UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

BUILDING DATABASE COORDINATION IN P2P SYSTEMS USING ECA RULES

Albena Roshelova

Jan 2004

Technical Report # DIT-04-002

Building database coordination in P2P systems using ECA rules

Albena Roshelova

DIT – Dept. of Information and Communication Technology
University of Trento, 38050 Povo, Trento, Italy

albena.roshelova@dit.unitn.it

Abstract

Recently, data integration systems and peer database management systems that attempt to model and integrate data in a peer-to-peer (p2p) environment have attracted the attention of researches. Such systems give opportunities to the local relational database management system to exchange data with other nodes in a p2p environment. The databases systems in p2p are completely autonomous, heterogeneous and independent, each maintaining its own data. We would like to use these databases to answer complex queries that go beyond the keyword searches. To accomplish this, we use database coordination as managing semantic interdependencies among databases at runtime. We define database coordination in fours basic notions: Interest Group, Acquaintances, Correspondence Rules and Coordination Rules. The work below is concentrates on implementation issues of the coordination rules. The coordination rules specify under what conditions, when and where to propagate queries in a decentralized environment. We are representing our current solution of the coordination rules implementation for data exchange in p2p based on ECA rules technology.

1. Introduction

An interesting issue nowadays is database management on top of decentralized network [5],[7],[8]. In a p2p environment, there is no central authority, only individual computers that are able to connect and communicate with any computer on the network. Every peer is responsible for the management of its own database and has no control over the other nodes in the system, which can be totally heterogeneous. There is no global schema and there is no restriction in the DBMS. Queries are recursively propagated over the network to some or all database nodes and results are collected and sent back to the client. Nodes are connected with links in any arbitrary way. The link between nodes enables a node to query another node. A node is an acquaintance of another node only with respect to a query. The acquaintance node knows how to propagate a query, to propagate result back and how to reconcile them with the results coming from the other acquaintances. The goal of database management over p2p is to support coordination among autonomous databases by offering propagation and distributed query processing.

In this environment it is getting more and more critical to develop methods for building systems that combine relevant data from many databases and present them in a form which is comprehensible for users. However, existing p2p applications support mostly file exchange and they do not deal with data management issues yet. The peers can join and leave at any time, which bring insecurity of database management systems. In p2p environment is possible that one of the nodes will not be available far any number of reasons such as services or due to software/hardware failure, etc. Each database stores data in a specific format and any field type translation processes must be coded into this solution. Building a global schema for all databases, or just those of acquainted databases is difficult task, while acquaintances keep changing.

Our work is focused of how we solve the problem of interoperability between different databases management system in decentralized environment. This project examines variants of manipulation of data stored on several pre-existing, autonomous and heterogeneous local database systems. In order to adhere to the p2p methodology the database management systems will be interacting between them with assistance of *database coordination definitions* [16]. The role of the coordination is the ability nodes to manage effectively, at runtime, semantic interdependencies among databases in a decentralized, distributed and collaborative manner. The underlying in p2p settings is on coordinating database management systems rather than the integrating their database schemas.

In this paper, we present the coordination rules mechanism as a basis for transitive propagation of queries through a chain of nodes. The nodes in the p2p network can use coordination rules to specify under what conditions, when and where to propagate queries or updates. A query can trigger one or more coordination rules that can lead to consequent propagation. Also, we define the coordination rules like a base for recursively decomposing the query in subqueries which are translated with respect to the databases of acquaintances. They can be used to describe cross-database views and cross-databases constraints. We implement the coordination rules as *Event-Condition-Action* (ECA) rules, which are part of an active database system technology. An Active DBMS provides a mechanism for the declaration of rules, often referred to as the knowledge model or the rule language, and a mechanism for the execution of the rules, often referred to as the execution model or rule execution semantics. The Active database systems, as opposed to “passive” ones, are able to recognize specific situations in the database where they react automatically, without an explicit external request.

The reminder of the paper is organized as follows: Section 2 presents related works. In section 3, shortly explains specific distributed database management architecture in p2p network. Motivation examples illustrate the requirements for building coordination rules in section 4. Section 5 draws Coordination Rules implementation and finally, the paper is concluded and remarks future work in Section 6.

2. Related work

Previous relevant work peer database management system is presented in [1] and [5] by introducing the Local Relation Model (LRM) as a data model specifically designed for p2p applications. Authors introduce the first prototype of building distributed architecture for p2p environment. LRM assumes that the set of all data in a p2p network consists of local (relational) databases, each with a set of acquaintances, which defines the p2p network topology. For each acquaintance link, domain relations define translation rules between data items, and coordination formulas defining semantics dependences between the two databases. The main goals of the data model are to allow for inconsistent databases and to support semantic interoperability in the absence of a global schema. The proposed work is described at a very high-level of details, which is introduced in section 3.

In the Hyperion project [12], the authors draw on architecture for peer database management systems (PDBMSs). They envision a PDBMS as conventional DBMS augment with a p2p interoperability layer, where this layer implements the functionality required for peers to share and coordinate data without compromising their own autonomy. Here, authors focus on the specification and management of the logical metadata that enables data sharing and coordination between independent, autonomous peers.

Another exciting and very recent development is discussed in [11] where Peer databases are presented as stand-alone and independent databases containing local data and a set of ECA rules that are used to exchange data among p2p environment. They also describe algorithmic issues related to establishing and abolishing acquaintances. Each Peer database is managed by a peer data management system (PDBMS).

In [17] is introduced other approach for query management in p2p through a *mediator system*. It is presented a P2P mediation architecture as an important requirement and that composable mediators can be implemented efficiently through query processing techniques. The focus on this work is on query processing methods to realize composability in the Peer Mediator

System architecture in an efficient way that scales over the number of mediators. Contributions of the paper consist of an investigation of the interfaces and capabilities for peer mediators, and the design, implementation and experimental study of several query processing techniques that realize composability in an efficient and scalable way.

3. Distributed Database Management Architecture in P2P network

Two Logical Architectures of Local Relational Model have been outlined so far. The first level is described in detail in this section. The second level architecture has been recently proposed in [6]. The architecture is extension of the first level as open up Query Manager and JXTA Layer.

In the Figure 1, a Local Relational Model is presented that interacts with databases through the p2p network [5]. The p2p network here consists of open-ended number of peers, where each peer has a local database, and extra layer, called p2p layer, or also LRM layer. However, the LRM layer interacts with the local database and interfaces it with the p2p network. Also, the LRM layer has the p2p functionality on the Local Database (LDB).

The LRM architecture has the following scenario: The User Interface (UI) module allows users to submit queries. Therefore, the queries will be answered by the local databases and the peer databases to receive the results and messages from the other nodes and to control the other modules of the LRM Layer. Query Manager (QM) and Update Manager (UM) are responsible for query and update propagation. The Wrapper module here, is a translation layer between QM, UM, and LDB.

White arrows show that the communication between peers and inter-module communication is realized by using XML messages. Communication between the Wrapper module and LDB is realized through DBMS. The main functionalities for coordination p2p databases are implemented within QM and UM and assisted by using the following notions: Interest group, Acquaintances, Coordination rule, Corresponding rule which are introduced in detail below.

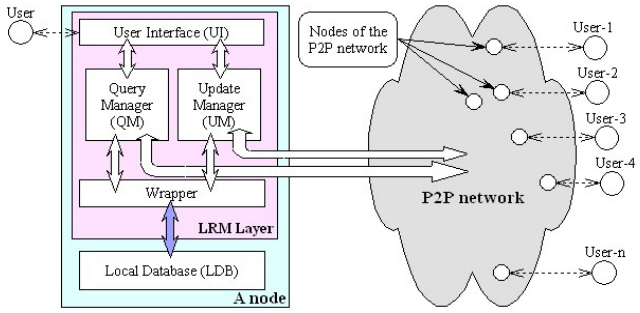

Figure 1. First Level Architecture.

From the figure above *Interest groups* are set of nodes, which are able to answer queries about a certain topic. Also, a node can belong to multiple groups. The notion of a group is introduced with the main goal of computing for any given input query, the Query Scope (QS) - the set of nodes, where a query should be propagated. The group satisfies two complementary requirements. First, groups deal with the complexity and the high number of nodes in the p2p network. A node searches for one or more group according to their topic. This means that input query should be associated with one topic. Second, groups are used to compute a bound on the number of nodes in the query scope, therefore guaranteeing termination. Each node has a Group Manager (GM) in charge of the metadata management needed in order to run the group. *Acquaintances* are nodes. An acquaintance node knows about that have data, which can be used to answer a specific query. Therefore, they can be thought of as links from one node to another, labeled by a (schematic) query. When a node is an acquaintance, then must be a way to compute how to propagate a query, to propagate results back, and to reconcile them with the results coming from the other acquaintances. The acquaintances are associated with one or more *corresponding rules*. These rules take care of the semantic heterogeneity problems. They are

implemented as rewrite rules and are called by coordination rules, in the body of the code implementing their action and condition components. Correspondence rules are used for the translation of queries and query result. For instance, they can be used to translate attribute or element names. Nodes in p2p can use *coordination rules*, which specify under what conditions, when and where to propagate queries or updates. The implementation of coordination rules is presented as ECA rules mechanism. A triggering event can be an update or a query coming from the user or from another node, condition, refers to properties of the query or update and action can be translation and propagation of a given update or query to a particular acquaintance.

4. Motivating Examples

In order to show the applicability of the system and query management in a p2p network, we give here a simplified real scenario. We draw an example from the Car Stock Exchange domain. Here we assume that we have three Nodes that could exchange data information about a rate of exchange of stocks. We have, Car Stock Exchange 'A', which is a Milan Car Stock Exchange, Car Stock Exchange 'B' from New York and 'C' is Frankfurt Car Stock Exchange as it is depicted on figure 2 below.

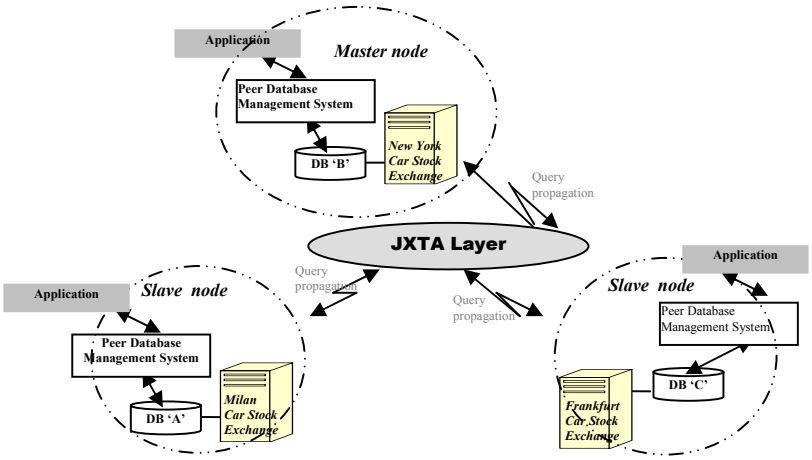

Figure 2: Car Stock Exchanges Example in p2p.

In the above scenario nodes create a virtual network which means that the nodes are acquaintances and can exchange data between them. In fact, the virtual network defines that these databases have data from the Car Stock Exchanges domain. Due to this network any node can interact with other nodes, which means that they can exchange data information through p2p network. The databases systems in p2p are completely autonomous, heterogeneous and independent, each maintaining its own data. In order to translate interoperability between two independent database systems we use the corresponding rules. Therefore, the databases could communicate freely in our virtual network. Through GUI-based applications on top of the nodes, a person could request data or create queries. Then suppose that a dealer from New York Car Stock Exchange create a query. From now on New Car York Stock Exchange node is a Master node. This means that it will be responsible for management on the global query process. The Frankfurt Car Stock Exchange node is a slave node. The slave node will interact with the master node by sending information when the query is executed. The involved database systems have the following schemas:

Node A: ORACLE

DB_A: orders(ISIN, acctId, automobileID, bs_ind, qty, prc, ccy, date);
automobiles (automobileID, auto#, automobile, automobileYear); Dealers (acctId, name, lname)

Table: orders

Ord#	acctID	Auto#	bs_ind	qty	prc	Ccy	date
3	1001	2	Buy	3	10 230	Euro	30/10/2001
4	1002	3	Sell	5	12 231	Euro	04/05/2002
5	1001	4	Buy	10	8 323	Euro	11/03/2003

Table: stockbrokers

acctID	Name	Lname
1001	Fausto	Muller
1002	James	Braun
1003	Cristin	Dores

Table: automobiles

Auto#	automobile	automobileYear
2	"FIAT"	2001
3	"BMW"	2003
4	"BMW"	2003

Node B: MS SQL

DB_B: autoorders (ISIN, acct#, modelID, bsInd, qty, price, ccy, date);
modelautos (autoID, modelId, autoName, autoYear); persons (acct#, name, lname);

Table: autoorders

Id	Acct#	modelID	bsInd	qnt	prc	Ccy	date
3	1001	12	Buy	4	8 230	Euro	03/10/2001
4	1002	13	Sell	4	11 731	Euro	12/05/2002
5	1001	14	Sell	15	7 932	Euro	01/02/2003

Table: persons

Acct#	Name	Lname
1001	Fausto	Muller
1002	Maria	Berlanda
1003	Paul	May

Table: modelautos

modelID	autoName	autoYear
12	"BMW"	2001
13	"RENO"	2002
14	"FIAT"	2003

Node C: DB2:

DB_C: CarTransactions(No#, acctNo, modelID, bsIndex, qnt, price, ccy, date);
CarModels (modelID, carId, model, modelYear); Dealers (acctNo, name, lname);

Table: cartransactions

No#	acctNo	carId	bsIndex	qty	price	Ccy	date
3	1001	5	Buy	3	8 230	Euro	12/11/2001
4	1032	6	Sell	5	8 731	Euro	14/12/2002
5	1005	7	Sell	10	12 622	Euro	10/03/2003

Table: brokers

acctNo	Name	Lname
1001	Fausto	Muller
1002	Anna	Georgieva
1003	Joanna	Vitcher

Table: carmodels

carId	model	modelYear
5	"BMW"	2002
6	"RENO"	2003
7	"BMW"	2003

Where the fields above have the following meaning:

Ord#, Id, No# – unique identifiers;

acctId, acct#, acctNo – dealer's account identifiers;

carId, modelId, auto# - car's unique identifiers;

bs_ind, bsInd, bsIndex – buy-sell indices;

qty, qnt – quantities of stocks to be sold/bought;

prc, price – corresponding prices;

ccy – currency of nomination;

date –date of the transaction.

automobile, autoName, model –models of the models;

automobileYear, autoYear, modelYear – year of manufacturing;

Then, let us consider a dealer (i.e. with acctID=1001) from node B want to execute the following query on the example databases: "Find how many car models in Car Stock Exchange domains over p2p network are for sale and manufactured is bigger than 2003 year."

```
Q1:     SELECT model, year_of_manifacture, quantity
        FROM ORDERS,CARS
        WHERE year_of_manifacture > 2002 AND  index = “sell”;
```

Suppose we have access to the online databases information sources shown in the databases schemas above. Here, we are interesting only in cars for which the year of manufacturing is bigger than 2002 and those available for sale cars. In order to obtain that information we have the following plan to answer:

- Ask from <year, index> tuple, that result, select only those cars where year > 2002 AND index = ‘sell’, for each database from the Query Scope where nodes are acquaintances.

Intuitively, the answer to a global query Q1 should be obtained by transforming it to an equivalent query of the views of the schema mappings between acquaintances. We assume that a translation algorithm exist which is sound and complete, i.e. it computed all the correct translation that exist. The query constrains are satisfied if they can be translated to constraints derivable from the mapped constraints. Cosider that we have got the query scope for this query Q1 and the nodes are already acquaintances. The coordination rules are responsible for the finding a path for query execution and answering. We assume, that each node consists of pre-defined coordination rules. We implement the coordination rules as ECA rules mechanism. The ECA rules examples given below are expressed like a database trigger for convenience.

Let us recall our example, the global query (Q1) has been received on the node B. This node B in collaborating with the QS finds three nodes which have equivalent domains and could satisfied our requirements. Then, suppose that the global query (Q1) can be executed locally on the Node B and on the other two remote nodes A, C, as well. Therefore, these nodes are acquaintances in accordance with our logical architecture from section 3. However, we have the following plan for query execution: first the query is executed locally and then the coordination rules propagate the query to the remote node A, and after that to node C. Assume that the predefined ECA rules, in our case database triggers are settled on those nodes (i.e. A, B and C). Therefore, we have the following pre-defined database trigger set up on the node B:

```
TRIGGER CARINFO
    AFTER RETRIVE ON ORDERS  Λ  CARS
  BEGIN
            IF   BSIND = “sell”   Λ    AUTOYEAR = 2003
                 (  SELECT bsind FROM ORDERS; )
                 (  SELECT autoyear FROM CARS; )

            THEN
               EXECUTE PROCEDURE
                     Retrieve data from the local database
               END PROCEDURE
            END IF
            EXECUTE PROCEDURE
              Take next node (i.e. node A);
            END PROCEDURE
            EXECUTE PROCEDURE
              Apply corresponding rules to the database schema DB_A;
            END PROCEDURE
            EXECUTE PROCEDURE:
              Propagate to node A;
            END PROCEDURE
  END
END TRIGGER.
```

Initially, we are trying to answer the query by evaluating the condition; the condition part tries to retrieve records which correspond to our criteria. When the condition procedure terminates, then the action part propagates the query to next node from the QS for local query processing.

The results returned from the databases after the queries execution need to be merged. The post-processing operations are done on the master node B in accordance with the global query.

The following result is received after execution of the query Q1:

Models	**Year**	**Quantity**	**Price**	
BMW	2003	05	12 231	→ coming from Node A
BMW	2003	10	12 622	→ coming from Node C
FIAT	2003	15	7 932	→ coming from Node B
Reno	2003	05	8 731	→ coming from Node C

Q2: “Then suppose that the dealer (i.e. with acctId = 1001) wants to buy all available for sale automobiles of “BMW" model which the year of manufacturing is bigger than 2002.”

In order to execute the query we have to obtain the information from the following plan:

- Ask from <quantity, model, year, index> tuple, that result, select only those cars where quantity > 0 AND model = ‘BMW’ AND year AND index = ‘sell’ > 2002, for each database from the Query Scope, where nodes are acquaintances.
- Then, a record is inserted in that database which satisfied the above requirements. (i.e.INSERT INTO cartransactions (No#, acctNo, carId, bsindex, qty, price, ccy, date) VALUES (6, 1001, 7 , “buy”, 10, 12 622, “euro”, “23/12/2003”)).

Then, assume that the dealer knows that the nodes C and A can answer the query Q2 and coordination rules propagate the Q2 to node C for local query processing. However, we have on the node C the following predefined database trigger for execution of the query:

```
TRIGGER CARTRANSACTIONS
AFTER RETRIEVE CARTRANSACTIONS Λ CARMODELS
    BEGIN
        IF  qty> 0 Λ  model = “BMW” Λ  modelYear > 2002  Λ  bsIndex = “sell”
           ( SELECT qty, basindex CARTRANSACTIONS; )
           ( SELECT model, modelYear FROM CARMODELS; )

        THEN
                INSERT INTO cartransactions (No#, acctNo, carId, bsindex, qty, price, ccy, date )
                        VALUES   ( 6, 1001, 7 , “buy”, 10, 12 622, “euro”, “23/12/2003”)
         END IF
        EXECUTE PROCEDURE
           Apply corresponding rules to the database schema DB_A;
        END PROCEDURE;
        EXECUTE PROCEDURE:
           Propagate to  node A;
        END PROCEDURE;
     END
    END TRIGGER.
```

From the rules example above, we have that if the condition is evaluate to true, which means that on the node B we have record(s) which satisfied our criteria. Then, we can insert a record into the CARTRANSACTIONS table and propagate to the next node (i.e. the query to Node A).

5. Coordination rules implementation

Generally, nodes in p2p act as master and slaves. A node can interact as a master and a slave simultaneously. Once a global query is created on a node this node becomes master, until execution of the global query completed. The master node controls the sessions, with the slave nodes performing subsidiary functions. Unlike from the master node techniques, the slave node interacts with the master as exchanging information about the query process (i.e. whether the node can answer the query). The query processing is entirely transparent to the application and the end-user. Mainly the coordination rules are responsible for global query coordination of the queries in a decentralized environment. The main idea is that, when a node receives a global query and query scope, then this node propagates to all acquaintances from the query scope in accordance with the coordination and correspondence rules [6].

The coordination rules process has an active role on top of distributed database management system in p2p. It monitors the query and automatically triggers consistency enforcement mechanism. Coordination rules are set of coordination activities which achieve a common goal. They are basic mechanism for a query management in p2p. The active database management systems which embed in a DBMS fulfill the requirements of the coordination rules process. Active databases provide the capabilities of: a) Defining one or more query to be monitored; b) Checking of the query constraints; c) Executing some action when some of the queries have been induced and propagation. In this section, we describe an implementation of coordination rules process applying the active database technology.

5.1 ECA rules

Active database management systems allow rules to be specified declaratively. We implement coordination rules as ECA rules mechanism. The ECA rules functionality is necessary to automatically manipulate queries between nodes. When a query is registered into the system, the coordination rules activate all components for evaluation of query and propagation the query to other node(s). At runtime, nodes use coordination rules which specify under what condition, when and where to propagate queries. The most common ECA rules format is based on the following model: “when ever *event* occurs, check the *condition* and if it holds, execute the *action*. The event part is a query received from the user or from another node in p2p network. The condition part refers to predicates or Boolean functions of the query and action part is the query transformation, and propagation of the query to the given node in p2p network. ECA rules provide a resolution for more flexibility implementation of the coordination functions.

- An *event type* describes situation to which a reaction must be shown. It is something that happens at a point in time. The event type can be simple and composite. Simple event defines a single low-level occurrences that belongs to one categories described in source. Composite event is defined by combination of simple or composite events using a range of operators that constitute the event algebra. An event can be a on of the following database operators: insert/delete/update/retrieve.
- A c*ondition part* formulates in which state the relevant part of the database has to be in order to execute the action. The condition start after the rule has been triggered. This part could be either a predicate on the database state, or a database query with empty or non-empty result. The condition verifies whether a query could be executed in the acquaintance node(s).
- An *action part* defines the reaction to an event and is executed after the rule has been triggered and its condition part is positive. The action part propagates to next node from the list of the query scope.

5.2 Coordination rules process

Given a user query, the coordination rules decide how to answer the query by synthesizing source views. In fact, the coordination rules control the logical process of the propagation of the query through a chain of nodes. They are in charge of transmission of the queries, when the data from multiple sites must be joined to satisfy a single query.

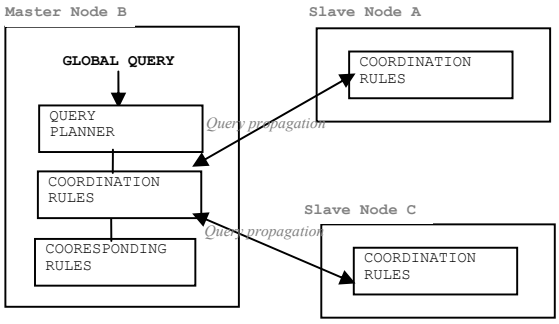

Figure 3. Coordination rules in P2P.

The base logical process of how query is manipulating by coordination rules is shown on Figure 3. When the global query is registered in the system (i.e. Node B) from now on this node became a master node for the query and this node keeps temporarily the query execution track (i.e. in case of rollback query transactions). However, the master/slaves technology is different from the centralized query processing and execution and schema integration technologies. The query planner (QP) processes queries by detecting the destination groups and computes the query scope. The responsibilities of the master node are to prune irrelevant nodes which might be contained in the Query Scope (QS) and starting the initial query process and propagation in the p2p. Firstly, when a node is an acquaintance, it computes for any given query, the paths from that node to other nodes from the QS. This node knows how to translate an input query into a specific query, formulated with respect to the database of an acquainted node. This specific query referred to as an acquaintance query [16]. The QP and coordination rules exchange information of whether a query to be propagated to a given node.

Let us take query Q1 from our motivation example in section 4. Then, suppose that node B and A are acquaintances with respect to an acquaintance query (i.e. $Q^{B(Xb_1....Xbn) \to A\ (\ Xa_1....Xan\ )}$), we have the following coordination rules represented as ECA rules form, they will execute the query on the node B and later on propagate to node A:

EVENT: $Q^{B\ (Xb_1....Xbn)}$;

CONDITION: IF $xb_2 > 2002 \wedge xb_4 =$ “sell”;
THEN
execute query;
END IF

ACTION: EXECUTE PROCEDURE:
take next node (i.e. node A);
if $Q^{B\ (Xb_1....Xbn) \to A\ (\ Xa_1....Xan\ )}$
then
apply Corresponding rules for query (i.e.$Q^{B\ (Xb_1....Xbn) \to A\ (Xa_1....Xan)}$);
propagate to the node A;
END PROCEDURE.

The coordintation rules system monitors events, evaluates conditions and triggered the corresponding actions when the condition become true. An event can be a registered user query or a query coming from other node, the condition part is a database constraint. The condition

maps input query to the target database schema of that node and evaluates the constraints part. There is a mechanism which provides the facilities to specify and evaluate query constraints. When the condition is satisfied, which means that the query could be executed on that node, the action part translates the query to the corresponding peer database schema with assictance on the corresponding rules. Then, the query is propagates to next node for local execution. In case that the coordination rules find a relation, for an example between two tables on different acquaintance nodes and the query have database constraints over these query. Its answer will be formed by joining the two views in the mapping.

5.3 Coordination Rules Architecture

The architecture and features of Coordination Rules System has the following basic functional components: 1) Query Pool; 2) Query Detector; 3) Scheduler; 4); Event; 5) Condition; 6) Action.

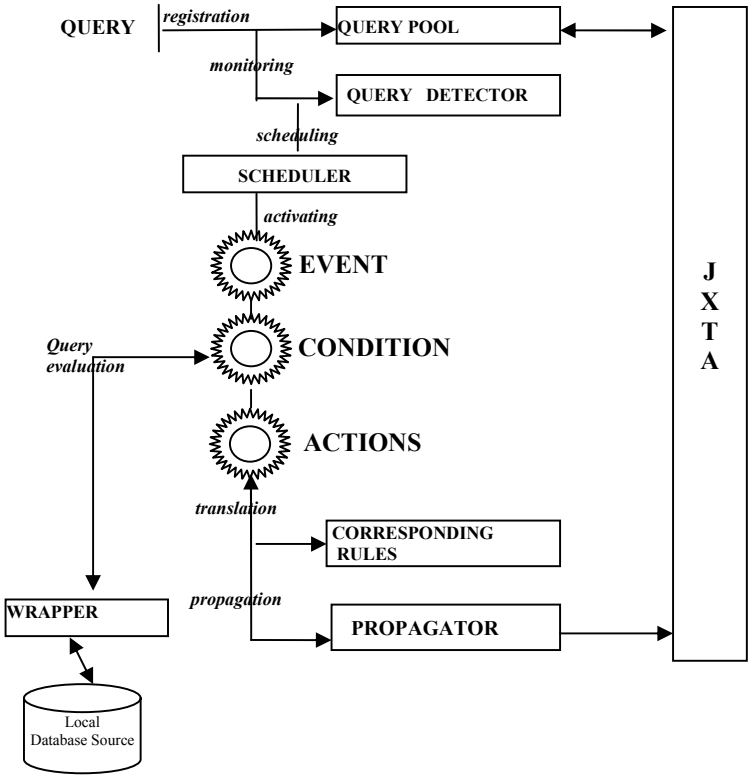

Figure 4. An Architecture of Coordination rules.

Whenever a node receives a query it is registered into the Query Pool. The Query Detector (ED) monitors the Query Pool for a new registered query. When the new query is registered into the Query Pool, the QD sends a query request to the query planner for computation of the Query Scope as we have described in the above sections. After receiving the query response from the QS, the query is scheduled (by the Scheduler) and waits to be activated from the Event Component (EC). Firstly, the EC checks-up whether the local database could answer to the query. In case that the query could be executed locally, the query is decomposed and sent to the Condition Component (CC). However, if the local database could not answer to the query, it is dispatched to the Action Component (AC). The CC is optional, it depends on whether the query has constraints or has not. In case that it holds one, the EC invokes CC. The CC consists of methods for evaluating the query constraints. These methods evaluate the constraints by

interacting with the local database system through a wrapper. The role of the Wrapper is to translate the query to the relevant query language for this DBMS. After the CC process has finished, the CC invokes the AC. The AC takes the next node from the QS, invokes Corresponding rules for a translation process in accordance to the database schema of that node. Then, the new transformed query is propagated by Propagator to the node in p2p network.

6. Conclusion

We have introduced coordination rules, a technique that can be used for data exchanges on top of different database systems in p2p network. The coordination rules process is general in the sense that it is logic-based and can be easily implemented in most active database systems. The process derives, in a systematic way, a set of rules that describe the query propagation in decentralized environment. Therefore, based on the coordination rules, nodes specify under what conditions, when and where to propagate queries for execution. We represent the coordination rules as ECA rules technology. The coordination rules mechanism is a part of the Local Relation Model (LRM)[5],[6],[16], which is a data model specifically designed for p2p applications.

Currently, we are working on the development coordination rules mechanism which is going to be integrated in our p2p database management systems prototype [16]. The next goal is focusing to the problem of guaranteeing the correct execution of interleaving queries across multiple database nodes. Likewise, we are aiming at exploring query processing in p2p issue. The contribution of this paper is a small but positive step toward building a stable architecture for peer database management systems in p2p environment.

References

[1] P. Bernstein, F. Giunchiglia, Anastaslos Kementsletsldis, J. Mylopoulos, L. Serafini and Ilya Zaihrayeu - Data Management for Peer-to-Peer computing: A vision. In Proceedings of the ACM SIGMOID, International conference on Management and Data, Madison, Wisconsin, June 2002.
[2] Fausto Giunchiglia - www.dit.unitn.it/~fausto – C2C Project
[3] L. Serafini, F. Giunchiglia, J. Mylopoulos, P. Bernstein - The Local Relational Model: Model and Proof Theory. Technical Report #DIT 03-002, University of Trento, Italy, January 2003.
[4] I. Zaihrayeu, Query answering in Peer to Peer Database Networks, Technical Report
[5] Fausto Giunchiglia and Ilya Zai hrayeu: Making peer databases interact – a vision for an architecture supporting data coordination. Technical Report #DIT-02-0012, University of Trento, Italy, June 2002.
[6] Fausto Giunchiglia, Ilya Zaihrayeu: Implementing database coordination in P2P networks. Technical Report #DIT-03-035, June 2003.
[7] V. Kantere, A rule mechanism for Peer to Peer Data Management, Master Thesis report, 2002
[8] Vasiliki Kantere, John Mylopoulos and Iluju Kiringa - A Distributed Rule Mechanism for Multidatabase Systems - University of Toronto, Canada, 2003.
[9] Chakravarthy S., Krishnaprasad V., Anwar E., Kim S.K.: Composite events for Active Databases: Semantics, contexts and detection. In Proceeding of the Twentieth International Conference on Very Large Databases, J. Bocca, M. Jarke, and C .Zanialo, Eds., Morgan-Kaufmann, San Mateo, Ca,
606-617
[10] N.W.Paton. Active Rules in Database System, ACM Computing Surveys, Vol 31, No.1 , March 1999
[11] V. Kantere, I. Kiringa, J. Mylopoulos: Coordinating Peer Databases Using ECA Rules, Department of Computer Science, University of Toronto, School of Information Technology and Engineering, University of Ottawa, 2003
[12] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, J. Mylopoulos: The Hyperion project: From Data Integration to Data Coordination. University of Toronto, Canada, 2003.
[13] Kumar K.M. Senthil: A Java Implementation of Coordination Rules as ECA Rules. Technical Report DIT 03-037 Informatics and Telecommunication, University of Trento, 2003.
Release 2 (9.2) http://www.csis.gvsu.edu/GeneralInfo/Oracle/appdev.920/a96590/adgp3act.htm
[14] A. Kementsietsidis, M. Arenas and R.J. Miller. Data mapping in peer-to-peer systems: Semantic and algorithmic issues. In ACM SIGMOD Int'l Conf. on the Management of Data, 2003.
[15] S.Gribble, A.Halevy, Z.Ives, M.Rodrig, and D. Suciu. What can databases do for peer-to-peer? In Proc. of Int'l Workshop on the WEB and Databases (WebDB), 2001.
[16] Fausto Giunchiglia, Ilya Zaihrayeu: Implementing database coordination in P2P networks. Technical Report #DIT-03-035, November 2003.
[17] Katchaounov, Timour: Query Processing for Peer Mediation Database. DIT, Uppsala Universitet, Sweden, November, 2003