Non-SQL 질의 데이터 서버 아키텍처

권기현*° Chakra Balayar* 천상호**

* 삼척대학교 정보통신공학과, ** 강원대학교 컴퓨터과학과
anyjava@empal.com, yeschakra@hotmail.com, sccheon@empal.com

An Architecture for Data Server of Non SQL Query

K. H. Kwon°* Chakra Balayar* S. H. Cheon**

* Dept. of Information & Communication, Samcheok National University
** Dept. of Computer Science, Kangwon National University

ABSTRACT

To develop enterprise architecture based distributed application needs consideration of various factors such as division of role between web-designer and software developer, defining entity and its usage, database connection and transaction processing. This paper presents DONSL(Data Server of Non SQL-Query) architecture that provides solution to above aspects through web-tier object modeling guaranteeing efficient transaction processing and performance between web-tier and DBMS through simplified usage of query logic property.

1. Introduction

Recently, researches related to the application of framework and pattern on any distributed system are getting wide interest [1][2][3].

The need is to maximize the interoperability, extendibility and reusability of the web-application making possible the division of once-written system into various classes to ease system upgrading and extension [5][6]. To accomplish this task, work division between web designer & software developer, sharing of entity within the system, automatic creation of SQL-query should be done properly[9].

In this paper, more effective and productive DONSL architecture is designed and implemented to replace existing WAS(Web Application Server). Chapter 2 describes N-tier structure and MVC framework. Chapter 3 proposes MVC Model based DONSL architecture. In chapter 4, we describe the implementation of DONSL architecture and its performance. Finally, we had conclusion.

2. Related Research

2.1 N-Tier Architectures(3-tier & Multi-tier,WAS-tier)

Browser, application server that is connected by servlet or CGI and database form 3-tier architecture which lacks solution to complex web application and processing time increases as user increases[4].

Multi-tier architecture-WAS tier solves the above weaknesses. But it requires the detail understanding of distributed architecture making object modeling complex which can lengthen development time, and can make maintenance handy.

2.2 MVC Framework & Entity: Roles and Weaknesses

MVC (Model-View-Controller) framework is the structure that minimizes the alteration effects through function encapsulation[10]. Model is expressed in terms of entity in distributed system. If entity needs modification, all other entity related items must be modified. And it's difficulty to implement presentation entity and DBMS entity separately.

These problems tighten the coupling between tiers and become obstacle on system development and maintenance. SQL query depends on the database. The changes in SQL query logic, entity and database affect all tiers.

2.3. MVC Model based Object Modeling

This architecture applies MVC concepts to web-tier in Web Application Server (WAS) using entity as Model, WorkBean as View and controller bean as Controller. It does the modeling of web-tier but WAS tier is handled by automatic transaction processing. Entity can remove tight coupling problem among tiers through common API.

3. DONSL Architecture based on MVC Model

3.1 DONSL Architecture

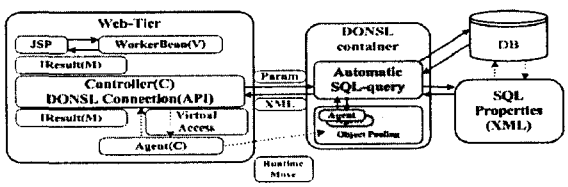

Fig. 1. Multi-Tier Architecture of DONSL

DONSL is composed of web-tier, WAS-tiered DONSL container and database. Web-tier uses IResult entity as Model, WorkerBean as View, and controller bean and agent as Controller. DONSL container connects object pool agent, executes the automatic SQL query, and provides communication among tiers on XML basis. In this architecture, programmer only develops XML property used from JSP, WorkerBean, agent and database. The advantages of this architecture are as follows:

- modeling from Web-tier
- globalizes entity(IResult)
- automatic SQL-query
- default agent to handle Flat Tx(API provides)
- addition of agent for handling complex Tx.

3.2 DONSL Distributed Transaction Specification

Transaction processing is done by making DONSL architecture accessible from web-tier through unit transaction and by including it in primitive SQL-query or unit transaction. Automatic processing of transaction unit is done by executing agent for container forwarding encapsulated transaction from DONSL container to agent.

3.3 DONSL Agent

DONSL agent is used for processing complex transaction. DONSL container forwards the encapsulated transaction object to agent and calls execution method of agent.

Ex.) Source Code of Agent Interface:

```
public interface IDonslAgent extends IObject {
public void unset();
public IResult execute(IUserTransaction ut)
 Throws SQLException;
}
```

3.4 Admin Tool

Admin Tool is the tool that manages all settings of DONSL-tier. Web-tier setting is done making accessible through unit Tx and are database setting, primitive SQL-query setting, transaction setting. It is used as developer's reference tool.

3.5 Development Roles

In DONSL, web-designer and developer handle JSP part where as WorkerBean, controller and agent are handled by developer alone. Developer also looks DONSL container connected with database.

3.6 Advantages of DONSL

First, One Side Object Modeling becomes possible in distributed environment. Second, it shortens developing time making Tx processing simple and it also makes error modification easy by converting(DAO class remove) SQL-query logics into SQL-properties. Third, error modification time can be shortened by converting object transmission among tiers into XML(entity class remove).

4. Prototype Implementation & Performance Evaluation

4.1 Prototype Implementation

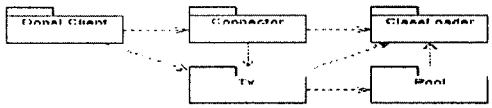

Fig. 2. Prototype Package

DONSL prototype can be divided into two parts, DONSL container and DONSL client. DONSL container is again composed of 4 packages as in the Fig. 2, connector package to communicate with client, Tx package to handle transaction, ClassLoader package to oversee dynamic class loading of agent and pool package to manage database connection and object resource.

In the Fig. 2, the arrow dotted line shows the dependency relation and the most important thing is weak dependency between ClassLoader and Tx package.

First, ClassLoader is implemented getting inheritance from J2SE API ClassLoader class since it is the weakest dependency package. Second, Tx package being the most core part of DONSL architecture manages the creation and processing of Transaction. Here is the class diagram of Tx:

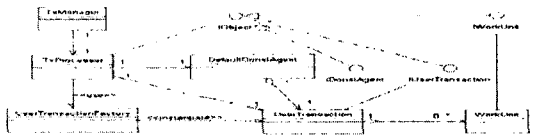

Fig. 3.Class Diagram of Transaction Processing

TxManager handles start and end of all unit Transaction as shown in Fig. 3. TxProcessor encapsulates transaction setting details of DONSL Admin Tool and passes UserTransaction to the Agent, and calls the agent's execution method. DefaultDonslAgent is implemented executing WorkUnit Collection that is included in UserTransaction using transmitted UserTransaction.

The relation between UserTransaction and WorkUnit is one-to-many. UserTransaction Class includes WorkUnit collection. As seen in Fig. 3, TxProcessor, DefaultDonslAgent, UserTransaction,

WorkUnit all commonly get inheritance from IObject..

4.2 Prototype Usage Procedure

The setting procedures of DONSL Admin Tool are: First, transaction list is registered. It is made on the basis of all transactions included in the project requirements giving unique name. The comment and details can be added later. Second, database is registered with unique name. Third, SQL is also enlisted with unique name, objective database, input parameter and type, SQL-query. Transaction details can be changed and agent code can be uploaded directly or after compilation.

The setting is completed after saving all Admin tool settings. And, client programming can be done almost similar like general JDBC programming.

4.3 Performance Evaluation

Performance Evaluation was done comparing with J2EE platform Java PetStore sample application. DONSL simply changed Java PetStore sample application into DONSL server accessible format.

For making overload reasonable, the following settings are done in the stress tool.

- Warm up Time: 1 minute
- Measurement Time: 5 minute
- Think Time: 5 ~ 35seconds(avg. 20 sec)

Measurements are taken after passing warm up time and user's think time to bring the best result.

Setting hardware and overload percentage properly, the test was done generating request from more than 50 virtual users till occurrence of Socket Error or Internal Server Error (HTTP Error code 500).

Here was the result obtained:

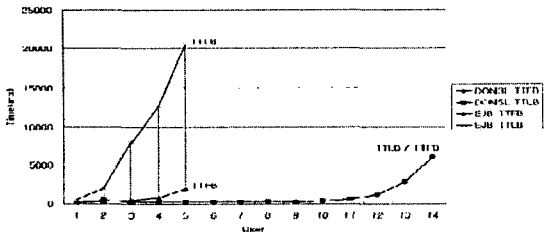

Fig. 4. Response Time Comparison Graph

- TTFB(Time to First Byte) : time from sending request till getting first 1 byte response
- TTLB(Time to Last Byte) : time from sending request and till getting whole response.

Analyzing the above graph it was proved that DONSL server yields the 5 times good performance when the users are 50 and if more, there is more significant result. The number of classes and code length per module was also measured for comparing productivity and, significant difference was seen due to agent. Hence, it eased programmer's coding job.

5. Conclusion

The rising complication in the web application has forced to improve MVC Model based Object Modeling. Hence, we proposed DONSL Architecture based on MVC Model to facilitate those concerns.

We classified DONSL Architecture into Web-Tier, WAS-tiered DONSL container and database. The developer only needs to develop XML property which is used from JSP, WorkerBean, agent and database which eases his job. And, DONSL architecture brought the efficient handling of complicated transaction through the usage of agent and hence, it was proved to be an ideal solution for complicated web application.

References

[1] R. Johnson, "Frameworks = Patterns + Components", Communication of ACM, vol. 40, Oct. 1997.

[2] F. Bushchmann, R. Meunier, H.Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture A System of Patterns", Willey and Sons, 1996.

[3] M. Jacyntho, D. Schwabe, G. Rossi, "A Software Architecture for Structuring Complex Web Applications", In International World Wide Web Conference(www2002), 2002.

[4] K. Iijima, J. Ivins, "An Alternate Three-Tiered Architecture for Improving Interoperability for Software Components", In International World Wide Web Conference(www2003), 2003.

[5] F. Marinescu and E. Roman, EJB Design Patterns Advanced Patterns, Processes and Idioms, Wiley and Sons, 2002.

[6] T. Fischer, J. Slater, P. Stromquist and C. Wu, Professional Design Patterns in VB.NET Building Adaptable Applications, Wrox, 2002

[7] D.Schwabe G. Rossi: An Object-Oriented approach to web-based application design. Theory and Practice of Obect Systems(TAPOS), Special Issue on the Internet, vol.4#4, pp.207-225, Oct. 1998

[8] Mowbray, Thomas J. and Ruh, William A. Inside CORBA: Distributed Object Standards and Applications, Addison Wesley, 1997.

[9] S. H Cheon, G. H. Kweon, H. J. Choi, "Developing a Automatic Components Creating System in Distributed Environment", Korea Digial Context, Vol. 2, 2001

[10] S Burbeck, "Application Programming in SmallTalk -80: How to use Model View Controller (MVC).http://st-www.cs.uiuc.edu/users/march/st-docs/mv.html. 1992

[11] G. Krasner, S. Pope, A Cookbook for using the model-view-controller user interface paradigmin small talk-80, Journal of OOP, 1(3), Aug/Sep 1998.