Design and Implementation of a Heterogeneous Relational Database Synchronization Mechanism Based on Tree Distribution Architecture

Fei Xu
Information Communication Academy
National University of Defense Technology
Wuhan, China
xufei_china@126.com

Linfei Ma
Information Communication Academy
National University of Defense Technology
Wuhan, China
malinfei1993@gmail.com

Feng Wen
Information Communication Academy
National University of Defense Technology
Wuhan, China
123826545@qq.com

Binde Luo
32336 troops of PLA
290616365@qq.com

***Abstract*—In allusion to the database server of tree distribution in the network environment, this paper proposes a heterogeneous database synchronization mechanism based on the shadow table method. In this paper, the application environment of the isolation netgap is described, and the overall architecture of the synchronization system is introduced, and the implementation of the module such as change data capture, incremental file generation, data synchronization operation and exception handling is proposed. Finally, the database synchronization mechanism is verified by the application of the data acquisition system of an organization.**

Keywords- Tree distribution architecture; Heterogeneous relational database; Database synchronization;

I. INTRODUCTION

As an efficient and convenient data management tool, relationship database is widely used in government, enterprise and army. With the continuous development of information technology and the need of the organization's own business development, the relationship database they used have gradually exhibited the features of dispersion and heterogeneity[1]. In order to guarantee the quality of service, the underlying server needs to meet the following 4 characteristics: low latency, high availability, high fault tolerance, high scalability (referred to as "three high and one low" or H3L)[2].

To meet these characteristics, the usual technical means is to "distribute" the data, the complete data is divided into multiple copies according to certain rules, and each piece of data as a subset can process part of the business requirements and provide certain data services. The tree database distribution architecture shown in Figure 1. is widely used in large enterprises, chain companies, government departments and other institutions. In this architecture, the central database can be divided into sub-database of different levels according to the related business, service object, geographical location, management level, security level (the following is divided into services, and other divisions are similar), and the sub-databases can continue to expand their own data needs. The advantages of this architecture include:

1) Data security: the security protection of data in this architecture is reflected in two points. First, the bearer network can select the network that meets the security requirements according to its own needs. Second, security devices or security protocols can be set up between the bearer network and the database to further improve data security, such as isolation netgap and secret machine.

2) Strong scalability: the database distribution architecture shown in Figure 1. can be expanded horizontally based on new business, the expanded sub-database only includes relevant data for newly expanding new services, and will not affect the functions of other sub-databases. Meanwhile, each sub-database can continue to expand vertically as a central database to further refine the business category. Similarly, the vertical expansion structure will not affect the functionality of existing structures.

3) Strong robustness: after the collapse of any database in the whole architecture, the data of the current database can be restored through other databases if the backup data is not available. Specifically, a central database with vertical expansion can synthesize data from all sub-databases to construct a new central database, while a sub-database without vertical expansion can reconstruct a sub-database by distributing corresponding business data from the central database. In the second case, the sub-database may lack data that was recently updated by the business system and not synchronized to the central database.

978-1-7281-5712-2/19/$31.00 ©2019 IEEE
DOI 10.1109/ICISCE48695.2019.00054

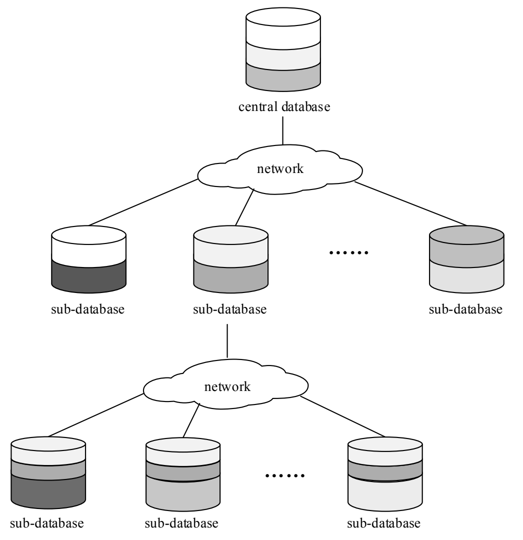


Figure 1. The database architecture based on tree distribution

4) Strong autonomy: in order to ensure the consistency of data in the entire architecture, each database's ability to manage data is reflected in two aspects. First, the relevant data of the sub-database are summarized. The second is to synchronize the related data of the database to the central server. This mechanism allows the sub-database to have a high degree of autonomy for the data.

In the tree distribution architecture, each database can provide data services for related business in its network. However, since the sub-database has snapshots of some data in the central database, in order to ensure the consistency of the services provided by the sub-database and the central database, it is necessary to ensure that the data in all the databases in the architecture shown in Figure 1. is consistent. Based on this, this paper designs a data synchronization system architecture.

II. Data Synchronization System Architecture

In some specific scenarios, there are security devices between the database and the bearer network, such as isolation netgap and secret machine. The information exchange between the two ends must follow the unique transmission mode. Therefore, each database is independent in the network and cannot directly perform data interaction. In order to ensure the data consistency between the central database and the sub-database and complete the data synchronization between the heterogeneous databases, the data synchronization system architecture shown in Figure 2. is designed. A dedicated transmission channel composed of components such as security devices and bearer networks serves as a means of information interaction between databases. The business system directly stores the generated data in the sub-database through the direct connection method. The data files that need to be synchronized are generated by the sub-database, and are synchronized to the central database through a dedicated transmission channel. After the central database synchronizes the data, the receipt files are generated and transmitted back to the sub-database.

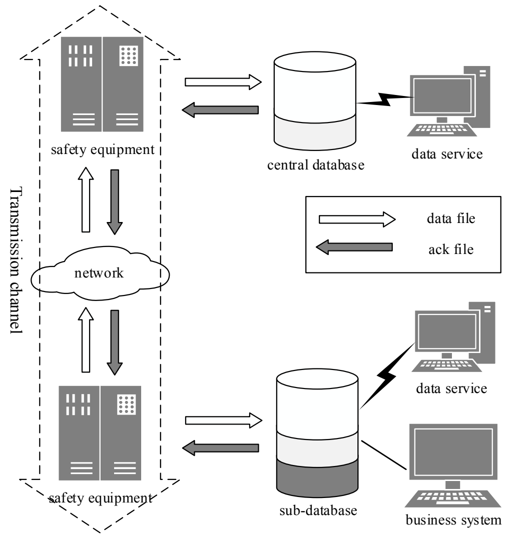


Figure 2. The data synchronization system architecture

III. Data SYNCHRONIZATION Mode Selection

The data synchronization methods are divided into full synchronization and incremental synchronization from the level of generating data content.

A. Full synchronization

When synchronizing data, all the data in the database of subordinate system is completely sent to the superior system, replacing the data associated with the subordinate organization stored in the superior system database to achieve data consistency.

B. Incremental synchronization

When synchronizing data, the changed data would be sent to the superior system after the recent successfully report process to achieve the consistency of the data in the systems of upper and lower levels.

IV. Functional Module Design

The system architecture based on incremental synchronization is composed of five sub-modules: change data capture, incremental data file generation, data transmission, data importing control and exception handling. These modules work together to complete data synchronization between heterogeneous databases.

A. Change data capture module

At present, the methods of change data capture mainly include system log method, trigger method, log analyze method and shadow table method, etc. These methods have their own advantages and disadvantages [3]. Combined with the information interaction ability of the dedicated transmission channel, comprehensively compare various change data capture methods. This paper uses the shadow table method to generate change data, mainly considering the following aspects:

1) The data transmission by the way of "reliable transmission" method between the netgap is quasi-real-time transmission, which has a natural disadvantage in network delay. No matter what kind of change data capture method is used, it cannot shorten the transmission delay.

2) The shadow table method can obtain incremental data through comparison between tables, which could reduce the demand for the bandwidth and improve the transmission efficiency.

3) The shadow table method only needs to grasp the final state of the database, independent of other information, and the feasibility is high.

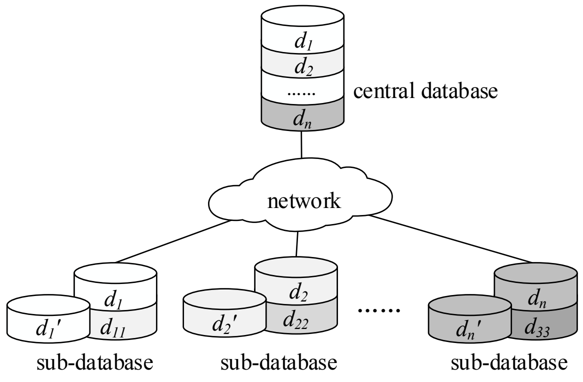


Figure 3. The database architecture based on shadow table

The two-level database setup using the shadow table method is shown in Figure 3. The central database divides data into n subsets $\{d_1, d_2, ..., d_n\}$ based on business, which meet $d_{center}=\bigcup_{i=1}^{n} d_i$, and $d_i \bigcap_{i\neq j} d_j = \phi$. N sub-databases are deployed through the bearer network, and each sub-database is built with a subset of the data d_i divided by the central database, and a shadow table database d_i' is constructed at the same time. The data structure in the shadow table database d_i' is exactly the same as in d_i, but the data content is the data after the last synchronization. That is to say, at any moment, the content in the shadow table database d_i' is completely consistent with the data d_i of the service in the central database, and the shadow database d_i' in the sub-database can be regarded as the mirror image of the central data about the service in the database of the current level.

In some cases, the sub-database can provide a more refined data service within the service, and can expand its data structure d_{ii} based on d_i, and the data generated by the corresponding business system is stored in $d_i \bigcup d_{ii}$ in the sub-database. Once the contents of $d_i \bigcup d_{ii}$ in the sub-database have been updated, the contents of the sub-database need to be synchronized to the central database.

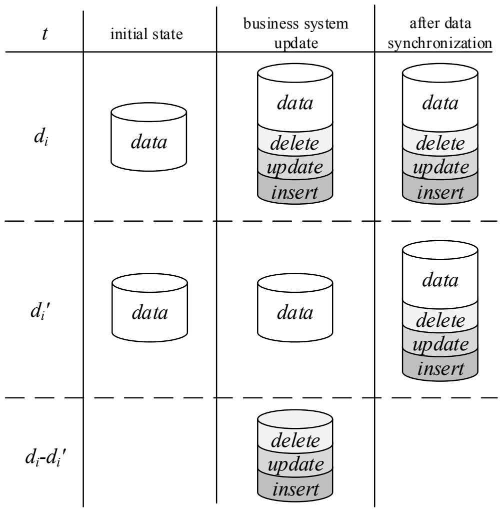


Figure 4. The state change diagram of sub-database

When capturing change data, the system of this level does not need to establish a connection with the superior system, and the data difference between d_i and d_i' can be directly compared. After generating the change data, the subordinate node sends the data to the superior node through the dedicated transmission channel, and the upper node completes the data importing operation. Superior nodes would generate acknowledgement (ack) documents, and send them back to lower nodes also by the dedicated transmission channel. A round of data synchronization has been completed for now.

For each non-root node, the contents of d_i and d_i' depend on the current state of the system. As shown in Figure 4. , the data in d_i and d_i' database is exactly the same at the initial time of system deployment. After a period of business processing, the data in d_i has changed due to operations such as insert, update and delete. After the data is synchronized, the data in d_i and d_i' become identical again, the same as the initial state.

The change data capture module is to find out the change data and create the data file. Let the difference data of d_i and d_i' at a certain time be $C_{increment}$, and the new added data set in d_i is C_{insert}, and the modified data set is C_{update}, and the removed data set is C_{delete}, which is easy to know:

$$C_{increment} = C_{insert} + C_{update} + C_{delete} \tag{1}$$

The incremental data generation work can be decomposed into calculating C_{insert}, C_{update} and C_{delete}.

1) Calculate C_{insert}

The C_{insert} is a set of records that exist in d_i but not in d_i'. For each table in d_i, calculating C_{insert} means to calculate the sum of the added records for all tables in d_i.

For a table t in d_i, there must be a table t' with the same structure in d_i'. The new record in t can be obtained by comparing the primary key values recorded in d_i and d_i'. Let t's primary key be *pk_t*, then the SQL statement *select * from di.t where not exists (select 1 from di'.t where di.t.pk_t = di'.t.pk_t)* can quickly query the new records in table t, which exist in d_i and not exist in d_i'. This kind of query is performed on all the tables in d_i, and the result set is C_{insert}.

2) Calculate C_{delete}

From the above, we can see that, C_{delete} is the sum of the deleted records of all the tables in d_i. For each table t in d_i, we need to calculate the record that exists in d_i' but not in d_i. Therefore, the calculation process of C_{delete} is similar to the calculation process of C_{insert}, and it is only necessary to exchange d_i and d_i'. The SQL statement *select * from d_i'.t where not exists (select 1 from d_i.t where d_i.t.pk_t = d_i'.t.pk_t)* can quickly query the deleted records in table t, to perform the same query statement on all the tables in d_i, we can get the result set C_{delete}.

3) Calculate C_{update}

From the above, it can be seen that, calculating C_{update}, that is, calculating the sum of the modified records of all tables in d_i. Compared with calculating C_{insert} or C_{delete}, C_{update}'s calculating process is more complicated, mainly including two steps:

a) Find out the records with the same primary key value in all di and di'. Through the SQL statement select * from di.t, di'.t where di.t.pk_t = di'.t.pk_t can query the same records of the primary key value in t. In the case that the number of records in t is large and the number of operations (insert, update and delete) is relatively small, most of the records in di and di' is the same, so they can be removed from the data set by preprocessing. Calculating Cupdate in the remaining data set can effectively reduce the amount of calculations.

b) Compare the records with the same primary key value by fields, and find the different fields and values. Field-by-field comparison can be implemented by code, and the resulting fields and their values can be added to the data file as part of the incremental data.

B. Data file generation module

Due to the existence of the encryption device, and the data interaction between networks can only be conducted through dedicated transmission channel. Incremental data synchronized between heterogeneous databases also need to be delivered in this way. As a result, the calculated $C_{increment}$ needs to be persistent and stored to a file, and then it can be transferred. The data file generation module is used to write the incremental data to the incremental file.

At present, there are two commonly used binary format files for data transmission between networks: XML and JSON. There are also binary serialization formats that have been specifically optimized for data compression, such as MessagePack[4].

XML is a sophisticated markup language developed from the Standard Generalized Markup Language (SGML). Based on the characteristics of markup language, XML has high readability [5], and it is efficient in node retrieval. It can optimize the parsing program for specific problems and achieve the effect of improving the parsing speed of XML files.

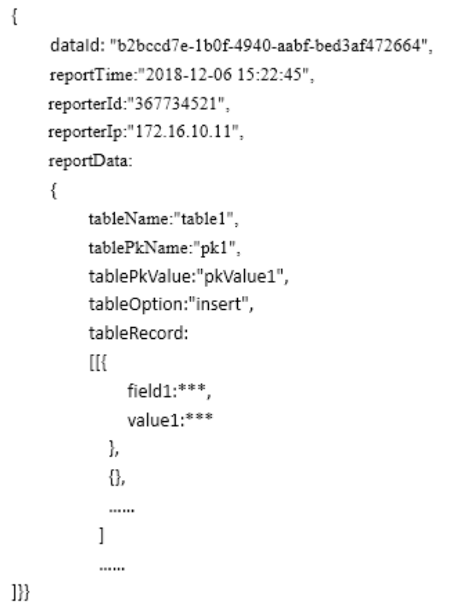
JSON is not a markup language, but a JavaScript object notation, a kind of syntax for storing and transferring text [5]. JSON uses a simple key-value method to store data, which is much smaller than XML. It can reduce the resource requirements such as network bandwidth in the transmission of network information, and thanks to the browser's support for JavaScript, it is easy to parse JSON data. The similarities and differences between XML and JSON can be found in [6]. Since JSON files only store data, compared to XML, there is a lack of description of the data structure, so the requirements for program developers are higher. At present, there are many mature third-party JSON libraries, which improve the parsing efficiency of JSON files and enhance the function of JSON library based on standard JSON [7], and this further strengthens the application of JSON format in network data exchange.

In the process of distributed systems, such as Web services, component server communication, Android or IoT terminal device data exchange, XML and JSON have similar research conclusions, as shown in TABLE I.

TABLE I. THE COMPARISON OF XML AND JSON IN DATA TRANSMISSION**ERROR! REFERENCE SOURCE NOT FOUND.**.

Comparison Item	XML	JSON
data formate	complex	simple
storage space occupation	large	small
required bandwidth	high	low
amount of code	big	small
coding efficiency	low	high
parsing speed	slow	fast
agility	low	hign
learning threshold	hign	low
language and platform support	good	good

Comprehensively comparing of the above two file formats, this paper uses JSON files to store incremental data. The file structure is shown in Figure 5. , which includes two types of data, one for the management of system, and the other is to pass the data changes in the business data. The management data mainly includes related information of the data transmitter and receiver, data file encoding, sending time. The business class data mainly includes the incremental data information generated by the comparison, such as table name, primary key name, primary key value, data operation type (insert, update and delete), field name, field value, and so on.



```
{
    dataId: "b2bccd7e-1b0f-4940-aabf-bed3af472664",
    reportTime:"2018-12-06 15:22:45",
    reporterId:"367734521",
    reporterIp:"172.16.10.11",
    reportData:
    {
        tableName:"table1",
        tablePkName:"pk1",
        tablePkValue:"pkValue1",
        tableOption:"insert",
        tableRecord:
        [[{
            field1:***,
            value1:***
          },
          {},
          ......
         ]
         ......
]}}
```

Figure 5. The JSON file format

C. Data transmission module

The data transmission module is used to perform the task of sending incremental data packets and receiving the ack files. Data transmission and reception operations are implemented by secure transmission in the encryption device environment. The data synchronization system architecture mainly implements calls to the sending and receiving interfaces of the encryption device.

Taking the netgap encryption device as an example, the secure netgap (SGAP) is a common physical isolation means, and physical separation is established between the internal and external networks that need to communicate through technical means. As shown in Figure 6. , the netgap 's support for the application is mainly achieved by stripping and rebuilding the application protocol, decomposing or reorganizing the data packets or files that need to be transmitted into static data, and performing security review on the static data. Then the data can flow to another network. In addition to data review, the security monitoring and control module also controls the switches on both sides to ensure that the internal and external networks cannot be connected to the netgap at the same time [1].

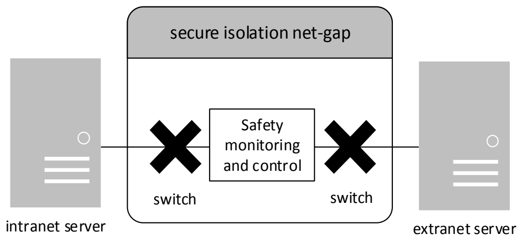


Figure 6. The logical control figure of secure isolation net-gap

The heterogeneous database synchronization architecture proposed in this paper only calls the common file interaction function of the netgap, which is mainly based on the following considerations:

1) Security devices can interact based on files. Not all devices have database synchronization. Netgap is the example of encryption security devices.

2) Database synchronization in the netgap environment requires the deployment of a database server on the external network, and there are still some security risks.

3) There is the process of receipt confirmation in the data synchronization process in the architecture, and the way of database synchronization cannot meet this requirement.

When invoking the file interaction function, the netgap needs to configure the file mapping corresponding to the internal and external networks, as shown in TABLE II.

TABLE II. THE DESCIPTION OF A CONFIGURATION FILE.

Folder Name	Meaning
OutSendFolder	the folders moved from extranet to intranet
OutRecvFolder	the folders copied from intranet to extranet
InSendFolder	the folders moved from intranet to extranet
InRecvFolder	the folders copied from extranet to intranet

The data transmission module is built in combination with the netgap environment, which mainly includes the following three aspects:

1) The data files flow in the internal and external networks in the netgap environment, including moving from the internal network to the external network in the sub-database netgap environment and from the external network to the internal network in the environment of the central database netgap. This process is performed automatically after configuring the file mapping of the netgap.

2) The flow of data files and receipt files in the bearer network, including data files sent by the sub-database netgap external network to the central database and the receipt files sent to the sub-database netgap external network by the central database. This process is executed by the file sending program, and the network address of the receiver needs to be configured.

3) The receipt files flow in the internal and external networks in the netgap environment, including moving from the internal network to the external network in the central database netgap environment and from the external network to the internal network in the sub-database netgap environment. Same as 1), this process is automatically executed after configuring the file mapping of the netgap.

D. Data importing control module

After receiving the incremental data sent by the subordinate system, the superior system will start a file importing process. The main steps include:

1) JSON file parsing

The receiver needs to parse JSON file content to obtain management class data and the business data. The management class data will be used for the operation log update in step4, and the business class data will be used for the generation of the SQL statement in step2.

2) Convert SQL statements

The receiver performs further processing on the business class data. According to the different data operation types, the business data will be converted into an executable SQL statement by using fields such as table name, primary key name and primary key value. These statements would be used in the data importing process in step3.

3) Perform importing operation

After the executable SQL statements are obtained, the receiver system executes them to synchronize the data of current level with the lower one.

4) Update operation log

After the synchronization is successful, the receiver inserts this synchronous information into the database log table and saves the operation information of this synchronization to the importing operation table for later analysis.

5) Generate and send ack files

After the importing operation is completed, the receiver generates an ack file according to the data importing result, and sends it to the lower system through the dedicated transmission channel to complete the data importing operation.

E. Exception handling module

The exception handling module is used to process exceptions in the synchronization process between heterogeneous databases to ensure the consistency of data. Figure 7. and Figure 8. show the state diagram of the sender and the receiver. In the data synchronization under the netgap environment, there are mainly two types of exceptions. One is the failure caused by network reasons, and the other is the failure of synchronization due to data conflicts.

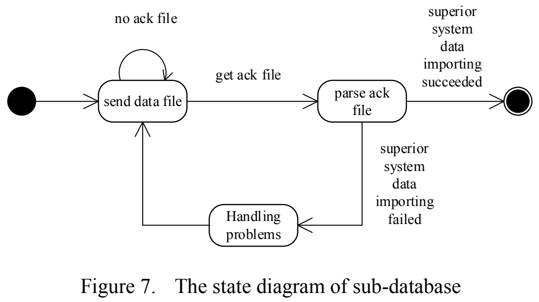

no ack file
get ack file
send data file
parse ack file
superior system data importing succeeded
superior system data importing failed
Handling problems

Figure 7. The state diagram of sub-database

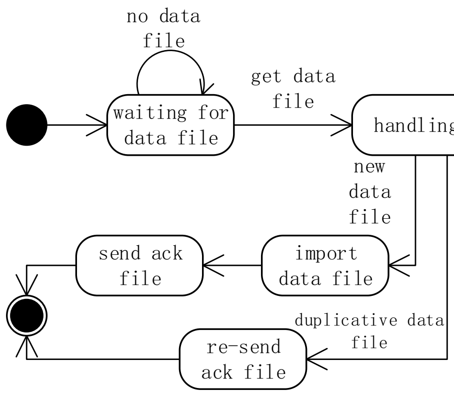


Figure 8. The state diagram of central database

1) Synchronization failure due to network reasons

Mainly refers to the failure of data synchronization due to the failure of transmission of data files or ack files. For the data sender, when the data synchronization fails, it doesn't know whether the data file has not been successfully received by the receiver or the ack file has lost, and its performance is that the receiving of the ack file has timed out. In this case, the data sender can only resend the data file again after a period of time. For the data receiver, if a new incremental data packet is received, the data importing operation will be performed directly. If a duplicate incremental packet is received, it will be discarded and the corresponding ack file for the packet is going to be resent to the data sender again.

2) Synchronization failure due to data conflict

Mainly refers to the failure of data synchronization due to the conflict of record's primary key or foreign key values, such as multiple execution of the insert statement with the same primary key value, or the value of a foreign key violates the foreign key constraint. The process is to roll back the latest reporting operation and indicate the reason of the error in ack file, and after the existing problems are solved, the data reporting operation would be performed again.

V. EXPERIMENTAL SYSTEM VERIFICATION

Based on the above architecture, this paper designs experiments to analyze the performance of each module. Since the efficiency and performance of the data transmission module and the exception handling module are greatly affected by hardware such as networks and devices, they are not considered in the experimental verification. This paper conducts experiments on the change data capture module, the data file generation module and the data importing control module, and analyzes the synchronization results of the insert, update and delete operations for the data when processing different numbers of data, wherein the change data capture module and the data importing control module takes the processing time as the evaluation basis, the data file generation module takes the data file size as the basis for evaluation, and the result takes the average of multiple experiments.

For the insert operation, the processing results of the data at the 1w, 10w, and 50w levels are shown in TABLE III. :

TABLE III. SYNCHRONIZATION PERFORMANCE OF INSERT OPERATION

data Size	data generation module	file generation module	data importing control module
10K	0.11s	0.34s	8.90s
100K	4.45s	3.90s	72.23s
500K	10.74s	21.57s	517.40s

For the update operation, the processing results of the data at the 1w, 10w, and 50w levels are shown in TABLE IV. :

TABLE IV. THE SYNCHRONIZATION PERFORMANCE OF UPDATE OPERATION

data Size	data generation module	file generation module	data importing control module
10K	0.13s	0.31s	5.62s
100K	5.82s	3.60s	44.82s
500K	17.85s	19.63s	351.79s

For the delete operation, the processing results of the data at the 1w, 10w, and 50w levels are as shown in TABLE V. :

TABLE V. THE SYNCHRONIZATION PERFORMANCE OF DELETE OPERATION

data Size	data generation module	file generation module	data importing control module
10K	0.10s	0.03s	4.22s
100K	2.31s	0.54s	42.83s
500K	13.22s	2.44s	251.79s

The data in TABLE III. , TABLE IV. and TABLE V. indicate that the efficiency of the data synchronization mechanism is related to the size of the data volume. The larger the data volume is, the lower the processing efficiency will be. The change data capture module and the data file generation module are more efficient, and the data importing control module takes more time under the same conditions. The efficiency of insert, update, and modify operations is slightly different, with the delete operation being the most efficient and the update operation being the least efficient. This result is compared with the results in [9]. The mechanism proposed in this paper has great advantages in the query operation (corresponding to the change data capture module) and the parsing operation (corresponding to the data file generation module).

VI. CONCLUSION

This paper studies the data synchronization between heterogeneous data in the encryption device environment and designs the experimental system to verify it. By using the contrast analysis of the shadow table and the incremental data synchronization, the data consistency between the databases can be satisfied, and the individualized requirements of the database can be also fully satisfied. The data synchronization problem of the heterogeneous system is well solved. The large-scale use of a business system based on this synchronization strategy also proves the accuracy and reliability of the synchronization method to some extent. It is necessary to further study how to improve the efficiency of incremental data generation and to design and optimize incremental data generation algorithms in case of large amount of data, and how to reduce the consumption of system computing resources.

REFERENCES

[1] CUI Hengxiang, FENG Jing, MA Weijun, ZHANG Qing, and PANG Kun, “Research and Implementation of Heterogeneous Database Synchronization Based on Netgap,” Software Engineering, vol 19, Feb. 2016, pp. 10-13.

[2] Wei Hengfeng, “Research on Distribution Data Consistency Technology” [D], Nanjing University. 2016.

[3] Wang Hao, Ge Ang, and Zhao Qing, “Spark-based database increment near-real-time synchronization”, Software and Algorithms, vol 35, 2016, pp. :9-10.

[4] MessagePack is a binary serialization format, which is commonly used for data exchange. https://msgpack.org/.

[5] Shu Yin, “Exploring Serialization and Deserialization” [J], Communication World, 2019, 26(01): 190-191.

[6] Liu Baolin, “Research and Comparison of Python Text Analysis” [J], Computer Programming Techniques and Maintenance, 2015(09): 14-15, 23.

[7] ZHANG Yuanyuan, ZHANG Qinyan, LI Feng, and YAO Xianglong. “Design and Implementation of Distributed Heterogeneous Data Synchronization Platform JTang Sync,” Computer Technology and Development, vol 26, Dec. 2016, pp. 169-175.

[8] Zhang Yifan, Liu Pingzeng, Ma Hongjian et al, “A Distributed System Architecture Based on JSON” [J], Chinese Journal of Agricultural Mechanization, 2015,36(05): 255-257, 266.

[9] Wang Xiaorui, Huang Xiangzhi, Shen Xiajiong, Zhou ke, and Wang Dong, “Study of the data synchronization model for heterogeneous databases”, Computer Era, No.5, May. 2016, pp. 12-15.