웹티어 오브젝트 모델링을 통한 non-SQL 데이터 서버 프레임웍 구현

정회원 권 기 현*, 천 상 호**, 최 형 진**

Implementation of Non-SQL Data Server Framework Applying Web Tier Object Modeling

Ki-hyeon Kwon*, Sang-ho Cheon**, Hyung-jin Choi** ***Regular Members***

요 약

엔터프라이즈 애플리케이션 개발을 위한 분산 아키텍처를 개발할 때는 여러 고려 사항 중에서 계층(tier)의 응집력(cohesion)을 높이고 계층간 연결 결합력(coupling)을 낮추기 위해 페이지 작성자와 소프트웨어 개발자의 역할을 명확히 분리하는 것과 비즈니스 로직의 단위가 되는 엔터티(entity)를 정의하고 데이터베이스 연결과 트랜잭션 처리에 엔터티의 사용 및 역할에 대해 정의하는 것이 우선적으로 필요하다. 이 논문에서는 DONSL(Data Server of Non SQL query) 아키텍처를 제시하여 이러한 문제점을 해결하고자 한다. 이 아키텍처는 웹 티어 오브젝트 모델링 방법을 사용하며 계층(tier)간의 결합도를 낮추고, 데이터베이스 연결에 반드시 사용되는 DAO(Data Access Object)와 엔터티를 효과적으로 분리하여 이러한 문제점을 해결 한다. 핵심 내용으로 DAO에서 엔터티 객체를 제거하는 방안을 통해 DAO 개발을 용이하게 하는 방법과 SQL 질의 자동 생성을 통해 트랜잭션 처리 자동화 방법, 그리고 트랜잭션 처리시 AET(Automated Executed Transaction)와 MET(Manual Executed Transaction)를 효율적으로 운용하는 방법에 대해 제시하고 시스템을 구현하였다.

Key Words : data access object, non-sql data server, AET(Automatic Executed Transaction)

ABSTRACT

Various aspects should be taken into account while developing a distributed architecture based on a multi-tier model or an enterprise architecture. Among those, the separation of role between page designer and page developer, defining entity which is used for database connection and transaction processing are very much important. In this paper, we presented DONSL(Data Server of Non SQL query) architecture to solve these problems applying web tier object modelling. This architecture solves the above problems by simplifying tiers coupling and removing DAO(Data Access Object) and entity from programming logic. We concentrate upon these three parts. One is about how to develop the DAO not concerning the entity modification, another is automatic transaction processing technique including SQL generation and the other is how to use the AET/MET(Automated/Manual Executed Transaction) effectively.

* 강원대학교 전자정보통신공학부 정보통신공학전공 (kweon@kangwon.ac.kr),
** 강원대학교 IT특성화학부(대학) 컴퓨터과학전공 (shcheon61@empal.com, choihj@kangwon.ac.kr)
논문번호 : KICS2005-10-405, 접수일자 : 2005년 10월 13일, 최종논문접수일자 : 2006년 3월 15일

Ⅰ. Introduction

There is significant increase in the researches relating application of framework and pattern on any system. in distributed environment[1-3]. The need to apply framework and pattern to any system development is for maximizing the interoperability, extensibility and reusability of the web application[4]. To maximize the reusability and to satisfy the interoperability, it is necessary to meet following conditions: First, the separation of role between page designer and page developer should be done. Second, the entity should be made common within the system. Third, the SQL query object that is needed to connect database must be created automatically. Fourth, transaction can be automatically processed by agent.

This paper is organized as follows. Section II presents MVC framework and object modeling. Section III describes non-SQL concept. Section IV presents DONSL transaction processing using agent. Section V discusses DONSL architecture based on MVC model, while section VI describes the implementation of DONSL architecture and its performance evaluation. The conclusion is in section VII.

Ⅱ. MVC Framework and Object Modeling

2.1 MVC Framework

MVC framework minimizes the alteration effects due to change in logic[5,6]. In distributed system, model is expressed in terms of entity and used for handling data[7]. Since entity related items must be modified whenever it needs modification. There is the problem to implement separately the presentation entity and DBMS entity which tighten the coupling between tiers. Since SQL query is always dependent on the database, DAO entity needs to be considered equally. The changes in SQL query logic or entity or database affects all related tiers.

2.2 Object Modeling

Object modeling based on MVC framework applies MVC concept to web-tier in web application server (WAS). It simplifies the modeling and makes WAS-tier transaction processing automatic. The entity can remove tight coupling problem among tiers through common API.

Ⅲ. Non SQL Concept

In non-SQL concept, a developer should not include SQL query logic in DAO though the common method to access any database is through DAO.

List 1. An Example of DAO

```
5 List getEmployeeList(String empno, name){
...
12 Statement stmt = con.createStatement();
13 String query="SELECT empno, name, address
   FROM EMP";
14 query += "where empno = " + empno + "";
15 query += "and name like '%" + name +
   "%'";
16 ResultSet rs = stmt.executeQuery(query)
17 }
```

The lines 13-15 in List 1 can be expressed in many different ways. If non-SQL concept is applied, we can remove these kinds of SQL query from DAO so that we don't need to reconfigure server whenever the query is reconstructed.

In the List 2, those SQL lines are replaced by single line i.e. function queryGenerator(). The developer only needs to write queryGenerator (param) function.

List 2. Removal of SQL Query.

```
1 public List getEmployeeList(Hashtable param){
2 Statement stmt = con.createStatement();
3
4 String query = queryGenerator(param);
5
6 ResultSet rs = stmt.executeQuery(query);
7 }
```

If the SQL query is "select empno, name, ad-

dress from EMP where empno='1111' and name='Kwon', it can also be expressed in following ways:

(1) "select empno, name, address from EMP"
(2) "select empno, name, address from EMP where name='Kwon'"
(3) "select empno, name, address from EMP where empno='1111' and name='Kwon'"

etc.

The above sample query can be divided into "select empno, name, address from EMP where 1=1", "and empno=111", "and name=Kwon". If the value of empno or name is null, then "and empno='111'" or "and name='Kwon'" needs to be removed.

Let's suppose, the SQL query is split and saved into XML as in List 3. '?' represents the part where parameter will be included. The order of '?' in 2, 3 lines and of 5, 6 lines must match. When SQL query is executed taking parameters(EMPNO, NAME), the automatic SQL query logic analyzes according to line and '?' and makes ready the next execution. After analyzing both parameters, it checks whether EMPNO parameter is NULL or not. If it is, it deletes line 2, similarly in the case of NAME too.

List 3. Generation of SQL query by Admin Tool.

```
1 <D-QUERY>select empno, name, address from
  EMP where 1=1
2 and empno='?'
3 and name='?'
4 </D-QUERY>
5 <D-INPARAM NO="1" TYPE="String">
  EMPNO</D-INPARAM>
6 <D-INPARAM NO="2" TYPE="String">
  NAME</D-INPARAM>
```

As in List 4, many SQL queries can be included into one transaction. DONSL container creates one transaction instance after analyzing the settings, and creates three SQL encapsulated instances for including in the transaction.

List 4. XML Setting of Transaction

```
1 <D-TX GNAME="bbs" NAME="DeleteBBS_TX"
  TYPE="required">
2 <D-AGENT>com.nosco.bbs.agent.BbsAgent
  </D-Agent>
3 <D-WORK TYPE="D-SQL">selectEtcOwner
  BBSSQL</D-WORK>
4 <D-WORK TYPE="D-SQL">deleteBBSSQL
  </D-WORK>
5 <D-WORKTYPE="D-SQL">selectDeleteStepSQL
  </D-WORK>
6 <COMMENT>Delete the article of BBS
  </COMMENT>
7 </D-TX>
```

The transaction management is done according to AET (Automatic Executed Transaction) and MET(Manual Executed Transaction) which are explained in detail in the next section. AET is used in DONSL by default. But, we used MET in this case. In List 4, there are three SQL queries and brings problem which to be executed first. MET is used to solve this problem which uses the result of first SQL as parameter to the second and similarly the third gets from second.

Ⅳ. DOSNL Transaction Processing using Agent

DONSL uses automatic managed transaction(AMT) where AMT manages automatically all the contents set in XML. The execution of transaction is done in two ways : AET(Automatic Execution Transaction) and MET(Manual Execution Transaction).

4.1 Automatic Agent Deployment

We suppose that DONSL server acts from remote for any web server. The programming is done in web server and the developed agent automatically moves to DONSL server. Since this deployment keeps inspection on changing of agent, it is effective to use only during project development but not after completion.

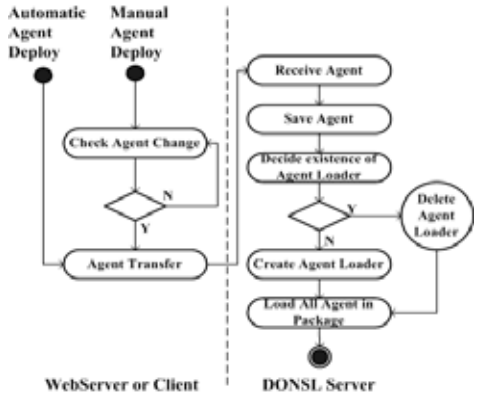

그림 1. AET와 MET 처리흐름
Fig. 1. Workflow of AET and MET

4.2 Manual Agent Deployment

The main difference is that it uses web admin tool provided by DONSL server to upload agent class. It becomes a problem if agents are more than normal. It is more effective after the project completion and can also reduce overload due to agent.

Ⅴ. MVC Model based DONSL architecture

5.1 DONSL Architecture

As in Fig. 2, DONSL architecture has web tier, WAS-tiered DONSL container and database. Web tier applies MVC concept and uses IResult entity as model, WorkerBean as view, and Controller and Agent as controller. The DONSL container connects an agent in the object pool and the agent executes the automatic SQL query, and provides communication among each tier on XML basis. In this architecture, the page designer and software developer develop JSP part but software developer alone develops WorkerBean, Controller, Agent, DONSL container and SQL properties(XML).

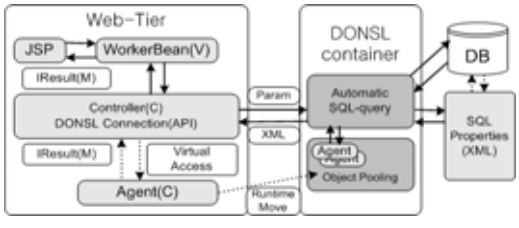

그림 2. DONSL 아키텍처
Fig. 2. DONSL Architecture

5.2 Advantages of DONSL

The advantages of DONSL architecture are as follows:

First, one side object modeling becomes possible in distributed environment and it shortens developing time making transaction processing simple and error modification easy by converting(DAO class remove) SQL-query logics into SQL-properties. Second, error modification time can be shortened by including object transmission among tiers into XML properties(Entity class remove). Third, it also supports distributed transaction and has advantages like performance maximization, WAS-tier development in low cost, low maintenance cost.

Ⅵ. Implementation & Performance Evaluation

6.1 Prototype Implementation

DONSL prototype can be divided into two parts, DONSL container and DONSL client. DONSL container consists of 4 kinds of packages. They are Connector package, Transaction package, Classloader package and Pool package. Transaction package being the core part of DONSL architecture manages the creation and processing of transaction.

TxManager handles starting and ending of all unit transaction as shown in Fig. 3. TxProcessor encapsulates the transaction setting details of DONSL administration tool item-wise. It passes UserTransaction to the agent, and calls the agent's execution method. DefaultDonslAgent is implemented executing WorkUnit.

The relation between UserTransaction and WorkUnit is one to many i.e. many jobs can be run in one transaction. It should be noticed from

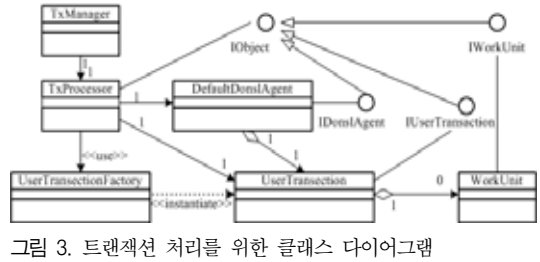

그림 3. 트랜잭션 처리를 위한 클래스 다이어그램
Fig. 3. Class Diagram of Transaction Processing

Fig. 3 that the TxProcessor, DefaultDonslAgent, UserTransaction, WorkUnit commonly get inheritance from IObject. And, connector package was designed using servlet which can execute DONSL container from any servlet container.

6.2 Prototype Usage Procedure

The followings are the setting procedure of DONSL admin tool:

First, transaction list is registered executing transaction menu at first which is made on the basis of all required transactions. Unique transaction name and comment should be given, and details can be added any time later. Similary, database and SQL details are enlisted. Then, unit transaction details are set. When the execution method of transaction SQL list is not simple, the developer defined agent must be implemented. At this time, transaction details can be changed after setting necessary agents from agent menu. The agent programming code can be uploaded directly or after compiling. In this way, DONSL server settings are completed. Only client programming is now required which can be done almost similar way as general JDBC programming. The following is the source code for accessing DONSL server from client(List 5):

List 5. Client Source Code

```
DSURL url = new DSURL("donsl:/test@localhost:8888:ds/Donsl");
DSConnection conn =DSConnectionFactory.getDSConnection( url, null );
ParameterString[] paramNames = { "COFFEE_NAME" }
NameString[] paramValues = { "java" };
DSStatement stmt = conn.createStatement();
stmt.setParams( paramNames, paramValues );
NameString txName = "Test1_TX";
stmt.execute( txName );

IResultSetHouse house = stmt.getResultSetHouse();
IResultSet result=house.getResultSet(0);
while(result.next()){
System.out.println(result.getString("COFFEE_NAME "));
}
```

6.3 Performance Evaluation

Performance evaluation was done comparing with J2EE platform Java PetStore[8] application. DONSL simply changed Java PetStore application into DONSL server accessible format. For making overload used in the test reasonable, the following settings are done in the Microsoft Web Application Stress Tool[9].

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

TTFB(Time to First Byte) : The time between sending request and getting first 1 byte response.

TTLB(Time to Last Byte) : The time between sending request and getting whole response.

For the best results, the actual test was done taking measurements after passing warm up time(1 min.) and each user's think time(5~35 sec.). After setting hardware and overload percentage properly, the test was done generating requests from more than 50 virtual users till occurrence of Socket Error or Internal Server Error(HTTP Error code 500). The result was obtained as in Fig. 4. Having analyzing the above graph we proved that the DONSL server yields 5 times good performance when the users are 50. When more than 50 there is more significant difference. In case of

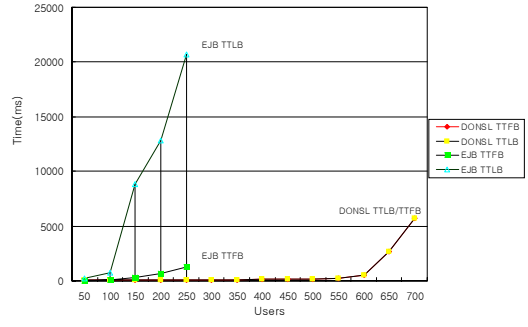

그림 4. DONSL과 EJB 응답시간 비교
Fig. 4. Comparison Graph of Response Time

J2EE environment, the measurement was not possible due to the occurrence of Socket Error when there are more than 250 users.

Ⅶ. Conclusion

In this paper, we have presented DONSL architecture which separates business logic and presentation logic using non-SQL data server framework. It also improves the productivity of complex web application and uses agent technology, supports distributed transaction. It removes SQL query from business logic and handles those queries by XML. Hence, it offers many advantages over traditional approach to web application development that relies solely on MVC model. Though DONSL architecture provided much relief to the enterprises who were looking for the solution to increasing complex web application, there is still a lot to do in order to improve broad acceptance, compatibility and standardization.

REFERENCES

[1] D. Schwabed, G. Rossidd, "An object-oriented approach to web-based application desing", *Theory and Practice of Object Systems (TAPOS)*, 4, 1998.

[2] R. Johnson, "Frameworks = Patterns+ Components", *Communication of ACM*, 40, 1997.

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

[4] D. C. Schmidt, "Experience using Design Patterns to Develop Reusable Object-Oriented Communication Software", *Communication of ACM*, 38, 1995.

[5] M. Jacyntho, D. Schwabe, G. Rossi, "A Software Architecture for Structuring Complex Web Applications", *Int. WWW conference*, 2002.

[6] K. Iijima, J. Ivins, "An Alternate Three-Tiered Architecture for Improving Interoperability for Software Components", *Int. WWW Conference*, 2003.

[7] S. H. Cheon, K. H. Kwon, H. J. Choi, "Developing an Automatic Components Creating System in Distributed Environment", *Korea Digital contents*, 2, 2001.

[8] Sun Java Software, "Java Pet Store", http://java.sun.com/reference/blueprints/, 2004.

[9] Microsoft Software, "Web Application Stress Tool", http://www.bridgeport. edu/sed/projects/cs597/Fall_2002/jishah/web_application_stress.htm, 2002.

[10] B. Goetz, "Java Theory and Practice : State Replication in the Web Tier", http://www-128.ibm.com/developerworks/java/library/j-jtp07294.html, 2004.

[11] 허미영, 이종화, 김용진, 진병문, "사이버교육을 위한 기반구조 시스템의 설계", *한국통신학회논문지*, 24(12B), pp.2225-2232, 1999.

권 기 현 (Ki-hyeon Kwon) 정회원

1993년 2월 강원대학교 전자계산학과 졸업
1995년 2월 전자계산학과 이학석사
2000년 8월 컴퓨터과학과 이학박사
2002년~현재 강원대학교 전자정보통신공학부 정보통신공학전공 조교수
<관심분야> 미들웨어, 임베디드시스템, 패턴인식

천 상 호 (Sang-ho Cheon) 정회원

1986년 2월 서강대학교 수학과 졸업
2002년 2월 컴퓨터과학과 이학석사
2005년 2월 강원대학교 컴퓨터과학과 이학박사
1985년~1989년 한국화약 전산실
1989년~1997년 (주)선경유통
1997년~현재 (주)오픈시스템서비스 대표이사
<관심분야> 컴포넌트시스템, 분산시스템, 미들웨어

최 형 진 (Hyung-jin Choi) 정회원

1982년 2월 영남대학교 물리학과 졸업
1987년 2월 일본동경 공업대학 정보공학과 공학석사
1990년 2월 공업대학 정보공학과 공학박사
1990년~1991년 ETRI선임연구원
1991년 현재 강원대학교 전자계산학과 교수
<관심분야> 인공지능, 화상처리, 패턴인식, 컴퓨터그래픽