A New Technology for Smooth Migration from Traditional Databases to Unstructured Databases

Zhan-hui XIAO, Jun CHEN * and Long ZHOU

Guangdong Gower Grid Company of China Southern Power Grid, DongFeng East Road No. 757, Yuexiu District of Guangzhou City, Guangdong Province, China

*Corresponding author

Keywords: Traditional database, Non-structured database, Migration solution.

Abstract. Using big-data technologies for more comprehensive corporate data analysis requires a migration of data storage from conventional databases to unstructured ones. As a result, technologies for smooth database migration are particularly critical for corporates. This paper proposes a technique for smooth migration from conventional databases to big-data systems, which is validated by using a traditional corporate as an application example. This paper first briefly introduces the database migration processes and then describes the solutions for the related technique problems in conventional database migration in details. At last, the proposed approach is validated using a corporate application example.

Introduction

With the development of technologies such as cloud computing and Internet, a tremendous amount of data are generated every day, which include not only structured but also unstructured data, such as video, audio, and text files[1]. However, traditional databases lack scalability due to their inefficiency in mass data storage and querying. The problems related to traditional structured databases can be solved by using unstructured databases, which are highly efficient in mass data read/write performance and can be easily expanded to store custom data formats at any time[2]. Using big-data technologies to comprehensively analyze corporate data requires data migration from a conventional database system to an unstructured database[3].

To our knowledge, no existing studies were focused the migration technologies from traditional relational databases to new unstructured databases[4]. Data migration technologies are facing three major challenges as follows: (1) The data migration process with single-system shutdown can cause serious data loss due to the lack of backup and restoration mechanisms. Data missing caused by the single-system shutdown during the entry of the migrating data in the write interface can result in unrecoverable loss to corporates. When the entry of the migrating data in the write interface fails, it is impossible to restore the original database. (2) The “dirty data” generated during migration tests cannot be processed. Storage of migrated data into a new database through new write interface might generate certain problems and lead to the entry of massive poor-quality dirty data that are difficult to repair[5]. (3) For traditional developers, learning big-data systems is too costly and switching data access layers is too difficult. Accessing new databases requires the developers to master the technology of big data systems, but the learning cycle is long and the cost is high, which in turn increases the difficulty in application transplantation of upper layers[6].

Migration Process

Database migration can be divided into four stages, including 15 steps. First, the history data are imported into the big data system based on the principle of incremental import in scale. The correctness is verified before each time of incremental import in scale. The operator should pay close attention to the error warnings during the importing process to determine the restore point in a timely manner.
Next, according to the structure of the production system, a compatible interface is developed to allow the entry of data into the new database through interface. After iteration debug testing of the interface, the data are written into both new and old databases in parallel and meanwhile the problems in the entry interface are repaired until real-time data can be stored into the new database to complete the entry interface migration.

Various unexpected events might occur during the process from parallel entry in the dual system to the completion of the entry interface migration, which can result in serious data quality issues in the new database. The low-quality data should be deleted from the new database. In addition, due to possible data loss during the process from the implementation of the new entry interface to the completion of database migration, it is necessary to conduct bidirectional repairing of missing data. Lossless data mending can be achieved by replacing the data from a particular time point before implementation of new data interface to a time point after the new application with the data from the original database in the same time window.

After entry interface migration, the history data are already imported into the big data system and the real-time entry of online data has been realized. The next step is the migration of the operations system. Because the upper layers often adopt the earlier application development frameworks such as JDBC, Hibernate, etc., it is necessary to design a query interface with a compatible threshold value. The system proposed in this paper encapsulates the big data system using the Table Function technique, through which an incorporation of word ‘from’ and integration of structured and unstructured data in the process of data storage are achieved and the application learning barrier for developers is reduced. After debug testing of the query interface, the migration of in-use production system and confirmation of the operations layer can be performed.

After successfully completing the procedures including history data import, modification of the entry and query interfaces, and database removal, and ensuring no problems in these procedures, the
original history database system can be removed. However, generally we recommend that it should not be removed immediately and instead, it should be safely removed after 3-6 months of trial operation.

**Application Example**
A traditional corporate is used as an example, which has 6000 operating devices passing back 50 million pieces of real-time data daily, including the working condition data from the working hosts and the electronic devices equipped on the hosts as well as the location data passed back from the hosts in various regions. The structure of data collection and processing platforms of this corporate is shown in Figure 4, including a M2M platform for data collection and processing as well as the upper-layer application system based on the M2M platform.

![Figure 4. Structural diagram of the data collection and processing platforms in the corporate example.](image)

As shown in Figure 4, the M2M platform is the distributed processing cluster responsible for receiving and parsing the working condition data from the operating devices in the corporate. The M2M platform is composed of interface PCs clusters, processor clusters, business databases and data warehouses. Data stored on M2M platform can be divided into three categories: (1) host data, mainly including the basic info of the hosts, the definition data of working condition data sources, device lock records, error types and other meta information, installation relationships between the core components and the hosts, data sources and working conditions of the hosts, group information of working conditions, etc.; (2) working condition data, mainly including real-time data passed back from the hosts across the country, condition data generated in the working process of the hosts and from the electronic devices equipped on the hosts, as well as real-time location data from the hosts all over the country; (3) working condition data warehouse, including mainly various types of host data statistics (e.g., workload and pumping throughput statistics), which are entered after analysis of the host and working condition data via data storage in a regular basis.

Considering the actual needs of the corporate, the main solutions targeting the aforementioned three technique challenges in the traditional database migration are as follows:

**Parallel System Migration**
The working condition data are updated and transmitted in real time, which requires a migration of data storage from a traditional Oracle database to a big-data system. Thus, a real-time LaUD interface for working condition data is implemented and connected to the M2M platform. The data, including the working condition data of both hosts and equipped electronics, are written into the Cassandra and Oracle system in parallel via the M2M platform. Further, a modification of the data warehouse of operation decisions and working conditions is conducted to replace the source of the original data source with LaUD. The registration information of hosts and definition information of working conditions are still stored in the Oracle database.
Application Transplantation for User-Friendly Data Access

Upper-layer applications of M2M are mainly divided into two categories: online and history-data application system. The former receives real-time working condition data delivered directly from the load-balanced network on the M2M platform and stores a small amount of data in its database, based on which it provides service to its upper layers. This system is not affected by the migration of M2M business databases. The history-data application system needs to access the M2M business database and M2M data warehouse, including querying the working condition data and assessing the data of remote maintenance system and operating decisions system. Because the working condition data can be written into the LaUD big-data system, the necessity of written the working condition data into the Oracle database is eliminated through modifying the working condition data query system, remote maintenance system and operating decisions system, thus realizing a “true” migration.

(1) Remote maintenance system is a set of Web application developed by Java. The Web server is Tomcat 6.0 and the iBatis migration framework is used for database access. The back-end database is established directly based on host data and working condition data on the M2M platform. The main functions include: 1) query the history data of the device working conditions, 2) diagnose faults and display the self-diagnosis results of the hosts by the smart components, 3) conduct factory tests and remote device lock.

For the remote maintenance system, if the function of history data querying for the working conditions of the hosts and electronics is affected by the migration process, the history working conditions of these components will all be replaced by the data extracted from the LaUD databases. Therefore, it is recommended to build up a web server supporting restful api and encapsulate the working condition data in LaUD into a series of external interfaces than can be remotely invoked. Next, the data in the LaUD database are extracted to the Oracle server using the function provided by Oracle UTL_HTTP package. Finally, the data are externally packaged as a storage process, using Oracle as shell for unified encapsulation.

(2) Working condition data query is a Java-developed web application, which uses Tomcat 6.0 as a web server and iBatis as a migration framework. The back-end database is established directly based on working condition data on the M2M platform.

Using the LaUD database query technology entirely for data query requires a modification of the DAO layer (a data access layer) of Web Service, which can be realized using an approach similar to that for the remote maintenance system above. The main challenge is the embedding of a Java program into Oracle, which enables the storage process to query the latest working condition in the corresponding tables. There are two critical query steps: one is to query the latest working condition, and another is to query the up-to-date time.

(3) Operation decisions warehouse is a Java-developed web application, which uses Tomcat 6.0 as a web server and iBatis as a migration framework. The back-end database is established directly based on the working condition data on the M2M platform.

The upper-layer application system of data warehouse acquires the statistics of the working conditions in all provinces and cities directly from daily updated fact tables, i.e., equipworkload, factdayworkload, and FactWorkload. Therefore, the aim of the migration is to update the three fact tables using the data in LaUD. From the perspective of long-term benefits, it is more meaningful to completely replace the function of the storage process by exporting the output data to the fact tables, because the future analysis and statistics of the history data will involve these business logics.

The post-migration status is shown in Fig.5. Up to date, 30 billion pieces of working condition data and 50 billion pieces of electronic data have been stored and updated in real time in the LaUD-based big-data system. Besides, the access time of the upper-layer applications to the big-data system is shortened to the second level, significantly increasing the production efficiency of the corporate.
Conclusions

Traditional databases cannot meet the increasing demand for data storage. Besides, data missing/loss caused by the traditional database migration techniques might lead to great loss for corporates. This paper designs and implements a new technology for smooth data migration from conventional databases to big-data systems. This technology possesses a variety of advantages. First, the parallel data entry through dual systems with the original database serving as data backup can avoid data loss. Second, the dirty data in the entry interface generated during testing can be deleted and replaced with the data imported from the original database in the same time window to ensure a lossless data migration. Third, a SQL-like statement encapsulation of data access to the big-data system reduces the modification of upper-layer applications, thus reducing the learning cost of developers. In the future research, we will focus on developing database migration technologies that are more efficient.

References


