INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 12 (24): 8165-8170, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.8165.8170

Design and Realization of Bank Non-site Auditing ETL

Wang Shan
Computer Engineering College, Shenzhen Polytechnic College, 518055, Shenzhen, China

Abstract: A self-developed ETL tool which is named Data Copy Tool (DCT) system is designed and realized under the background of the commercial bank non-site auditing. Based on SOA design methodology, with XML technology to achieve heterogeneous data sources, the paper proposes a method of using the configuration files to enhance the system scalability and to complete the ETL function more stably. DCT system can also be easily extended to the ETL tasks of other industry data warehouse.

Key word: SOA, extract-transform-load, data copy tool, off- site audit

INTRODUCTION

With the rapid development of the industry and the fierce competition, banks are faced with a number of potential risks. These risks are a serious threat to the financial security of commercial banks, which may affect their daily robust operation. Commercial banks require the technologies of using data warehouse technology urgently to extract and analyze the business data accumulated by the system, to automatically audit the business compliance and to monitor and warn the operational risks for banks.

One of the future direction of ETL tools is the application integration package. This paper presents a SOA-based ETL design ideas and develops an ETL tool-DCT (Data Copy Tool) to explore the ideas of integration the SOA's modular, high-cohesion, low-coupling and interfaces technics in ETL designs (El Akkaoui *et al*., 2013), which makes ETL towards SOA direction, that is, not to move the code but with something like "Software Configuration" idea to achieve changes in demand. This paper introduces a non-site audit system architecture design of a commercial bank first. The architecture is designed to support modular development, to integrate the common application packages. Furthermore, the paper elaborates the DCT key technologies of the development of structural design and the extended interface realizations.

OFF-SITE AUDIT SYSTEM

Off-site audit system is a system involving multiple services, multiple technologies, along with business processing, knowledge management and decision analysis. System needs huge works of data collection, data processing and data analysis.

Features and benefits of SOA: Service-Oriented Architecture (SOA) is a "service-oriented" design methodology with standards-based, coarse-grained, loosely-coupled software architecture. All the application functions are defined as callable, independent services, which communicate with precise-defined standard interface. The SOA architecture has features of good defined interfaces and independent loosely-coupled realization (Abdul-Manan and Hyland, 2013).

So with the underlying SOA-based architecture (Luo *et al*., 2012), we can guarantee to develop an entire loose coupling and flexibility ETL system architecture. No massive changes to other parts of the system are required in the need of system expansion to avoid duplicated development of the basic functional modules, which builds a better foundation for future expansion of the business logic.

Off-site audit system architecture design based on SOA: Bank systems have the features of high-efficiency, stability, scalability and advanced technique. It could achieve electronic banking supervision system and the establishment of the auditing knowledge database; improve the efficiency and quality of the audit work and to strengthen the risk prevention capability. According to the above objectives, combining SOA methodology, we design the logical architecture for the system shown in Fig. 1.

The system is divided into background subsystems and foreground subsystems.

The background subsystems are designed for data collections and data processing to achieve the automation of data collection; the foreground subsystems are to achieve the functions of the audit work. By separating the front and back subsystems and providing flexible variable system interfaces for back subsystems, the

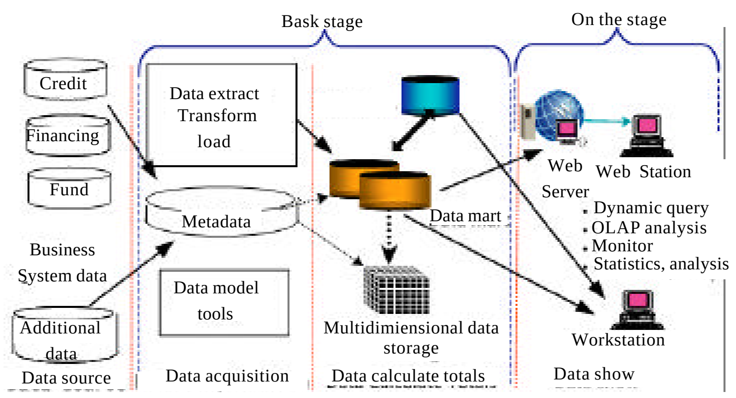


Fig. 1: The logical architecture of the off-site audit system

system achieves the independence of the underlying system, that is, it can connect business systems and data structures on different platforms without the need for program modifications, which adapts to the commercial Bank's existing business system running model.

The foreground subsystems include Web services and database application services. Web services face directly to users, all programs and services are implemented on a centralized server, users access the system functions through browsers. Database application services are responsible for the customer data collection, data processing and data presentation.

ETL is the core date process in the database application, which is the key to ensure the quality and the process capabilities of the entire system. In the following, I describe the design and implementation method of the SOA-based ETL tools (Data Copy Tool) developed by myself.

DCT DESIGN IDEAS AND FUNCTIONAL MODULES

SOA-based ETL architecture design: We propose that the development of ETL should rely on SOA, that is, it should not change the codes but should realize the requirement changes with something like “Software Configuration”. The flowing includes the advantages of doing so:

- To develop with standard. This uniforms the style of the development team and facilitates the future maintenance
- To lower the development costs. “Configuration" is much cheaper than "modifying codes”
- To reduce the maintenance costs significantly. Maintenance personnel even can do the maintenance without viewing codes

Therefore, in accordance with the project data collection requirements of the commercial banks’ non-site audit information system, based on the idea of SOA we proposes the modular ETL design architecture shown in Fig. 2.

Data access interface: The established Bank business systems choose many different database products. The data sources vary widely including database files, non-database data sources such as text files and so on. To make better scalability, the system adopts XML technology, uses XML language to describe the data source and achieves universal access interface to heterogeneous data sources. ETL programs simply read the XML tags in the configuration files to access the various data sources and establish the data connections.

The parametric configuration file: With the parametric methods and visible MapForce tool, the system achieves the dynamic management of metadata, such as software configuration of data sources and target databases, data tables, extraction period, the source data to target data transformation rules, data cleansing and so on. When the configuration files are re-set or changed, the real-time loop ETL programs will read the contents and update the changes to the ETL server buffer. Based on the contents of the buffer, ETL data extraction programs will start new tasks. This running method makes the system a good adaptive capacity.

ODS layer design: The audit system has to extract data from many business systems which have various data structures and various encoding; therefore, it is necessary to encode the data with unified encoding before the data come into the audit DW. We add an ODS (data store) layer. The data is collected into the ODS, then cleaned and converted in the ODS. After a series of treatments,

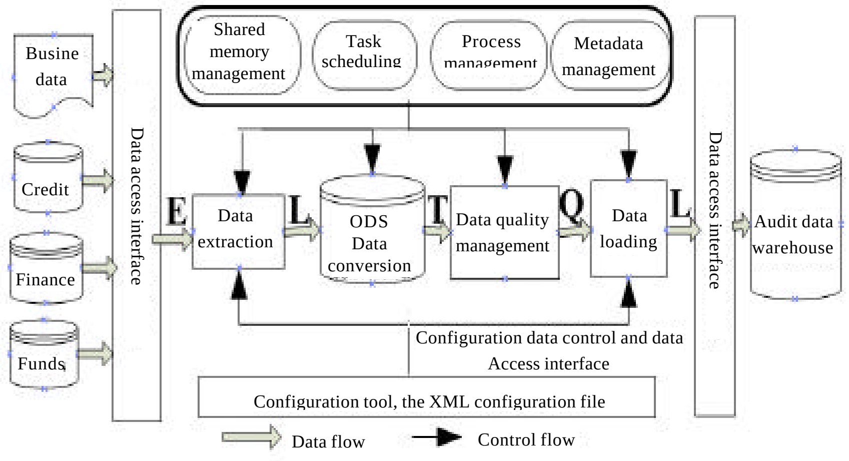


Fig. 2: ETL architecture

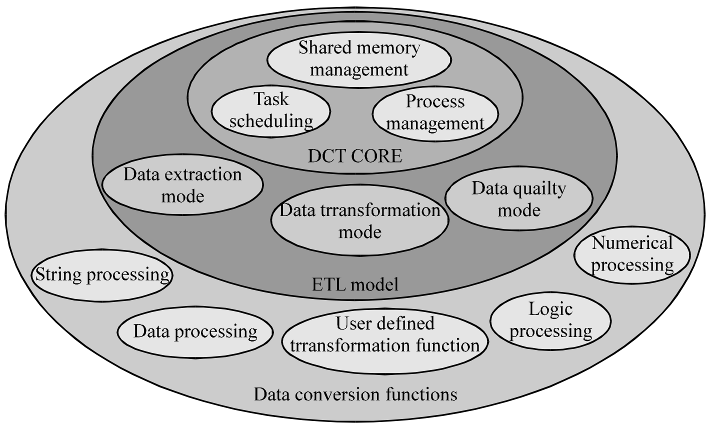


Fig. 3: DCT development structure

the quality of the data in ODS are significantly improved to fit the basic requirements of the auditing system.

During the design of ODS, most parts of the tables in ODS have their corresponding tables in the audit system with the same structures. Some structures of the tables in DOS cannot be the same with audit system due to special reasons. The structures of these tables are designed similarly with each other by increasing the redundancy data fields to make them basic same. The above designation makes the system more conducive to the generation of audit DW.

DCT function module design: DCT development structures are divided into shared memory and signal management, task scheduling, process management, data extraction, data loading, data conversion and data quality management module.

Shared memory and signal management, task scheduling, process management are the fixed core modules. Shared memory module manages the shared memory and signal management by creating static parameter settings. The function of task scheduling module is to read the tasks needed to be performed from the shared memory area, generate all the sequence of steps according to the data loaded into relational tables, execute each of these steps one by one until the end of the procedures. The three modules exchange and synchronize data via shared memory and signal management.

Function interfaces are provided for Data extraction, data loading and data conversion, data quality management. Other modules use the interfaces to call them. These other modules can be extended as shown in Fig. 3.

KEY TECHNOLOGIES OF DCT IMPLEMENTATION

DCT is an ETL framework. It uses common interfaces to access heterogeneous data sources through XML technology. It executes ETL tasks through the parameters stored in configuration files. For different applications, programmers can develop different functions to handling data extraction, data loading, data conversion and add these functions to the DCT framework. Due to the above reasons, the system has good adaptability and scalability. In the following, I describe the major technologies with which DCT achieves good scalability.

XML-based configuration files technology: Extensible Markup Language (XML) is a simple data storage language that can describe data structures with an open self-defined way. While describing the data content, it can also highlight the descriptions of the structures. In this way XML can reflect relationships of data. XML allows the application systems to exchange data easily without knowing the specific case of data representation format (Gu *et al*., 2007).

DCT configuration files use XML language. A master configuration file is created to manage the shared memory areas and the signals; a static routing table configuration file is created to define the network topology; a data-loading relational table configuration file is created to define ETL of data (i.e., how the data flow from one source to another source) and an task initiation and termination configuration file is created to define the batch commands to be executed before or after all the tasks. One fragment of the DCT master configuration file is shown in the following:

```
<DCT>
<IPCS comment="IPCS Parameter">
        <!—KEY in Shared memory area, KEY of signals -->
        <ShmFile>$PRJDIR/etc/DCT.SHM</ShmFile>
        <SemFile>$PRJDIR/etc/DCT.SEM</SemFile>
       </IPCS>
       ¡-…
<!-- Data loading relational table -->
<ELRalation>$PRJDIR/etc/etl.xml</ELRalation>
<!-- The maximum number of processes ruuning imultaneously -->
<MaxProc>10</MaxProc>
</DCT>
```

DCT configuration files use XML documents as the data exchange carriers to exchange data.

ETL model development mechanism: By developing ETL model (Song *et al*., 2013), we can make DCT support more applications. ETL development model (Zhang *et al*., 2012) can be summarized as the following steps:

- Develop ETL model handling codes in the corresponding index
- Register a new ETL model in the DCT system, including stating the function prototype, registering to the function array etc
- Use the new ETL model in the ETL configuration file

Below, I describe the details of a data extraction model to show the development processes of an ETL model.

Developing codes of a new ETL model: Each data extraction format corresponds to an extraction module. For each extraction module we design three function interfaces:

```
Int extractOpen(DCTCTRL *c, ETLITEM *e);
int extractFetch(DCTCTRL *c, ETLITEM *e, unsigned char **b, size_t *l,
ETLITEM s);
int extractClose(DCTCTRL *c, ETLITEM *e);
extractOpen () function is to counect to the database, open files or open other
data extraction source.
extractFetch () function is to read the data one by one from data sources and
write the data to the buffer. The parameter b is a pointer pointing to the data
buffer. The data buffer is allocated in the extractFetch() function itself but
must be released after the function call.
extractClose() function is to close the cursors and data sources, release
resources.
```

Moreover, because the same module in an ETL task (Song and Yan, 2010) may be used in multiple steps, we cannot use a static variable to store global information. Therefore, we must use the extended parameter structure variable and store it in the variable e-> ExtRun.

Each module can customize the structures related with the extended parameters. In order to allow all modules have similar program structures, after defining the extended parameter structures, we design to use the following macro definitions to unify the extended parameter names:

```
#define EXTPARSTR struct    extpar_e_septext_dbtb
```

Register a new ETL model: Register a new ETL model in DCT system, including stating the function prototype and registering to the function array.

- State the ETL model function prototype in the header file $PRJDIR/src//lib.src/etlfunc.h, for example:

```
/* Load the delimited text string into the database */
#define ETL_L_SEPTEXT_DBTB "L|DATABASE|SEP-TEXT"
int etlLSepText_DBTB_Open(DCTCTRL *, ETLITEM *);
int etlLSepText_DBTB_Put(DCTCTRL *, ETLITEM *,
unsigned char **, size_t *,ETLITEM);
int etlLSepText_DBTB_Close(DCTCTRL *, ETLITEM *);
```

- In the corresponding source codes, register the three ETL functions into the DCT function array. For example the code of registering Open() function with ETL_REGOPENFUN macro is shown in the following

```
ETL_REGOPENFUN(ETL_E_SEPTEXT_FILE,etlESepText_File_Open)
```

Configuration files using the new ETL model: With the new ETL model, all we have to do is to specify ELFunction element and its corresponding Extension element in etl.xml file. The main codes are as follows:

```
<ELFunction>L|DATABASE|SEP-TEXT|TRANSFER</ELFunction>
<Extension>
 <mfd remark="MapForce document name">
 $PRJDIR/etc/mfd/dcttest.mfd
 </mfd>
 <xsdpath remark=" Data structure definition document search path">
 $PRJDIR/etc/xsd/amis
 </xsdpath>
 </Extension>
```

Conversion functions development mechanism: DCT provides an array of data conversion functions and predefines some system conversion functions. For users or specific data conversion, users can define their own conversion functions in practice. For those user defined conversion functions, users have to comply with the designed DCT interface standards while writing and register the functions into the function array to use the functions in the DCT ETL tasks.

Design of the data conversion function interface standard: The function prototype designed for all data conversion functions in DCT is shown as follows:

```
int TRF_foo(int argc, char **argv);
Where:
argc: The number of parameters
argv: Parameter array.
Return value£º
0: Conversion function is executed successfully.
<>0: Conversion function execution error, the error information is written to the log file.
```

Statement of the function prototype: State the function prototype in header file $PRJDIR/src/lib.src/datatran.h£¬for example:

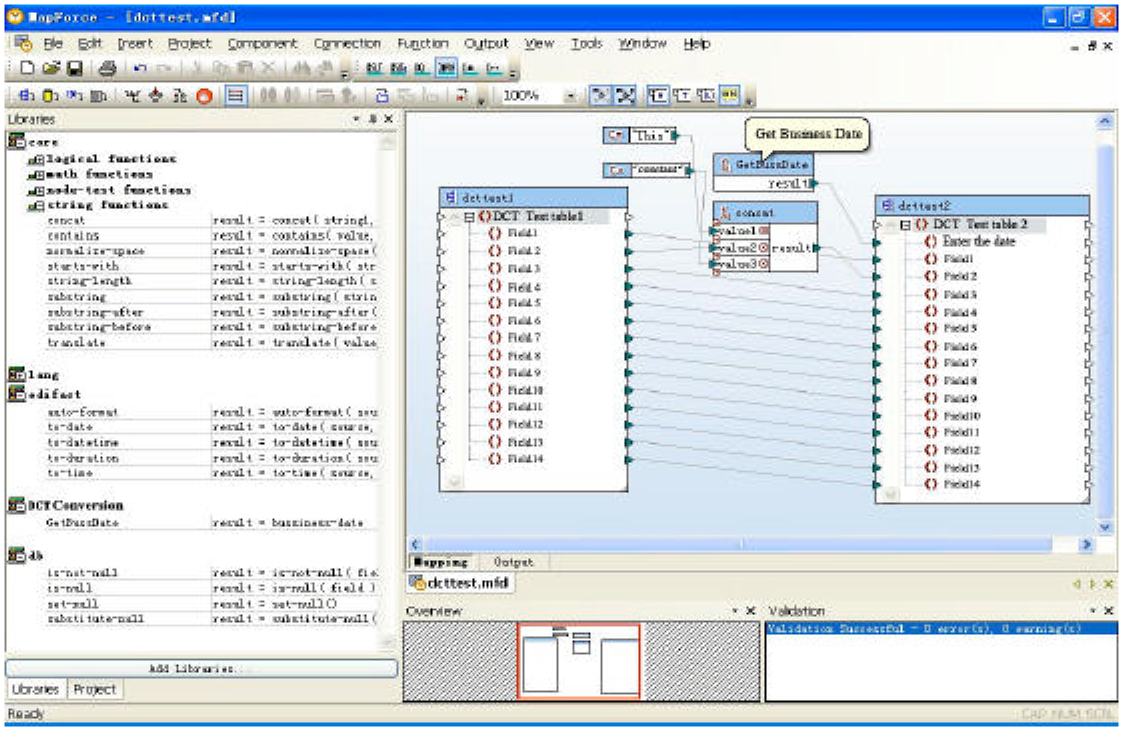
```
DAT_DCLTRANFUN(GetBussDate) /* get current business date */
```

Data conversion function array registration: The code of registering a data conversion function in dctInitDataTranFun is shown below:

```
DAT_REGTRANFUN("GetBussDate", GetBussDate)
```

Use a user defined data conversion function in mapforce: Define data conversion rules with MapForce first; then analyze the rules by DCT and call the appropriate data conversion functions in accordance with the conversion rules in the data conversion module.

First, import the function library into MapForce, Then, use the imported function in the conversion function definition, shown in Fig. 4.



```
core
  logical functions
  math functions
  node-test functions
  string functions
    concat               result = concat( string1,
    contains             result = contains( value,
    normalize-space      result = normalize-space(
    starts-with          result = starts-with( str
    string-length        result = string-length( s
    substring            result = substring( strin
    substring-after      result = substring-after(
    substring-before     result = substring-before
    translate            result = translate( value
lang
edifact
    auto-format          result = auto-format( sou
    to-date              result = to-date( source,
    to-datetime          result = to-datetime( sou
    to-duration          result = to-duration( sou
    to-time              result = to-time( source,
DCTConversion
    GetBussDate          result = business-date
db
    is-not-null          result = is-not-null( fie
    is-null              result = is-null( field )
    set-null             result = set-null()
    substitute-null      result = substitute-null(
```

```
Validation Successful - 0 error(s), 0 warning(s)
```

Fig. 4: Call the user defined conversation function

RESULTS AND CONCLUSION

The system has run stably in the recent three years. System errors and program crashes hardly occur. The time costs for data extraction, transformation and loading are less than 3 h day^{-1} and less than two days per month. The query response time for data traversal of a table with 60 million records in a database is 1 min. The system makes it possible for regulatory authorities to monitor the banking activities comprehensively, sustainably, efficiently and accurately.

In this study, based on SOA design methodology, a reusable ETL architecture design strategies are proposed and DCT which is a framework for ETL tool is implemented for the ETL part of off-site audit system of a commercial bank. With the modular design and common interfaces implemented by XLM technologies, the problems of complex data sources, data transfer efficiency and other issues are resolved. Using parametric configuration files to enhance the flexibility of the system and providing secondary development of standard interfaces make it much easier to develop new ETL models for new application and develop new data conversion module for new conversion rules. This design ensures the DCT system has better flexibility, system scalability and adaptability.

ACKNOWLEDGMENT

This study was supported by Natural Science Foundation of China (NSFC50505036). The authors also thank for the support by Science and technology project in Shenzhen (06KJce037).

REFERENCES

Abdul-Manan, M. and P. Hyland, 2013. A framework for assessing enterprise-wide SOA implementation readiness. Int. J. Intelli. Inform. Technol., 9: 21-37.

El Akkaoui, Z., E. Zimanyi, J.N. Mazon and J. Trujillo, 2013. A BPMN-based design and maintenance framework for ETL processes. Int. J. Data Warehousing Mining, 9: 46-72.

Gu, T.Z., J. Shen and X.H. Chen, 2007. Research on heterogeneous data integration based on XML. Appl. Res. Comput., 24: 94-96.

Luo, H., W. Zhou, D. Ye and J. Yu, 2012. A buffer-based parallel ETL data flow processing framework. Comput. Appl. Software, 1: 88-91.

Song, J., W.J. Hao and G. Chen, 2013. Research of distributed ETL architecture based on mapreduce. Comput. Sci., 40: 152-154.

Song, X.D. and X.L. Yan, 2010. Design ETL meta-model in datawarehouses. Comput. Simul., 9: 106-108.

Zhang, J., H. Lei, X.F. Tang and S.P. Meng, 2012. Study on the optimization design and implementation of ETL appilcation. Microelect. Comput., 29: 134-137.