BioEng Project

Design/Implementation Report

Version 2.0

By

Jinating Zhang

August 14, 2001
## Content

1. Introduction ................................................................................................................... 4
2. Architecture .................................................................................................................. 5
3. Database ........................................................................................................................ 7
   3.1 Conceptual Design ..................................................................................................... 7
   3.2 SQL Server 2000 Implementations ............................................................................ 12
   3.3 SQL Scripts for Creating Tables .............................................................................. 13
      3.3.1 LearningObject (Appendix A-1) ....................................................................... 13
      3.3.2 LORelation (Appendix A-2) ............................................................................ 13
      3.3.3 TextObject (Appendix A-3) ........................................................................... 13
      3.3.4 ImageObject (Appendix A-4) .......................................................................... 13
      3.3.5 AudioObject (Appendix A-5) .......................................................................... 13
      3.3.6 VideoObject (Appendix A-6) .......................................................................... 13
      3.3.7 UserInfo (Appendix A-7) ................................................................................ 13
      3.3.8 RLOCont (Appendix A-8) ............................................................................... 13
      3.3.9 Trackinglogging (Appendix A-9) ................................................................... 13
      3.3.10 KeywordMapping (Appendix A-10) ................................................................ 13
      3.3.11 RKeyMeSH (Appendix A-11) ....................................................................... 13
      3.3.12 MT2001 (Appendix A-12) ............................................................................ 13
4. Functional Modules ....................................................................................................... 14
   4.1 Overall Design Diagrams ......................................................................................... 14
   4.2 Detailed Design ......................................................................................................... 14
      4.2.1 Login (Appendix C-1) .................................................................................... 14
      4.2.2 Course Center (Appendix C-2) ....................................................................... 14
      4.2.3 Resource Center (Appendix C-3) .................................................................... 14
      4.2.4 Section Center (Appendix C-4) ..................................................................... 14
      4.2.5 Multimedia Center (Appendix C-5) ................................................................ 14
      4.2.6 View Resource (Appendix C-6) ...................................................................... 14
      4.2.7 Metadata Management (Appendix C-7) ........................................................... 14
      4.2.8 Search Engine (Appendix C-8) ....................................................................... 14
      4.2.9 Tracking/Logging Statistics (Appendix C-9) .................................................... 14
      4.2.10 Authorization (Appendix C-10) .................................................................... 14
      4.2.11 Packaging (Appendix C-11) ......................................................................... 14
4.2.12 Uploading/deleting (Appendix C-12) ........................................................... 14
5 Integration ColdFusion with Java/COM ........................................................................ 15
  5.1 Integration ColdFusion With Java Class .................................................................. 15
    5.1.1 Zipping Class for Content Packaging (Appendix D-1) .................................. 15
    5.1.2 Getting Image Property Class (Appendix D-2) ............................................. 15
  5.2 Integration ColdFusion With Java Applet ............................................................. 15
    5.2.1 A List Java Applet for Accepting CFTree Selection (Appendix D-3) ............ 15
    5.2.2 Interacting Between Java Applets Using Java Script .................................... 15
  5.3 Integration ColdFusion With COM Object ............................................................ 15

6 Experiences and Future Work ..................................................................................... 16
  6.1 Design experiences ............................................................................................... 16
    6.1.1 Horizontal VS. Vertical Segmentation of Metadata in Table Formalization .. 16
    6.1.2 BigInt Identity VS. GUID ............................................................................... 17
    6.1.3 Leave Space for LO types for Later Expansion ............................................. 17
  6.2 Implementation Experiences ................................................................................. 17
    6.2.1 Session Management .................................................................................... 17
    6.2.2 Dynamic SQL Query String Building ......................................................... 18
  6.3 Future Work ........................................................................................................... 18
1 Introduction

BioEng project is a cooperated research project between College of Medicine and School of Computer Science at the University of Oklahoma. The primary objective of this project is to build a proof-of-concept prototype of a multimedia medical repository for online education. The techniques involved in this project are shown in Fig. 1.

Fig. 1 Related Techniques in Building the BioEng Project Prototype

There are several goals to achieve in the prototype:
1. Multimedia Resource Repository with Full Database Support
2. Metadata Support based on IMS (Instructional Management System) Specification and MeSH (Medical Subject Headings) Standard
3. Course Material Searching Engine Using Metadata
4. Improving Exchangeability of Online Courses by Using XML
5. Data Mining on Learning Patterns Using Tracking/Logging data

This report is about the design and implementation of the prototype. The reset of the report is arranged as follows: Section 2 overviews the system architecture and components. Section 3 deals with database design and implementation. Section 4 presents design and implementation documentation for functional modules. Section 5 discusses issues in integrating ColdFusion with Java/COM. Finally Section 5 presents some experiences and suggestions for future work on implementing a production system.
2 Architecture

The overall architecture is as Fig. 1.

Fig. 2 System Architecture

The proposed architecture adopts a three-tier architecture based on web-database technologies. The first tier is Web Browser. Although any mainstream Web browser can be used, Microsoft Internet Explore 5.0+ is recommended to reduce compatibility problems. The second tier is the middleware tier or business logic tier and is implemented using ColdFusion Markup Language (CFML). Based on the content model and metadata model, several modules are also developed, including Authorization, Tracking and Logging, Interface Template, Searching Engine and Content Packaging. The third tier is the database server. Although major DBMS systems can be used in the architecture, we use SQL Server 2000 Standard Edition for its high performance/price ratio.

Before we go over each module, we will explain the overall data flow fist as shown in Fig. 2. The data flow goes through the following steps:
1. Client sends URL request to Internet Information Server (IIS)
2. IIS sends request to File Management System (FMS) if the request is for static web page.
3. FMS sends static web page to IIS.
4. IIS sends request to CFAS if request is ColdFusion scripts (with extension .cfm)
5. If CF a page involves database operations, then CFS send query string to SQL Server.
6. SQL Server send the retrieved table to CFAS.
7. CFAS generate standard Web page dynamically.
8. The generated Web page is sent back to IIS.
9. IIS send a Web page to client browser for display.
There are totally five logic components in this architecture based on content and metadata models, namely Authorization, Tracking and Logging, Interface Template, Searching Engine and Content Packaging. Three types of users, i.e., Learner, Instructor, and Administrator are identified. The privileges of a Learner is a subset of an Instructor, i.e., Learner is only allowed to view the content of a learning object while Instructor is able to add/delete/edit the content, edit metadata and view usage statistics of a learning object he/she owns. Similarly, the privileges of an Instructor is a subset of an Administrator, i.e, Administrator is responsible for all learning objects and system usage statistics.

The tracking and logging module is designed for data mining purpose to discover learning patterns for future system improvement. The Interface Template module generates Web contents dynamically. All the dynamically generated Web pages are context-aware, i.e., different GUI and content for users with different roles. The search engine module divides metadata elements into several logic groups and allows user to combine them arbitrarily. Within each group and for each metadata element in the group, the search engine allows a user to specify metadata element specific values (or value ranges) to query against metadata database. Finally, the content packaging module extracts explicitly stored metadata elements as well as those that are implicit between tables and generates IMS content packaging specification compatible zipped package for export and exchange.
3 Database

3.1 Conceptual Design

We define the smallest reusable course material as the Minimum Reusable Learning Object (MRLO). The whole system is made of those MRLO through different levels of aggregation. Although the architecture allows arbitrary number of hierarchical levels, currently we have five levels of learning objects (LO) in our system that are Course, Subheading, Resource, Section and Multimedia. Fig. 4 shows a typical example.

![Fig 4. Hierarchy of Course Material](image)

An upper-level LO can have multiple lower-level LOs while a lower-level LO can be used by multiple upper-level LOs. Therefore their relationship is many-to-many. In this application, the classification of Course, Subheading, Resource, Section objects are more of semantic because the content of LO is refined from upper level to lower level. They are all in the form of HTML document and can be generated using any popular HTML authorizing tools by instructors. On the other hand, the classification of multimedia file is more on syntactic, i.e., based on media type. Usually a multimedia file is a sub-LO of a Section type LO while in some cases a large multimedia file itself could be a standalone Resource type LO. For all types of LOs, they have a set of common mandatory metadata attributes while each type of LO has its type-specific ones. For example, size (in kilobytes) is common to all LOs while we have number of pages (slides) and number of words specifically for text LOs. Fig. 5 shows the LO hierarchy in our system and type-specific metadata attributes for each type of LOs. Fig. 6 shows a set of common metadata attributes and their relation with related entities, such as contributor, funding agency and most importantly, the taxonomy system – Medical Subject Heading (MeSH).
Fig. 5 Learning Object Hierarchy and Special Metadata Attributes Associated with a Learning Object

See Fig. 6 for common metadata elements of a learning object

ISA

Course Object
Subheading Object
Resource Object
Section Object
Multimedia Object

Text

Image
Audio
Video

Number of Pages (Slides)
Number of Words

ColorDepth
RunTime
SampleRate
RunTime
# of Frames

ParentID
ParentType
ChildID
ChildType

LO Relation

n Learning Object
m
Fig. 6 Common Metadata Attributes of a Learning Object

Most of the metadata attributes shown in Fig. 5 and Fig. 6 are self-explanatory. We put Microsoft Word/PowerPoint/Excel, PS/PDF along with HTML as text LO. Text LO has two basic type-specific metadata attributes, namely number of pages (slides) and number of words. For Image type LO, we have Horizontal Size (HSize) and Vertical Size (VSize) to denote width and height of an image. ColorDepth means how many bit are used for a single pixel. For video type LO, we have Number of Frames and Run Time. For audio type LO, currently we are only concern on Sample Rate and Run Time.

Metadata attributes shown in Fig. 5 and Fig. 6 can be classified into two categories, those can be automatically obtained when the LO is created/accessed and those have to be explicitly specified (The later category is bolded in the two figures). Metadata attributes that are specific to multimedia types of LOs can be retrieved when they are uploaded to the system, such as using Microsoft COM object to retrieve number of pages/slides/words of Microsoft Word/Excel/PowerPoint, Java Advanced Imaging (JAI) package to retrieve HSize/VSize/ColorDepth. Similarly metadata attributes for video/audio can be retrieved by commercialized or open source packages. Some of the metadata attributes common to all
types of LOs can also be automatically generated by recording system events, such as creation date, last edit date, last access date, number of user hit, format (in terms of MIME type) and physical file size, etc. The other category is that metadata attributes has to be explicitly input by user. They are listed in table 1. For the attributes that have one to one relation with a LO, its value can be input through a simple input box (free text) or by selection from a limited choices (controlled vocabulary). For attributes that have many to one relationship with a LO, a more complex interface in a repeated manner is used.

Table 1 Metadata Attributes Need Explicit Input

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reproduce</td>
<td>Whether Allow other users to reproduce this LO</td>
<td>Yes, No or need special requirement</td>
</tr>
<tr>
<td>Post</td>
<td>Whether this LO is ready to be accessed by a learner or act as instructor’s draft</td>
<td>Yes or No</td>
</tr>
<tr>
<td>Restriction</td>
<td>Whether this LO is restricted to be accessed</td>
<td>Yes or No</td>
</tr>
<tr>
<td>BroadBandNeed</td>
<td>Whether broad band is needed to access this LO</td>
<td>Yes or No</td>
</tr>
<tr>
<td>GoLiveDate</td>
<td>When this LO is ready to be accessed</td>
<td>Date</td>
</tr>
<tr>
<td>ExpireDate</td>
<td>When this LO is going to be expired</td>
<td>Date</td>
</tr>
<tr>
<td>ExpirePerson</td>
<td>Who is responsible for the expiration of the LO</td>
<td>Person (ID)</td>
</tr>
<tr>
<td>DeveloperNote</td>
<td>Note from Developer. Providing extra information besides Title</td>
<td>String</td>
</tr>
<tr>
<td>Duration</td>
<td>How long will it take to go through this LO</td>
<td>Time Span</td>
</tr>
<tr>
<td>IntendedEndUser</td>
<td>Types of intended end user</td>
<td>Multiple Types (ID) A Subset of all possible combination is used in the implementation</td>
</tr>
<tr>
<td>MeSH</td>
<td>MeSH IDs associated with LO</td>
<td>Multiple MeSH IDs</td>
</tr>
<tr>
<td>Contributor</td>
<td>Contributor and his/her role associated with the LO</td>
<td>Multiple Pairs of Person (ID) and Role (ID)</td>
</tr>
<tr>
<td>Agency</td>
<td>Agencies fund the development of the LO</td>
<td>Multiple Agency (ID)</td>
</tr>
<tr>
<td>Catalog/Entry</td>
<td>Key/Value pair to record additional metadata attributes that are important</td>
<td>Multiple Catalog (String) and Entry (String) Pair</td>
</tr>
</tbody>
</table>
IMS Global Learning Consortium recently released IMS Learning Resource Metadata Specification version 1.2 which includes a working document from an IEEE standards committee, of which IMS member organizations have been key contributors, and a number of modifications that have been approved by the IMS Technical Board. The mapping between IMS/IEEE metadata specification and our metadata table is shown as Table 2. It is easy to see that the mapping is an intersection between the metadata specification and database schema. The reasons are two-folds, first not all data fields are needed by the specification and some of them are for internal use and/or other purposes such as the number of user hit, GoLiveDate and expiration related attributes. Second, not all metadata attributes in the specifications are needed and supported in this application.

Table 2. Mapping between IMS/IEEE Metadata Specification to Database

<table>
<thead>
<tr>
<th>IMS/IEEE Metadata (Num/Name)</th>
<th>RDBMS Table/Data Fields</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Identifier</td>
<td>LearningObject/LOID</td>
<td></td>
</tr>
<tr>
<td>1.2 Title</td>
<td>LearningObject/Title</td>
<td></td>
</tr>
<tr>
<td>1.5 Description</td>
<td>LearningObject/DeveloperNote</td>
<td></td>
</tr>
<tr>
<td>2.2 Status</td>
<td>LearningObject/Post</td>
<td></td>
</tr>
<tr>
<td>2.3 Contribute (Role/Entity/Date)</td>
<td>RLOCont/(LOID,userID,ConType,Date)</td>
<td></td>
</tr>
<tr>
<td>4.1 Format</td>
<td>LearningObject/sFormat</td>
<td></td>
</tr>
<tr>
<td>4.2 Size</td>
<td>LearningObject/nSize</td>
<td></td>
</tr>
<tr>
<td>4.4 Requirement</td>
<td>LearningObject/BoardBandNeeded</td>
<td></td>
</tr>
<tr>
<td>5.5 intendedenduserrole</td>
<td>LearningObject/IntendedEndUser</td>
<td></td>
</tr>
<tr>
<td>5.9 Typicallearningtime</td>
<td>LearningObject/Duration</td>
<td></td>
</tr>
<tr>
<td>6.2 copyrightandotherrestrictions</td>
<td>LearningObject/(Restriction)</td>
<td></td>
</tr>
<tr>
<td>7 Relation</td>
<td>LORelation(ParentID, ChildID)</td>
<td></td>
</tr>
<tr>
<td>8 Annotation (Person/Date/Description)</td>
<td>LearningObject/(Annotation)</td>
<td>Reduced from 1:m to 1:1 according to specification</td>
</tr>
<tr>
<td>9.2.2 taxon</td>
<td>KeyWordMapping/(LOID,MeSHID)</td>
<td></td>
</tr>
<tr>
<td>9.4 Keyword</td>
<td>KeyWordMapping/(LOID, Keyword)</td>
<td></td>
</tr>
</tbody>
</table>

Note that some of metadata attributes in IMS/IEEE specification are automatically supported because their values are implicitly determined, such as the language (1.1) is always English, the source of taxonpath (9.2.1) is always “MeSH”, etc.
3.2 SQL Server 2000 Implementations

Fig. Overall Database Diagram

Fig. Table Scheme Definition 1
3.3 SQL Scripts for Creating Tables

3.3.1 LearningObject (Appendix A-1)
3.3.2 LORelation (Appendix A-2)
3.3.3 TextObject (Appendix A-3)
3.3.4 ImageObject (Appendix A-4)
3.3.5 AudioObject (Appendix A-5)
3.3.6 VideoObject (Appendix A-6)
3.3.7 UserInfo (Appendix A-7)
3.3.8 RLOCont (Appendix A-8)
3.3.9 TrackingLogging (Appendix A-9)
3.3.10 KeywordMapping (Appendix A-10)
3.3.11 RKeyMeSH (Appendix A-11)
3.3.12 MT2001 (Appendix A-12)
4 Functional Modules

4.1 Overall Design Diagrams

Modules in the system are inter-related. For example, authorization determines Web content and tracking/logging runs along with LO viewing. From design and implementation perspective, we divided the system into several programming blocks:

1. LO Centers that render management interface and list the sub-LOs belong to a LO.
2. LO Viewers that view LOs. Among them, Resource viewer is the most complicated one since it allows choosing template and use dynamic frames. Unlike Course, Section and Multimedia Viewers that only involve a single LO, Resource viewer could involve multiple LOs. There are some associated attributes with Resource type LOs, such as index, credit, introduction, etc.
3. LO management modules, such as adding/deleting a LO.
5. Metadata Searching.
6. Query on Tracking/logging statistics.

The overall module relationship is shown in Appendix B-1 and B-2.

4.2 Detailed Design

4.2.1 Login/Logout (Appendix C-1)
4.2.2 Course Center (Appendix C-2)
4.2.3 Resource Center (Appendix C-3)
4.2.4 Section Center (Appendix C-4)
4.2.5 Multimedia Center (Appendix C-5)
4.2.6 View Resource (Appendix C-6)
4.2.7 Metadata Management (Appendix C-7)
4.2.8 Search Engine (Appendix C-8)
4.2.9 Tracking/Logging Statistics (Appendix C-9)
4.2.10 Authorization (Appendix C-10)
4.2.11 Packaging (Appendix C-11)
4.2.12 Uploading/Deleting (Appendix C-12)
5 Integration ColdFusion with Java/COM

5.1 Integration ColdFusion With Java Class

Setting ups in ColdFusion administration at http://127.0.0.1/CFIDE/administrator/index.cfm
On the left of the page, under Server/Java
- Setting JVM: C:\Program Files\JavaSoft\JRE\1.3\bin\hotspot
- Setting Classpath: d:\BioEngNew\java

Two important tips:
- Stop and restart CFAS in control panel for any classpath chance.
- The classpath that contain java classes to be used must have read/access right to everybody (not IUSER_MachineName). Restart CFAS service too.

5.1.1 Zipping Class for Content Packaging (Appendix D-1)
(c.f. PackageLO.cfm)

5.1.2 Getting Image Property Class (Appendix D-2)
(c.f. AddLOComponent.cfm)

5.1.3 Data Mining Class (Appendix D-3)
(c.f. WebMining.cfm)

5.2 Integration ColdFusion With Java Applet

Register an applet at http://127.0.0.1/CFIDE/administrator/index.cfm on the left of the page, under extensions/applet. Specify codbase (where the class is located, must be URL addressable) and code name.

5.2.1 A list Java Applet for Accepting CFTree Selection (Appendix D-4)
(C.f. ProcessKeyword.cfm, EditKeyword.cfm, SearchForm.cfm)

5.2.2 Interacting Between Java Applets Using Java Script

We use Java Script to integrate several Java applets. (C.f. ProcessKeyword.cfm, EditKeyword.cfm, SearchForm.cfm).

5.3 Integration ColdFusion With COM Object

We use a COM object from Microsoft called DSOleFile.PropertyReader that can retrieve several attributes of OLE documents (WORD, EXCEL, POWERPOINT, etc.)
(c.f. AddLOComponent.cfm)
6 Experiences and Future Work

6.1 Design experiences

6.1.1 Horizontal VS. Vertical Segmentation of Metadata in Table Formalization

Suppose we have three LO types (A, B and C) and four attributes (I, II, III, IV). The association between LO types and attributes are given below:

<table>
<thead>
<tr>
<th>LOType</th>
<th>AttrI</th>
<th>AttrII</th>
<th>AttrIII</th>
<th>AttrIV</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>•</td>
<td></td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>•</td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

The horizontal segmentation of the above metadata schema is as follows:
- Table A (AttrI, AttrIII) which only include LO of type A.
- Table B (AttrI, AttrIII) which only include LO type of B
- Table C (AttrI, AttrIV) which only include LO type of C

The vertical segmentation of the above metadata schema is as follows:
- Table I only includes Attribute I but includes all three LO types LOs
- Table II only includes Attribute II and only includes LO of type A
- Table III only includes Attribute III and only includes LO of type B
- Table IV only includes Attribute IV and only includes LO of type C

The connection between segments in horizontal segmentation is natural and we can use “UNION” to combine query results of common attributes from different segments.

The connection between segments in vertical segmentation must through “JOIN” and need common IDs from both segments.

Initially we use horizontal segmentation to reduce storage and improve query efficiency. However, this doesn’t scale well when the number of LO types grows. The major problems are:

- The length of query string is often multiplied by the number of LO types. It is cumbersome from programming perspective. Often the length of our query string generated by searching engine is more than 2k that might exceed the DBMS server limit eventually.
- Often we need to get the query result according to common metadata attributes and then selectively further go down to specific attributes. Thus we need multiple accesses to the same table in this schema that is not efficient.

We then change to the vertical segmentation scheme. The design ideas are as follows:

- The learning objects have recursive many-to-many relationship. This reflects the requirements that a higher-level LO has multiple lower-level LOs while a lower-level LO might be reused in multiple higher-level LOs. We put the metadata elements common to all learning object into one table and each learning object has a unique ID. The relationships between LOs are recorded in a separate table that has four attributes (ParentID, ParentType, ChildID, ChildType). Given ID and Type we
can go to the corresponding tables to retrieve metadata elements that are specific to a LO.

- Separating common metadata elements and special metadata elements into different tables has the following advantages 1) Conceptual Simplicity. 2) Query efficiency. In many cases, queries are only on generic metadata elements. Thus we only need to work on the base table and no union operation is needed. 3) Coding Simplicity. We greatly reduced the length of query string by only query on the single generic table in most cases. 4) Flexible extensibility. User can add additional learning object types and redefine their relationships. The SQL query code on the generic metadata table can be reused without any modification even new types are added.

6.1.2 BigInt Identity VS. GUID

Initially we use bigint SQL Sever 2000 data type with automatic increment (i.e. identify to guarantee uniqueness in the table) as LOID and user ID. We found two major disadvantages for this design:

- We want to associate the independent LOs (e.g., raw course material) that are not belong to any other LOs with UserID. However, it is difficult to make sure that UserID doesn’t conflict with other LOIDs since both of them increase automatically and independently.
- We need another query to get the IDs of latest added record. To avoid concurrent control problem, we have to lock the two queries. This is a waste of computation power.

Then we change bigint to Global Unique Identifier (GUID), also a special SQL Server 2000 data type for all identifications. Although it takes a little bit more storage space and computing, it provides global uniqueness. One problem associated with GUID is that it cannot be used in aggregation operations in SQL query. The solution is to convert GUID to varchar first before aggregation operations.

6.1.3 Leave Space for LO types for Later Expansion

Initially we use 1,2,3 and 4 to represent LO types. However, we found that we need to add another LO type between 1 and 2 later on. Thus we had to go over almost all codes to make the change. In the later design, LO type codes for course, subheading, resource, section and multimedia file are 10,20,30,40 and 50 respectively. Thus we allow further expansions of new learning object types. However, this benefit only goes with vertical segmentation design. We will have to create new tables for new learning object types and parent/child relationship if horizontal segmentation design is used. We also need to rewrite all codes related to LO queries in the design.

6.2 Implementation Experiences

6.2.1 Session Management

Unlike Active Server Page (ASP) that can let user capture session timeout event, it is very difficult to do so in ColdFusion if not possible. Currently we just let user continue their previous session if a user doesn’t log out before exiting the system. It might cause a lot of problems especially for mining on tracking/logging data.
CF session ID is determined by two cookies (CFID/CFTOKEN). However, even session is timed out, these two cookies are still in the client side unchanged. When a user logs in next time, the session ID will remain the same if they are not explicitly cleared.

We can put the cookies in the server memory instead of at the client side machine. When client browser is closed they are gone and the server will generate new session ID when a new browser is opened. This is what we are using in our system. However, this implementation will not allow user to open multiple browser in a session. It might bring inconveniences. We are hoping ColdFusion version 5 will make significant progress in session management. Another solution is to rewrite session management by ourselves that are very costly.

6.2.2 Dynamic SQL Query String Building

In building metadata searching engine we need to build dynamic SQL Query String and submit to DBMS. We need dynamically decide tables to be joined, criteria in WHERE clauses, putting ANDs to connect several criteria, etc. Putting WHERE and ANDs in the right place is not as simple as it seems to be since there might be situations we didn’t think of. When WHERE or ANDs are missed. In these cases, a query error will be encountered.

Initially we put all INNER JOINs at the top level and specify criteria by using multiple ANDs in WHERE clauses, then we find the following transformations of SQL query string make our task easier.

```
Select * from
A inner join B on A.ID=B.ID
inner join C on B.ID=C.ID
inner join D on C.ID=D.ID
Where A.a1=x and D.a2=y
=>
Select * from
(select * from A inner join B on A.ID=B.ID where A.a1=x) s
inner join
(Select * from C inner join D on C.ID=D.ID where D.a2=y) t
on s.ID=t.ID
```

6.3 Future Work

As far as the functionality of this prototype system is concern, the following is an incomplete list of to-do work:

- Web Content Authorizing tools. We only provide course material upload in this prototype system so far.
- Changing independent LOs into normal ones. We need to put them into proper hierarchy and insert relationship records into Table LORelation.
- Immigrating course material from remote site to local machine. Some code modifications are needed in replacing links.
- Make post/restriction metadata attributes into active in LO selections.
- Make contact/SME of contribute types into active. Currently we assume all contributors are creator. New GUI is needed to add this information.
- Java classes/COM objects are needed to retrieve metadata attributes for audio/video LOs. So far we only work on OLE type text object and image object supported by JAI.
• Right now the ExpirePerson is varchar type and should be PersonID of GUID. Sponsor is not used either. A table or LDAP directory is needed to lookup the needed information.

Appendixes
Appendix A: SQL Script for Creating Tables (Appendix A.doc)
Appendix B: Overall Module Relationship Diagram (Appendix B.doc)
Appendix C: Detailed Design (Appendix C.doc)
Appendix D: Java Classes Source Codes (Appendix D.doc)
Appendix E: ColdFusion Source Codes Listing (Appendix E.doc)
Appendix F: Naming Conventions (Appendix F.doc)