ROSA: A Data Model and Query Language for e-Learning Objects

Fábio Porto[1], Ana Maria de C. Moura[1], Adriana P. Fernandez[1], Abílio Fernandes[1], Fábio José Coutinho da Silva[1], Gilda Helena Bernardino de Campos[2], Laura Coutinho[2]

[1]Military Institute of Engineering
Rio de Janeiro, RJ, Brazil
{fporto, anamoura}@de9.ime.eb.br

[2]CCEAD / PUC-Rio
Rio de Janeiro, RJ, Brazil
{gilda, laura}@ccead.puc-rio.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, Tamino. Together, ROSA and Tamino, offer instructional designers a semantic view of e-learning content. In this paper, we present ROSA Data Model and Query Language, designed as an extension to RDF data model and RQL query language. The Data Model is structured around the LO modeling and their relationships, adapted to the e-learning domain. An algebra defines valid operations over LO data. Queries are formulated in ROSAQL that extends RQL with joins, graph navigation and recursion.

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 [1]. This education mode resorts to various educational technologies to facilitate both the learning and the communication processes between tutors and learners.

LCMSs are being developped 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 [2]. A LO is identified by a set of metadata descriptors established by an international metadata standard, such as LOM (Learning Object Metadata) [3] and SCORM (Sharable Content Object Reference Model) [4], 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 project uses Semantic Web technology to enable semantic query capability.

The Semantic Web is built upon two W3C standards, corresponding to the second and third layers respectively of Berners-Lee's architecture [5]: the XML language [6], which provides standard syntax for data representation, ensuring data exchange; and the framework Resource Description Framework – RDF [7], a standard data model for expressing structure and representing metadata vocabularies, serving as a metadata language for the Semantic Web. This model allows the user to define relationships between resources as semantic graphs on the Web, using XML as the syntax language to transport data. In addition, RDF includes a schema, named RDF Schema (RDFS) that can be used to define application specific vocabularies, defining taxonomies of resources and properties used by specific RDF descriptions.

In this paper, our attention is focused on proposing a data model and query language 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 associations. The latter are modeled as an extension of RDF statements. ROSA data model provides a powerful algebra that makes it possible to query over LOs metadata, as well as over the semantic associations between LOs. ROSA is designed as a semantic layer middleware on the top of a DBMS. The project is sponsored by Consist, which provides the Tamino XML native DBMS as a storage layer [8]. As a result, some of the propositions in this paper are influenced by the target XML data model, although mappings to other data models may also be envisaged.

The rest of the paper is organized as follows: section 2 presents a use case that serves as a query benchmark 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 presents the ROSAQL query language and section 5 presents related work. Finally, section 6 concludes the paper with additional comments and future work.

2. Use case

This section presents a use case for ROSA project. It comprehends a RDF style representation of a LO conceptual map for a Master program in Computer Systems, a class diagram and a list of representative queries that should be answered by this system.

The use case data and queries reflect ROSA objective of supporting the design of e-learning courses. The LOs metadata and their relationships offer a rich semantic model for searching the repository during class preparation. On the other hand, it is not designed as a support for running courses, which would require execution time primitives not found in this use case.

2.1 The LO Conceptual Map

Figure 1 presents a simplified LO conceptual map for the Master Program in Computing Systems and the corresponding schema describing and classifying LOs and their relationships. The map is represented through a directed-labeled graph where nodes represent LOs, identified by their names. Directed labeled edges model relationships between LOs, like predicates in RDF. In this map, some LOs are associated to their classes in the schema through a dashed line.

The classes Program, Course and Topics represent domain specific types of LO. They model the main composition structure of courses in a certain school program. Other relationships, such as *prerequisite* and *is basis for* also appear in the map representing a specific domain knowledge defined by the user. In addition, the classes e-Learning Program, Master Program and PhD Program specialize the Program class. A LogicalLO represent a collection of LOs and may contain many PhysicalLOs. A PhysicalLO contains physical media used to prepare a dicdatic material, such as image, documents, ppt files, etc. PhysicalLOs are not represented in the Conceptual map.

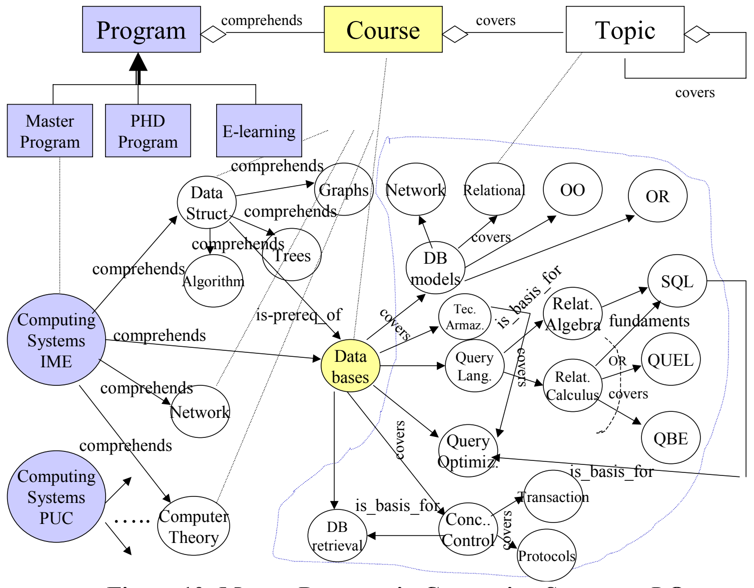

Figure 10: Master Program in Computing Systems – LO conceptual map

Metadata descriptors of *LO* classes correspond to a subset of the *LOM* metadata standard[1] [3].

In order to give a general idea of the semantic meaning expressed by the *LO* conceptual map in Figure 10, the following assertions can be made:

- the Database course covers the topics Database Models ;
- the topic Relational Calculus fundaments the topic SQL;
- the topic Storage Techniques is basis for the topic Optimization;

The verbs in these assertions (relationships) are taken from a controlled vocabulary (thesaurus) specified within an application domain. The next subsection lists relationships considered relevant to the use case domain.

2.2 Domain predicates

The relationships considered in the *LO* conceptual map in Figure 10 and their synonyms include:

a) Aggregation predicatescomprehends (has, covers, is part of , is made of, characterizes, contains, includes, etc.);Domain specific predicates
   a. prerequisite
   b. fundaments (is basis for, is condition for, etc.)
   c. requires (opposite of prerequisite)
   d. implies (determines, leads to, derivates, influences, allows for, etc.)

[1] LO attributes (metadata descriptors) have been omitted in this paper due to space restrictions

2.3 Queries

In this section, queries covering e-learning design activity complete the use case with data access requirements. The use case supports queries exploring LO metadata and the semantic relationship between LOs. Queries are classified and listed bellow:

1. LO metadata projection
    a) Which are the *difficulty levels* of the registered *Topics?*
    b) What are the *formats* of PhysicalLOs of type *File?*
2. LO projection
    c) Which are the *Program LOs* registered*?*
3. Semantic relationships projection
    d) Which are the *associations* established with *Program LOs* ?
4. Projection of semantically associated LOs
    e) Which are the prerequisite *Courses* for the *Database Course*?
5. Projection of aggregated LOs
    f) Which Topics does the Database Course comprehend?
6. Aggregation navigation
    g) Which courses are parts of each Program?
7. Selection on metadata
    h) Get *LOs created* since '01/01/2003'?
    i) Which Topics are classified with level 'b'?
8. Selection on relationships
    j) Which LOs are associated to the Database Course?
    k) Which LO does comprehend the Database Course?
    l) Which LOs do fundament the Topic 'SQL'?
    m) Which LOs classified with level 'b' does the Database Course comprehend?
9. Selection on relationship identification
    n) Which are the fundamental LOs?
10. Recursive navigation
    o) Which Topics do fundament SQL?
    p) Which Topics do the Program MS in Computing Systems comprehend?
11. Merge
    q) Which *Topics* are present at both IME and PUC institutions?
12. Physical LO retrieval
    r) Which Physical LOs does the Topic Transaction Management cover?

This system does not intend to support a natural language query interface as these queries may suggest. Rather, use case queries are useful to measure system capabilities and to validate the algebra and query language proposed in this paper. Queries are directly submitted by a user interface or by an ad hoc interface using the ROSAQL query language (see Section 4.1).

3. Data Model

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

A database comprehends a database schema and database instances for a certain user domain. A database schema, or simply schema, identified by a *name*, includes definitions and rules used to create objects and relationships between them in a certain domain.

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

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

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 can not be composed by other LOs, whereas LogicalLOs may contain a collection of other LOs associated by aggregation and semantic relationships.

Associations between *LOs* are specified through *Relationships*. There are two types of relationships: aggregation and semantic. Aggregation relationship models composition of *LOs*. Its semantic is fixed and known to the model. On the other hand, semantic relationships establish specific domain associations between *LOs*. Relationships are also identified by a unique complex resource identification corresponding to the verb it represents, as specified in the domain thesaurus.

The associations of *LOs* expressed through relationships conceptually define triples *t(subject, property, object)*, where *subject* and *objects* are of type *LO* and *property* is classified as a *relationship*. The schema specifies valid associations through triples T(class, relationship type, class). Thus, valid instances of *t* are in accordance to *T*.

The cardinality of associations is instance dependent. Typical association cardinality is *one to many* but restrictions may limit the number of elements playing the role of objects in a association. For this reason, relationship instances (properties) are modeled as collections with objects as their elements.

Relationship types are specialized according to the type of collection they implement. Aggregations may implement the semantics of bag, set, list and 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.

Note that the collection abstraction provides a modeling construct for a whole complex LO structure. This is important as long as the subject LO requires a specific navigation over aggregated LOs . This construct is essential; otherwise we cannot ensure that required LOs will be followed during an instruction.

The classes set and bag 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 10, for instance, the *exclusive or aggregation* relationship between *Relational Calculus* and *SQL, 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 *minimum cardinality=unbounded.*

Finally, relationships implement semantic 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 [9]. Aggregation and semantic relationships are modeled as collections of LOs. This is more general than in RDF where only aggregation type of relationships are defined as collections. Moreover, relationship cardinalities in ROSA may restrict the participation of members in collections, which is not available in the RDF statement semantic.

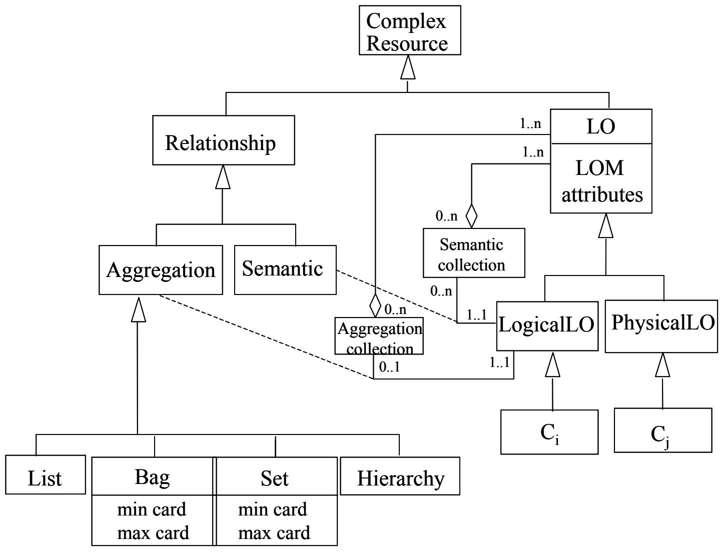

Figure 11: The Rosa Data Model

3.2 Algebra

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

Many algebras have been proposed covering operations in different data models ([10], [11], [12], [13], [14]). 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 [11].

In LO 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. Aggregation relationship receives special treatment once its semantic is fixed and known to the systems.

Semantic relationship operators provide the processing of domain specific associations. This processing considers the classification of predicates 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.

A list of operators and their semantics are presented next:

Object functions

Object functions implement per object properties access. Three types of object functions are proposed corresponding to access to LO attributes, aggregated members and semantically associated members.

- **gettype()** – obtains a literal corresponding to the type of the current LO.
- **getLOM_attribute$_i$()** – invoked on a complex resource, returns a literal corresponding to the value of the attribute named *LOM_attribute$_i$* in the current object.
- **semantic() –** invoked on a complex resource, returns a bag of LOs associated to the former through any type of semantic relationship.
- **semanticrelationship$_i$() –** invoked over a complex resource, returns a collection of LOs associated to the current complex resource through the semantic relationship named *semanticrelationship $_i$*.
- **comprehends()** - invoked over a complex resource, returns a collection of LOs that take part in an aggregation relationship within the current complex resource.
- **property() –** invoked on a complex resource, returns a literal corresponding to the *ids* of the associations found in the complex resource.

Relationship fuctions

- **getId()** – invoked on relationship objects, returns the literal corresponding to the value of its *id* attribute.

Collection Operations

Operations on collections are type preserving in respect to the input collection. The collection resulting from a semantic relationship is always of type bag. LOs obtained from navigating through an aggregation relationship form a collection with type and restrictions inherited from the relationship. The union of collections of different types results in a collection of type bag.

The processing of a collection may require the invocation of object functions over collection members. When these functions return collections, a set-collapse operator groups all collections into a single one.

Source ***($c_{(expression)}$)*** – This operation produces a collection of LOs from an expression. The expression syntax is left to be defined by an implementation. It is equivalent to its homonym operation in [15].

getNext ***($_{[r1,..]}c$)*** – This function obtains the next LO from a collection *c*. The navigation order is defined by the operation implementation according to the collection type. If the collection has a maximum cardinality *n* specified, then a set of LOs {r_1, $r_{2,..,}$ r_k}, where k ≤ n, identifies the interesting LOs, otherwise, the first *k* LOs are retrieved.

Projection $\Pi_{[p1[^\wedge p2[^\wedge..^\wedge[pn]]]\ ]}$ (c) – This operation receives a collection *c* of type *C* of LOs and produces a new collection with complex resources according to *p*, which defines the projection behavior. The behavior associated to *p* modifies the resulting collection structure. The specification of a list of p_i defines a conjunction of each p_i behavior. A list of projections p_i leading to collections with elements of incompatible types are invalid.

The different projection behaviors are presented bellow, including the formulation of the projection operation for the corresponding use case queries.

a) Structure projection - $\Pi_{(a1,a2,...,an)}$ (c)
   if *p* is a list of LO attributes (i.e. (a_1,a_2,...,a_n)) then the resulting complex resources structure follows the attribute list and the attribute values are obtained by invoking the *getLOM_attribute$_i$()* function on each object in *c*, for **≤** i **≤ n.**
   - Which are the *difficulty levels* of the registered *Topics?*
     - $_{(difficultylevel)}$ (*IME.Topic*)
   - What are the *formats* of PhysicalLOs of type *File?*
     - $_{(format)}$ (*IME.File*)

b) LO projection - Π (c)
   If p is not present, then the resulting collection is a copy of the input collection.
   - Which are the registered *Programs*?
     - (IME.Program)

c) LO projection through generic association - $\Pi_{(semantic())}$ (c)
   If p is the generic *semantic()* function the resulting collection is the result of the union of all the collections obtained by invoking the respective function on each element of *c.*
   - Which are the *Topics* associated to the *Database Course*?
     - $_{(semantic())}$ (*c*) – where c is a collection composed of the *Database course* LO.

d) Semantic relationship projection - $\Pi_{(property())}$ (c)
   if p is the property() function, then the resulting collection will include all complex resources of type *Relationship* with id attribute value obtained from invoking the *getid()* function on the semantic relationship objects associated to the objects in c.
   - Which are the *associations* established with *Program LOs*?
     - $_{(Property())}$(IME.Program)

e) Projection of semantically associated LOs - $\Pi_{(semanticrelationshipi())}$ (c) if *p* is a semanticrelationship$_i$() function, the resulting collection is the union of the collections obtained by invoking the corresponding function on each element of *c*.

- Which are the prerequisite *Courses* for the *Database Course*?
    - $_{\text{(prerequisite)}}$ (c) – where *c* is a collection composed of the *Database Course* LO.

f) Projection of aggregated LOs - $\Pi_{(comprehends())}$ (c)

if p is the comprehends() aggregation relationship function, then the resulting collection will contain LOs obtained by applying this function on each object in c.

- Which Topics does the Database Course comprehend?
    - $_{\text{(comprehends())}}$ (c) - where *c* is a collection composed of the *Database Course* LO.

g) Projection on a path expression - $\Pi_{\text{(semanticrelationshipi()..semanticrelationshipn ())}}$ (c)

- Which Topics are covered by the *Database Course* and *are the basis for* other Topics?
    - $_{\text{(comprehend().isbasefor())}}$ (c) – where c is a unitary collection containing the DatabaseCourse LO.

Selection $\sigma_{(\delta ai)}$ *(c)* – evaluates predicates over LO metadata attributes in a collection *c*. The collection may be obtained by traversing semantic or aggregation types of relationships. In each case, selected LOs are those that evaluate true in an existential quantification predicate. Predicates are formulated as a combination of logical operators (and, or, not) and comparative operators ($<,>,\geq,\leq, = ,\neq$) on attribute values. Predicates over LOs taking part in relationship collections require the invocation of collection functions. This algebra considers an instance variable associated to the collection objects so that conjunctive predicates are implicitly defined over instance variable attributes.

Relationship properties provide alternative paths for the selection operations. Firstly, reflexive relationships include the subject LO in the collection of LOs to be evaluated by the selection predicate. Selection operation on collections associated through symmetric relationships may opt for evaluating predicates without having to follow the relationship association, as both ends will appear as subjects of that relationship.

a) Selection on metadata attribute $\sigma_{(\delta ai)}$ (c)

For each object in c, evaluate the predicate δ on LOM attribute a_i, including in the resulting collection LOs that evaluate as true.

- Which Topics are classified with level 'b'?
    - ($\sigma_{\text{(difficultylevel= "b")}}$ LO)
- Get LOs created since '01/01/2003'.
    - ($\sigma_{\text{(creationDate>"01/01/2003")}}$ LO)

b) Selection on elements in relationship collections $\sigma_{(\delta\text{semantic().attribute}i())}$ (c).

For each LO, apply the *semantic()* function obtaining a collection of LOs semantically associated to the subject LOs. For each LO found obtain the value of attribute$_i$ and evaluate the δ predicate over it. For those evaluated as true, return the subject LO.

- Which are the *LOs* associated to the *Database Course*?
    - (σ ($_{\text{(semantic().id='Database')}}$ (IME.Course))

c) Selection through a specific relationship σ (δsemanticrelationshipi ().attributei) (c)
For each LO in *c*, apply the semanticrelationship() function obtaining a collection of semantically associated LOs. For each LO found obtain the value of attribute$_i$ and evaluate the δ predicate over it. For those evaluating true, return the subject LO.
  - Which *LOs* does comprehend the *Topic* "SQL"?
    - Π (σ ($_{\text{(comprehend().id='SQl')}}$ (IME.Topic))
  - Which are the *LOs* that *fundament* the *Topic* "SQL"?
    - Π (σ ($_{\text{(fundament().id='SQl')}}$ (IME.Topic))

d) Selection on relationship identification σ $_{\text{(δproperty ())}}$ (c)
For each LO in *c*, evaluate the existence of an association with a specific *id*. If it evaluates as true, return the LO to the resulting collection.
  - Which are the fundamental LOs?
    - Π (σ ($_{\text{(property()='fundament')}}$ (IME.LO))

e) Selection on a path expression σ $_{\text{(δ(semanticrelationship}i\text{ ()..semanticrelationshipn ().ai)}}$ (c)
For each LO in *c*, obtain *the* collection by iteratively invoking the semantic relationships in the predicate list. The predicate is evaluated over attribute a_i of LOs found at the end of the relationship path.
  - Which Topics created by "Ana Maria Moura" are covered by the Database Course and are the basis for other Topics?
    - Π σ $_{\text{(δ(comprehend().isbasefor().author='Ana Maria Moura')}}$ (c) – where *c* is a unitary collection containing the *DatabaseCourse* LO.

f) Selection on all relationship components
  - Which LOs classified with *level* 'b' *does* the *Database Course comprehend*?
    - Π (σ ($_{\text{(level='b')}}$ (Π $_{\text{(comprehends())}}$ (σ $_{\text{(id='Database')}}$ (IME.Course)))))

Transitive *Closure* φ $_{\text{(relationship [iteration max], up|}\underline{\text{down}}\text{)}}$ (c) – obtain a collection by recursively invoking the referred relationship over LOs. Relationships must qualify as transitive property. Aggregation relationship is transitive by definition. Other semantic relationships must have transitive property set. The transitive closure iteration ends when either there are no more paths to follow or, no more nre LOs are produced, or an iteration limit, such as *iteration max*, is reached. The path from a LO may follow a *down* direction, default, where relationships are followed from LOs playing the subject role towards LOs playing object roles, or the opposite direction, up, conversely.

a) Transitive closure down
  - Which *Topics* do the *Program MS in Computing Systems comprehend*?
    - Π φ $_{\textit{(comprehend())}}$(c) – where *c* is a unitary collection containing the LO *MS in Computing Systems*.

b) Transitive closure up
  - Which *Topics* do *fundament SQL*?
    - Π φ $_{\textit{(fundament(), up)}}$(c) – where *c* is a unitary collection containing the *SQL* LO.

Union ($c_1 \cup c_2$) - The resulting collection contains the distinct LOs in collections c_1 and c_2. The LO equality operation considers the identification attribute value for duplicate elimination.

- Which Topics and Courses were created by 'Ana Maria Moura'?
  - Π ($\sigma_{(autor=\text{"Ana Maria Moura"})}$ (*Topic*) $\cup$ $\sigma_{(autor=\text{"Ana Maria Moura"})}$ (*Course*))

Join $\bowtie$ $(c_1)_p(c_2)$- The resulting collection includes the LOs in collections c_1 and c_2 whose attribute values match the predicate in *p.*

- Produce LOs from collections *Topic* and *Course* that agree on their authors?
  - Π ($\bowtie$(t $_{Topic}$) $_{(\text{t.author=c.author})}$ (c Course))

4. Query Language

Ad-Hoc data base queries are directly submitted to the ROSA system through the 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 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,<u>down</u>} [until *limit*]]
[**order by** {LOM attribute } {<u>asc</u>,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 a *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 identifies a database and class 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 occur: no more relationship 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.

e) Order by – orders LOs according to a list of LOM attributes. If needed, the resulting collection is coerced to be of type *list*.

4.2 Query Examples

a) Which are the difficulty levels of the registered Topics?

```
        return difficulty-level (IME.Topic)
```

b) Which are the registered Programs?

```
        return  (IME.Program)
```

c) Which are the Topics associated to the Database Course?

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

d) Which are the prerequisite *Courses* for the *Database Course?*

```
        return prerequisite() Course (IME.Course) c
               where c.id='Database
```

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

```
        return comprehend().isbasefor() Topic  (IME.Course) c
               where c.id='Database'
```

f) Which Topics are classified with level 'b'?

```
        return Topic (IME.Topic) t
               where t.level='b'
```

g) Which *LOs* does comprehend the *Topic* "SQL"?

```
        return (IME.LO) l
               where l.comprehend().id = 'SQL'
```

h) Which Topics created by "Ana Moura" are covered by the Database Course and are the basis for other Topics?

```
        return Topic
        from IME.Course c in c.id=´database'
               where c.comprehends().author='Ana Moura' and
             c.isbasisfor()
```

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

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

j) 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'
k) Produce LOs from collections Topic and Course that agree on their authors?
        return  LO
        from Topic t, Course c on t.author=c.author
```

5. Related Work

The research on algebra languages has been very intense since Codd [10] presented the relational algebra. The majority of 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 an algebra for query languages in this domain. A reasonable comparison could be made with algebras proposed for the RDF data model. RAL is an algebra for querying RDF [13]. 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 represent relationships as one-to-many associations that take part in 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. Regarding the algebra, ROSA includes path expressions and transitive closure operations that provide inferences on the intentional data model.

6. Conclusion

Nowadays, Distance Education (DE) is a rapidly expanding mode of education. Many courses traditionnally taught in a classroom environment by a teacher, are now being transformed into an electronic sibling, in order to be adapted to this new education methodology. Learning objects have been used as a promising solution to develop standard infrastructures enabling tutors to create and organize their didactic material.

This paper presented ROSA, a repository system to store and access LOs, based on their semantic properties, expressed in RDF. The system includes a powerful data model whose algebra is completed described. In order to test the algebra capability, a query language, named ROSAQL is under development. We use the semi-structured DBMS Tamino 4.0 [8], a native XML DBMS to store LOs. The use of the XML data model for representing RDF sentences provides an easy data interchange and query capabilities.

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. There is also a study, already in course, for integrating a Thesaurus to the system helping users with terms relationships.

References

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

[2] Jacobsen, P. E-learning Magazine http://www.elearningmag.com/elearning/article/articleDetail.jsp?id=5043.

[3] IEEE: "Draft Standard for Learning Object Metadata"; 15 July 2002.

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

[5] The Semantic Web - XML2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html.

[6] W3C (World Wide Web Consortium). XML - Extensible Markup Language. http://www.w3.org/XML/ - Last access Jan., 2003.

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

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

[11] 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.

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

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

[14] 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

[15] Jeffrey Naughton, David DeWitt, David Maier et all, The Niagara Internet Query System, IEEE Data Engineering Bulletin 24(2): 27-33 (2001)

[16] M. Fernandez, J. Simeon, P.Wadler, A semi-monad for semi-structured data, Int'l Conference on Database Theory, London, January 2001