웹에서의 LCL 화물 정보 시스템 구현

이달상* · 김태화** · 반상우**
* 동의대학교 기계산업시스템공학부 교수
** 동의대학교 산업공학과

LCL Cargo Information System on Web

Dal-Sang Lee* · Tae-Hwa Kim** · Sang-Woo Ban**
* Division of Mechanical and Industrial System Engineering, DongEui University
** Dept of Industrial Engineering, DongEui University

요 약

LCL은 컨테이너에 화물을 적재할 때 한 화주가 컨테이너의 공간을 전부 사용하는 FCL과 달리 여러 명의 화주가 각자의 화물을 한 컨테이너에 적재하는 방식을 말한다. 기존의 LCL 화물 선적은 화주의 요청에 의하여 선주가 Forward를 통하거나 이전에 거래하던 화주를 통하여 화물을 모집하여 남는 CBM(Cubic Miter)를 채우게 된다.

본 논문에서는 이들을 웹 상에서 서비스하는 LCL 화물 정보시스템을 제안한다. 이 시스템은 화주와 선주의 정보를 시스템을 통하여 다른 화주나 선주에게 제공하는 한편 이들 정보들을 바탕으로 LCL 선적할 화주의 결정에 필요한 적절한 정책은 시스템의 Agent들에 의해서 이루어진다. 그리고 데이터베이스의 접근에 있어서도 CORBA를 이용하여 이 기종의 시스템들에 분산된 데이터베이스에 쉽게 접근할 수 있도록 하여 보다 많은 정보를 바탕으로 시스템을 운영할 수 있다.

1. 서론

우리나라에서는 1996년부터 해양수산부의 항만정보운영시스템과 관련업계와의 항만 물류업무처리 효율화를 위해서 전자문서교환(Electronic Document Interchange)을 도입하였다. 그리고 현재 물류망(KL-Net)을 중심으로 관련업계와 관련기관들의 시스템들이 연결되어있다[1]. 외국에서는 일찍이 1980년대 중반에 데이터베이스를 구축하여 모든 수출입정보와 화물재고관리 서비스를 비롯한 수출입 화물과 관련한 모든 정보를 제공하고 있다.[2] 그러나 화주가 수출을 위해 선사에 의뢰하려면 각 선사의 항로와 선적마감시간, 컨테이너 등을 검색해야하는 불편이 따른다.

배에 선적하는 화물은 FCL(Full Container Load)과 LCL(Less of Container Load), 그리고 Bulk로 나뉜다. FCL은 선적마감시간 전에 선사에 의뢰해서 배에 선적해 정해진 기한에 보내면 되지만 특히 LCL에 있어서는 먼저 배에 선적된 컨테이너의 화주의 요청에 의해 먼저 LCL을 만들고 그리고 선주가 남은 컨테이너의 CBM을 화주나 Forward에게 의뢰하면 화주가 선사에 의뢰하거나 Forward는 다른 화주들을 모집한다. 하지만 화주의 입장에서는 선사나 대리점에 CBM이 남아있는지의 여부에 대해서는 의뢰를 하기 전에는 알 수가 없다. 본 논문에서 제안하는 LCL 화물 정보 시스템은 선사가 화주의 결정에 필요한 정보들과 화주의 정보들을 적절히 검토하여 화주를 성공적으로 결정하여 주며 화주는 원하는 항로와 선적마감시간, 그리고 화물을 선적할 컨테이너를 검색할 수

가 있다. 데이터베이스의 접근에서 CORBA를 이용하여 각 선사나 화주의 데이터베이스에 동시에 접근하여 화주가 원하는 정보를 다양하게 검색할 수 있으며 많은 정보를 바탕으로 하므로 화주의 선정에도 보다 성공적인 결정을 내릴 수 있다.

2. LCL 화물 정보 시스템

하나의 컨테이너에 여러 화주의 화물을 모집하는 방법에는 2가지로 처리될 수 있다. LCL 화물 정보 시스템은 이들 방법들을 모두 웹을 통해 제공할 수 있도록 하는 시스템이다.

2.1 화주로부터 웹 상에서 직접 신청을 받는 방법

배에 화물을 선적한 화주는 선주에게 다른 화주의 화물을 선적해 달라는 요청을 하면 선주는 LCL 화물 정보 시스템에 접속하고 인터페이스를 통하여 항로와 잔여 CBM에 대한 정보와 가격 정보 등을 입력하게 된다. 이러한 정보를 바탕으로 화주들은 잔여 CBM이 자신의 화물을 선적하는데 적당한지, 가격은 적절한지 등을 결정하여 화물에 대한 정보를 인터페이스를 통해 입력하게 된다. 이렇게 입력된 화주의 정보들은 데이터베이스에서 관리되어지고 시스템은 신청한 화주의 화물이 컨테이너에 선적이 가능한 화물인지에 대한 정보를 확인한다. 그리고, 빠른 시간에 적당한 화주들을 선정하게 된다. 그림 1은 LCL 정보시스템 중의 화물 정보인터페이스와 잔여 CBM 정보 인터페이스를 통하여 서로 간의 정보를 교환하는 것을 나타낸다.

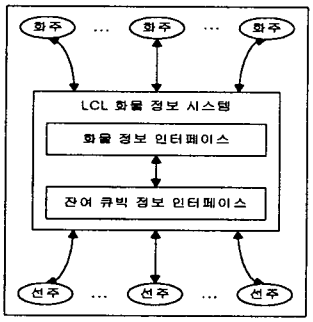

그림 1 시스템을 통한 화주와 선주간의 정보 교환

2.2 화주가 LCL 신청을 위해 미리 등록하는 방법

화주는 선적할 자신의 화물이 하나의 컨테이너에 대한 전체 CBM 보다 작고 화물이 거의 손상되지 않는 화물이라면 LCL 선적을 위하여 LCL 화물 정보 시스템의 화물 정보 인터페이스를 통하여 화물 정보를 입력하여 미리 신청을 하게 된다. 이렇게 신청된 화주와 화물의 정보는 데이터베이스에 정정되게 되고 선주로부터의 LCL을 신청할 화주의 모집 신청이 오게 되면 이들 화주들 중 선정하게 된다.

3. 웹 상의 분산 데이터베이스 화물 정보

LCL 화물 정보 시스템에서는 화주나 선주로부터 정보를 입력받아 이들 정보를 일정의 데이터베이스에서 관리하게 된다. 그리고 이들에게 정보를 제공해 주기 위해서는 웹 상에서의 데이터베이스의 연동이 필요하다. 그리고 이들 정보들 중에는 각 선주나 화주 측에서 제공하는 정보들을 이들 각자의 데이터베이스에서 직접 가져와야 하는 경우도 있다. 하지만 이들은 그 운영 시스템이나 데이터베이스가 서로 각기 다르다. 그러므로 이들을 통합할 수 있는 기술이 필요하다.

3.1 웹 상에서의 데이터베이스 연동 방식

웹 상에서 데이터베이스를 연동하는 기술로 초기 사용하였던 CGI(Common Gateway Interface) 방법은 모든 클라이언트의 데이터베이스 접근과 결과 전달에 있어 HTTP 서버를 통하여야 하므로 질의 처리에 드는 비용과 부담이 크다. 또한 CGI는 세션 관리 기능을 구현하기 어렵게 하기 때문에 사용자가 데이터베이스에 접근할 때마다 새로 데이터베이스에 로그 인을 수행하게 되어 질의 응답 시간이 느리다.

이러한 문제점을 해결하기 위하여 Java를 이용하는 방법으로써 애플릿의 JDBC(Java Database Connectivity)와 데이터베이스를 직접 연결하는 ODBC(Open Database Connectivity)사이의 드라이버를 이용하여 질의, 응답하는 구조로 2-tier, 3-tier, ..., n-tier 등의 구조를 가진다[6,7,8]. 2-tier 방식은 클라이언트에서 오직 브라우저를 통해 데이터베이스에 접근하는 구조이다. 그러나 이러한 구조는 접근하고자 하는 데이터베이스에 대한 함

수를 클라이언트가 포함하고 있어야 함으로 애플릿의 크기가 커지고, 따라서 다운로드받는 기간이 증가하게 된다[5,9].

3.2 분산 데이터베이스 연동

앞에서 언급하였던 3-tier나 n-tier의 구조가 이에 해당한다. 이 구조는 클라이언트와 서버사이에 CORBA 등의 미들웨어를 두는 구조로 이는 데이터베이스 접근을 위한 함수나 라이브러리 등을 포함하지 않음으로써 2-tier 구조에서의 문제점을 해결할 수 있다[4,5]. 즉 데이터베이스에 접근하는 함수들을 객체로 구현함으로써 서로 다른 플랫폼에 위치한 이종의 데이터베이스에 대한 접근이 가능하게 된다. 각각의 데이터베이스에 접근하는 서버 객체들을 만들고 이들을 관리함으로써 클라이언트는 각각의 데이터베이스에 대한 API 및 승인이 없더라도 CORBA를 통해 클라이언트 객체가 서버 객체를 호출함으로 원하는 질의를 전달하여 분산 환경 하에 산재된 데이터베이스로부터 질의 응답을 얻을 수 있다. 이렇게 분산 환경의 데이터베이스로부터 정보를 직접 가져오게 됨으로 많은 정보들을 바탕으로 한 정책이 이루어져 보다 성공적인 결과를 기대할 수 있으며, 신속한 응답 시간을 얻을 수 있다. 그림 2는 LCL 화물 정보 시스템에서의 분산 데이터베이스의 연동 구조이다.

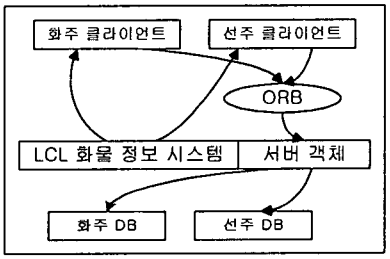

그림 2 클라이언트들이 분산 DB에 접근하는 형태

3.3 서버 객체

서버객체는 각기 접속하는 데이터베이스에 대한 연결하고 질의를 전달하여 응답을 가져오는 API로 되어져 있다. 객체들을 구분하면 다음과 같다.

- LCL 시스템 DB 연결 객체들
- 선주 데이터 베이스 연결 객체들
- 화주 데이터 베이스 연결 객체들

객체들은 서로 접근하는 데이터베이스나 목적에 따라서 그 형태가 틀려진다. 예로 Oracle 데이터베이스에 접근하고 JDBC 드라이브로 Thin 드라이브를 사용한다면 데이터베이스에 접근하는 API는 다음과 같다.

```
try {Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" ); }
catch(ClassNotFoundException ex){}
  String url = "jdbc:msql://203.241.200.180:1880/LCL";
  Connection con = DriveManager.getConnection (url, uid, pass);
...
```

4. 화주를 선정하기 위한 정책 Agent의 연동

앞에서 언급하였던 LCL 화물 정보 시스템에서의 컨테이너에 선적할 화주를 결정하는 필요한 정보를 제공받는 방법은 LCL 선적 요청 이후 화주의 정보 입력과 사전에 신청한 화주의 정보를 이용하는 방법 그리고 데이터베이스로부터 직접 정보를 이용하는 방식 등이다. 이들 방법들은 배타적으로 행해지는 것이 아니라 비어있는 CBM에 대한 최대한의 선적을 목적으로 동시에 고려되어 진다. 때문에 각각의 방법들에서 얻어진 정보를 바탕으로 하는 정책은 어떠한 정보에 더 많은 비중을 두느냐와 그 상황에 따른 정책을 어떻게 적절히 변화시키느냐가 중요하다. 이러한 정책의 결정과 변화를 위해서는 각각의 Agent들의 역할이 필요하다.

Agent들의 구성은 다음과 같은 Agent들로 이루어진다.

- 각각의 인터페이스로부터 정보를 수집하는 Agent들
- 분산 데이터베이스에 접근하는 서버 객체로 부터 정보를 전달받아 관리하는 Agent
- 정보들을 바탕으로 정책을 결정하는 Agent

이들 Agent들 간의 통신을 위해서는 Agent 통신 표준안인 ACL(Agent communication Language)로서 KQML(Knowledge Query and Manipulation Language)을 사용하며 Agent들 간의 중재를 위해 Facilitator들 두게 하여 서로간의 통신을 돕는다

[3,10]. Facilitator에는 에이전트들의 이름 및 그 에이전트가 할 수 있는 기능과 주소를 등록하게 된다. Facilitator는 어떠한 Agent가 문제를 요청했을 때 그 문제를 해결함에 있어 가장 적합한 Agent를 메타 지식을 이용해 선택하여 요청한 에이전트에게 알려주는 역할을 한다.

실제 시스템에서 사용되는 화주 Agent와 Facilitator 간의 KQML의 메시지 전달의 예를 들면 다음과 같다.

```
(ask-one :sender CargoAgent :receiver facilitator
        :reply-with LCL1 :language KQML
        :content(com_name INS,
                cargo_name shoes,
                cubic_num 5,
                destination germany,
                shipping_date 2001-4-27))
(tell : sender facilitator :receiver CargoAgent
        :in-reply-to LCL1 :language KQML
        :content(com_name INS,
                (cargo_name shoes,
                (state inquiry))
```

그리고 이러한 Agent들간의 통신을 통한 결정 시간은 화주의 대기 시간을 줄이기 위해 짧아야만 한다. 그림 3은 Agent들 간의 통신을 통한 협력하는 구조를 보이고 있다.

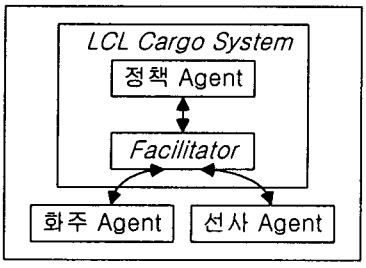

그림 3 각 에이전트간의 통신

5. 결론

기존의 LCL 선적에서는 문서를 통하여 화주나 Forward들 간의 정보전달이 이루어져 화주의 결정에 대해서 정보들의 내용을 바탕으로 한 정확한 정책을 할 수 없으며 이로 인해 이러한 결정이 성공적이라 말 할 수 없다. 그리고 이들 정보들에 대한 관리 또한 어려웠다. 본 연구에서 제시한 LCL 화물 정보 시스템은 모든 정보의 흐름을 네트워크를 통하여 시스템 내에서 처리되므로 정보의 관리가 효율적으로 이루어 질 수 있으며 이들 정보들을 바탕으로 한 Agent들의 협력을 통해 보다 성공적인 결정을 내릴 수가 있다. 그리고 무엇보다 각각의 분산된 데이터베이스로부터 정보들을 바로 가져올 수도 있기 때문에 실제 사용자들을 통한 정보입력을 대신할 수 있고 보다 많은 정보들을 바탕으로 하여 성공적인 결정을 기대할 수 있다.

참고 문헌

[1]최형림, "수출입 컨테이너화물 통합데이터베이스 구축", 한국해운학회지, Vol. 31, 2000, pp. 157-174

[2]박남규, 손형수, 최형림, 이태우, "항만물류에서의 원스톱서비스 시스템구현 방안", 한국해운학회지, 제28호. 1999, 6, pp. 127-151

[3]Pattie Maes, Robert H. Guttman, Alexandros G. Moukas. "Agents that Buy and Sell: Transforming Commerce as we Know It", Software Agents Group. MIT Media Laboratory, Submitted to the Communications of the ACM, 1999

[4]"VisiBroker for Java programmer's Guide Version 3.2," Visigenic Software, Inc., 1998.
"CORBA: catching the next wave", availble from http://developer.netscape.come:80/docs/wpapers/corba/index.html.

[5]Robert Orfail, Dan Harkey, "Client/Serve Programming with JAVA and CORBA", 2nd Edition, wiley, 1998.

[6]"Java Media and Communication APIs Intergration", available from http://java.cun.com/marking/collateral/media.html

[7]Core Java, 2nd Edition, Sun Soft Press,1997.

[8]A. Hobbs, "Teach yourself Database Progrmming with JDBC in 21 DAYS", sams net, 1996.

[9]박 재 현, 한 상 만, " ObjectWeb 기반의 시스템 통합 기술," available from http://cs.woosong.ac.kr/study/web/8/objectweb.html.

[10]김현수 외 "인터넷 환경하에서의 부품제조업체를 위한 판매 에이전트 개발", 한국정보시스템 학회, 논문지 제 7권, 2호, 1998년, pp215-233.