International Journal of Grid and Distributed Computing
Vol. 3, No. 3, September, 2010

A Study on Cooperation System design for Business Process based on XMDR in Grid

SeokJae Moon, GyeDong Jung, YoungKeun Choi
Department of Computer Science and Engineering,
Kwangwoon University, South Korea
{msj8086, gdchung, yhchoi}@kw.ac.kr

Abstract

Interoperability between legacy systems is essential for various processes to be effectively operated in terms of cooperation in business. A process, however, may include various queries, and thus data interoperability based on queries might not be properly executed due to problems of heterogeneousness. This paper proposes a cooperation system for the business process based on XMDR in Grid. The proposed system solves the problem of heterogeneousness that may take place regarding interoperability of queries in a XMDR-based business process. Heterogeneousness in an operation of a business process may involve metadata collision, schema collision, or data collision. This can be handled by operating a business process by making use of XMDR-based Global Query and Local Query.

Keywords: *Business Process, XMDR, Heterogeneous, Cooperative, Interoperability.*

1. Introduction

As a way of operating various processed, a technology called business process is proposed[1,2]. For a process needed for cooperation in a business environment to be operated effectively, interoperability between legacies is vital. Legacy systems distributed individually, however, have different goals, and thus it might be difficult to operate them cooperatively. Besides, a service-based enterprise data integration is essential to operate processed between legacy systems in a business environment, but the processes might not be designed properly for data integration or might cause problems of heterogeneousness due to data interoperability. Heterogeneousness can be divided to schema collision and data collision in the semantic classification in terms of metadata information. Schema collision involves the semantic process of collision to semantics, structures, and expressions between database schemas, while data collision to units, formations, and validations between instance values. This paper proposes the XMDR-DAI[4] based system as a method for a business process to be effectively operated cooperatively. XMDR-DAI(Data Access & Integration) enables interoperability between legacy systems needed for data integration. In addition, collision between different units of metadata information among queries in the business process, that is, schema collision, is handled by making use of XMDR[5]. XMDR, in this paper, consists of global schemas that designate as the standard schemas used in legacy systems, and uses them to solve collision problems between units of metadata information upon business process execution in connection with local schemas. This paper states the definition of XMDR, GQBP and LQBP in Chapter 2, and the cooperative system design and execution by means of XMDR-DAI in Chapter 3.

Chapert 4 compares the proposed systems with others, and Chapter 5 presents conclusion of the study.

2. Definition of XMDR and GQBP, LQBP

XMDR[5] is a technique to save relational database metadata in the object-oriented DB to solve the collision problems in the data integration. In other words, this is a system to combine MDR and ontology and integrate the data to solve the problem of collision of distributed data[7]. Thus, XMDR in this paper consists of MSO(MetaData-Semantic Ontology), InSO(Instance-Semantic Ontology), MLoc(MetaData-Location), and MDR(MetaData Registry). MDR consists of GS(Global Schema) and LS(Local Schema)[4]. This paper includes LQBP(Local Query Business Process) and GQBP(Global Query Business Process) with the inner queries of a business process as the basis for XMDR[4]. GQBP involves GQquery(GQ:Global Query) based on the global schema in a business process logic while GQ is classified to SELECT, INSERT, and UPDATE. It is defined by the generation rules according to ANSI. LQBP includes LQquery(LQ:Local Query) in which GQBP is reformed based on the local schema according to the conversion rule in connection with GQ in a business process logic. LQ as well is classified to SELECT, INSERT, and UPDATE. It is defined by the generation rule according to ANSI. GQBP adopts GQ generation rules 1,2, and 3 as defined in.

3. Proposed System

The system proposed in this paper enables the business process necessary for cooperation to be effectively executed. To this end, data integration and interoperability services are provided. **Figure.1** shows the XMDR-based system for cooperation, which consists of Client Zone, Data Service, XMDR-DAI, Data Integration Service, Legacy DBS, and Analytic DBS. The major roles are as follows:

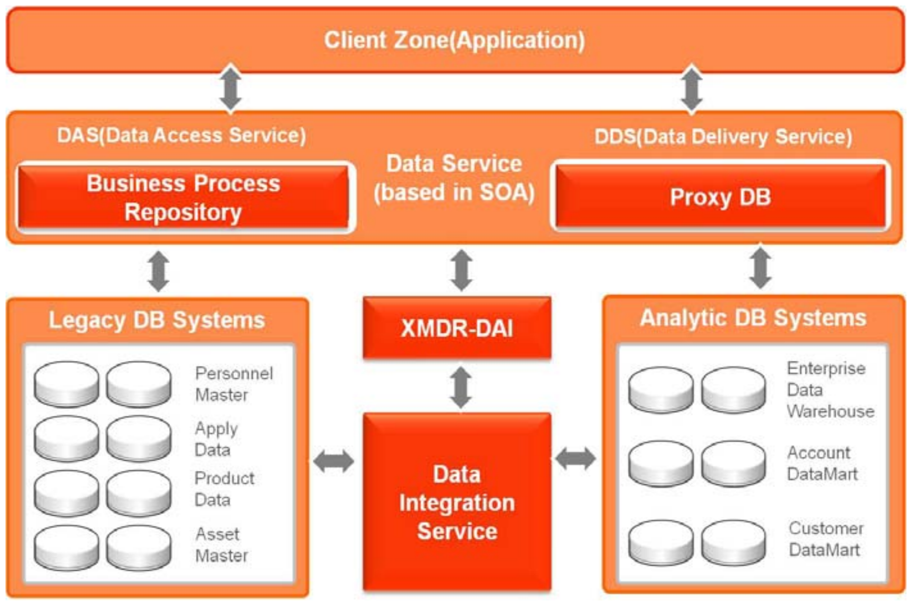


Figure.1. A Cooperative System by Means of XMDR-DAI

- Clinet Zone: On This layer is an application area where a business process can be formed by means of a user interface.
- Data Service: This layer defines and executes a data service based business process, and stores the result data after the execution.
- XMDR-DAI: This element is the key part of this system that can solve collision problems of metadata information produced while the business process is being performed, and enhances data interoperability in the process between legacy systems by converting GQBP to LQBP.
- DIS(Data Integration Service): This element acquires the system access, performs transaction execution, and collects data so that the business process delivered in XMDR-DAI can be executed in the legacy DBS and Analytic DBS as the mediator between XMDR-DAI and DBS.
- Legacy DBS, Analytic DBS: This is an area where there are existing legacy DB and Analytic DB.

3.1. System execution process

In Figure.2, the Client requests BP(Business Process) of the BP Agent necessary for cooperation. BP Agent searches for information registered in BPR(Business Process Repository), and returns the selected BP information to the Client. The Client that received BP information prepares GQBP and delivers it to the DQP Agent. DQP Agent confirms whether there is FROM in the query after parsing GQBP, and executes the following step upon finding any FROM phrase. It inquires the Proxy Agent of generating a Temp Table if there is no FROM phrase. Then the DQP Agent requests the XMDR Agent of Global Meta information based on the parsed GQBP information. XMDR Agent returns the requested Global Meta information and related Local Meta information to the DQP Agent.

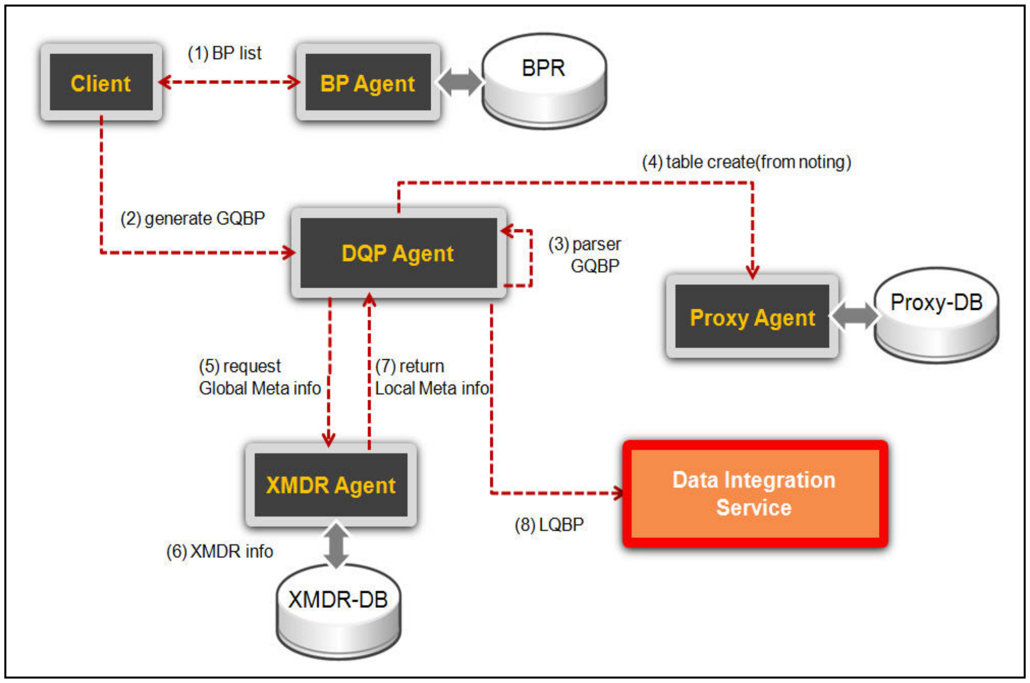


Figure.2. Execution Process 1

Afterwards, the XMDR Agent refers to the returned Local Meta information and delivers it to DIS after remaking it LQBP. **Figure.3** shows that the DQP Agent delivers LQBP to the BP

Analyzer Agent. BP Analyzer Agent delivers to Connector Agent the connection information for LQBP parsing and DBS access.

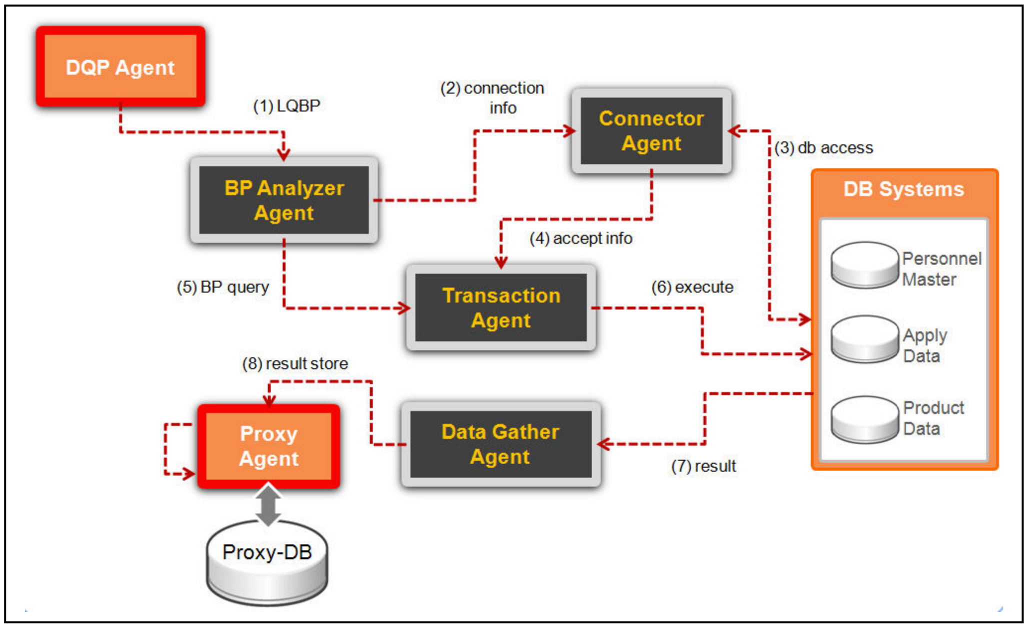


Figure.3. Execution Process 2

The Connector Agent acquires the DBS access with regard to LQBP by means of the connection information, and delivers it to the Transaction Agent. BP Analyzer Agent delivers the BP queries in the parsed information to the transaction agent upon receiving DBS access from the Connector Agent. Transaction Agent executes the execution queries by means of BP queries and access. Upon the completion of executing the BP queries, the execution results (data) are collected by the Data Gather Agent, delivered to the Proxy Agent, and stored in the Proxy DB.

3.2. Configuration of XMDR-DAI(Data Access and Integration)

BP Agent: When the business process requested by the user exists in BPR, the process is searched and the BP_list is provided accordingly. However, when the business process is to be prepared by the user, the global schema information provided through the XMDR Agent is utilized so that a newly defined business process can be adopted in a template form. Figure.4 shows the BP Agent execution algorithm. When the Input is (BP_list, *id) the parameter, the existing process is used by means of BPR_serach() when it is found to be in BPR. When it is not in BPR, however, BP_list parsing and extraction of parseTable, parseField, and parseValue follow:

```
ALORITHM: BP_Agent
INPUT: request of BP_list
OUTPUT: LQBP
   BEGIN
```

```
      IF (BP_list exit in BPT) THEN   //BPR: Business Process Repository
        BP := BPR_search(BP_list);
        return BP;
      ELSE parser(BP_list);
        gTable := XMDR_Agent(parserTable, 0);
        gField := XMDR_Agent(parserField, 1);
        gValue := XMDR_Agent(parserValue, 2);
        GQBP := GQBP_generate(gTable, gField, gValue, BPformat);
        BPR_reg(*GQBP, *id); //register BRP
      END
    END
```

Figure.4. BP Agent Algorithm

The extracted values call XMDR_Agent, and they are provided with the global schema information. Then GQBP_generate() generates GQBP by means of global schema information and BPtemplate, and registers it by calling BPR_reg(). The generated GQBP is returned to the user.

XMDR Agent: Figure.5 is an XMDR Agent execution algorithm. For Input, (itemList, handle) parameter values are inserted, and the hadle is divided into 0,1,2, and 3 to extract the table, field, value, and loc schema information from XMDR from XMDRExtract().

```
ALORITHM: XMDR_Agent
INPUT: request of itemLists, handle
OUTPUT: XMDR information
  BEGIN
       CASE handle OF //metadata info
            0 := xmdrTable = XMDRExtract(XMDR, 0);
               return;
            1 := xmdrTable = XMDRExtract(XMDR, 1);
               return;
            2 := xmdrTable = XMDRExtract(XMDR, 2);
               return;
            3 := xmdrTable = XMDRExtract(XMDR, 3);
               return;
  END
```

Figure.5. XMDR Agent Algorithm

DQP Agent: Figure.6. shows the DQP Agent execution algorithm, which needs GQBP as the input, and GQBP is parsed by parser(). The results are divided to GQtable, GQfield, GQvalue, and GQloc respectively. The separate schema information calls createTable() to generate a table in the Proxy DB to save the GQBP execution results. Each GQtable, GQfield, GQvalue, and GQloc extracts LQtable, LQfield, LQvalue, and LQloc, which are part of the schema information needed to generate LQBP after XMDR_AGENT() performs the mapping with the local schema information. The extracted local schema information is conversed to LQBP by GQ2LQ(), then generated in an XML document form by SpliteXML(), and delivered to each legacy system.

```
ALORITHM: DQP_Agent
INPUT: GQBP
OUTPUT: LQBP
  BEGIN
      token[n] : = parser(GQBP);
      FOR i:=1 TO i<=n DO
           CASE token[i] OF
              table[i] := GQtable[i] : = token[i];
                  return;
              field[i] := GQfield[i] : = token[i];
                  return;
                      value[i] := GQvalue[i] : = token[i];
                  return;
                      loc[i] := GQloc[i] : = token[i];
                  return;
           END
       END
       gsTemp := (GQtable[i], GQfield[i], GQvalue[i], GQloc[i],id);
       createTable(gsTEMP, 1); //proxy-db table create
       LQtable := XMDR_Agent(GQtable, handle);
       LQfield := XMDR_Agent(GQfield, handle);
       LQvalue := XMDR_Agent(GQvalue, handle);
       LQloc := XMDR_Agent(GQloc, handle);
       LQBP := GQ2LQ(LQtable, LQfield, LQvalue, LQloc);
       SpliteXML(LQBP2XML(LQBP); //send to legacy systems

   END
```

Figure.6. DQP Agent Algorithm

Proxy Agent: Proxy DB is a temporary saving area for the executed data. Figure.7 shows the Proxy Agent execution algorithm, and with (resultXML, gsTEMP) parameter as the input value. ResultXML is the result of executing LQBP in the legacy while gsTEMP is the table information to be generated in the Proxy DB that includes the global schema delivered from the DQP Agent. The two elements - resultXML and gsTEMP - are divided by the y value. When the y value is 1, create_tabel() is called to generate a table in the Proxy DB while when it is 0, the result of processing the resultXML document in DOMProcoess() is saved in the Proxy DB. Then the user is informed of the result.

```
ALORITHM: Proxy_Agent
INPUT: LQBP execute result data(resultXML, gsTEMP, e, y);
OUTPUT: notice to UI;
  BEGIN
    IF (y!=null) THEN
      create_table(gsTEMP);
      return;
    ELSE
       IF (e!=null) THEN
         resultXML := DataGather();
```

```
            DOMProcess(resultXML);
            **return** true; //notify UI
          ELSE
            return false;
          END
      END
  END
```

Figure.7. Proxy Agent Algorithm

3.3. Configuration of DIS(Data Integration Service)

Data Integration Service (DIS) proposed in this study consists of four elements: BP Analyzer Agent, Connector Agent, Transaction Agent, and Data Gather Agent. The definition, functions and algorithms of each Agent are as below:

BP Analyzer Agent: It analyzes LQBP sent from DQP Agent of XMDR-DAI and legacy DB accessinfo and query. Fig.8 shows BP Analyzer Agent Algorithm by which input LQBP in entered as a parameter. Each parser parses LQBP and then divides accessinfo and LocalQuery. Accessinfo calls Connector_Agent and then obtains legacy DB access right, which is sent to Transaction_Agent, and delivers LocalQuery as a parameter, so it can work.

```
ALORITHM: BPAnalyzer_Agent
INPUT: LQBP
OUTPUT: local-db connection info, localQuery
  BEGIN
        //parser DB access info, query info
        accessInfo := parser(LQBP, id);
        localQuery := parser(LQBP, id);

        local_conndb := Connector_Agent(access_info);

        //execute local query
        Transaction_Agent(local_conndb, localQuery);

  END
```

Figure.8. BP Analyzer Agent Algorithm

Connector Agent: It receives legacy DB access right using DB information sent from BP Analyzer Agent. Fig.9 describes Connector Agent Algorithm, which obtains returned legacy DB access right based on access_info parameter from BP Analyzer Agent using LegacyServer.CreateObject().

```
ALORITHM: Connector_Agent
INPUT: access_info
OUTPUT: local-db connection info, localQuery
  BEGIN
        strconn := parser(access_info);
        dbconn :== LegacyServer.CreateObject();
        dbconn.ConnectionString = strconn;
        conndb := LegacyDB.access(dbconn);

           return conndb;

   END
```

Figure.9. Connector Agent Algorithm

Transaction Agent: It carries out processes using BP query and local DB access right. Fig.10 is Transaction Agent Algorithm by which input (local_conndb, LocalQuery) parameter from BP Analyzer Agent is entered. It calls out legacyTransExecute() using this parameter and connect to actual legacy DB to transact the query. The transaction process continues while there is a local query in LQBP. The results (data) of transactions of all queries are sent to DataGather_Agent.

```
ALORITHM: Transaction_Agent
INPUT: local_conndb, LQBPQuery
OUTPUT: query execute result
   BEGIN
        strconn = parser(local_conndb);
        strquery = parser(localquery);

        WHILE (query Execute) DO
        BEGIN
            TRY
               Execute_result := legacyTransExecute(strconn, strquery);
               DataGather_Agent(execute_result, legid);
            EXCEPT
               queryTransException.error.message;
            END
     END
```

Figure.10. Transaction Agent Algorithm

Data Gather Agent: It collects business process results (data) and sends them to Proxy Agent. Fig.11 is Data Gather Agent Algorithm by which results of local query by Transaction Agent are collected using RecordsetResult() function and sent to Proxy Agent.

```
ALORITHM: DataGather_Agent
INPUT: query execute result, legid
OUTPUT: recordset result
   BEGIN
          resResult := new RecordsetResult();
          resResult := execute_result;

          Proxy_Agent(resResult., legid, 0, 1);

          return true;

  END
```

Figure.11. Data Gather Agent Algorithm

3.4. Sequence Diagram

In this chapter, we explain sequences of each Agent at XMDR-DAI and DIS. As shown in Fig.12, Client requests necessary BP list to BP Agent, which then finds relevant process and returns the BP list to Client. Then, BP Agent asks to XMDR Agent for Schema info and delivers any returned Schema info to Client. Client creates GQBP using BP list and Schema info and delivers it to DQP Agent. DQP Agent uses GQtable and GQfield from Schema info where the delivered GQBP is parsed and requests Proxy Agent to create Table. Now DQP Agent obtains LQ Schema info returned from XMDR Agent using GQ Schema info and then converts GQBP to LQBP, which is sent to DIS.

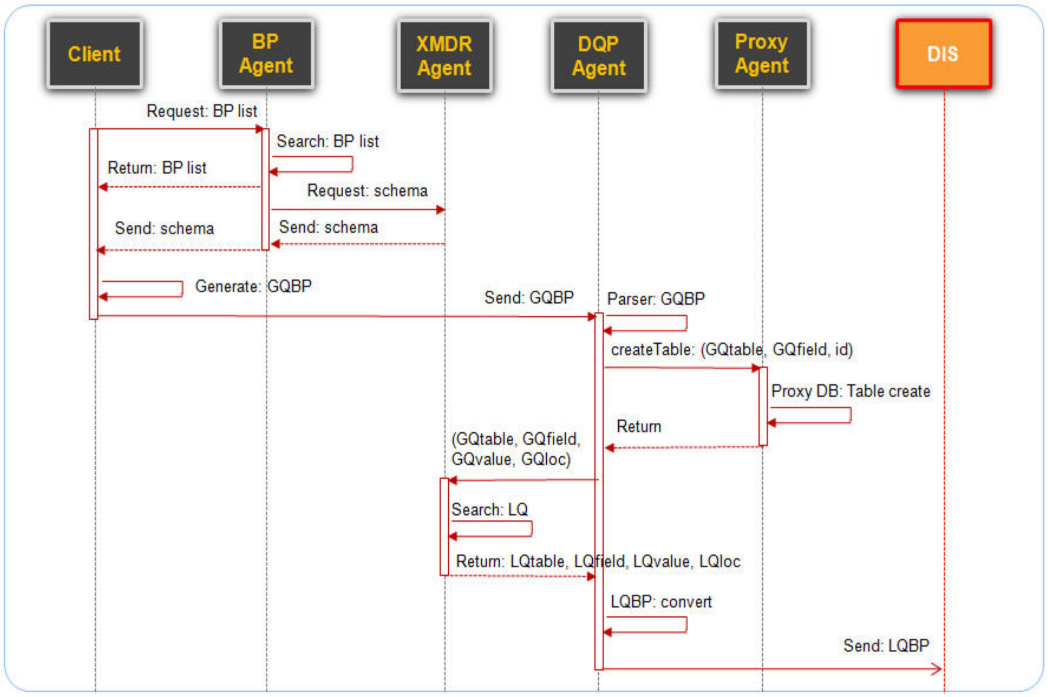


Figure.12. Diagram 1 of Sequence between Agents

Fig.13 shows how LQBP is processed, once delivered. LQBP sent from DQP Agent as in Fig. 12 is analyzed by BP Analyzer Agent. From the results of the analysis, Connection info is delivered to Connector Agent, and access info is returned from DB System upon request. Then, BP Analyzer Agent delivers LQBP query to Transaction Agent and carries out relevant transaction at DB Systems on the basis of DB access. The results of LQBP are gathered by Data Gather Agent and sent to XMDR-DAI.

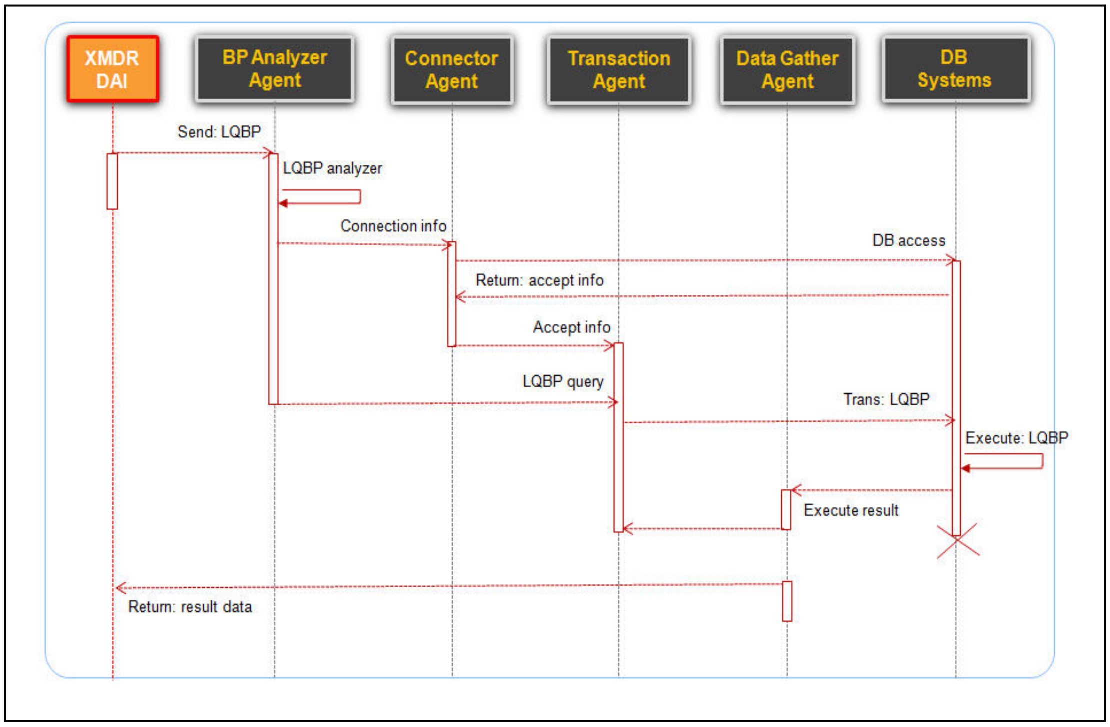


Figure.13. Diagram 2 of Sequence between Agents

4. Comparison of Systems

Table 1 summarizes the results of comparison of existing OGSA-WebDB[10] and Oracle ODS[11]. In this study, the proposed collaboration system is compared in terms of business process support, interoperability type, data integration application technology, data exchange technology, solution to heterogeneousness, data scalability, and data consistency.

Table.1. Comparison of Systems

Item	Oracle ODS	OGSA-WebDB	XMDR-DAI
Business process support	Support process services based on master data	Support query-based process services using web service	Support process services using GQBP and, LQBP
Data integration application technology	Mediator type	Wrapper type	Mediator, Agent type
Data access and exchange	Data Hub	OGSA-GDS	XMDR-DAI

technology			
Solution to heterogeneousness	Use of data standardization and consistency	Use of meta data schema	Use of XMDR-based schema
Data scalability	Easy to expand using uniformed data collection system	Easy to expand using meta data schema	Easy to expand using XMDR-based global schema
Data consistency	Modify data and maintain consistency using Data Hub	Modify data and maintain consistency using GDS	Maintain consistency of modified data on the basis of XMDR Setup

In regards to system interoperability, first, Oracle ODS and OGSA-WebDB support MD (MaterData)-based process services and query-based process services using web service. In terms of data exchange technology, Oracle ODS enables data access and exchange using Data Hub-based MD, while OGSA-WebDB provides access to data using Globus middleware-based[13] WebDB. However, the proposed system supports business processes for data exchange and sharing and allows for data access and exchange using XMDR-DAI at legacy system. Second, when it comes to data interoperability, Oracle ODS solves heterogeneousness by standardizing data, while OGSA-WebDB solves syntactical and structural heterogeneousness using metadata schema. In addition, Oracle ODS maintains consistency of modified data using Data Hub, while OGSA-WebDB uses queries without FROM clause at GDS to maintain online data modification. Meanwhile, XMDR-DAI tackles the heterogeneousness of data syntax, structure, and meaning using XMDR and uses GQBP and LQBP to maintain consistency after data modification and transfer.

5. Conclusion

This paper proposes a cooperative system for an XMDR-based business process in Gird. This system is advantageous in that the user does not need to consider the metadata collusion when a business process is executed for cooperation between legacy systems. Besides, additional legacy systems do not need to modify the local schema for data access to other legacy systems in consideration of data interoperability and cooperation. However, it has yet to seek the extension to corporate systems such as ERP, EAI, and DataWareHouse so that it can be executed for integration as the backbone of the workflow process.

References

[1] Smith, H., Fingar, P.: Business Process Management. In: The Third Wave. MK Press (2003)

[2] Indulska, M., Chong, S., Bandara, W., Sadiq, S., Rosemann, M.: Major Issues in Business Process Management. In: ACIS 2006 (2006)

[3] Dustdar, S., Treiber, M.: Integration of heterogeneous web service registries – the case of visr. WWW Journal (2006)

[4] Moon, S., Jung, G., Choi, Y.: XMDR-DAI Based on GQBP and LQBP for Business Process. In: AST 2010 Proceedings, pp. 72–85 (2010)

[5] Keck, K.D., McCarthy, J.L.: XMDR: Proposed Prototype Architecture Version 1.01(February 2005), http://www.xmdr.irg

[6] Arjuna, et al: Web service business activity framework (2005), http://specs.xmlsoap.org/ws/2004/wsba/

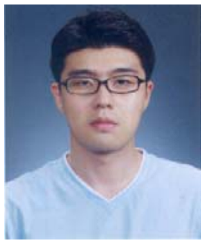
[7] ISO/IEC IS 11179, Information technology -Specification and standardization of data elements (2003)

[8] Fred A. Cummins, "Enterprise Integration: An Architecture for Enterprise Application and Systems

Integration", Wiley;1st edition, pp. 496 (February 1 2002).
[9] Eyhab Al-Masri, Qusay H. Mahmoud, "Interoperability among Service Registry Standards", Published by the IEEE Computer Society, pp. 210-220 (June 2007).
[10] Said Mirza Pahlevi, Isao Kojima, "OGSA-WebDB: An OGSA-Based Systems for Bringing Web Database into the Grid," itcc, Vol.2, pp. 105-109 (2004)
[11] Corinne, B, Marty, M, Cellan, M, Olaf, M, Andrew, P, Kiho, Y, "Building the Operational Data Store on DB2 UDB Using IBM Data Replication, WebSphere MQ Family, and DB2 Warehouse Manager",(2001), http://publib-b.boulder.ibm.com/Redbooks.nsf/redbooks/
[12] Andrew White, David Newman, Debra Logan, John Radcliffe, "Mastering Master Data Management", Garther, (2006)
[13] Globus Toolkit, http://www.-unix.globus.org/toolkit/.12

Acknowledgment This present has been conducted by the research grant of Kwangwoon University in 2010.

Authors



SeokJae Moon received the BS degree in computer science from Korea national open University, Seoul, Korea, in 2002, the MS degree in computer software from Kwangwoon University, Seoul, Korea, in 2004. Now, he is a PhD candidate in the Department of Computer Science and Engineering at Kwangwoon University, Seoul, Korea. His current research interests include eXtended metadata registry (XMDR), business process, distributed computing, grid computing, interoperability, cooperation systems.

GyeDong Jung received the BS, the MS, and PhD degrees from the University of Kwangwoon, Seoul, Korea, in 1985, 1992, and 2000, all in computer science. Now, he is a Professor in the institute of Information and Science Education, Kwangwoon University, Seoul, Korea, His current research interests include metadata registry(MDR), mobile agents, web service, grid computing, XML.

YoungKeun Choi received the BS degree from in mathematics education from University of Seoul, Seoul, Korea, in 1980, the MS and PhD degrees in computational statistics from Seoul University, Seoul, Korea, in 1982 and 1989. Now, he is a Professor in the Department of Computer Science and Engineering at Kwangwoon University, His current research interests include object oriented design, mobile agents, interoperability, parallel processing, ontology.