Sharing Clinical Information in P2P Environment with RBAC Mechanism

Meghdad Mirabi, Hamidah Ibrahim, Md. Nasir Sulaiman, Mostafa Nikpour Kermanian, and Razali Yaakob

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

Summary

This paper proposes a Peer-to-Peer (P2P) distributed database system (PDBS) with Role Based Access Control (RBAC) mechanism for sharing the clinical information. In the proposed system, peers as health centers have their own local databases and information in these local databases can be shared among health centers through user queries. As a fully decentralized P2P distributed database information sharing system, each health center has both server and client functions. Each health center can act as a client that sends queries to others and also can act as a server which responses to the client health center. The proposed system supports health center operators to send queries and update queries to other health centers and integrates the distributed query results and displays the results through a user interface based on the role of the user in the system. Furthermore, health centers are able to discover other online health centers to send queries and receive responses. The proposed system is implemented with the JXTA platform in P2P environment.

Key words:

P2P, PDBS, RBAC mechanism, DDBS

1. Introduction

A distributed database system (DDBS) is a combination of database technology and network technology [1, 2]. A distributed database system involves multiple sites or nodes connected together via communication network. The sites may be in the same building or geographically far apart. Data are a collection of logically interrelated databases distributed over a computer network. A user can access data stored locally as well as from any other node. DDBS promises to improve reliability and availability because of the replication. If one site is down, the others are able to operate. The system scalability can be easily expanded by adding processing and storage power to the network.

In database systems, most users need to access only a small part of the database to carry out their tasks. Allowing users unrestricted access to all the data can be undesirable [3]. Access control means making sure that data or other resources are accessed only in authorized ways [4].

Role based access control (RBAC) is a mechanism that provides for relatively simple and flexible administration while allowing users to have access to the resources that they need. In an RBAC-based system, users are associated with roles, and roles are associated with operations. In addition, RBAC allows users access to resources based on their competencies and responsibilities within an organization.

Peer-to-Peer (P2P) computing is a distributed computing paradigm, in which participates rely on one another for services. The P2P computing promises that many resources can be combined together to build distributed systems with greater reliability than the sum of its parts without a central authority or server. Each participating computer or node in a P2P system is called a peer, and each peer may play a variety of roles. When accessing information, they are clients; when serving information to other clients, they are servers; when forwarding information for each others, they are routers [5].

Since P2P systems usually do not require special administrative or financial arrangements, the cost to start and grow a P2P system is low. P2P computing applications can tolerate the dynamically joining of individual peers; the system is amplified as new peers join in. Since new peers may introduce new resources that the system was previously lacking, it will improve the richness and diversity of the system [6, 7].

Distributed database technologies can be applied to retrieve information that is distributed to peers. The semantic meaning of shared information in peers can be queried at various content levels [8]. For users in a distributed online community, or peer group, who have common interest in a subject domain, it is essential for them to exchange information that is stored in their peers by semantic queries. Conventional distributed database techniques were designed for a network of distributed, large scale, database servers with centralized control. The emergence of P2P computing technology provides a new platform for new application of distributed database technology.

In a P2P platform, we can consider a peer group as an online distributed community; each peer in the community has local information that can be shared with other community members [9]. A distributed community is dynamic; since peers can join and leave the community

Manuscript received January 5, 2009
Manuscript revised January 20, 2009

any time. Local information in a peer may be stored in a personal database, instead of centrally managed database server. Peer's personal databases are loosely coupled without central control. However, from the community point of view, we can logically consider the collection of local databases of all community members as a single distributed database.

Clinical information contains data about the various health centers that patients visited, the family genetic pathologies, clinical treatments, and also personal information of patients regarding incomes necessary for bills.

Currently, clinical information is organized in a stand-alone way and decisions in a health center are taken by using only local data, resulting in incomplete information. To enable effective decision-making, clinical and administration units need to access distributed data with good access policy [10].

Recently, the use of XML [11] as a standard language for data exchange has been proposed for sharing the clinical information. Clinical records are mapped into a simple XML-based record. It is a lightweight data structure defined to contain relevant and aggregate information extracted from the different health centers.

Decentralized and Peer-to-Peer (P2P) approaches have been proposed for data access and integration [12] for XML based records. P2P systems are largely used to share information among distributed data sources. Nevertheless, sharing and accessing remote data requires that each peer must know the data model used by the others. The problem in using a P2P framework for sharing clinical information is that no standard has been yet adopted.

Also, the lack of a central administration P2P network brings challenges with respect to access control. That is, without central administration, it is more difficult to restrict access to specific resources. Of course, an application can include a mechanism to restrict access to resources, but this approach requires the developer to focus on access control in addition to the development task at hand.

This paper proposes a P2P architecture allowing different health centers to share clinical information and the health center operators can access to the shared data based on their roles that are defined in the system.

The rest of the paper is organized as follows: in section 2, we describe about related works of this research area. The requirements of the proposed system are described in section 3. Section 4 presents XML message protocols which are used in this system. The proposed system architecture is described in section 5. Section 6 explains about proposed distributed query processing mechanism. It supports single table query processing, multi table query processing and update query processing. Furthermore, the proposed Role Based Access Control mechanism is presented in section 7. P2P infrastructure is adapted as the cooperation framework of this system which is described in section 8. Section 9 presents the experimental results of this paper. In section 10, we summarize our work and describe future works.

2. Related Works

Recently, the use of XML as a standard language has been proposed for data sharing among hospitals. For instance, [13] proposes a system that allows for querying XML data in a P2P environment. The user is able to add new XML data in the P2P environment and efficiently querying them by adding semantics to the document. The proposed system shares common structure XML data, so that semantics is well known to the peer nodes. Similarly, [14] treats the problem of managing general purpose XML resources in a P2P environment.

Also, the use of P2P platforms for sharing clinical information has been proposed. In [15] a P2P system that enables a community of radiologists to share radiological images and their associated diagnoses is presented. The use of P2P in such system allows overcoming one of the main limitations of centralized approaches, that is, the fact that data being shared can grow so much to make storing all the information on a single machine inefficient or infeasible.

In addition, many approaches have been proposed to secure access to XML data. The problem with XML is that control is based on rules that are defined on the server that hosts database, while due to the nature of XML; the desideration is that privacy management rules are defined in the XML document itself. There have been several proposals of XML access control methods such as [16, 17]. In our proposed system, a simple access control mechanism based on the role of user in the system is proposed. In the other hand, other health centers that want to response to querying health center filter some shared data based on the role of the user who queried and then query results are sent to the querying health center.

3. The System Requirements

The requirements of the system are specified as follows:

3.1 Conceptual Data Model and Query Semantics

In the proposed system, all health centers share a global schema. The local database of each health center is an instantiation of the shared schema; it contains the relational tables that are instantiations of the relations in the schema.

Theoretically, the conceptual centralized data model implements the global schema as a single relationship database. Each table in conceptual centralized data model

is a union of corresponding tables in all peers.

3.2 System Functions

The proposed system should be able to support the following operations:

(i) New Query
A user can issue an SQL query at any health center. The system disseminates the query to all the other health centers. The health centers that hold the queried data in their local database respond to the query and send the corresponding query results back to the querying health center.

(ii) Update Queries
An update query is a repeated query processed by the same health center. The result of an update query is the integration of the local query results of these health centers:

a) Which have not been queried previously for the same query.

b) The database tables of the local databases which are modified since the last query which is sent to them.

3.3 Decentralized, Platform Independent P2P Application

The proposed system is required to provide facilities to establish peer groups, and allow health centers joining and leaving without restriction. Health centers should be able to find other health centers and directly interact with other health centers independent of their network locations.

4. Message Protocols

We use two message protocols for communication between health centers. Messages are represented in XML format.

4.1 DBQueryMessage

The DBQueryMessage is sent by a client or querying health center to other health centers to retrieve database data. The XML schema for DBQueryMessage is shown in Fig. 1 and an example of DBQueryMessage is shown in Fig. 2.

4.2 DBResponseMessage

The DBResponseMessage responds to a DBQueryMessage. It is also represented in XML format. The XML schema for DBResponseMessage is shown in Fig. 3 and the example of DBResponseMessage is shown in Fig. 4.

```
<xsd:element name="DBQueryMessage">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="QueryID" type="xsd:string"/>
            <xsd:element name="RoleName" type="xsd:string"/>
            <xsd:element name="TableName" type="xsd:string"/>
            <xsd:element name="QueryMessage" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Fig. 1 DBQueryMessage Schema

```
        <DBQueryMessage>
            <QueryID>q001</QueryID>
            <RoleName>Doctor</RoleName>
            <TableName>PatientTable</TableName>
            <QueryMessage>
                    Select PatientID, FirstName, LastName, Sex
                    From PatientTable
            </QueryMessage>
        </DBQueryMessage>
```

Fig. 2 Example of DBQueryMessage

```
<xsd:element name="DBResponseMessage">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="PeerID" type="xsd:string"/>
            <xsd:element name="QueryID" type="xsd:string"/>
            <xsd:element name="RoleName" type="xsd:string"/>
            <xsd:element name="TableName" type="xsd:string"/>
            <xsd:element ref="QueryResult"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="QueryResult">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Row" minOccure="1"
            maxOccure="unbounded"/>
        </xsd:sequenvce>
    </xsd:complexType>
</xsd:element>
<xsd:element name="Row">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Column1" type="xsd:string"/>
            <xsd:element name="Column2" type="xsd:string"/>
                                    …
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Fig. 3 DBResponseMessage Schema

```
<DBResponseMessage>
    <PeerID>PDB001</PeerID>
    <QueryID>q001</QueryID>
    <RoleName>Doctor</RoleName>
    <TableName>PatientTable</TableName>
    <QueryResult>
        <Row>
            <PatientID>P0001</PatientID>
            <FirstName>Peter</FirstName>
            <LastName>Jackson</LastName>
            <Sex>M</Sex>
        </Row>
        <Row>
            <PatientID>P0002</PatientID>
            <FirstName>Emily</FirstName>
            <LastName>Eland</LastName>
            <Sex>F</Sex>
        </Row>
    </QueryResult>
</DBResponseMessage>
```

Fig. 4 Example of DBResponseMessage

5. System Architecture

In order to form a fully decentralized P2P distributed database information sharing system, we design that each health center as a peer should have both server and client functions. In this section, we introduce the overall architecture of the proposed system and its components.

The overall architecture of the system is composed of the Peer Server Component, the Peer Client Component, and database for each peer. The Peer Server Component is always running on the health center to send and receive messages. The Peer Client Component is an application; it is initiated by a user who uses and interacts with the peer. Fig. 5 illustrates the architecture of the peer in the system.

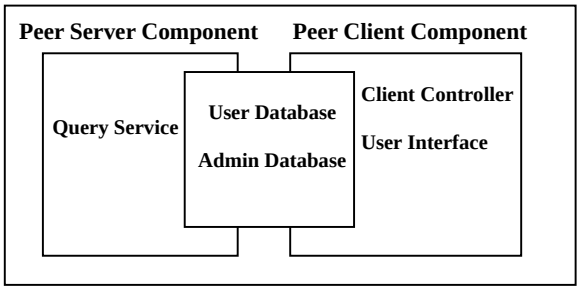

Fig. 5 Architecture of the Peer in the System

The Peer Server Component contains Query Service subcomponent. The primary function of Query Service is to handle database queries. When the Query Service receives a query message, it determines whether it is a new query or an update query. If it is a new query, the Query Service queries the User Database to generate the query result and sends the query results to querying health center. It also saves the received query message into the Server Query Log. If it is an update query, the Query Service determines whether the information in the database has been changed. If it has not been changed, the service does not respond to the query to reduce the network traffic. Otherwise it sends the query result back to the querying health center and updates its query log.

The Peer Client Component contains the following subcomponents: User Interface and Client Controller. The User Interface provides a user friendly visual environment for the users of the health center to send a query and view the results. The Client Controller is responsible to send database queries to all health centers. When a user submits an SQL query, the Client Controller constructs a DBQueryMessage and sends the message to all other health centers. If the SQL query involves multi tables, the Client Controller decomposes the query into several single table queries and a local query. It individually constructs each single table query into a DBQueryMessage and sends constructed DBQueryMessage to all other health centers. The local query is used for the local join operation. The Client Controller integrates the query results returned from other health centers. To integrate query results, the Client Controller keeps retrieving query results from the Admin Database, generates temporary tables for integrating query results, and applies the reformulated local query on these temporary tables to generate final results. The integration also involves handling the cached results in the Admin Database of the querying health center. When a health center starts the Client Controller, the Client Controller integrates cached results in the Admin Database with the previous query results.

The databases in each peer (health center) are divided into two categories: User Database and Admin Database.

The User Database stores:

- User Data: contains user tables.

The Admin Database stores:

- Cached Results: the query results which are arrived after timeout would be saved into the Admin Database.
- Client Query Log: it saves the SQL queries which are sent by the health center.
- Server Query Log: it saves the SQL query messages which are received from other health centers with time stamps.
- Temporary Query Results: the query results which are arrived before timeout would be temporarily stored in the Admin Database and wait for the Client Controller to integrate them.

6. Distributed Query Processing

The proposed system allows users of the health center to issue single table queries, multi table queries, and update queries. In this section, we describe how each of such queries is processed in the system.

6.1 Single Table Query Processing

When a user query is submitted through the User Interface, the Client Controller analyzes the submitted SQL statement to determine whether it is a single table query or not. If there is only one table in the *from* clause of the SQL statement, this query is a single table query. A user submits a single table SQL query through the User Interface then the Client Controller structures the SQL statement into a DBQueryMessage. The Client Controller sends the DBQueryMessage to all other online health centers. Also, the client health center saves the SQL statement into the Client Query Log for future repeated query. When the query is sent out, the timeout is also setup to indicate when the Client Controller stops receiving query results from other health centers.

Upon receiving a DBQueryMessage, the Query Service of the receiving health center parses the message and extracts the SQL statement. It queries its User Database with the same SQL statement and retrieves the query result. It structures the query result into a DBResponseMessage with its own PeerID, the QueryID and the RoleName as well as the name of the queried table. It then sends the DBResponseMessage back to the client health center. Meanwhile, it logs the query with the current time in the local Admin Database.

When the Query Service of the Client health center receives a DBResponseMessage, it saves the message with the query result into the local Admin Database. Meanwhile it informs the Client Controller form the arrival of a query result. The Client Controller keeps retrieving query results from the Admin Database and integrates them until all query results are received or it is timeout. After the integration of the query results, the Client Controller displays the integrated query results on the User Interface.

6.2 Multi Table Query Processing

When a user query is submitted through the User Interface, the Client Controller analyzes the submitted SQL statement. If there is more than one table in the *from* clause of the SQL statement, this query is a multi table query. When a user submits an SQL query involving multiple tables to the Client Controller through the User Interface, the system uses the following mechanism to process the query.

When a multi table SQL query is issued, the Client Controller decomposes the SQL query into a local query and several single table queries. The remote queries are individually wrapped into DBQueryMessage and sent to other health centers. The original multi table SQL query will be saved into the Client Query Log. The new local query is kept for integrating the query results to be received.

When the Query Service of the receiving health center receives the DBQueryMessages which carry the decomposed single table queries, it handles them separately in the same way as handling single table queries; it parses each message, extracts the single table SQL query, and queries its local User Databases. It then constructs the query result into DBResponseMessage and sends it back to the Client health center. It also saves all the received queries into the Server Query Log.

Upon receiving DBResponseMessage containing single table query results, the Query Service of the client health center saves the query results into the Admin Database and informs the Client Controller form the arrival of query results. The Client Controller retrieves the query results from the Admin Database and integrates the single table query results into separate temporary tables and then applies the created local query to those temporary tables to produce the final joined query result. After the integration of the query results, the Client Controller displays the results to user through the User Interface.

6.3 Update Table Query Processing

When a user selects one SQL query from the Client Query Log through the User Interface, the Client Controller considers whether it is an update query or not. For an update query, the system only retrieves new data from health centers that has not been retrieved or has been updated since the last query. When a user issues an update SQL query, the Client Controller creates a new version of the same QueryID, and decomposes the query if it is a multi table query. It then structures the SQL query or queries into DBQueryMessage(s) in the same way as processing new queries, and sends the query message(s) to other health centers.

When the Query Service of other health centers receives a DBQueryMessage, it first checks the local Server Query Log to determine whether this is a new query or not. If it is a new query, the Query Service directly query the local User Database using the single table SQL statement in the message, and saves the query message in the Server Query Log. Otherwise, the Query Service checks with the local User Database to determine whether the queried table has been updated or not. If not, the process is end, and no response is sent back to the querying health center. If the table has been updated, the Query Service queries the table from the User Database, and sends the query back to the querying health center. Meanwhile, it updates the corresponding query record in the Server Query Log.

On the Client health center, the Client Controller processes the received update query results in the same way as processing new query results.

7. Role Based Access Control Mechanism

RBAC mechanism allows users access to resources based on their competencies and responsibilities within an organization. In an RBAC-based system, users are associated with roles, and roles are associated with operations. In RBAC, an administrator can define a role and assign users to—and remove users from—that role as necessary.

In the proposed system, RBAC mechanism is done on server side. Thus, the DBQueryMessage that Client Controller of client health center sends to each health center has an element that is called RoleName. The RoleName is the role of user in client health center that queries through its User Interface. When the Query Services of the other health centers receive the query, they gets query results from their local User Databases, and filter some fields based on the element of RoleName in DBQueryMessage and then send the query results back to the querying health center.

There are two assumptions in this model.

(i) All health centers must have the same role definition and role mapping.
(ii) Each health center can only function in one role at a time.

8. Peer-to-Peer Infrastructure

The P2P infrastructure is adopted as the cooperation framework of this system. From an administrative point of view, the system is composed of a set of homogeneous databases of the health centers. Communication among health centers are implemented using the JXTA API. JXTA provides a set of XML-based protocols that allows computers and other devices to communicate and collaborate in a P2P fashion [18, 19]. These protocols establish a virtual network on top of the internet and non-IP network, allowing peers to directly interact and organize independently of their network locations.

In this system, we use two protocols of JXTA that are Peer Discover Protocol to enable health centers to discover other health centers on the network and Peer Resolver Protocol to allow a health center to send a generic query to one or more health center and receive corresponding responses.

9. Experimental Results

The experimental environment consists of three PCs with Intel Pentium IV processor and 1MB RAM. All the PCs are running the Windows XP Professional as operating system. Each PC (health center) can act as a client to send a query to others and also can act as a server to response to the client health center. The overall layout of the health centers is shown in Fig. 6. Health center 1, health center 2, and health center 3 set up the data set according to the User Database schema and Admin Database schema.

The User Database includes tables to store the information about patients, doctors, co-medicals, diseases, diagnosises, and therapies. The Admin Database stores cached results and temporary query results in temporary tables and also includes Client_Query_Log table to save SQL queries which are sent by the health center and Server_Query_Log table to save the SQL query message which are received from other health centers with arrival time.

Also, we use simple access control model to test the system. In this model, the users of the User Database are categorized into three classes that are medical doctors (Doctors), who work for the health centers, co-medicals (Nurses) who work for the medical doctors and database administrations of each health center (Administrations). Each class of the user is allowed to access a specific set of data of the User Database.

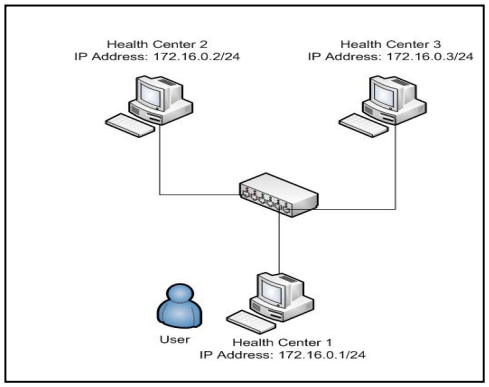

Fig. 6 Layout of Experimental Environment

9.1 Single Table Query Testing

We assume a user with the role of doctor in health center 1 wants to query the information of all doctors. He/she uses the User Interface to query to all online health centers (health center 2, and 3).

After a few seconds, health center 2 and health center 3 receive the request and parse the DBQueryMessage and then extract QueryID, RoleName, TableName, and QueryMessage from the DBQueryMessage and based on this information request to their local User Databases and provide the DBResponseMessages and send them back to health center 1. Health center 1 receives the responses and extracts the query results and stores them into its local Admin Database and then displays them on its User Interface. Fig. 7 shows the results through the User Interface.

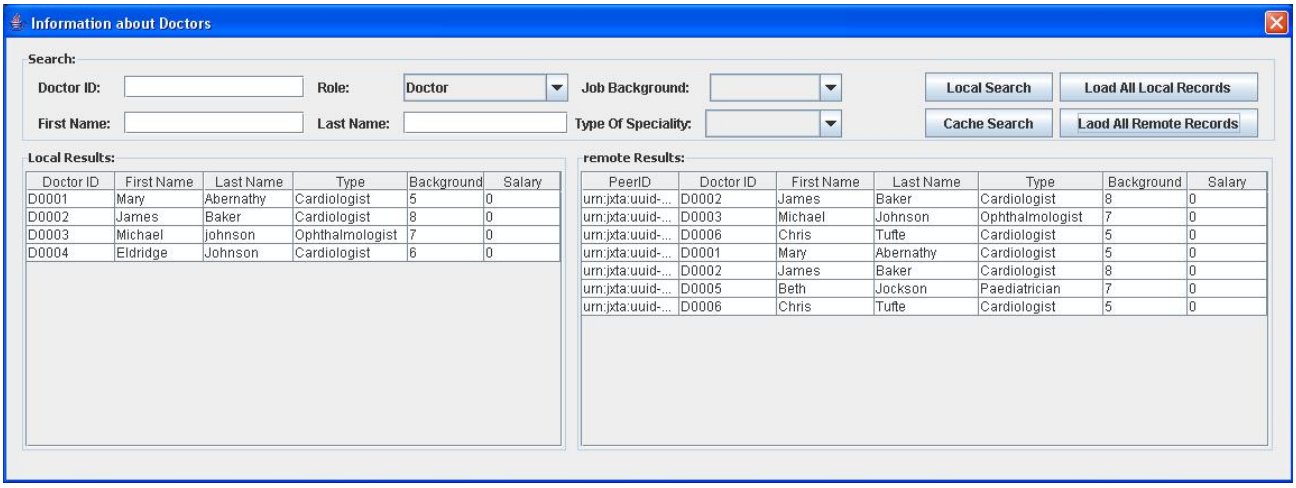

Fig.7 Single Table Query Testing

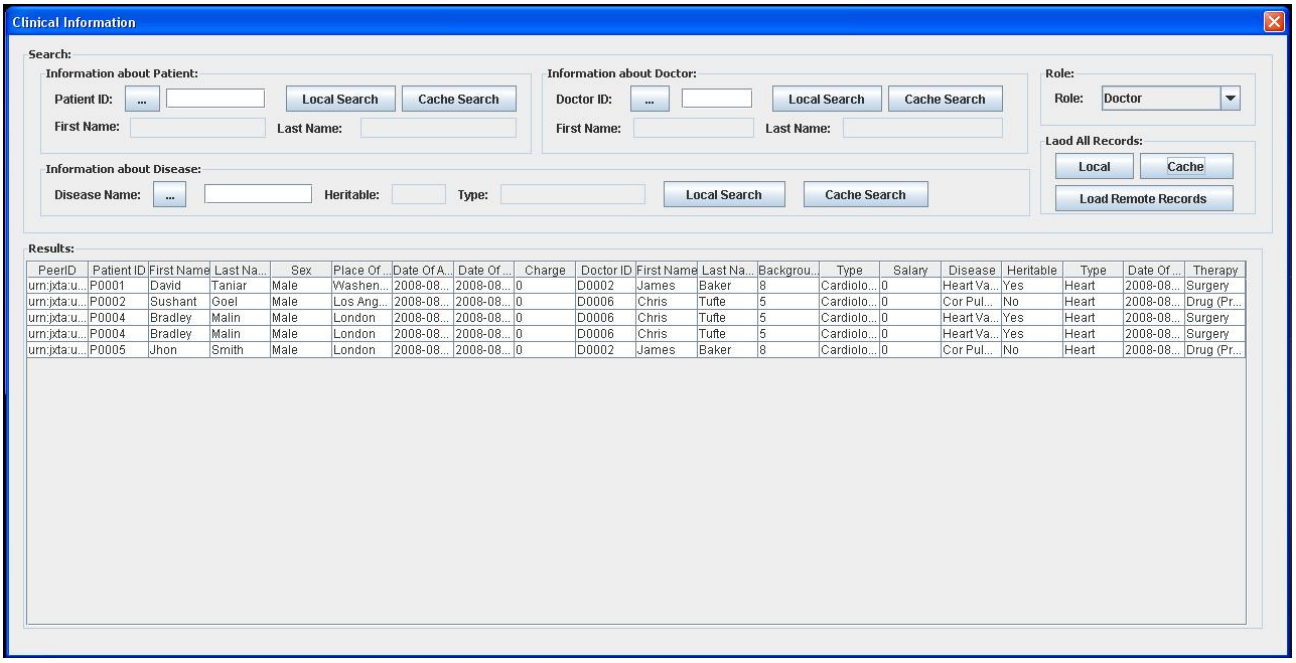

Fig. 8 Multi Table Query Testing

Although, the doctor has sent query to list all online health centers' doctors, he/she doesn't have enough permission to access to the salary field of doctors.

9.2 Multi Table Query Testing

We assume a user with the role of doctor in health center 1 wants to query all clinical information about patients in other health centers. He/she uses the User Interface to query to all online health centers (health center 2 and 3). The clinical information about patients are stored into more than one table, so the system automatically creates related temporary tables in its Admin database and sends DBQueryMessages to all online health centers.

After a few seconds, health center 2 and health center 3 as online health centers receive DBQueryMessages and request to their local User Databases and provide DBResponseMessages and send them back to health center 1. When health center 1 receives DBResponseMessages from health center 2 and 3, it extracts the query results and saves them into temporary

tables in its Admin Database. The final query results are integrated by health center 1 and are displayed through the User Interface to user.

Fig. 8 shows the query results through the User Interface. Although, the doctor has sent query to list clinical information, he/she doesn't have enough permission to access to the salary field of doctors and the charge field of health center.

9.3 Update Query Testing

We assume that a user with the role of nurse in health center 2 updated a patient record in User Database of health center 2. After that the user who has the role of doctor in health center 1 wants to query all information about patients. Like the previous scenario, the user uses the User Interface to query to all online health centers.

After a few seconds, health center 2, and health center 3 receive requests. Because only one table of health center 2 is updated, the health center 2 sends response back to health center 1. The health center 1 receives the response and updates temporary table that is related to response message and then displays the query results through the User Interface to user.

9.4 Testing the Time Out of Integrating Query Results

We replace the health center 3 with a slower machine and set the time out of health center 1 to 100 milliseconds. Now, we assume a user with the role of doctor in health center 1 wants to query the information of all doctors. He/she uses the User Interface to query to all online health centers (health center 2, and 3).

After a few seconds, health center 2 and health center 3 receive the request and parse the DBQueryMessage and then extract QueryID, RoleName, TableName, and QueryMessage from the DBQueryMessage and based on this information they request to their local User Databases and provide the DBResponseMessages and send them back to health center 1. Health center 1 receives the response of health center 2 and extracts the query results and stores them into its local Admin Database. Although health center 3 received the request and sent the response back to health center 1, it is too late for Client Controller of health center 1 to integrate the query response of health center 3. The response of health center 3 that carried the query result is stored in health center 1' Admin Database. Later when the user loads cache results, the system integrates previous results with cache results and displays them to user through the User Interface.

10. Conclusion and Future Works

This paper implements the interoperability and clinical information sharing between different health centers. It proposes a distributed Peer to Peer (P2P) based framework that enables health centers to share and aggregate clinical information. Clinical records are mapped into a simple XML-based clinical record. Health center operators that have different role in the system send queries; queries are then distributed to the online health centers, through a P2P infrastructure. The query results which are generated based on the role of the operator, are sent back to the querying health center and are displayed through the User Interface.

The proposed system is implemented only based on a simple RBAC mechanism. The JXTA security team has developed a trust based security model for JXTA. It uses public key cryptography to allow groups of peers to communicate securely [20]. In future, the system may be extended to create secure group using JXTA security technology. So, messages sent within the secure peer group will stay within the group. Each message can also be protected from the sender to the receiver if the peer does not trust some peers in the peer group. The future version of the system could allow each peer group to construct its own security schema based on the type of authentication, the degree of decentralization of the trust model, and the threat model from which the peer group is being protected.

References

[1] Silberschatz, A., H.F. Korth, and S. Sudarshan, Database System concepts (4th ed.). 2001, WCB/McGraw-Hill: Boston. p. 1-24.

[2] Ozsu, M.T. and P. Valduriez, Distributed and Parallel Database Systems. ACM Computing Survey, 1998. 28(1): p. 125-128.

[3] Ramakrishnan, R. and J. Gehrke, Database Management System (3th ed.) 2003, McGraw-Hill: New York. p. 694-695.

[4] Ricardo, C.M., Databases Illuminated 2004, Jones and Bartlett Publisher: London. p. 470-471.

[5] Parameswaran, M., A. Susarla, and A.B. Whinston, P2P Networking: An Information-Sharing Alternative. Computer, 2001. 34(7): p. 31-38.

[6] Anderson, T.E., et al., Serverless Network File Systems. ACM Transactions on Computer Systems (TOCS), 1996. 14(1): p. 41-79.

[7] Pitoura, E., O. Bukhres, and A. Elmagarmid, Object Orientation in Multidatabase Systems. ACM Computing Survey, 1995. 27(2): p. 141-195.

[8] Kossmann, D., State of the Art in Distributed Query Processing. ACM Computing Survey, 2000. 32(4): p. 422-469.

[9] Fan, L., et al., Summery cache: A scalable wide-area web cache sharing protocol. IEEI ACM Transaction on Network, 2000. 8(3): p. 281-294.

[10] Cannataro, M., et al., SIGMCC: A system for sharing meta patient records in a Peer-to-Peer environment. Future Generation Computer Systems, 2008. 24(3): p. 222-234.
[11] WorldWideWeb Consortium. XML Language. [cited; Available from: http://www.w3.org/XML.
[12] Abiteboul, S., et al. An electronic patient record "steroids": Distributed, Peer-to-Peer, secure and privacy-conscious. in Proceedings of the Thirtieth international conference on Very large data bases. 2004. Toronto, Canada.
[13] Abiteboul, S., I. Manolescu, and N. Preda. Constructing and querying Peer-to-Peer warehouses of xml resources. in Proceeding of the International Conference on Data Engineering. 2005. Tokyo, Japan.
[14] Bonifati, A., et al. Heptox: Marrying xml and heterogeneity in your p2p databases. in Proceedings of the 31st international conference on Very large data bases. 2005. Trondheim, Norway.
[15] Blanquer, I., V. Hernandez, and F. Mas. A p2p platform for sharing radiological images and diagnoses. in DiDaMIC Workshop. 2004.
[16] Damiani, E., et al., A fine-grained access control system for xml documents. Transactions on Information and System Security (TISSEC), 2002. 5(2): p. 169-202.
[17] Miklau, G. and D. Suciu. Cryptographically enforced conditional access for xml. in Fifth International Workshop on the Web and Databases. 2002. Madison, Wisconsin.
[18] Sun JXTA v.2.0 Protocols Specification [cited; Available from: https://jxta-spec.dev.java.net/JXTAProtocols.pdf
[19] Wilson, B.J., JXTA. 2002.
[20] Karjoth, G., D.B. Lange, and M. Oshima, A security model for aglets. IEEE Internet Computing, 1997. 1(4): p. 68-77.