Performance of Using Oracle XMLDB in the Evaluation of CDISC ODM for a Clinical Study Informatics System

Shaohua Alex Wang[a], Yang Fann[b], Huey Cheung[a], Frank Pecjak[a], Barg Upender[a], Adam Frazin[a], Raj Lingam[b], Sarada Chintala[a], Gladys Wang[b], Marc Kellogg[a], Robert L. Martino[a], and Calvin A. Johnson[a]

[a] *Center for Information Technology, National Institutes of Health, Bethesda, MD*[b]

National Institute of Neurological Disorders and Stroke, National Institutes of Health, Bethesda, MD

Abstract

This paper describes potential strategies for data modeling and implementation as part of the general architecture of CSIS, a Clinical Study and Informatics System that has been developed for the National Institute of Neuroscience and Stroke (NINDS). We discuss the NINDS requirements and how they influenced the system design, with an emphasis on dynamic form creation. We also evaluate open standards such as "CDISC ODM" and several database technology choices, namely conventional relational "EAV" tables and the new "XMLDB" from Oracle. We describe performance test results, which show that it is feasible to implement CDISC ODM in Oracle XMLDB and the schema-based storage option for storing XML content is better than "CLOB"-based for applications that require high numbers of queries for XML fragments. For storage and retrieval of whole XML documents, CLOB storage is optimal. CSIS implements a hybrid approach of combining CLOB storage for storing XML documents with relational tables for storing metadata. Due to the complexity of the CDISC schema and related performance and implementation concerns, we have decided not to implement CDISC internal storage in our system.

1. Introduction

Promoting clinical research is a major priority in the new strategic plan for the National Institute of Neurological Disorders and Stroke (NINDS) [1]. The web-based Clinical Study Informatics System (CSIS) is a major component of an integrated Clinical Informatics and Management System (CIMS), which is being developed for NINDS intramural clinical researchers. In addition to CSIS, CIMS contains the Protocol Tracking Management System (PTMS), which supports protocol submission, approval, and monitoring of the protocol review process; and a data integration module, which provides data warehousing services to collect data from a variety of data sources for analysis [2].

CSIS provides supporting tools for conducting clinical studies including patient recruitment, screening, enrollment, data collection, monitoring, and reporting. Figure 1 shows

Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems (CBMS'04)
1063-7125/04 $ 20.00 © 2004 IEEE

the software architecture design of CSIS. It is a web-based, three-tiered architecture that has been deployed on IBM's WebSphere™ application server. The first tier, the presentation layer, could be a thin client such as Microsoft Internet Explorer or a thick client such as Microsoft *Infopath* [3]. The second tier, the business logic layer, employs J2EE standards-based technology (including Java Server Page, and open source packages such as Jakarta Struts, and Jakarta ObJect-relational Bridge). The third tier, the persistent data layer, uses Oracle Corporation's Oracle 9i (9.2.0.4) database with the XMLDB package installed.

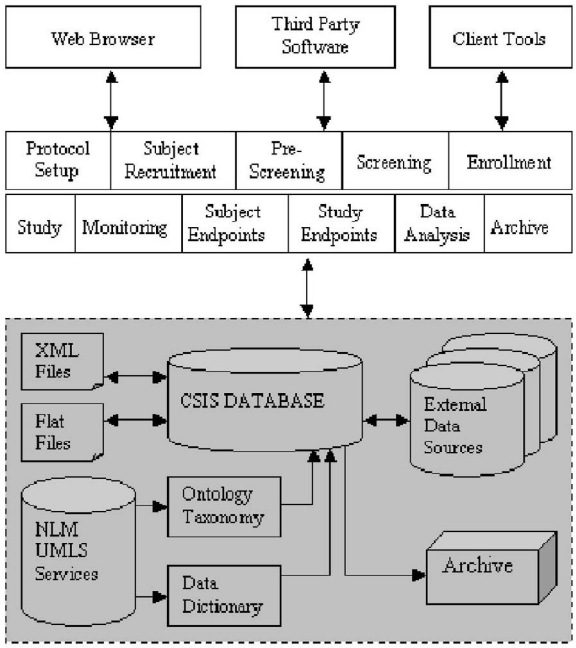

Figure 1. CSIS software architecture

Common to all the eleven modules in the second tier of Figure 1 is the utilization of electronic forms. Clinical data were collected by filling out the electronic form either by patient or by a caregiver. The forms are created when a p*rinciple investigator* (PI) creates or amends the protocol. One of the challenges in designing a process for the dynamic creation of electronic forms is the complete flexibility that the process must accommodate. In the next section, we will discuss various database designs to store the form data. In Section 3 we will present the performance test results of each database design. In Section 4 we will consider design tradeoff resulting from these performance test. Finally we draw some conclusions affecting our system design in Section 5.

2. Database design considerations

User and organizational requirements dictate that the database be generic enough to allow investigators to create arbitrary clinical forms without the intervention of a programmer or database administrator. If a new entity or attribute is needed, the appropriate structure must be created automatically with all necessary relationship constraints and proper indexing to ensure data integrity and optimal performance. The dynamic nature of such a system leads us to consider a metadata approach to data management. A metadata approach utilizes a general structure where only high-level relationships are defined. Specific information and relationships are maintained as row elements rather than column elements in the structure.

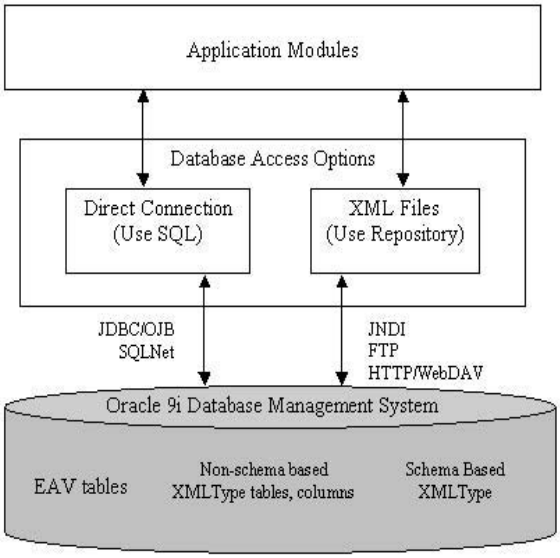

Figure 2. Database architecture

Another requirement is easy transport and representation of the data to external systems. To promote information exchange among researchers and to ease clinical trial data submission to the Food and Drug Administration (FDA), we have considered the potential of implementing the standard Operational Data Model (ODM)[4] created by the Clinical Data Interchange Standards Consortium (CDISC). The XML-based ODM model supports electronic acquisition, exchange, submission and archiving of clinical trial data and metadata for medical and biopharmaceutical product development. The model represents study metadata, study data and administrative data associated with a clinical trial. A potential disadvantage is that in some cases, large amounts of data will be required to represent some simple concepts. This is because CDISC is quite generic in nature, having been designed for use in diverse range of clinical trial situations.

Figure 2 depicts the current data storage model design. Clinical data can be transported to and from the database via an XML file through the FTP and HTTP protocols. The XML-based CDISC model fits well with this transport mechanism. The database layer can also be accessed via SQL statements through direct JDBC or Oracle network services connections. For the storage structure in this metadata approach, we consider two options: (1) Entity-attribute-value (EAV), and (2) XML documents stored in Oracle XMLDB.

2.1. Entity-attribute-value (EAV)

In an "EAV" design, the attributes for an entity are not hardwired into the database as table columns. Rather, they are stored as data, one row for every attribute. This design is often referred to as “vertical design” or “row modeling”. In addition, metadata describing each data element are stored in a data library, where the data item definitions can be readily created, viewed, and edited by the user. The EAV design makes it possible to accommodate new protocols (with new data items) without the additional programming that would be required in a “horizontal” database design. One needs only to add a description of each new data element to the data library. A good example of clinical information system implementing an EAV design is Yale University’s TrialDB [5]. This design also allows investigator to reuse and match common entities across forms and protocols.

Oracle XMLDB implements a number of SQL/XML standard based functions enabling us to query relational data and return XML documents (SQL/XML is an emerging part of ANSI/ISO SQL standard). We are able to generate the ODM compliant documents for each patient encounter stored in the EAV tables.

2.2. XML documents stored in Oracle XMLDB

Oracle XMLDB [6] is a set of utilities in Oracle 9i Release 2 that provides native support for storing and retrieving XML elements from XML documents. It stores information within the Oracle database and represents underlying data dually both as sets of XML elements within XML documents and as cells within relational tables. This structure allows for fine-grained queries on the data contained in the XML document, utilizing the traditional RDBMS tuning mechanisms (e.g. indexes and partitions), while maintaining DOM fidelity for viewing the entire document at once. This structure also lends itself to utilizing the CDISC ODM data format as a metadata definition. In fact, since the ODM structure is already available as an XML schema, we feel that utilizing the Oracle XMLType is more natural choice than EAV, provided that the performance of Oracle XMLDB is acceptable. (XMLType is an Oracle XMLDB defined opaque type for handling XML data. XMLType has predefined member

functions to extract XML nodes and fragments. One can create columns of XMLType or tables of XMLType and insert XML documents into them.)

Oracle XMLDB provides two options for storing XML in the database. The first, referred to as u*nstructured storage*, uses the Character Large Object (CLOB) data type to persist the XML document as a string of bytes in the database. The second, referred to as s*tructured storage*, involves shredding the XML document and then persisting the content as a set of SQL objects and tables. Structured storage is only available when the XML conforms to an XML schema. Oracle XMLDB uses the XML schema to generate the set of SQL objects required to persist the content of the instance documents. Database administrators and application developers can tune performance by annotating the XML schema to control how collections are managed. The naming of tables, SQL objects, SQL attributes, and the mapping between XML Schema data types and SQL data types among others can also be specified in the annotation. In our experience, a significant effort was required to annotate CDISC to make it work. Both storage options also support XPATH queries, with the relational storage option translating the XPATH into traditional SQL. By default, the underlying storage model for XML schema-based XMLType columns and tables is structured storage.

3. Oracle XMLDB performance test

In order to test the performance of implementing CDISC ODM in Oracle XMLDB structures, we have created 10,000 simulated patients and associated "Brief Psychiatric Rating Scale" (BPRS) [7] electronic forms, which conform to CDISC ODM. These records were inserted into a CLOB-based table, and an object-relational table, respectively. The object-relational table along with hundreds of other objects and nested tables were created automatically when the CDISC ODM schema was registered with Oracle XMLDB. To compare with the EAV data model, the data contained in these XML documents were inserted into EAV tables as well. The EAV tables are structured so that a complete CDISC document can be created with SQL/XML.

Table 1 shows the performance test results of a set of experiments. A comparison of such performance results allows us to get an appreciation for the most appropriate storage architecture. The experiments were performed on a 2-GHz dual processor Dell Pentium 4 server equipped with 12GB RAM. The installed operating system and database are Redhat Linux 9 and Oracle 9i version 9.2.04 respectively. In first test, we loaded 10,000 ODM XML documents into the database. Each document is about 10KB in size. It took less than 2 minutes to store 10,000 documents into CLOB-based storage and less than 18 minutes into schema-based storage. Clearly, inserting XML content into a CLOB-based table is faster than inserting it into the schema-based object-relational tables. Loading an XML document into schema-based storage is a fairly complex transaction because each XML document that was loaded had to be parsed, loaded into a SQL object, and inserted into the ODM object table.

Retrieval of 10,000 documents from the two different implementations exhibited the same characteristics. The unstructured CLOB type storage allows for higher rates of ingestion and retrieval because it avoids the overhead associated with parsing and recomposition during storage and retrieval operations. Retrieving XML documents from EAV tables involves a large number of joins, which slows down its performance. We conducted a full table scan in the second experiment followed by conditional query and update experiments. Each experiment is run once for each table. The result shows that transactions against the

structured storage model are much faster than the CLOB-based storage model, as expected.

Table 1. Database performance test results

Experiment	EAV	CLOB	Structured
Insert 10000 records	5 m 7.00 sec	1 m 56.16 sec	17 m 32.85 sec
Select 10000 records	19 m 5.00 sec	1 m 33.72 sec	14 m 29.00 sec
Select all SubjectKey	0.35 sec	20.76 sec	0.41 sec
Select number of patient	0.16 sec	21.43 sec	3.39 sec
Update SubjectKey	0.01 sec	16.69 sec	0.14 sec

An aggregate mix of database transactions was tested in a throughput test, including insertion and retrieval of a complete form, updating a field within that form and returning a list of patients that match a statistical criterion. We used the software "Benchmark Factory for Oracle" from Quest Software to conduct the throughput benchmark test. A summary of the test results is shown in Figure 3. We scaled up to 20 simultaneous users, which represented an upper limit workload. The test database with EAV design was able to support an average of 24.4 transactions per second. The schema-based structured storage and CLOB-based storage model yielded an average transaction rate of 10.5 transaction per second and 1.4 transaction per second respectively. The slow update statement in the case of CLOB-based XML caused a reduction in its overall transaction rate.

4. Discussion

The timing differences between CLOB-based and structure-based storage are the result of how Oracle XMLDB accesses the data in these two storage options. The common way of referencing XML documents is via XPath statements. The Oracle9i XMLType enables the querying of collections of XML documents through the *extract* function, which takes an XPath parameter. When a query is run against CLOB tables, all 10,000 records have to be brought into the memory, parsed and examined. On the other hand, when the XMLType is stored in structured storage (object-relationally) using an XML schema and queries using XPath are used, they are rewritten to go directly to the underlying object-relational columns. This enables the XPath to be evaluated against the XML document without having to ever construct the XML document in memory, resulting in a much faster transaction. To speed up the transaction even more, we have annotated the schema so that we can access the tables directly in the "select" and "update" statements. Note that even though transactions against the CLOB-based table are slower than that of structured-based storage, they are still within acceptable range. Also, function indexing against CLOB tables may improve performance. Transactions against EAV tables are still the

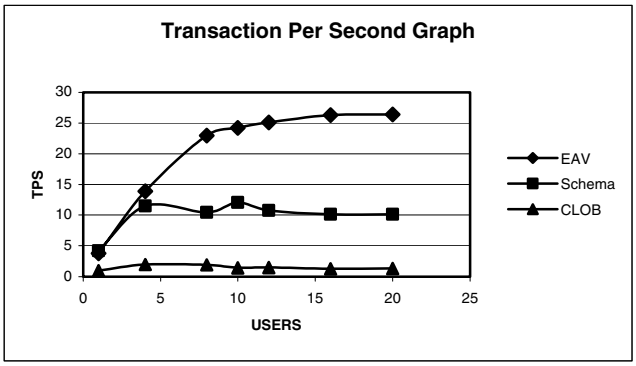

Figure 3. Load testing results. Transactions normalized among EAV and XPATH queries

fastest among the three different models. This can be attributed to not having the query-rewrite overhead as well as the design being optimized for the data at hand.

Even though CDISC ODM schema can be used to create database structures to collect and store clinical data, we have decided not to implement CSIS in that way for the following reasons: (1) We anticipate that ODM schema will change in the near future as a result of harmonizing with the HL7 reference information model. Schema change will require us to migrate the data and update the application, which is costly in terms of time and resource. (2) CDISC ODM contains a comprehensive list of XML tags that were not essential for NINDS electronic clinical forms. This added complexity makes any XPath query against tables containing CDISC ODM documents unnecessarily slow. This is true for both CLOB-based and structure-based storage. (3) We have chosen to use thick client technology such as Microsoft *Infopath* and XForm plug-ins for creating and filling the electronic clinical forms to make the process as user friendly as possible. However, since these clients require that a schema be bound to each form they create we feel that the CDISC ODM schema is too general for that purpose. (4) Based on the complexity of the CDISC ODM schema and our performance test, we speculate that for a simpler schema the performance discrepancy between CLOB-based and structure-based storage options should be much smaller than the CDISC ODM schema.

5. Conclusions

We have demonstrated that CDISC ODM schema can be used to collect and store clinical data into Oracle XMLDB in either structured or CLOB storage. Although there are costs associated with structured storage, it provides a number of advantages over CLOB storage for managing XML content as discussed above. For a variety of reasons, most notably the high likelihood that the CDISC ODM schema may change in the near future, we have decided not to implement a CDISC ODM schema-based storage system in order to avoid difficult data migration in the future. Instead we have implemented a hybrid data storage system for CSIS where content from electronic clinical forms were stored in an XMLType table as CLOB and metadata were store in relational tables. For reports and data analysis, the form will be parsed and the data will be brought into a data warehouse that has an EAV type structure.

References

[1] Neuroscience at the New Millennium, Priorities and Plans for the National Institute of Neurological Disorders and Stroke Fiscal Years 2000-2001, 1999

[2] Huey Cheung, Yang Fann, Shaohua Wang, Barg Upender, Adam Frazin, Raj Lingam, Sarada Chintala, Frank Pecjak, Gladys Wang, Marc Kellogg, Robert L. Martino, and Calvin Johnson, A Web-Based Protocol Tracking Management System for Clinical Research. 17th IEEE Symposium on Computer Based Medical Systems, 2004

[3] Thomas Robbins, Programming Microsoft InfoPath. Charles River Media, 2004

[4] Kush R, "A Multidisciplinary Approach to Data Standards for Clinical Development - Progress Update", Applied Clinical Trials, April 2002, pp. 35--44.

[5] Nadkarni PM, Brandt C, Frawley S, Sayward F, Einbinder R, Zelterman D, Schacter L, Miller PL: "Managing attribute-value clinical trials data using the ACT/DB client-server database system", Journal of the American Medical Informatics Association, 1998, 5(2):139-151

[6] Oracle9i XML Database Developer's Guide - Oracle XML DB Release 2, Part Number A96620-02, 2000

[7] Ventura, Green, Shaner & Liberman, "Training and quality assurance with the brief psychiatric rating scale: The drift buster". International Journal of Methods in Psychiatric Research, 1993.