ROSA: A Learning Content Management Systems with Semantic Access

Fábio Porto[1], Ana Maria de C. Moura[2], Fabio José da Coutinho da Silva[2]
EPFL -IC- Ecole Polytechnique Federal de Lausanne[1]
Military Institute of Engineering - IME/RJ[2]
Department of Computer Engineering - Rio de Janeiro - Brazil
fabio.porto@epfl.ch, [anamoura,fabiojcs]@de9.ime.eb.br

Abstract

Learning Content Management Systems (LCMS) supports e-learning applications with storage and efficient access for e-learning objects (LO)s. ROSA is a LCMS built as a semantic layer on the top of an XML native DBMS. In this paper, we present ROSA Data Model designed as an extension to RDF data model with ordered collections and relationship properties and cardinalities. An algebra is proposed with operations supporting queries over LO metadata and navigation through LO conceptual maps. Equivalence properties of relationships are implicitly considered, during navigation, by a query engine that supports ROSA operations and data structure. Additionally, a thesaurus is used during query processing extending the domain vocabulary with synonyms, generic/specific and associated terms. Finally, the system implements web semantic technology based on a formal data model tailored for an efficient storage and access of e-learning resources.

1. Introduction

Distance Education, also known as open, flexible or distributed learning, is a mode of education whereby learners are physically separated from the institution, and where the learning process takes place outside the education establishment. Students learn where and when it suits them, at their own pace [3]. This education mode resorts to various educational technologies to facilitate both the learning and the communication processes between tutors and learners.

LCMSs are being developed with the intention of abstracting e-learning material storage and access from applications. The fundamental unit of reference on e-learning data modeling has been specified as a Learning Object (LO). A LO is a collection of reusable material used to support learning, i.e., an entity that can be digital or not, and can be used for learning, education or training [8]. A LO is identified by a set of metadata descriptors established by an international metadata standard, such as LOM (Learning Object Metadata) [9] and SCORM (Sharable Content Object Reference Model) [2], an extension of LOM. The data elements that constitute a metadata instance of a LO are organized into a hierarchy, providing information about: its general characteristics, life cycle, meta-metadata, technical requirements, educational characteristics, intellectual property, relationships between other LOs, LOs annotations and LO classification system.

In a LCMS, the process of designing e-learning courses may be supported by query processing techniques that explore metadata and semantic relationships to aid in LOs search. This is especially useful in an academic community willing to publish and to share their courses materials. Thus ROSA has been conceived to support the design phase of e-learning courses, where tutors want to share and search for course material.

In this work we propose an environment that provides the ability to store and retrieve LOs based on their semantic contents. Hence, an academic user such as a professor, could create or improve his own course exploiting subjects not contemplated in other courses already published in the system, as well as a student might access the system to get information on a subject based on specific characteristics. So, queries such as: "*which topics are **relevant to** database learning*?"; "*which course material **comprehends** an OO Database topic?"; "which subjects are **basis for** teaching Query Optimization?"; or even "which courses are **prerequisite** for a database course?"* are supported by ROSA system, which takes into account predicate semantic content. Questions related to synonyms, specific/generic and associated terms are dealt by a domain thesaurus that helps during query processing. The ability to express queries over ROSA conceptual maps with thesaurus support leads to simple and powerful search engine, adequate for supporting large communities of tutors. In addition, students may also navigate through conceptual maps looking for interesting LOs. The relationships defined among LOs aid in both discovering materials and in establishing a didactic sequence of learning.

This paper focuses on presenting ROSA data model as a formal basis for the development of ROSA LCMS, in the context of the Semantic Web. LO is the main structure of the data model, which is complemented with semantic and aggregation types of relationships. Thus ROSA has been conceived to support the design phase of e-learning courses, where tutors want to share and search for course material associations. These are modeled as an extension to RDF statements [12]. ROSA data model provides a powerful algebra that makes it possible to query on LOs metadata, as well as on the semantic associations between LOs. ROSA is designed as a semantic layer middleware on the top of the Tamino XML native DBMS.

The rest of the paper is organized as follows: section 2 presents a LO conceptual map that serves as a use case to describe the data model in section 3. This section gives a comprehensive description of its structure and algebra, with examples based on the queries presented before. Section 4 describes ROSAQL, the ROSA query language to manipulate LOs. Section 5 gives an overview of the system implementation. Section 6 presents related work, and finally, section 7 concludes the paper with additional comments and future work.

2. The LO conceptual map

This section presents a use case for the ROSA project. It comprehends a RDF style representation of a LO conceptual map for a Master program in Computer Systems and a class diagram representing its schema.

Use case data and queries reflect ROSA objective of supporting the design of e-learning courses. LOs metadata and their relationships offer a rich semantic model for searching the repository during class preparation.

A conceptual map is an abstract mechanism that helps the e-learning designer to graphically model relationships among LOs of his e-learning application. It is represented by a directed-labeled graph, where nodes represent LOs, identified by their names, and directed labeled edges model relationships or semantic predicates the origin node has with the target one, like predicates in RDF.

Figure 1 presents a partial LO conceptual map of the Master Program in Computing Systems design at IME and the corresponding schema describing and classifying LOs and their relationships. The classes Program, Course and Topics, which correspond to domain specific types of LO, represent the schema. In fact, they model the main composition structure of courses in most school programs in Brazil. In this map, some LOs are associated to their classes in the schema through a dashed line. Relationships among LOs can express, for example, how courses of a specific program can be related to the topics they cover, or when it should be taught, whether before or after a certain topic or course according to the corresponding semantic of predicates, such as *is-prerequisite for, is basis for or depends on.*

In addition, the classes e-Learning Program, Master Program and PhD Program specialize the Program class.

In order to give a general idea of the semantic meaning expressed by the *LO* conceptual map of Figure 1, the following assertions can be made:

- IME Computing Systems program comprehends some courses, such as Data Structure and Databases, where the first is prerequisite of the latter;
- the *Database* course *covers* some topics such as *Storage Techniques,* Query Languages, Query Optimization, and Concurrency Control; however, *Storage Techniques* should be taught before *Query Optimization,* since the first *is basis_for* the latter*;*

- Query languages *cover* Relational Algebra and Calculus topics and these fundament SQL learning.
- Relational Calculus *covers* one of the query languages (exclusive OR): *QUEL* or *QBE*.

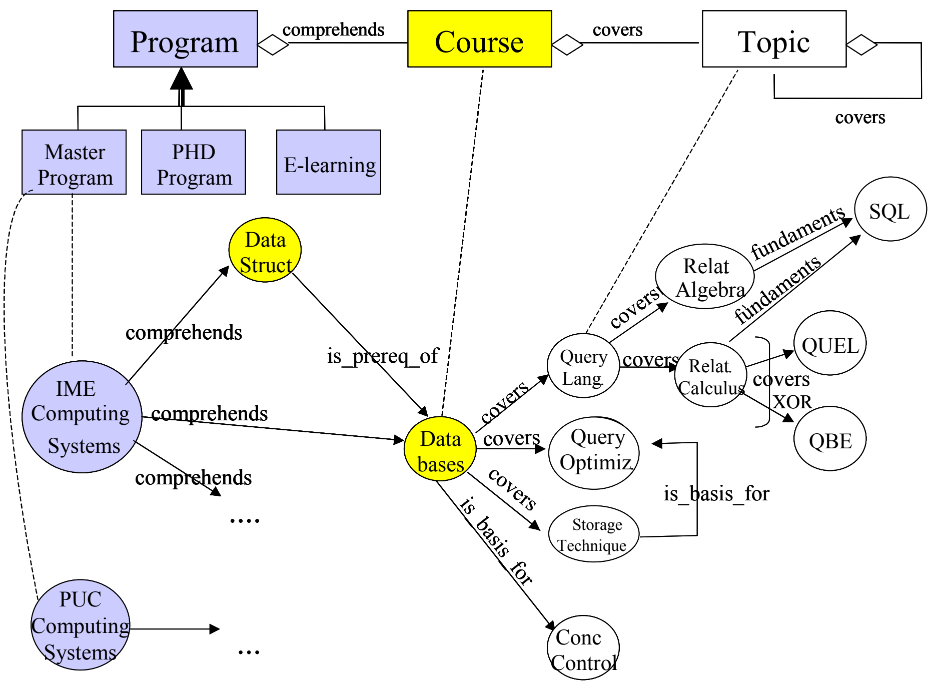

Figure 1: Master Program in Computing Systems – a LO Conceptual Map

2.1. Relationships

Relationships in a conceptual map are also referred to as predicates. They can be classified according to their semantic as aggregation type or domain specific. They place LOs into context by meaningfully associating them with other LOs. Users may choose terms to represent predicates that best represent their semantic in a domain. The computability of such terms is obtained by classifying them according to known types for which equivalence properties (reflexive, symmetric and transitive) have been defined.

In the context of e-learning, aggregation type relationships are specially relevant because they express the set of LOs to be visited when covering a certain subject (LO). Thus, predicates of type aggregation appear very frequently in conceptual maps. They present a transitive characteristic that is implicitly explored during query evaluation by the query engine.

Predicates may also be defined in a thesaurus that extends the conceptual map with associated terms, synonyms, specific/generic terms, etc. To better illustrate the use of predicates, consider those appearing in the conceptual map of Figure 1:

- **Aggregation type predicates**
  a. comprehends and its synonyms: has, covers, contains, includes, etc.;
- **Domain specific type predicates**
  b. is_prerequisite or is preceded _by;
  c. fundaments and its synonyms: is basis for, is condition for, etc.;

3. The ROSA Data Model

In this Section, a data model for the ROSA system is proposed. This data model is characterized by two basic structures: LOs and relationships associating them. The LO structure is composed of a set of metadata attributes in the Education domain and of a **N**avigational **S**earch **C**ontext (NSC). A NSC is a runtime attribute updated during the conceptual map navigation that links a source to a target LO through all the predicates traversed during the navigation.

The set of LOs metadata have been extended from the IEEE-LOM standard [9] and it contains information such as *identifier, title, language, contributors, version,* etc.

In this Section, a Data Model for the ROSA system is proposed. Initially, the structural aspect of the model is specified, followed by an algebra defining a set of valid operations.

3.1. Structure

Figure 2 presents the *LO* data model expressed in a UML class diagram. The *ComplexResource* class is the root of the class hierarchy, providing a common representation for *LOs*, *Relationships* and *Dynamically created objects*, which represent views in the database.

LO class derives directly from *ComplexResource* class. It includes metadata attributes according to the LOM metadata standard. *LOs* are uniquely identified by a *resource identification* attribute and are classified according to a *type* attribute.

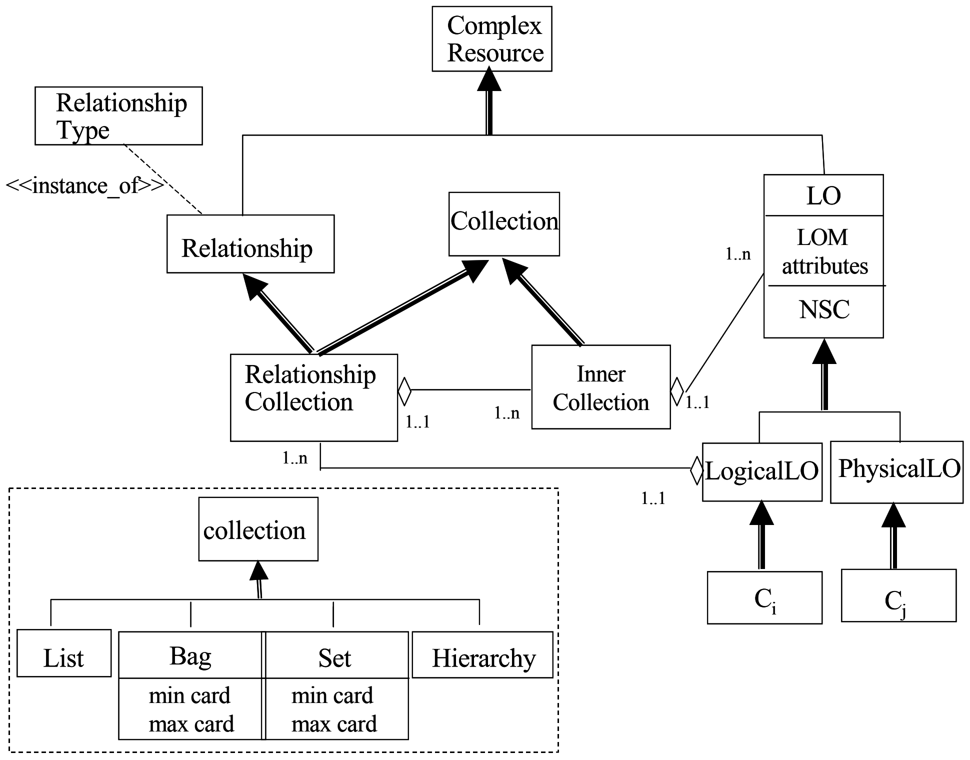

Figure 2: The Rosa Data Model

The NSC structure of a LO_i includes a set of pairs <LO_id ~ relationship>. Each pair represents a partial path navigated from a LO (identified by its *id*) through a relationship (r_i). Hence, concatenated pairs comprehend a complete path arriving at a specific LO_i.

In fact, a single LO can be retrieved from different navigational paths. Thus a NSC is represented by the following format: {<id_a~r_1~id_b~r_2~...~id_e~r_5>, ..., <id_r~r_7~...~id_v~r_9>}. When a LO is directly retrieved from the database repository, the corresponding NSC is empty, represented by the symbol ε.

Figure 3 shows an example of LOs association. Considering that LO_3 will be retrieved, the set of expression paths in the NSC are:

- NSC = ε if LO_3 is retrieved without navigation;
- NSC = {<LO_2 ~is_pre_requ_of>}if LO_3 is recuperated through LO_2;
- NSC = {< LO_1 ~fundaments~ LO_2 ~ is_pre_requ_of >}if LO_3 is recuperated via navigation from LO_1.

LOs instances represent concepts (Logical LO) and documents (PhysicalLO) that take part in an e-learning repository. LogicalLOs correspond to different levels of aggregations in the domain, such as: Topic, Course and Program in Figure 1. PhysicalLO corresponds to files and their metadata. An instance of PhysicalLO is considered atomic, which means that it cannot be composed by other LOs, whereas LogicalLOs may contain a collection of other LOs associated by aggregation and semantic relationships.

The Relationship class contains all verbs employed in a conceptual map. Its instances are classified according to a relationship type that defines common equivalence properties for its instances. As mentioned in Section 2.2, relationship types are classified according to the equivalence properties they present: reflexive, symmetric and transitive. The system explores implicitly such properties when evaluating a query over relationships. As an example, a conceptual map may use the verbs *comprehends* and *cover* indistinctly since they are classified as being of the same type, thus providing the same transitive equivalence property.

Collection is a class that represents physical collections of objects. A collection may impose a certain order to be followed during a visit to contained LOs. Thus, the model specializes the collection class into: set, bag, list or hierarchy. Set and list represent the corresponding mathematical concepts, whereas bags may contain duplicates in the collection. Hierarchy corresponds to *LO*s whose contents follow a tree structure. The set and bag classes include attributes to identify maximum and minimum cardinalities. Those attributes introduce restrictions over the collections that specify, respectively, the maximum and minimum number of elements that may be extracted from the collection. In Figure 1, for instance, the *exclusive or* aggregation relationship between *Relational Calculus* and *QUEL* and *QBE* can be modeled by a set collection with maximum cardinality 1 and minimum cardinality 1, meaning that at least one and at maximum one of the LOs in the set should be exhibited when presenting *Relational Calculus.* The representation of sets and bags with unlimited access to its elements is represented by *maximum cardinality=unbounded.*

Associations among LOs, whose cardinality is typically *one to many,* are modeled as relationship collections, which inherit both from Relationship and Collection classes conceptually define triples *t(subject, property, object)*, extended to n-ary relationships, where *subject* and *objects* are of type *LO* and *property* is a *relationship*. The schema specifies valid associations through triples T(class, relationship type, class). Thus, valid instances of *t* are in accordance to *T*.

User queries may join LOs from different database collections (or database repository), according to a certain boolean predicate. The Inner collection Class presents the similar concept of tuple in the relational model. It is used to support combination of LOs resulted from a join operation. As an example, considers a query that wishes to retrieve LOs stored at IME repository and at LNCC repository whose author is "Fabio Porto". For pairs of LOs that agree on the join predicate, an inner collection is formed containing both LOs. Initially, when a LO is retrieved from a repository a single LO inner collection is created to hold it. Subsequent join operations would eventually fill inner collection with joined LOs.

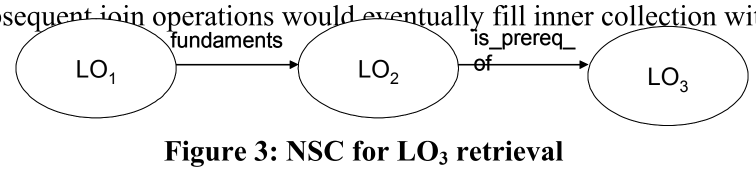

Figure 3: NSC for LO_3 retrieval

Finally, relationships implement semantics similar to that of statement in RDF. The data model, nevertheless, differ from RDF data model. Firstly, LOM metadata attributes are specified as properties of the LO class, rather than statements about the LO resource. Hence, attribute values are not identifiable and are not considered first class objects, as in the case of literals in RDF [6]. Aggregation and semantic relationships are modeled as collections of LOs. This is more general than in RDF, since it requires a bag resource and a specific list of predicates to model collections, leading to a complex representation and query evaluation. Moreover, relationship cardinalities in ROSA may restrict the participation of members in collections, which is not available in the RDF statement semantic. When representing, for example, the predicate *covers* of Figure 4a, which associates Query Language to Relational Algebra and Calculus in RDF, it is not possible to ensure that all the collection members would be visited when reaching the Query Language LO. On the other hand, the proposed model considers the collection members as a whole unit, as shown in Figure 4b, assuring that all its elements are effectively visited.

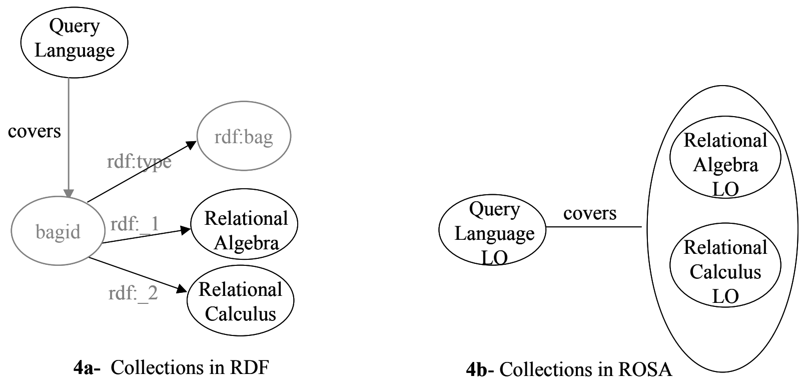

Figure 4: Relationship Collections

In addition, ROSA data model represents relationships as part of their instance data. So users may query associations and data indistinctly. This differentiates ROSA from OO data model, where relationships are part of the schema and cannot be queried.

3.2. The ROSA Algebra

In order to be complete, a data model for *LOs* requires the specification of valid operations, which is considered the model algebra. The algebra definition makes it possible to express a high-level query language into a formal canonical representation, as well as the conception of optimization strategies.

Many algebras have been proposed covering operations in different data models [4,5,13,14,6,7,]. This work presents an algebra, which, in some sense, borrows from all these works. Firstly, the majority of the operators are adapted from their relational algebra counterparts. Path expressions through LO relationships are similar to path expressions in the OO data model [5].

In this algebra, operators receive collections of *LOs* as input and equally produce collections of *LOs* as output. The closure of the model enables the composition of operators in an algebra expression.

The algebra includes operators to explore relationships between *LOs* as one of the semantic enrichments provided by the proposed model. This is achieved by exploring relationship type equivalence properties within the semantics of navigation operators. Relationships of aggregation type receive special treatment once their transitive property is known to the systems. Operators over domain specific type of relationships consider the equivalence class of its corresponding relationship type, such as: reflexive, symmetric and transitive. As an example, reflexive predicates include, implicitly, the association between a LO and itself through that predicate. Transitive predicates make it possible the evaluation of transitive closure operators and a symmetric predicate enables answering queries independently of the LO role in an association, as subject or object. The algebra includes operators to manipulate objects and collections selection and project operators, etc. [13].

4. The Query Language

Ad-Hoc data base queries are directly submitted to the ROSA system through ROSAQL query language. The system also includes a QBE like environment for submitting queries following a user-friendly interface. In the next subsection the ROSAQL query language is briefly presented.

4.1. The ROSAQL

return [$_{[p1[\wedge p2[\wedge..\wedge[pn]]]]}$ [C] (c)]
[**from** $E_i(C_i)$ [v] [in δ a_i] [, $E_j(C_j)$] [on δ]
[**where** {$_{(\delta ai)}C_i$, δ (collection-function),
$\delta_{(p1\wedge p2\wedge...\wedge\delta pn)}$}]
[**start at** [C_i with δpi] through [*relationship*, .] {up,$\underline{\text{down}}$} [until *limit*]]
[**order by** {LOM attribute } {$\underline{\text{asc}}$,desc]
[**union**]

a) *return* – specifies the complex resource structure to be produced. Its behavior is defined by the p_i term according to the algebra. In the absence of the *from* clause, a collection *c* may be specified as a source for the database collection.

b) *from* – specifies the input collections to be operated on by the query. A collection is specified by a pair (schema, class) where schema (E) identifies a database and class c is the collection class. A list of *n* terms in this clause should be followed by *n-1* join predicates between the collections. An instance variable *v* may be defined to iterate over a collection, in which case a selection may restrict the collection elements.

c) *where* – specifies a selection predicate over collections involved in the query. Selections may be specified over LOs metadata attributes, or over relationship identifications. The user may also write path-expressions to navigate through a list of relationships and define a terminal collection over which a predicate would operate.

d) *Start* – specifies the transitive closure operation. The input collection may be specified in the *from* clause in conjunction with the *where* clause or directly in the *start* clause. The semantic relationship defined to guide the inference must have the transitive property set, unless a dot '.' is specified, in which case an aggregation relationship is assumed. The inference stops when one of the events occurs: no more relationships to follow; no new LOs are obtained; a *limit* of recursion is achieved. The resulting collection contains LOs obtained from each recursive navigation. The clauses, *up* and *down,* indicate the path navigation direction to be followed. A *down* clause processes relationships from a *subject* LO to an *object* LO, whereas an *up* clause works in the opposite direction.

4.2. Some Query Examples

(a) Which Topics are covered by the *Database Course* and *are the basis for* other Topics?

```
return comprehends().isbasisfor() Topic  (IME.Course) c
 where c.id='Database'
```

(b) Which Topics do the Program MS in Computing Systems comprehend?

```
return Topic
from IME.Program p in p.id=  'MS in Computing Systems'
start at comprehends()
```

(c) Which Topics and Courses were created by 'Ana Maria Moura'?

```
return Topic
  where author= 'Ana Maria Moura'
union
return Course
  where author= 'Ana Maria Moura'
```

5. Implementation Issues

ROSA is being implemented as a semantic query processing system based on ROSA data model. The prototype extends the framework QEEF [1], designed to support new data processing requirements. This Section, briefly introduces QEEF and discusses the extensions required to adapt it to ROSA data model.

5.1. QEEF (Query Execution Engine Framework)

The objective of QEEF is to specify a framework that abstracts common aspects of query execution engines that take part in most of know implementations, as well as identify those that depend on specific requirements of application data model.

QEEF identifies five main components of a query engine that must be implemented for a new application: logical operators, physical operators and control operators, tuple data structure and data sources. Logical operators correspond to the implementation of algebraic counterparts; physical operators correspond to data access operations; control operations provide for flow control between operators and data source correspond to data providers. Developers of a new query engine identify the components that need to be implemented or borrowed from other implementations.

In order to implement ROSA using QEEF, a major concern was to model a tuple for representing LO structure and its relationships and the implementation for logical operators of ROSA algebra.

5.2. ROSA extension in QEEF

This prototype considers an XML representation for the storage of LO and its associations. Basically, a data source collection corresponds to the root of the XML tree, followed by LOM metadata and associations. LOM metadata data is represented in a subtree and each predicate leaving a LO is represented by an element with predicate cardinality represented as attributes. The ids of LOs taking part in a predicate collection are specified as child elements.

5.2.1. A LO tuple

Once an internal representation for objects in the data model has been conceived, it is possible to specify an adequate mapping for a tuple data structure, which is the basis for specifying the implementation of logical operations. As presented in other works[10], the representation of XML data into a tuple data structure is considerably more complex than in the Relational model. This is mainly as a result of the variable structure of an XML document, with missing or duplicate elements, which could produce tuples of different formats and sizes.

As a result, this implementation considers the concepts of pattern and witness trees as presented in [9]. Pattern trees are produced from references to data elements in the user query. It can be seem as a function *q(d):p*, that maps a document *d* by a query *q* into a pattern tree *p*, with the following properties:

- The root element of *d* is directly mapped into the root element of *p*;
- Each reference for a data element in *q* has its path included into *p*;
- If a reference to an element occurs as part of a final projection, then all referenced element subtree is also included into *p*, in the same relative position;

A resulting tree *r* is produced by mapping the pattern tree *p* against an input document *d*. A mapping may produce many *r* trees, as is the case of LOs with many authors (repeating elements), in which case, each occurrence of the author element produces a new tuple.

Having a model for representing LO tuples, it was possible to conceive an efficient data structure for holding tuples and accessing their values. The main concern was to have a structure where one could conduct a direct access to element values, without having to traverse the whole XML hierarchy.

Again, the intuition was to use access patterns defined by user queries as the basis for an efficient design. As an example, if a selection predicate is specified against a LO title attribute (LOM/general/title – LOM attribute hierarchy example), then the corresponding selection operation requires direct access to the title element. As such, the conceived tuple data structure is composed of two parts. The first part uses java *MapTree* data structure to implement a B+tree based access pattern to nodes in the *r* tree. The access keys for the *MapTree* correspond to the ids associated to the *r* tree nodes, traversed in pre-order, and assigned to them during the tree construction. A value is stored associated to each leaf node in the MapTree. For LOM attributes, this corresponds to the respective attribute value, whereas for associations (predicates), this value includes: cardinality attributes, and an array with ids of LOs taking part in that collection. Observe that the MapTree provides both direct access to elements according to their *ids* and a indexed sequential access for retrieving a whole subtree of LOs.

The second data structure offers direct association between a referenced element path and its corresponding *id* in the MapTree data structure. This is obtained through a java Hashtable collection, having access *paths* as keys and ids in the *r* tree as objects.

As a result of these two data structures, access to data during logical operation evaluation is standardized even having a variable XML representation of a LO. Access to each element node in the *r* tree is achieved by first getting its *id* in the Hashtable, using the access path and probing the MapTree using the *id*. Finally, a tuple stores the content of inner collections (refer to Section 3.1). When a join combines pairs of LOs they are physically store in a java *Vector* data structure.

5.2.2. Thesaurus support

Since ROSA data model and query language have been presented, we are now able to describe how a query is processed by the system, taking into account a domain thesaurus support.

In order to describe this processing, consider the query below, followed by its representation in ROSAQL:

a) Which LOs does the topic “Query-By-Example” cover?

ROSAQL (Q1): Return (IME.LO)l
Where l.comprehends().id= ‘Query-By-Example’

During execution, the following steps are performed:

Step1: Q1 is submitted to the system;

Step2: *Plan Generator* transforms Q1 into a QEP with operations in ROSA algebra:

$$\Pi\ (\sigma\ (_{(\text{comprehends().id='Query-By-Example'})}\ (\text{IME.Topic})\ )$$

Step3: Query engine evaluates the plan and returns results to user;

Now suppose the user is not satisfied with the LOs retrieved by Q1, and he wants to continue his search using a thesaurus support, from whom synonymous and associated terms can be derived. Figure 5 presents some terms of a thesaurus, designed in the context of the database domain. In ROSA, the user selects a thesaurus to support queries in a certain conceptual map.

The thesaurus is also modeled according to ROSA data model, where its association types are represented through domain specific relationships. The thesaurus is stored in Tamino using the same schema as the one used for storing LOs conceptual maps, allowing queries on thesaurus terms and associations to be processed uniformly within the system.

Step 4: the user chooses a reference term for his search, selected from RS1 or Q1. This corresponds to rephrasing his initial query with synonymous or associated terms, which are associated to the reference term.

Step 5: the new request is received by the query processing module that submits a query to obtain the synonymous and the associated terms from the thesaurus. In our example, the reference term “Query By Example” corresponds to “QBE”, as it is preceded by the association *USE* (or preferred term), and its associated term (*AT*) refers to “Relational Database”, as illustrated in the thesaurus (Figure 5);

Step 6: the query re-writing module rewrites Q1 with the terms obtained from the thesaurus search. Hence,

Q2: Return (IME.LO)l
Where l.comprehends().id= ‘QBE’

Q2 is transformed into ROSA Algebra and then resubmitted to the system. The same procedure is followed for the associated terms;

Step 7: after executing step 3 the user can still continue filtering his initial query Q1 as previously, or by visualizing the thesaurus in its systematic representation from a term selected from Q1 or RS1.

Step 8: the systematic thesaurus is presented to the user, who eventually re-submits the query using one of its terms.

Languages	Programming Languages
Programming Languages	BT Languages
OO Programming Languages	ST OO Programming Languages
C^{++}	QBE
JAVA	BT Query Languages
Relational Query Languages	UF Query By Example
SQL	AT Relational Database
QBE	Query Languages
QUEL	BT Languages
....	AT Database
	Query By example
	USE QBE BT: broader term
	ST: specific term
	…….. AT: associated term
(a) Systematic View	UF: use for
	(b) Alphabetic View

Figure 5: Terms of a Database Thesaurus

6. Related Work

The research on algebra languages has been very intense since Codd [4] presented the relational algebra. The majority of the algebras proposed since then adapted the basic relational operators set to new data models, in addition to proposing new operators to deal with special aspects of the data model.

Research on e-learning data models is still in its infancy and, to the best of our knowledge, no previous work has proposed a complete algebra for query languages in this domain. A very interesting project, however, is EDUTELLA [11]. Edutella is a RDF infra-structure for P2P-networks aiming at supporting educational resources sharing between institutions. Resources and their relationships are represented through a data model, based on the Datalog non-procedural query language. It expresses a RDF model as logical predicates in DataLog, which forms a knowledge base. Queries and results are expressed in accordance to the Datalog data model. Comparing to ROSA, EDUTELLA is built with the intention of integrating peers collections of educational material following a metadata standard, such as IEEE LOM. Its data model reflects a more generic view of LOs associations, whereas in ROSA, our intention is to conceive a data model that will give support to a flexible and meaningful LO conceptual model. In addition, the ROSA data model builds on the established success of relational and object-oriented data models.

In respect to works on algebras for the Semantic Web, RAL is an algebra for querying resources described through RDF [6]. RAL models RDF as a finite set of triples composed of resources associated through properties, which form a directed labeled graph. The model integrates RDF and RDFS definitions. Nodes in the graph are either resources or literals, both labeled by unique identifiers. The algebra operators receive a collection of nodes and process then with adapted relational operators, which include: projection, selection, cartesian product, join, union, difference, intersection and loop operators that iterate over collection nodes. The operations navigate through the RDF graph nodes using uniformly the projection operation to access object values from resources, playing the role of subjects in triples.

ROSA data model differs from RAL in many aspects. Firstly, the model accepts attribute definition for LOM metadata attributes, whose literal values are not identifiable. In addition, although conceptually forming triples, the data model represents relationships as one-to-many associations that take part in the LO structure. Moreover, associations in ROSA may be restricted by maximum and minimum cardinality definitions, and properties qualify associations as reflexive, symmetric and transitive.

These aspects extend RDF data model towards a more semantic model. In the context of the algebra, ROSA includes path expressions and transitive closure operations that provide inferences on the intentional data model. Another important contribution in our work concerns the query processing supported by a domain thesaurus, not contemplated in other similar works.

7. Conclusion

This paper presented ROSA, a data model and a query language to access LOs, based on their semantic properties. The system includes a powerful data model whose algebra is completed described. A collection of LOs is the main processing unit in the model, in which LOs are described according to IEEE LOM metadata attributes and LO relationships. The proposed data model extends RDF by: simplifying the representation of attributes, which simplifies query formulation; considering relationship equivalence properties, implicitly used during query evaluation; modeling relationships as n-ary ordered collections and restricting LO access in a collection through maximum and minimum cardinalities. Applications interface with ROSA through ROSAQL, a query language adapted from SQL, OQL and RQL to support ROSA algebra operations.

A first protoype of ROSA has been implemented using Tamino native XML DBMS Tamino 4.0 as a repository of ROSA objects according to ROSA data model. The QEEF, query execution engine framework, was extended to support ROSA algebra and data structures.

There is a vast spectrum of research issues to be addressed in the near future. The algebra can be extended to support inferences and analysis on the database schema. Equivalence rules and heuristics must be conceived in order to enable queries optimization.

References

[1] Ayres, F.M., Porto, F., Melo, R., Na Extensible Machine for Supporting New Query Execution Models (In Portuguese), XVIII Brazillian Database Symposium, Manaus, Oct., 2003.

[2] Advanced Distributed Learning Sharable content Object Reference Model Version 1.2 – The Scorm Overview, http://www.adlnet.org.

[3] The British Council, Educação a Distância, 2001 available at www.britishcouncilpt.org/education/distance.htm.

[4] Codd, E.F., A relational model of data for large shared data banks, Comm. Of the ACM, 13(6), June 1970, pp. 377-387.

[5] Catell, R.G.G., Barry, D.K., Berler, M., Eastman, J., Jordan, D., Russell, C., Olaf, S., Stanienda, T., and Velez, F., editors. Object Data Standard ODMG 3.0. Morgan Kaufman, January 2000.

[6] Francincar,F., Houben G., Vdovjak R., RAL: an Algebra for Querying RDF, 3rd Conf. on Web Information Systems Engineering, Singapore,12-14 Dec. 2002.

[7] Francincar,F., Houben G., Pau C., XAL: an Algebra for XML Query Optimization, in Database Technologies 2002, 13th Australasian Database Conference, Volume 5 of Conferences in Research and Practice in Information Technology, pp. 49-56. Austraolian Computer Society Inc., 2002.

[8] Friesen N., What are Educational Objects?, Interactive Learning Environments, Vol. 9, No. 3, Dec. 2001.

[9] http://ltsc.ieee.org/wg12/, last access 31/01/2004

[10] Jagadish, H.V., Al-Khalifa, S., Chapman, A., et al., TIMBER: A native XML database, The VLDB Journal, V(11), N(4), Dec., 2002.

[11] Nejdl, W., Wolf, B., Qu, C., et all, "EDUTELLA: A P2P Networking Infraestruture Based on RDF, 11th Intl. World Wide Web Conference, Honolulu, Hawaii, USA, May, 2002.

[12] Resource Description Framework (RDF) Model and Syntax Specification. 1999. http://www.w3.org/TR/PR-rdf-syntax/, 1999. Last access Oct. 2001.

[13] Fábio S. Coutinho, Fábio André M. Porto. Processamento de Consultas no Modelo ROSA. XIX SBBD out. 2004, Brasilia, DF.

[14] W3C (World Wide Web Consortium), The XML Query Algebra, Working Draft, May 2000,http://www.w3c.org/TR/2000/WD-query-algebra-20001204, last access, June 2003.