Design and Implementation of Embedded Database Security and Reliability

Zhang Gao
Sichuan Aerospace Vocational College

Abstract: This article analyzes the classification of common embedded databases, including SQLite embedded database, Berkeley embedded database, eXtreme DB embedded database, OpenBASE Lite, etc. By studying data replication and backup, embedded database recovery, embedded database backup design, detailed backup design and implementation, the purpose of this article is to improve the security of the database application process and the integrity of the information stored in the database.

1. Introduction
With the advancement of science and technology, embedded devices have been widely used in military, electronic communications, industrial control, medical and other fields. At the same time, the amount of data processed by embedded devices is also increasing, which promotes the development of embedded database systems. The embedded database system has strict time and space constraints, and requires high reliability. How to ensure that the embedded database meets the running space and running time constraints while ensuring its high security and reliability has become another subject in the field of embedded database research.

2. Common Embedded Database Classification

2.1 SQLite Embedded Database

![SQLite Embedded Database Architecture](image_url)

Figure 1 SQLite Embedded Database Architecture
The SQLite embedded database is an embedded database that uses the C programming language to implement basic attribute functions. The database architecture is shown in Figure 1. As a whole, the embedded database can be divided into four sections, namely the core section, the compiler section, the back-end section and the accessory section [1]. The plate can be further subdivided in the work process. For example, the core plate can be divided into interface module, SQL command processing module and virtual machine module. Compared with traditional embedded databases, SQLite embedded databases have application advantages such as fast running speed, small kernel plate, and strong compatibility. They can be scientifically organized in accordance with the specific application of the operating system. And in the database module, we not only can simultaneously provide the required database files for multiple groups of machinery and equipment, but also can combine the actual needs to encrypt the content of the files in order to be protected, thereby we can improve the security of database information and embedded database reliability [2].

2.2 Berkeley Embedded Database

Similar to the SQLite embedded database, the Berkeley embedded database is also a database designed using open source code for programming. The database architecture is shown in Figure 2. The system consists of five parts: transaction subsystem, storage management subsystem, lock subsystem, memory pool subsystem, and log subsystem. Each group of subsystems has its corresponding service task. For example, the storage management subsystem is a branch structure that assists the overall system to store data information. It is responsible for collecting the correlation information of the system operation, and then centrally backing it up according to a certain application mode to provide a basic application environment for subsequent data information applications. The database has strong compatibility, which can meet the application needs of different structures, thereby improving the practical value of data information [3].

2.3 eXtreme DB Embedded Database

In the embedded database composition system, compared to other groups of databases, eXtreme DB embedded database has relatively strong functional attributes and can be used in a variety of system application environments. At the same time, the database will pass through in the specific application
process. The kernel correlates related data information in the form of a link library. From the perspective of the overall application, the memory overhead of the database structure is small, and in the specific application process, it also has very strong data compatibility, which can adapt to the user-defined API interface, so that in the process of database management, it can very flexible. In addition, eXtreme DB embedded database has application advantages such as strong performance attributes, low investment costs, and strong structural operation stability in the actual application process. At present, the embedded database is in the real-time database management process. Has been well applied in [4].

2.4 OpenBASE Lite
In addition to the above-mentioned embedded database types, OpenBASE Lite also belongs to a database system structure with very comprehensive functional attributes [5]. Its functional attributes are relatively strong, and it can be used in a variety of system application environments. At the same time, in the specific application process of the database, the related data and information are linked through the kernel in the form of a link library. Besides, the size of the structure database can be customized, which can meet the application of various types of database information processing. And in the actual application process, it will also support the standard ODBC and JDBC interfaces, and complete the targeted processing of the formulated data information according to the corresponding type template of the interface. In addition, the database model supports multiple types of embedded systems and processor structures for practical applications, and it can also provide transactional sub-functions externally. It is currently well used in many fields.

3. Design and Implementation of Embedded Database Security and Reliability

3.1 Data Replication and Backup

3.1.1 Data Replication
According to the time limit for updating the primary database and the backup database, data can be copied in two ways, one is synchronous and the other is asynchronous. Synchronous data replication means that the local and remote databases are updated in a fully synchronized manner, that is, when the local update is performed, the remote database is immediately updated. Asynchronous replication means that when the local data is changed, the offsite data is not updated immediately, and the offsite data can be updated at fixed intervals [6]. Although the synchronous replication method can ensure that the primary database and the backup database can be updated in real time, the synchronous replication method places higher requirements on bandwidth. Asynchronous replication does not require complete synchronization of local data and offsite data updates, and is less affected by network bandwidth. However, in this way, there is a delay in updating local and offsite data, which is often used in places that do not require high real-time performance.

3.1.2 Data Backup
When the data information copying process is completed, data backup related operations need to be performed. Based on past application experience, currently commonly used data backup methods can be divided into the following two types. The first is a full backup. The so-called full backup refers to the need to back up the contents of all files in the embedded database when backing up data information. However, when the application efficiency of the full backup and the total amount of database file information are large, the time required to back up the data is relatively long, and the backup speed is relatively slow. The second type, incremental backup, is different from full backup in that in the process of incremental backup, each time a backup operation is performed, the content of the backup is the same as the changed part, but the first time data is backed up At this time, all data needs to be backed up. From the overall situation, this backup method consumes less time and costs, and the speed of backing up data is relatively faster [7].
3.2 Embedded Database Recovery

3.2.1 External Recovery
During the normal application of the embedded database, some data corruption sometimes occurs, such as incomplete copying of data by the user, and a database system crash caused by a virus intrusion. At this time, you can use external recovery to recover data information. The working principle of the recovery method is that in the single-user mode or the multi-user mode, if system data is damaged, the transaction file can be used to recover the content. After opening the transaction subsystem structure, write the log file to the transaction subsystem structure. If you encounter a system crash again during the recovery process, you need to scientifically complete the content of the transaction file before the next database is opened. If multiple log files are found in the transaction file, the database recovery operation needs to be performed in time to improve the integrity of the database information content.

3.2.2 Automatic Recovery
Corresponding to external recovery is automatic recovery. If the main target of external recovery is individual users or multiple users, the main service target of automatic recovery is multi-user mode. In the specific application process, users can use the transaction subsystem to access the embedded database. After a certain transaction subsystem structure fails, the transaction system, log system, and lock management system can be used to complete the recovery of database information. After the database starts the update operation, the system will complete the operation management in a stand-alone manner. At the same time, the log file name is also passed to the lock manager, and a corresponding update instruction is issued to it. At this time, the system will automatically recover when the task submission process is completed. If the entire update process fails, the system will maintain the initial state and perform structural updates again to achieve the ultimate application management purpose.

3.3 Embedded Database Backup Design

3.3.1 Backup System Requirements Analysis
In the actual application of data information, it is necessary to do a good job of application analysis related to the backup system requirements, so as to improve the practical value of the embedded database. In the process of actual application, the process of applying the actual project needs to be used as a basis to carry out the analysis operation of the backup system requirements. Many embedded databases only support local-level backup, and cannot perform data backup at different levels. Therefore, in the backup system requirements analysis process, it can be divided into two cases: synchronous backup mode and asynchronous backup mode. At the same time, the network system is used to perform backup operations across platform areas, thereby realizing the requirements of data off-site backup.
3.3.2 Overall Design of the Backup System

As shown in Figure 3, in the overall design of the backup system, its core content is to complete the cross-platform output of data information, that is, to implement off-site backup of the local database through the network system. To realize this functional requirement, during the normal operation of the local embedded database, participating users can combine the actual needs of themselves to apply the remote mirror database system. And when the system structure is opened, the embedded database located in a different place is also opened at the same time. With the path provided by Socket, the target data information is directly transmitted to the specified mirror path. At this point, the two sets of databases will be associated, that is, under the guidance of the model, the relevant information in the local database will be smoothly distributed to the off-site database, thereby realizing the requirements of data across platforms.

3.4 Detailed Design and Implementation of Backup

3.4.1 Remote Mirror Database Switch Settings

In the actual design process, a corresponding switch structure needs to be set in the embedded database, and whether the switch is open or kept closed is selected by the user in accordance with his own needs. In order to improve the stability of the mirror database operation, a global variable needs to be determined throughout the design process. There are two options for the value of the variable switchmir. One is switchmir = 0, which indicates that the user has performed the decision whether to perform the backup and cannot complete the backup of the database information by changing the path. The other is switchmir = 1. At this time, the entire system structure has formed a channel, and the user performs the mirror copying work uniformly. At this time, the data can be copied from local cross-platform to off-site systems. The specific application steps of this system are shown...
in Figure 4.

![Figure 4 Schematic Diagram of the Initial Mirroring System Operation](image)

When the program starts running, it will evaluate the parameter information in the embedded database. At this time, it will check whether the relevant information in the main database and the mirror database exists. If it exists, it will further determine it and determine the two sets of structures. Whether the .dbd files are consistent. After it is determined that the structure meets the application requirements, switchmir = 1 will be judged. If the judgment result is yes, then turn on the structural switch and start the relevant data copy. If the judgment result is no, then in the specific application process, you need to re-enter the data information and re-judge it. After the requirements are met, then proceed to the next stage.

4. Conclusion
In summary, due to the particularity of the embedded environment and the needs of actual projects, the encryption and backup functions of the embedded database need to be studied. The article introduces the theoretical knowledge related to database encryption, introduces common encryption algorithms, and implements field-level encryption for embedded databases. The embedded environment requires the database to have high reliability. Through research on the embedded database self-recovery and backup, the off-site backup function of the database is realized.

References
[1] Zhang Dongchen, Li Wenxin, Xia Jiagao. Design and implementation of space-borne high reliability embedded database system [J / OL]. Computer Technology and Development, 2019 (11): 1-9 [2020-01-03].
[2] Chen Xuefan, Zhang Peng. Research on storage device management strategy for embedded database [J]. Information Communication, 2019 (02): 199-200.
[3] Chen Xiaonian. Research and Application of Data Query in Embedded Mobile Database System [J]. Information and Computer (Theoretical Edition), 2019 (02): 145-146 + 153.
[4] Yang Junli. Influencing factors of database system security and design of security mechanism [J]. Electronic Design Engineering, 2018, 26 (16): 185-188.
[5] Liu Mouli. Design and implementation of synchronization system based on embedded database [J]. Application of Automation, 2016 (12): 58-60.
[6] Sun Cheng. Research on security of embedded database based on smart devices [D]. Jilin University, 2015.
[7] Yang Huihui. Design and implementation of embedded database security and reliability [D]. Xidian University, 2014.