fig.1 Structure of Embedded System

From the above diagram, we can see that, as a software middleware of embedded system, the embedded database will be bound to be restricted by the speed of embedded systems, resources, application and other factors. Embedded database system is essentially similar to the usual database system, which can be level, network, relational database, and be also object-oriented style. The differences between the embedded database and traditional database are as follows: embedded database is directly driven by procedure, i.e., call the corresponding API function to access data, while traditional database is driven by engine response approach. Since embedded database system and universal database system are not the same in the running environment, and many other applications, the embedded database can't be simply regard as a miniature of the universal database, it has its own characteristics [2][3][4].

(1) Occupy the less space and small memory resources

The disk space required in embedded database is very small, and we can limit the size and quantity of the data structure such as data tables, records, and other technologies to reduce the disk space for embedded database. The memory space of embedded devices is also very small, so embedded database must be able to run in the limited memory space.

(2) Reliability, manageability and security

The embedded database is generally used in the mobile environment, the requirements of embedded systems not only must be reliable, but also can run normally without artificial management.

(3) Interoperability and portability

In general, embedded database is designed and implemented for the specific development platform and operating system, in order to ensure communicate with other databases, in the development process, the interoperability among the databases is of great importance; at the same time, the operating system and hardware environment used in various embedded devices are different, the portability of embedded database must also be considered in design.

(4) Scalability

Due to its characteristics of embedded systems, embedded database must support the scalability, so that we can save disk space and improve efficiency.

2.2 The Architecture of Embedded database

Embedded database is similar to traditional database, and it consists of three levels, namely, internal level, conceptual level and the external level [5], as is shown in Fig.2.

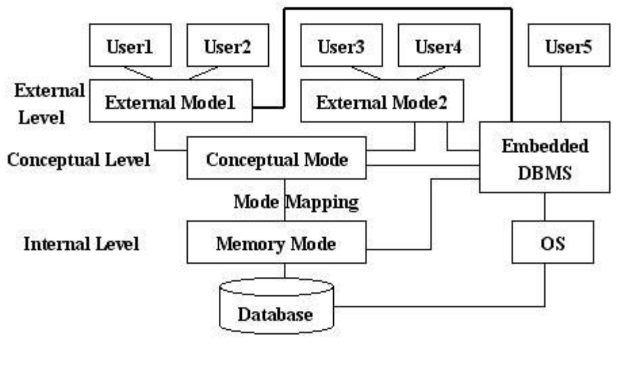

Fig.2 Architecture of Embedded database

The conceptual mode is the global logic description of DB and it is independent of all the others; The external mode is the interface between users and DBS and programmers only to store and manipulate data according to the framework of external mode and need not pay attention to conceptual mode; The internal mode is the description of DB in physic storage and it gives the definitions of all internal record type, index, file arrangements and the methods how to control the data. The conceptual mode provides a relatively steady middle part to restrict the other two parts (external and internal parts).

If the requirements of Real-Time are very high, we can directly deal with the physical files of database without through conceptual level, and it no longer carries out mapping from external mode to conceptual mode and from conceptual mode to memory mode.

Embedded DBMS is a software system which manages the data in the embedded database system, and which consists of three program modules, namely, the language compiler processing programs, system running control program and the service program. Embedded DBMS is the core components of the database system. All of its operations are executed through the Embedded DBMS. The descriptions of the three modules are given as follows [5]:

(1) Language compiler processing program, it includes language processing program at all levels, compiler processing program of database manipulation language and query language explanation program.

(2) The system running control program is the heart of Embedded DBMS running program, it is used to control and coordinate the activities of the program, mainly including concurrent control program, real-time scheduling program, transaction processing program, access and update program of database, storage management program and so on.

(3) Service-oriented program, including error recovery program, the work program log, database re-organization program and so on.

2.3 The Key Technologies of Embedded Database

In the design of embedded database, because of constraints of equipment resources, embedded database usually integrated with application system together, as the front-end, its data may be the subset of data sets in the back-end server. So, to design a good embedded database need to deal with various key technologies [6].

(1) Replication and Synchronization

When mapping with server database, embedded database usually takes some kind of data replication mode, so users can access to any data at all times. Because of data replication, a variety of necessary synchronization processes are required between various application front-ends and various back-end servers in the system.

(2) Backup recovery

The backup of embedded database is different from the usual database, it can’t be independent of the service, but in accordance with a simplified manner.

(3) Transaction processing

Transaction processing in embedded database system can be simplified in the front-end, but in the entire application system, it may need to be combined with the characteristics of mobile computing environment to control and deal with transactions.

(4) Security

Since the embedded devices have a high mobility, portability and non-fixed working conditions, which often bring potential insecurity factors, the database system is very strict with the control of access authority.

(5) Quick start-up of the system

The probability of fault in the embedded devices is higher than that of fixed host, so it is necessary to ensure that the system can carry out quick start-up through the system's hardware in the case of Software Error non-correction.

(6) Real-time processing

In many areas, the real-time performance is a basic requirement of embedded system. In order to avoid too long processing time, the requirement of real-time processing must be considered.

3 Features, Architecture and the Main Interface Functions of SQLite

3.1 Features and Data Structure of SQLite

3.1.1 Features of SQLite

SQLite is an open-source embedded database, which is written in C language by D. Richard Hipp. Compare 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 [7][8][9]:

(1) SQLite is an open-source embedded database, the library is self-contained and implemented in less than 30,000 line 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 in size. The open-source codes not only reduce production costs of products, more importantly, but also provide the most thorough means to resolve for the well maintenance and stable running of products.

(2) SQLite doesn't need to be installed and configured, doesn't need thread to start and stop, and doesn't need to create database or distribute user authority by administrator, in addition, SQLite can restore automatically after system collapse or loss of power. SQLite 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. A complete database corresponds to files on disk. The same database files can be used on different machines, but also can be freely shared among machines with different byte orders.

(4) No data type. The largest characteristics of SQLite lie in its no data type. This means that it can preserve any data to any column of any table that needs to be preserved, no matter what the data type declared is.

(5) 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.

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

(7) It provides many supports for the API, and supports the major programming languages including C/C++, PHP, Perl, etc. These programming languages communicate with the files of databases through the API.

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

3.1.2 Main Data Structure of SQLite

SQLite has several data structures to use in the design of program. They are as follows:

(1) SQLite is a data structure pointing to a database，in the design of program, sqlite * XXX is usually used to define a handle pointing to a database.

(2) Sqlite_stmt data structure, it includes VDBE byte codes compiled by SQL statements and all the necessary resources which execute the byte codes. In the design of program, sqlite_stmt *XXX is usually used to define a data structure pointing to some SQL statements.

(3) Sqlite_value. SQLite supports dynamic data type, many data types are entities of sqlite_value.

3.2 Analysis on the Architecture of SQLite

3.2.1 The Architecture of SQLite

The design of SQLite follows the hierarchical design thinking of software engineering, it takes modular design, to facilitate the maintenance upgrades, but also easy to port in different platform, it is essential to divide the whole database system into several different level modules to implement respectively. It can be divided into eight primary subsystems: Interface, Tokenizer, Parser, Code Generator, Virtual Machine, B-Tree, Pager and OS Interface [8]. 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.3.

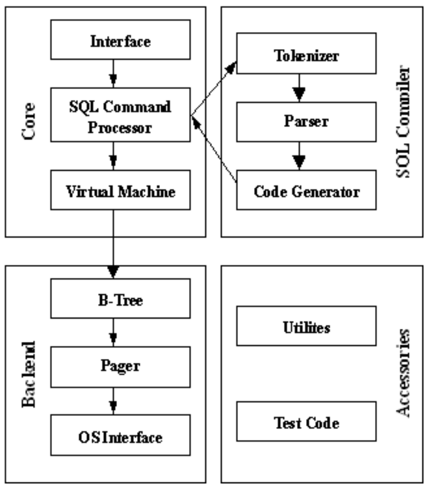

Fig.3 Architecture of SQLite

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 are received from the interface, which go into the high-performance SQL compiler, and be decomposed into various identifiers by tokenizer. The parser, also called lemon analyzer, which can identify the identifiers, and then recompose 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.

Compiler consists of three independent modules, namely Tokenizer, Parser and Code Generator. When strings containing SQL statements are to be executed, the interface transfers these strings 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. Tokenizer and Parser receive the standard SQL statements in accordance with the format of text, and then transform the SQL statements into data structures that can be dealt with by bottom layer. Parser is produced by self-defined parser-generator, namely the lemon of SQLite. When a statement has been converted to symbol that identified by SQLite, and then it be recombined in the form of parse tree, while paser transfers the parse tree to code generator.

Virtual Machine is the heart of the internal structure of SQLite, 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 [9]. In addition, it can do the stack operation processing for completing this operation. The instruction sets of VDBE can analyse any SQL statements, many statements of SQLite are firstly transformed into the corresponding virtual machine statements, and then to complete the tasks that completed by given SQL statements.

The core works of Code generator is to transform the received parse tree into the assembly language identified by SQLite, and then transfer these assembly language programs to the virtual machine to execute.

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 parts: B-tree, Pager and OS interface [10][11]. As a transit station for the database pages, B-tree and pager transfer the unified size of data blocks to OS interface. The task of B-tree is ordering, it maintains the complicated relationship among the data pages, and ensures the connection among the data pages, so that each part of them is very easy to find. 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. The task of pager is to transfer management, database locked and the collapse recovery etc., which require OS interface to collaborate with.

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. Therefore, what the SQLite see is the unified interface.

We have discussed several modules above in detail. In addition, the architecture of SQLite also includes Accessories, namely Utilities and Test code.

Utilities mainly solve some questions such as memory allocation, comparison of no type string, lexical analysis, storage of the symbol table in grammatical analysis, professional printing function and the random number function of SQLite.

Test Code. If you count regression test scripts, more than half of SQLite functions of the test code can be tested here, there are many assert( ) statements in the main code files. The os_test.c backend interface is used to simulate the fault recovery mechanism of SQLite when the system collapses.

3.2.2 Implementation process of SQL statement

When the SQL statements implemented by SQLite , the implementation process often take three stages to complete, namely, Prepare, Step and Finalize. The relational process among the three stages is shown in Fig.4.

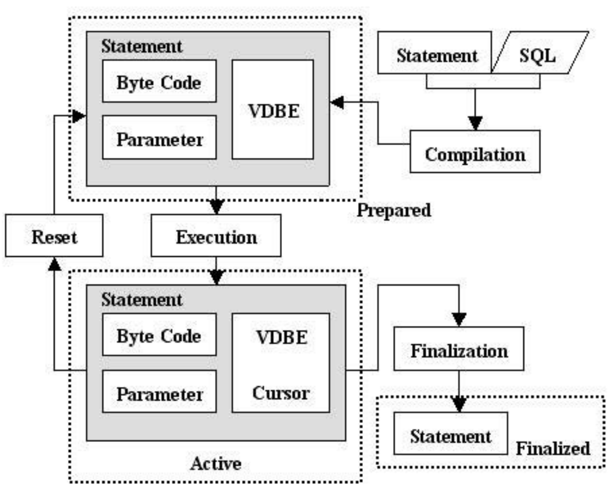

Fig.4 The relational process of SQL statement in the course of implementation

(1) Preparation stage: paser, tokenizer and code generator compile SQL statements into VDBE byte code, in the C API function, the function sqlite_prepare ( ) completes this task. Sqlite_prepare ( ) function directly associates with the compiler, and compiler generates sqlite_stmt, Byte-codes and all the necessary resources that required by executing SQL command are included in sqlite_stmt.

(2) Execution stage: At this stage, the Byte-codes are implemented step-by-step by VDBE, the initialization of every step is completed by the sqlite_step( ).

(3) Finalization stage: At this stage, the implementation of SQL statement is still finished by VDBE, at the same time the resources are released. This process is completed by sqlite_finalize( ).

As is shown in the above diagram, preparation, execution and finalization of the three parts are expressed as prepared, active and finalized respectively. Their implication can be described as follows: prepared means that all the resources have been obtained, SQL statements have been compiled into VDBE byte-codes; Active means that SQL statements are executed from the function aqlite_step( ); Finalized means that SQL statements are executed to end and related resources have been released.

3.3 The Main Interface Functions of SQLite

The API function 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. Four of them are the core API function[12], that is, sqlite_open ( ), sqlite_exec ( ), sqlite_close ( ) , and sqlite_errcode( ), which are used to execute SQL and acquire data, and realize efficient data storage and management. In addition, it is extensible, allowing the programmers to 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 descriptions are the composition of four API functions.

(1) Sqlite3_open ( )

The prototype of sqlite3_open ( ) is:

*int sqlite3_open(const char *filename, sqlite3 **ppDb);*

*int sqlite3_open16 (const void *filename, sqlite3 **ppDb);*

*int sqlite3_open_v2 (const char *filename, sqlite3 **ppDb, int flags, const char *zVfs).*

The first function is to open the database with UIF-8 encoding format, the second function is to open the database with UIF-16 encoding format, and the usage of the third function is similar to the first one in the case of not accepting additional parameters, the two additional parameters is one of these parameters, i.e. *SQLITE OPEN READONLY, SQLITE OPEN READ-WRITE, SQLITE OPEN READWRITE/SQLITE OPEN CREATE.*

Sqlite3_open ( ) opens the specified database, the path and filename of database files are determined by parameters. If the file doesn’t exist, it will be established; If the open or establishment of files are succeed, the function will return to SQLite_OK, meanwhile, it will also return to the legal database handle through the ppDb parameters, or else, the functions will return to the corresponding abnormal code.

(2) Sqlite_close( )

The prototype of Sqlite_close( ) is:

*int sqlite_close (sqlite3 *db)*

The function Sqlite_close ( ) shuts down the database that has been opened, while the handle of the database that has been closed is transferred by db parameters.

(3) Sqlite_exec ( )

The prototype of Sqlite_exec( ) is:

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

This function is used to deal with SQL query, it 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.

(4) int sqlite3_errcode(sqlite3 *db)

The prototype of Sqlite_errcode ( ) is:

*const char * sqlite3_errmsg(sqlite3 *);*

*const void * sqlite3_errmsg16(sqlite3 *);*

Error handling function includes above three functions. The first function returns to error code; the second function obtains error information of the latest operation, it corresponds to UTF-8 encoding format; the third function also obtain error information of the latest operation, it corresponds to UTF-16 encoding format.

4 Porting of SQLite in the ARM-Linux Platform

Embedded database takes the open source SQLite, it is a lightweight relational database, and has three-level modes of architecture, namely, user mode, logic mode and storage mode. In order to make database application run on the ARM-Linux environment, SQLite must be ported to the corresponding Linux platform, and then it is possible to do further application development.

Due to good features and architecture of the SQLite, the porting of a database will be possible. In addition, SQLite takes C language to develop, so the portability is very good. For the porting of SQLite, we can do cross compiling to the source code according to the different platform. This paper takes ARM-Linux as the bottom operating system, to port SQLite on the basis of ARM-Linux kernel, the corresponding ARM-Linux tools chain must be needed, the tools chain is usually installed in the directory- / usr / local / arm-linux / bin /, it usually begins with arm-linux. Porting will concern three tools: arm-linux-gcc, arm-linux-ar and arm-linux-ranlib. The following is a specific course of porting [12][13]:

(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- O6 amended to read:TCC = arm-linux-gcc-O6 ;

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 = - I/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 that have been modified, and then use the commands *make & & make install* to generate what we want to, 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 program 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.

5 A Case about SQLite Application in the Home Gateway Based on ARM

In the remote monitoring system of information appliances, a large number of real-time data need to be collected and processed, the diversity of data storage and management need a backend 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, and SQLite will be firstly considered to choose.

5.1 The Whole Design

In this paper, we construct an embedded home gateway based on S3C44B0X + uCLinux + SQLite + Boa, as shown in Fig.5.

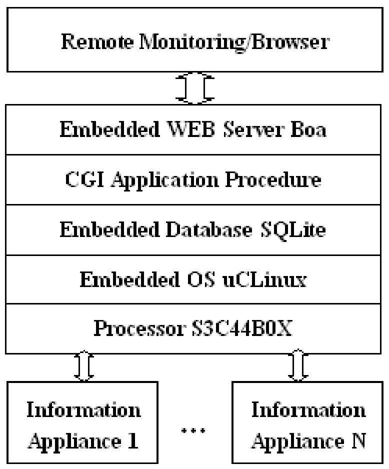

Fig.5 Structure of Embedded Home Gateway

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 [14]. Home gateway provides a unified platform for information appliances, and it can realize the interconnection and information exchange between information appliances or between information appliances and remote control terminals.

AS is shown in Fig.5, 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 stores 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.

In the design of embedded home gateway, the main task of system is completed by two procedures, namely, acquisition and processing program of real-time data, Network service program. The whole design diagram of embedded home gateway system is shown as Fig.6.

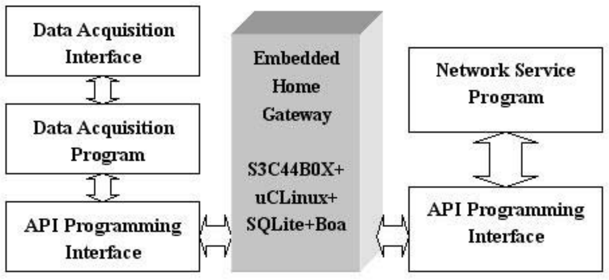
Data Acquisition Interface
Data Acquisition Program
API Programming Interface
Embedded Home Gateway
S3C44B0X+ uCLinux+ SQLite+Boa
Network Service Program
API Programming Interface

Fig.6 the whole design diagram of embedded home gateway system

(1) Real-time data acquisition program. This program includes several modules: data acquisition module, data processing module and data storage module. Data acquisition module is responsible for external signals acquisition, and the data is sent to the data processing module; Data processing module is responsible for having digital filter to data, and then stores the data in the public data buffer; Data storage module is responsible for calling API functions provided by SQLite, and storing the data of buffer in database.

(2) Network service program. This program includes two modules: embedded Web server Boa and CGI program. Embedded Web server is responsible for monitoring the request of network users, when need to request, corresponding CGI program will be started up to analyse and deal with parameters; CGI program convert the results into the format identified by Web browser in accordance with CGI specification, and then send the format back to the client as the HTTP response information.

In the above process, it involves a number of important protocols, such as HTTP protocol between client browser and Web server, CGI protocol between Web server and CGI application program, SQL protocol between CGI application program and embedded database SQLite.

5.2 Generation of SQLite Data Sheets

SQLite contains two types of data sheets, user information table and information appliances table. User information tables store user's name and password, and interact with the user login program and the user registration program; Information appliances tables store the running state data of appliances, mainly interact with information appliance control program and query program.

Information appliances register to the home gateway through the interface definition language, home gateway compiler scans and analyses the IAIDL, and then interprets it as a kind of intermediate information. By calling API function provided by SQLite, the information is stored in the related data table of SQLite, the home gateway has completed the process. After scanning and interpreting IAIDL, the home gateway compiler eventually transforms IAIDL into the corresponding database tables, 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.7.

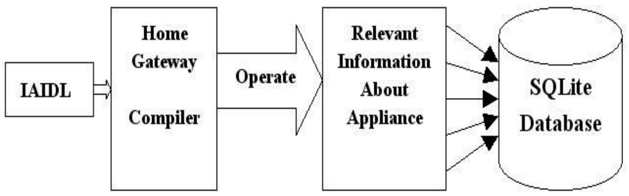

Fig.7 Generation of SQLite data sheets

5.3 Application Program Design Based on SQLite

Application development based on SQLite is how to use CGI program to establish, access, and update the SQL database [15]. This process can be divided into three steps to complete. The description of every step is given as follow:

(1) Establish the backend database of home gateway; Compare with the general database, the backend database needn't to develop front-end interfaces. Due to all operations of database come from network, and all requirements in networks exist in the form of HTML data stream, the front-end interfaces of database are written directly in HTML documents, and integrate with Web, therefore, the database has some advantages such as friendly interfaces, simple development.

(2) Make a Web page and build-in table lists with HTML documents. From the angle of Web page, the table lists provide an interactive interface, browser submits the contents of table lists to Web server application, and the application program operates database according to the user's requests. From the angle of application, the build-in table lists provide an interactive mechanism that application program can have an interaction with customers through the Web documents. The table lists are the front-end interface of database. Because the access to database is only relate to the browser, the embedded database can be used by different machine type and operating system, so that it can achieve the purpose of cross-platform.

(3) Edit CGI program includes decoding, function and output. Every part can be described as follows: In decoding part, environment variables acquired from client will be analyzed, and then obtain the necessary information that support CGI program to run; In function part, make use of these information to complete functions that implemented by CGI, such as the establishment of database connection, database access, etc.; In output part, the running results will return to the server, and eventually return to the customer.

In this paper, we 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 can call sqlite_open( ) to connect with a database, in which you provide the filename and access mode. Next, calling 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 program is how to write the clients' registration information into the table *users* of database file *IA_database* through the CGI program in the home gateway.

```
# 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)
{
fprintf(stderr,"Cound not open
database:%s\n",errmsg);
sqlite3_freemem(errmsg);
exit(1);
}
sprintf(qry,"INSERT INTO
USERS(name,password)\n",
"VALUES('%s','%s')",name,password);
ret=sqlite3_exec(db,qry,NULL,NULL,
&errmsg);
if(ret!=SQLITE3_OK)
{
fprintf(stderr,"SQLerr:%s\n%s\n", errmsg,qry);
}
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 database SQLite 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 realizes monitoring control functions of information appliances, which in turn will affect the user to access embedded database through the browser.

6 Conclusion

With the popularity of intelligent appliances, the formation of mobile computing environment, and the rise of mobile commerce, embedded database has become the focus of study currently, SQLite has small core, open source, and database is a file, it is very easy to realize the copy, move and cross-platform sharing of database files, 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 firstly to the definition, basic characteristics, structure and the key technologies of embedded database, and mainly discuss the features, the architecture and the key interface functions 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, intelligent mobile terminal, information appliances control, home medical equipment, etc.

References:

[1] Li anyu, Lin lijie, Embedded Moblie Database: From Research to Application, *http://news.csdn.ner/news/newstopic/21357.shtml*, 2005.

[2] Christtophe Bobineau, Luc Bouganim, Philippe Pucheral, Patrick Valduriez, "Pico DBMS: Scaling down Database Technique for the Smartcard", *Proceeding of 26th International Conference on Very Large Databases*, Cairo, Egypt. 2000.

[3] Surajit Chauduri, Gerhard Weikum, Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System, *Proceeding of 26th International Conference on Very Large Databases*, Cairo, Egypt, 2000.

[4] Mon-Fong Jiang, Shian-Shyong Tseng, Chang-jiun Tsai, Discovering Structure from Document Databases, *Proceeding of the second International Conference on Knowledge Discovery and Datamining*, 1999.

[5] Rick F, van der Lans. Introduction to SQL: Mastering the Relational Database Language, *Addison Wesley*, 2006.

[6] Hector.Garcia-Molina, JeffreyD.Ullman, JenniferWidom, Database System Implementation, *Prentice Hall*, 2001.

[7] Michael Owens, The Definitive Guide to SQLite, *USA: Apress*, 2006, 341-362.

[8] SQLite homepage [EB/OL] , *http://www.sqlite.org.*

[9] Wang Jinqin, Wan Lixin, The comparison of Embedded Database Berkeley DB and SQLite, *The application of SCM and Embedded System*, 2005, 28(2). 5-7.

[10] Michael Owens, Embedding an SQL Database with SQLite, *Linux Journal*, 2003, June (1).

[11] Hongjun Lu,Yuet Yeung Ng etc, T-Tree or B-Tree: Main Memory Database Index Structure Revisited, *Australasian Database Conference*, 2000:65-73.

[12] Michael A.Olson, Selecting and implementing an embedded database system, *IEEE Computer*, 2000,33(7). 27-34.

[13] Ling-yun, Li-Jun, The Development and Application of Script Program Based on SQLite, *China Academic Journal Electronic Publing House*, 1994-2008.

[14] Zhang qiuyu, Wei zheng, Study and Application of Embedded Home Gateway, *Computer Technology and Development*, 2003

[15] I.Seltzer, Michael A.Olson, Challenges in Embedded Database System Administration, *In Proceeding of the Embedded System Workshop*, Cambridge, MA, 1999.

[16] Zeng Ziming, Zeng Yuanyuan, Research on the Model of Agent-based Embedded Database for Mobile Embedded System, *The Seventh international Conference on Electronic Measurement and instruments*, ICEMI' 2005.