LMDQL:Link-based and Multidimensional Query Language

Paulo Caetano da Silva
Central Bank of Brazil
Av. Garibaldi, 1211, 40.176-900
Salvador – Bahia – Brazil
paulo.caetano@bcb.gov.br

Valéria Cesário Times
Federal University of Pernambuco
Center for Informatics
PO BOX 7851, Brazil
vct@cin.ufpe.br

ABSTRACT

The current commercial and academic OLAP tools do not process XML data that make use of XLink. To develop OLAP systems for helping in the analysis of such data, this paper proposes an analytical query language, namely LMDQL (Link-based and Multidimensional Query Language). Before presenting LMDQL, a multidimensional data model based on XLink is given. Then, the LMDQL syntax is detailed together with its operators and a discussion on its processor architecture. Finally, a case study based on XBRL Dimensions documents is given. XBRL can be seen as a standard used in the publication of financial data over internet and contains linkbases based on XLink.

1. INTRODUCTION

The use of XML (eXtensible Markup Language) as an alternative for integrating heterogeneous data sources has made this technology a standard for data interchange on the Internet. Therefore, XML documents are a rich source of information for organizational decision making. Similarly, the use of Data Warehouses (DW) and OLAP (On-Line Analytical Processing) tools [3] allows the identification of tendencies and standards, to better conduct the companies' businesses.

In XML, it is possible to represent information semantically similar in different ways. This leads to three kinds of data heterogeneity: (i) semantic, where similar information is represented through different names, e.g. enterprise and company, or dissimilar information through equal names, e.g. virus in the informatics field and in the health field; (ii) syntactic, where the semantically equal content is represented in several ways. For example, in different languages or in diverse measure units, e.g. meters and feet; and (iii) structural, in which data is organized in diverse structures, e.g. in different kinds of hierarchies, in attributes or in elements [12]. This representation flexibility is important, however it makes the use of OLAP concepts about XML data a complex task.

Applications and technologies derived from XML use XLink (XML Linking Language) [17] as an alternative for representing the information semantic and structure, expressing relations between concepts, which are usually defined in a scheme based on XML Schema [18]. A problem that occurs when processing documents that have chains of links, based on XLink, is the fact that the available query languages do not provide support for navigation on them. Although the XPath (XML Path Language) [19] is widely adopted as query standard in XML documents, it does not provide such resource. Several works have been developed with the analytical queries purpose (OLAP) over XML data [1], [2], [12], [20], [9], [21]. However, these researches do not take the use of XLink in XML into consideration.

This article presents LMDQL (Link-based and Multidimensional Query Language), an XML data query language, based on XPath+ [14], [15], an XPath extension for navigation over links based on XLink. From the analysis of the related works discussed in this article, the existence of a query language to XML data, which allows the following, has not been verified: (i) use of a collection of XML documents; (ii) perform queries based on the value or on structure of the XML document; (iii) consider the existence of linkbases, groups of links, as a source of information; (iv) perform analysis based on indexes determined by the data cube; (v) express values that indicate percentage relations among other values in the data cube; (vi) create operators based in others previously created, in execution time, and which can be stored for future queries; (vii) be based on a data model, defined by XML Schema and XLink, to solve heterogeneity conflicts in XML; (viii) be an extension of the MDX [10], a standard market language for performing OLAP queries. For this reason, the development of an OLAP system for XML, which uses XLink, consists in the motivation to perform this work.

In section 2, the specification of the LMDQL query language is presented. Section 3 approaches a case study with XBRL (eXtensible Business Reporting Language) [6]. Conclusions and future works are listed in section 4.

2. LMDQL

To perform navigation over links and recover information contained in linkbases, LMDQL uses XPath+. The performance of one XPath+ expression, detailed in [14], [15] supplies a node list with information about link's origin and destination. XPath+ provides a group of axes and functions to retrieve information contained in the XLink links. The LMDQL characteristics are discussed next. Section 3.1 discusses the data model, in which LMDQL is based. In section 3.2, syntax, semantics and operators of LMDQL are given. Its processor architecture and the implementation aspects are presented in sections 3.3 and 3.4.

2.1 A Multidimensional Data Model based on XLink

Because of the flexibility inherent to the XML technology, different forms of cube definition and data model for DW can exist. For this reason, the heterogeneity problems of XML data are made evident. However, the use of XLink and XML Schema may solve such problems through the specification of dimensions, facts and cubes. The data model on which LMDQL is based, extends the XBRL Dimensions concepts to other domains, through the inclusion of linkbases, elements, and attributes.

An XML data base with Xlink is made of schemas, linkbases and instances. The first ones specify the elements that represent the facts, the dimensions, the dimension members and the cubes. The linkbases establish the relation between members and dimensions and between dimensions and facts, establishing the possible cube combinations that can exist in the instance. In the instance occur the observed facts, which, combined with the measure dimension members, determine a data cube. The XML instance document, which may contain one or more cubes, has a dimensional

structure, on which the contexts with the dimension members are represented, and another one not dimensional, with the facts measures. Figure 1 illustrates the data organization according to this model. The elements to be used in the XML instance are specified based on XML Schema. The relations among the instance elements and among these and other resources are expressed in linkbases. The *Fact Schema* and *Hypercube Schema* documents define the elements that represent the facts and member, dimension and cube. Regarding the model organization, these schemas could be specified in one or in distinct documents. In the instance elements that represent the facts, the presence of the *contextRef* and *unitRef* attributes is obligatory, which refer to the multidimensional context and to the fact unit, represented by the elements *context* and *unit* of the dimensional structure.

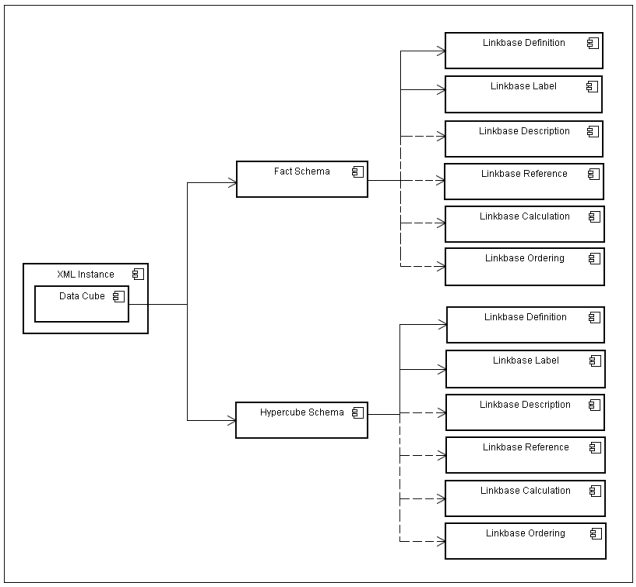

Figure 1. Data Model for LMDQL

Regarding structural aspects, the definition of relations among dimensions, the members and the cubes is performed by linkbases. The relations expressed by the arcrole *domain-member* sorts the possible members of a domain, which is associated with the dimension through the *arcrole dimension-domain*. The cross product from dimensions and facts, to establish the possible cubes to be used in the instance, is defined by the arcrole *hypercube-dimension*. For including the measures in the cube, the *arcrole all* is used. For excluding the member of a domain in a cube specification, the *arcrole notAll* is used. These relations are specified in the Definition linkbase.

In this data model, the heterogeneity questions on XML data are solved as follows: (i) in semantics, the Label linkbase establishes one or more labels for an element defined in the scheme. Consequently, one element can have several labels, and distinct elements, in different domains, can have the same label. Thus, one application can perform processing either through the element or its label; (ii) in syntactic, the Label linkbase allows the definition of names in different languages for the same element and the *unit* attribute allows informing the unit regarding the measure; and (iii) in structural, the scheme defines elements, its attributes and deriving elements. The Definition linkbase specifies the hierarchy among the elements, determining the structure of XML data. Regarding the other linkbases, the Ordering linkbase determines the presentation and the processing orders of the elements. The Description linkbase supplies a textual description of a relation. The Reference linkbase defines the elements that represent references related to the instance elements. The Calculation linkbase expresses arithmetic relations.

Besides solving the heterogeneity problems, this data model is flexible enough to be used in other domains and to incorporate new relations not predicted by its linkbases.

2.2 The LMDQL Syntax

The LMDQL syntax is based on MDX, with changes performed to become suitable to the XML documents context with XLink support.

2.2.1 Definition of Query Statement

The LMDQL queries extend MDX to perform queries based on the value or on the XML document structure. To do so, the clause $VARIABLE has been specified, which declares the paths that will be used on recovering the members of the document. The other clauses of the query forming expression remain as originally defined in MDX. One query based on the structure can be evaluated through the trees shown in Figure 2, displaying two representations of bank assets, considering that, in the tree on the left, the classification between public and private banks occurs. The LMDQL query, shown in Chart 4, is performed on a collection of XML documents, named *Banks*, so as to recover the total of the banks assets in both structures.

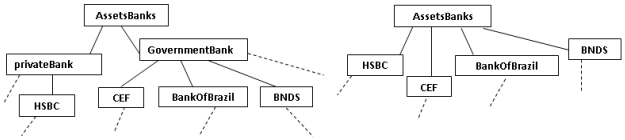

Figure 2. Organizational Structure Tree Representations.

Chart 4. LMDQL Query Statement

```
$VARIABLE [e] = [assetsBanks] |
  [assetsBanks].[privateBank] |
  [assetsBanks].[governmentBank]
WITH
MEMBER [Measure].[totalAssets] AS 'SUM ([e].Members)'
SELECT {[Measure].[totalAssets] ON Axis(0)}
FROM Banks
```

2.2.2 LMDQL Operators

Before listing the LMDQL operators, it is important to highlight some definitions that will be used in its specification (Chart 5). For these definitions, the instance element defined in the schema or a label related to the element is considered as a member.

Chart 5. Definitions used for specifying LMDQL

<MemberSet>	Set of members in a cube
<Member>	One member in a cube
<DimensionName>	Dimension name (e.g. [Location]).
<LevelName>	A level name (e.g. [Location].[State]).
<MemberName>	A member name (e.g. [Location].[state].[Bahia]).
<NumericExpression>	Any numeric value
<IntegerExpression>	Any Interger value.
<NumericSet>	A set of any numeric values.

The language operators are categorized due to the need of having operators that allow the query performance based on comparative values, once the use of indexes is of great importance in any

domain. However, the MDX language does not cover this kind of analysis completely, because it does not define one operator that allows the use of the Separatrix, a statistic position measure. On the other hand, the use of Xlink implies the need of one operator category that allows the navigation through linkbases. Thus, the LMDQL operators were categorized the following way: (i) Relative Evaluation Operators, which allow the creation of operators for determining indexes and the performance of horizontal and vertical analyses based on statistics information; and (ii) Navigation Operators, which allow the recovery of the information based on the document structure and on the linkbases.

2.2.2.1Relative Evaluation Operators

The use of indexes is justified when the analysis of certain relations is more significant than the direct analysis on data. The LMDQL language allows the user to construct operators that express any relations, and its applications in analysis techniques. In this work, the horizontal, vertical and standard index techniques are considered. The horizontal analysis highlights the variation of certain information over the time. The vertical analysis allows one to identify the participation percentage of certain information on the composition of another. The standard index analysis determines the relative location of information in a statistic distribution.

In MDX, the creation of operators through the use of calculated members can be performed. However, it is not possible to save the created operators, leading to redefinitions, in case they are necessary for further use. This feature is available in LMDQL. The LMDQL Relative Evaluation Operators are described below.

OperatorDefinition (String, String [, param(String+)]): given two strings, one for defining the expression that specifies the relation between the members of a cube, and another one that identifies the new operator, this operator creates another one based on the previous relation given. Optionally, parameters to be used in the given expression can be supplied. The operator creation occurs in execution time, allowing the immediate use of just created operators. The operators can be created in the internal context of a query or as an independent statement. In both cases, the operators created are stored in the operator library for future use. Chart 6 illustrates the use of this operator for creating and using the *assetsReturn* operator.

Chart 6. Use of the *OperatorDefinition* operator

```
OperatorDefinition('([Net Profit]/[Assets])', assetsReturn)
SELECT {[Time].[2007]} ON COLUMNS,
{Descendants ({[company].Children}, [geographic].[São Paulo])} ON
ROWS FROM Banks WHERE (assetsReturn)
```

HAnalysis (MemberSet, MemberSet^{+2}, [NumericSet] *): given a reference set, two or more sets of members and, optionally, the indexes for correcting value in each set, this operator calculates each set's evolution in relation to the reference. Chart 7 shows the use of this operator.

Chart 7. LMDQL Query for horizontal analysis

```
SELECT {[Time].Members} ON COLUMNS,
{HAnalysis([Time].[2006],{[Measure]. [Assets].Children},1)}ON
ROWS
FROM Banks WHERE ([bank].[BankOfBrazil],
     [geographic].[São Paulo].)
```

VAnalysis (Member, MemberSet): given a reference member and a set of members, this operator calculates the percentage of each member in relation to the referenced one. In Chart 8, the use of VAnalysis is illustrated.

Chart 8. LMDQL Query for vertical analysis

```
SELECT {[ Time].[2008]} ON COLUMNS,
{VAnalysis([Measure].[Assets].[Operational Assets],
[Measure].[Assets].[Operational Assets].Children})} ON ROWS FROM
Banks
```

Separatrix (MemberSet, Member, IntergerExpression): from a set of members, a member to be evaluated and an integer that represents the separatrix, this operator determines the intervals for this set and returns the evaluation member relative position. The intervals are calculated through the members ordering, which are divided according to the integer value given as parameter. The use of this operator is exemplified in Chart 9, where the query returns the position of Bank of Brazil assets in relation to the position of other banks from Brazil.

Chart 9. Use of the Separatrix operator

```
SELECT {assetsReturn} ON COLUMNS,
{Separatrix({[company].Children,
[geographic].[Brazil]},[bank].[BankOfBrazil],3)}
ON ROWS FROM Banks
```

2.2.2.2 Navigation Operators

The operators presented next allow the retrieval of information contained in the relations expressed in linkbases, schemas and instances. For example, it is possible to find instances with different elements, but with the same meaning and same label. Then, the query is performed by checking the elements that relate with the label.

Cross (Member, String*): given a reference member, this operator returns the members that have relation with it. Optionally, the kind of relation to return can be specified. This is given by the names of the linkbases. One example of the Cross operator is illustrated in Chart 10.

Chart 10. Use of the Cross operator

```
SELECT {[Time].[2007], [Time].[2007]} ON COLUMNS,
{Cross([Measure].[Assets], 'Definition')} ON ROWS
FROM Banks
```

NNearestValues (Member, IntergerExpression [,String]): from a reference member and a precision numeric value (N), this operator returns the closest N members to the reference member. Optionally, it is possible to define, using ASC or DESC, whether this amount of members refers to the members with closest values above or below the reference value. One example of the use of this operator is shown in Chart 11.

Chart 11. Use of the NNearestValues operator

```
SELECT {[Time].[2005].Children} ON COLUMNS,
{NNearestValues([Measure].[BankOfBrazil].[Assets], 3, DESC) ON ROWS
FROM Banks
```

NNearestValuesPercentual (Member, NumericExpression [,String]): given a reference member and a numeric value (N), this operator returns the present members at a percentage variation rate (N%) based on the reference value. Optionally, the ASC or DESC values are also used. Chart 12 illustrated the use of this operator in one LMDQL query.

Chart 12. Use of the NNearestValuesPercentual operator

```
SELECT {[Time].[2005].Children} ON COLUMNS,
{NNearestValuesPercentual([Measure].[BankOfBrazil].
         [Assets], 10) ON ROWS FROM Banks
```

2.3 LMDQL Processor Architecture

The architecture of the LMDQL processor, which can be seen in Figure 3, is divided into three layers, which perform the interface function with the user, LMDQL query processing and data acquisition. These architecture layers are: (i) *Interface*, which contemplates the GUI used for command submission in LMDQL. The two components from this *Interface* layer are the *Query Editor*, that validates the queries and forwards them to the *Processor* layer, and the *Query Viewer*, that displays the results; (ii) *Processor*, where the analytic-dimensional processing mechanism is. It contains the modules *Query Processor*, *XPath+ Processor*, which can be viewed on the right of Figure 3, and the *Optimizer*, which contains optimization mechanisms. The *Query Processor* is meant to perform the lexical and syntactic analysis on the query and send it to the execution manager, which will be detailed in section 3.4. *XPath+ Processor* is responsible for controlling the query performance, receiving requirements in the XPath+ expression form, performing them and retrieving the necessary data to query execution; (iii) *Data*, data acquisition layer, composed by: (a) *Data Cube*, stores the instance documents; (b) *Metadata Repository*, the repository of metadata, in which *schemas* and *linkbases* are; and (c) *Operators Library*, where the operators created by the user are stored.

2.4 Implementation Aspects

One analytical processing environment based on open source code tools was developed aimed at making the analysis over XML data accessible to a large number of users. Therefore, the *mondrian* [11] was chosen to compose the desired environment, since it is an OLAP server used in several Business Intelligence (BI) solutions, and is developed in Java, which makes it computational platform independent. Another aspect that has influenced this decision was the use, by this server, of the MDX query language, which provides resources that allow performing a great variety of queries on multidimensional data.

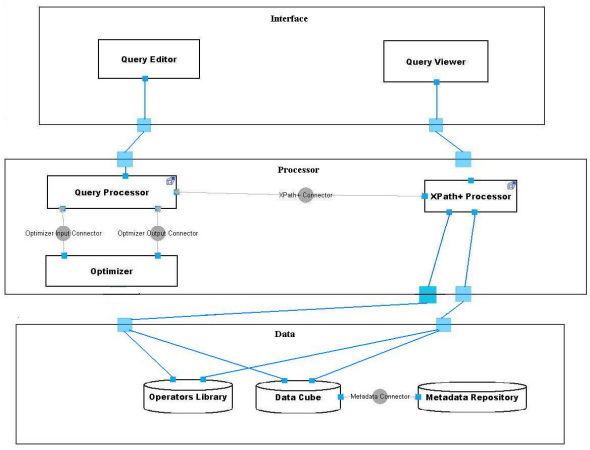

Figure 3. Architecture of the LMDQL Processor

The *mondrian* architecture is formed by four layers. Its extension occurred in the database layer which contains the facts and dimensions table, once LMDQL acts on XML. Then, an open source native XML SGDB was used, eXist [5]. This SGBD allows the creation of collections and sub-collections of XML documents, functionality of great importance for performing LMDQL queries in more than one document. Another extended layer was the dimension one, which analyses grammatically, validates and executes MDX queries. The metadata stored in it, represented by an XML document, describe the data cube and mapping in a relational model. Then there is the processing of the MDX query transforming it into SQL. This SQL query returns the values from the cube cells and the dimension members of the cube. The LMDQL query processing keeps the current cube mapping system. However, the LMDQL processor intercepts the SQL queries generated and converts them into XQuery queries, using XPath+ expressions. This was the main challenge of this approach, since XQuery and SQL are based on different data paradigms (i.e. ordered sequences and tuples sets).

The interception procedure of the SQL queries and its conversion into XQuery queries is performed in the *Query Processor* module, from the *Processor* layer, through a JDBC driver [8] developed to map the relational cube representation used by the OLAP server for XML documents. This gives the OLAP server an interface to access the data present in XML documents identical to the one used to access relational databases. This makes indifferent for the OLAP server the database environment that is being used, XML or relational, for performing MDX queries.

The JDBC driver performs the cube mapping defined in the OLAP server for the XML documents through a metadata, represented by a configuration file. After the translation of the SQL queries, the JDBC driver applies the XQuery resulting queries on the corresponding XML documents stored in the native XML SGBD and sends the returned data from these queries, in JDBC format, to the OLAP server. The JDBC specification was chosen for the implementation of this access driver to database to make it independent from the computational platform and compatible with *mondrian*.

The architecture of the JDBC driver, which can be viewed in Figure 4, is divided in three layers, which perform interface functions with the OLAP server, SQL queries conversion in XQuery queries and interface with native XML SGBD. These architecture layers are described next: (i) *JBDC*, which contemplates the interface with the OLAP server. The components of this layer are the *Statement*, used to perform a SQL query and return the result generated, the *ResultSet*, which contains the data table representing the SQL query return, the *Connection*, which represents the session through which the SQL queries are performed, and the *DriverManager*, which represents the JDBC *driver* management service; (ii) *Parsing*, where the conversion mechanism from SQL to XQuery queries is. The modules *SQL-Tokenizer*, *XQuery-Generator* and *Cube-Metadata* are in it. The *SQL-Tokenizer* has the function to perform the lexical and syntactic analysis on the SQL query to generate the tokens of this query. *XQuery-Generator* has the responsibility to create the XQuery queries from the SQL *tokens*. *Cube-Metadata* is the metadata that maps the cube defined in the OLAP server for the XML documents; (iii) *XQJ*, interface layer with the native XML SGBD, composed by the components: (a) *XmlConnection*, session through which the XQuery queries are performed; (b) *XmlDataSource*, where the sessions established with the native XML SGBD are managed; (c) *XmlExpression*, used to perform one XQuery query and return the result generated from it; and (d) *XmlSequence*, which contains the return SQL query data.

The approach chosen for the *mondrian's* extension has the advantage to cause the smallest possible coupling to this OLAP

server. This implies in the possibility of LMDQL use in BI applications that do not make use of the *mondrian*. Furthermore, besides the data model proposed in section 3.1, other DW models for XML (without XLink) can be used, being enough the cube mapping in the *mondrian*. So, it is transparent for the user the fact that the data base is relational or based in any DW solution for XML.

3. LMDQL QUERY on XBRL DOCUMENTS

In order to validate the LMDQL operators, an example based on XBRL documents will be used. XBRL is a XML-based language used to create financial reports. It uses plenty of XLink links in order to express the semantics of instance elements. Figure 5 illustrates a DW structure, based in the taxonomy defined in COREP/FINREP XBRL Project [4]. This is a model where the hypercube *section exposure types hypercube* relates with the *credit risk* and *exposure type* dimensions and where the hypercube *securitization positions SA hypercube* relates with the *securitization type* and *credit risk* dimensions. Based on these definitions, the XBRL instances, which represent a data cube, are stored in the *Financial* collection of the data base. The metadata, linkbases and schemas, are also stored in the *Data* layer repository.

The MDX language allows performing vertical and horizontal analyses through queries made by the user, using the calculated member operation. In Chart 16, one MDX expression to perform one horizontal analysis on three distinct periods is shown. It is observed that it is necessary the definition of members referring to the desired periods and the association between them and the tuples (*Unit Requirements*, *Horizontal Analysis*). In one analysis with a higher number of comparative periods, the query complexity clearly increases. For this kind of query to be easier to write, operators were created for vertical and horizontal analyses in LMDQL, which allow the construction of queries that are more concise and simpler to use and keep. Figure 6 illustrates one identical query using the *HAnalysis* operator of the LMDQL language, being the result gotten in the instances relative to each period, present in the *Financial* documents collection.

Figure 7 exemplifies the use of the MDX DrillDownLevel operator, which performs a search for the members of one level below the one represented in the query, allowing a more detailed view of the data set.

4. CONCLUSION

The LMDQL language allows the multidimensional queries performance in XML documents that make use of XLink. LMDQL extends MDX for performing relative analyses, making the queries more concise, through the operators for vertical and horizontal analysis, besides including the analysis based on statistic positional measures, through the *Separatrix* operator. Furthermore, LMDQL allows the creation of operators based on others previously defined, in execution time, speeding the definition of new operators and making it possible for them to be stored in a library for further use. This allows the creation of operator libraries for specific domains. LMDQL defines operators for obtaining information in data containing in linkbases, allowing the performance of analytical queries in applications based on XLink.

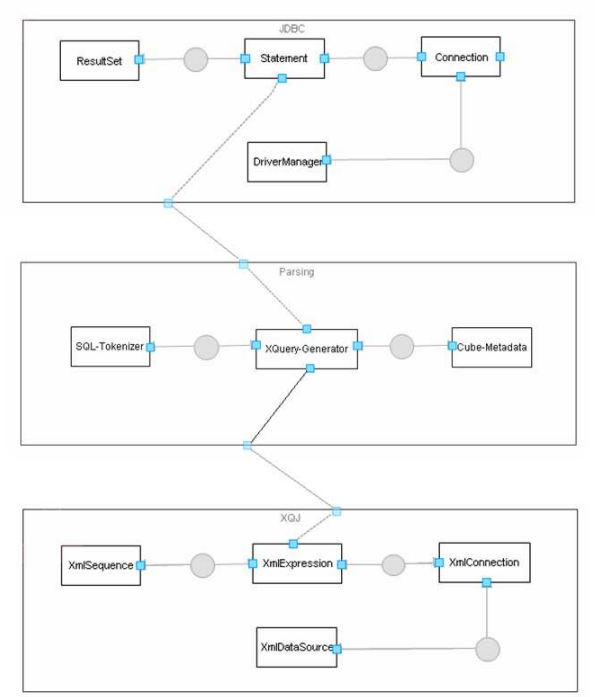

Figure 4. Architecture of the JDBC driver

Chart 16. MDX Query for horizontal analysis

```
WITH
MEMBER [Measures].[HA 2006] AS
    ((([Time].[2006]-0) * 100) / [Time].[2006])
MEMBER [Measures].[HA 2007] AS
  ((([Time].[2007]-[Time].[2006]) * 100) /
     [Time].[2006])
MEMBER [Measures].[HA 2008] AS
  ((([Time].[2008]-[Time].[2006]) * 100) /
     [Time].[2006])
SELECT Union(Crossjoin({[Time].[2006]},
  {[Measures].[Unit Requirements],
   [Measures].[HA 2006]}),
Union(Crossjoin({[Time].[2007]},
 {[Measures].[Unit Requirements],
  [Measures].[HA 2007]}), Crossjoin({[Time].[2008]},
 {[Measures].[Unit Requirements],
  [Measures].[HA 2008]}))) ON COLUMNS,
{[Credit Risk].
 [Credit Risk SA Securitization].Members} ON ROWS
FROM [Capital Requirements]
```

The LMDQL language processor implementation allows the reuse of the MDX operators implemented in existing BI solutions, favoring the use of LMDQL in those solutions. This implementation allows its use in the OLAP server, no matter the data storage environment, XML or relational. It is intended to expand the LMDQL functionalities to incorporate operators for performing OLAP query on text. As the time and space efficiencies were not evaluated in this paper, the implementation of the processor optimization module and the definition of a taxonomy for financial OLAP operators will also be performed.

5. REFERENCES

[1] Beyer, K., Chamberlin, D., Colby, L., Ozcan, F., Pirahesh, H., Xu, Y. (2005) “Extending XQuery for Analytics”. In: SIGMOD.

[2] Bordawekar, R. R., Lang, C. A. (2005), “Analytical Processing of XML Documents: Opportunities and Challenges”. SIGMOD Record, Vol. 34, No. 2.

[3] Chaudhuri, S., Dayal, U. (1997) “An overview of data warehousing and olap technology”. SIGMOD Record 26(1),

65–74.
[4] COREP, COREP/FINREP XBRL Project. www.corep.info/.
[5] eXist Native XML Database. exist.sourceforge.net/
[6] Hernández-Ros, I., Wallis, H. (2006), "XBRL Dimensions 1.0" www.xbrl.org/Specification/XDT-REC-2006-09-18.htm
[7] Hümmer, W., Bauer, A., Harde, G. (2003) "XCube – XML for Data Warehouses", In: Proc. The 6th ACM Intl Workshop on Data Warehousing and OLAP, p. 33–40.
[8] JDBC 2.0 Standard Extenxion API (1998). java.sun.com/products/jdbc/jdbc20.stdext.pdf
[9] Jian, F. M., Pei, J., Fu, A. W. (2007) "IX-Cubes: Iceberg Cubes for Data Warehousing and OLAP on XML Data", In: *CIKM'07*, Lisboa, Portugal.
[10] MDX Function Reference, (2008), msdn .microsoft.com/en-us/library/ms145506.aspx.
[11] Mondrian (2008), mondrian.pentaho.org
[12] Näppilä, T., Järvelin, K., Niemi, T. (2008) "A tool for data cube construction from structurally heterogeneous XML documents", In: Journal of the American Society for Information Science and Technology (JASIST), Vol. 59, Issue 3, Pages: 435-449.
[13] Park, B. K., Han, H., Song, I.Y. (2005) "XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses". In: 7th International Conference in Data Warehousing and Knowledge Discovery, LNCS 3589, pp. 32–42.
[14] Silva, P. C., Aquino, I. J. S., Times, V. C. (2008), "A Query Language For Navigation Over Links", In: XIV Brazilian Multimedia Systems and Web Symposium – WebMedia.
[15] Silva, P. C., Times, V. C. (2009), " XPath+: A Tool for Linked XML Documents Navigation", In: XSym - Sixth International XML Database Symposium at VLDB'09 in Lyon, 24 August 2009.
[16] XQuery 1.0: An xml query language, (2007), www.w3.org/TR/xquery.
[17] XML Linking Language (2001), www.w3.org/TR/xlink.
[18] XML Schema (2004) www.w3.org/TR/xmlschema-1.
[19] XML Path language (2007), www.w3c.org/tr/xpath20/.
[20] Wang, H., Li, J., He, Z. and Gao, H. (2007). "Flexible and Effective Aggregation operator for XML Data". In: Information Technology Journal 6 (5), Asian Network for Scientific Information, pages 697-703.
[21] Wiwatwattana, N., Jagadish, H., Lakshmanan, L. and Srivastava, D. (2007), "X^3: A cube operator for xml olap". In: Int. Conf. Data Engineering (ICDE'07).

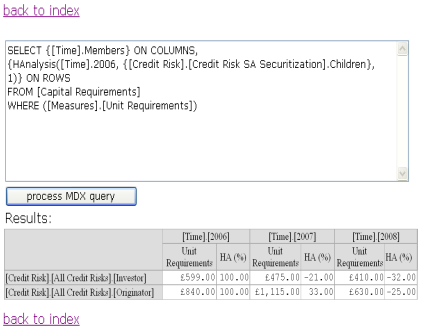

Figure 6. LMDQL Query for horizontal analysis

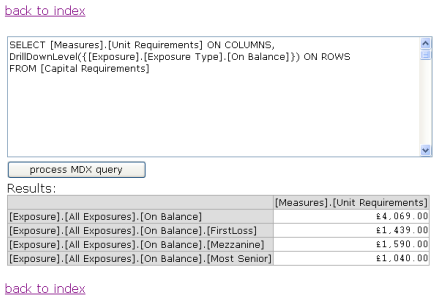

Figure 7. LMDQL Query for the DrillDownLevel operator

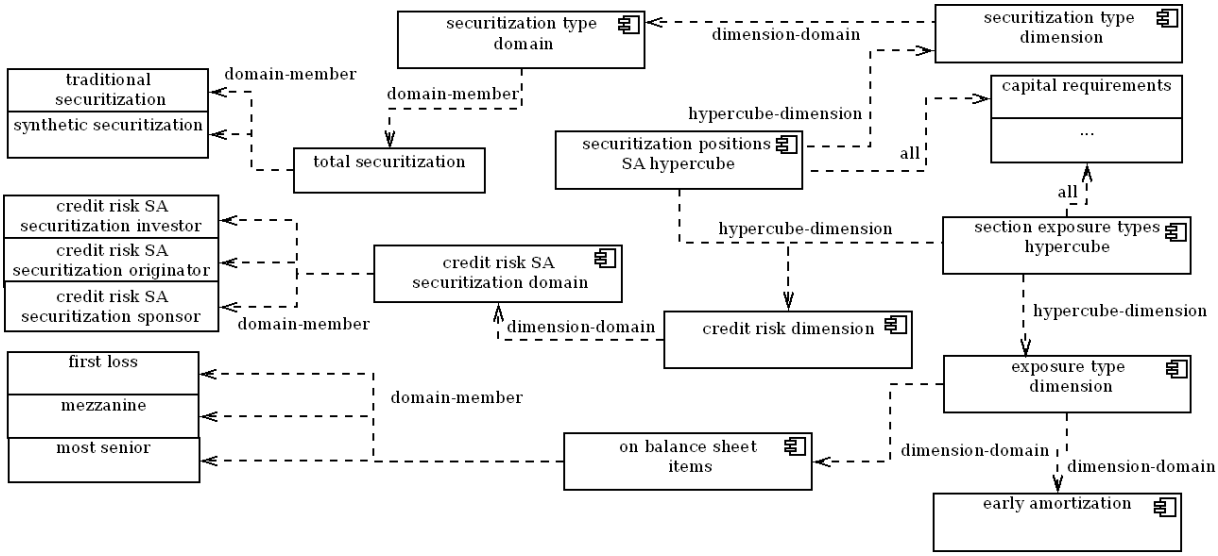

Figure 5. DW Model for the Financial cube