MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 13

XML Query Processing

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction	-	-	3
	1.1 Objectives	-	-	
2	XML Query Languages	-	1, I	4-13
	2.1 XML-QL	-	-	
	2.2 Lorel	-	-	
	2.3 Quilt	1	-	
	2.4 XQL	-	-	
	2.5 XQuery	2	-	
3	Approaches for XML Query Languages	1	2	14-18
	3.1 Query processing for relational structure	3	-	
	3.2 Query processing on storage schema	4	-	
4	XML Database Management Systems	-	3, II	19-21
5	Summary	-	-	21-22
6	Glossary	-	-	22
7	Terminal Questions	-	-	23
8	Answers	-	-	23-24
9	References	-	-	24

1. INTRODUCTION

In the previous unit, you studied the concept of object relational and extended relational databases which included design techniques used in RDBMS, extension techniques in RDBMS, standards for OODBMS products and applications, etc. In this unit, we will reflect on the concept of XML Query Processing.

XML is used to divide presentation and data. Therefore it provides independency and flexibility for association of the content. Because of this temperament of flexibility, data interchanged among two dissimilar systems can utilise XML as the data format. XML represents a tree-like structure which is instinctive, understandable and simple to recognise. By means of XML schema or DTD (document data type), the type and attributes of every tag which is functional for some XML document can be properly defined.

The function of an XML query engine or processor is to translate the syntaxes and carry out the operations coded by the query. After the process and processing time is estimated to be minimum, output is returned. This therefore provides proficient processing.

In this unit, you will study various XML query languages like XML-QL, Lorel, Quilt, XQL, and XQuery. You will recognise the approaches used for XML query processing like query processing on relational structure and storage schema. Also we will discuss the concept of XML database management system.

1.1 Objectives

After studying this unit, you should be able to:

- *Recognise XML query languages like XML-QL, Lorel, Quilt, XQL, etc.*
- *Discuss the approaches used for XML query processing*
- *Explain XML database management system*

2. XML QUERY LANGUAGES

Nowadays XML is turning out to be the most significant new standard used for the representation of data and exchange on the World Wide Web. There are many new languages that have been projected for extracting and reforming the content of XML.

Some of the languages existing are conventional languages (such as SQL, OQL), and other languages are more recent and very much motivated by XML. Till now, no standard for XML query language has been decided. However research is going on within the World Wide Web association and among various academic foundations and in the main companies associated with internet.

Now we will discuss various XML query languages as below.

2.1 XML-QL

XML-QL is an XML query language which is used for the process of querying, constructing, converting and incorporating the data of XML. XML- QL language basically displays XML as semi structured data comprising of irregular or speedily developing structure. For matching data in an XML document, patterns are used by XML-QL. An extension of XML-QL known as Elixir was projected to assist ranked queries depending on textual resemblance.

In case of XML-QL, the client is permitted to query an XML document similar to a database, and illustrate an output construct.

XML-QL syntax comprises two sections, one of which is a WHERE clause and a second one is a CONSTRUCT clause. We use a WHERE clause to illustrate the data to search for, and a CONSTRUCT clause is used to illustrate the process of returning the data that is found. Now we will discuss these two clauses as below.

1. WHERE

The syntax for WHERE clause is shown as below:

WHERE XML-searchstring [ORDER-BY variable [DESCENDING] [variable [DESCENDING]]] IN 'filename'

You can divide the WHERE clause into various parts. The first part is known as the search string, and the other part is known as ORDER-BY clause which is used optionally. ORDER-BY clause is similar to ORDER BY in SQL. The last part includes the necessary XML document file name. Let us now discuss all these parts.

XML-searchstring: The search string must be a valid XML section. In this part, the module varies from the specification. It is implemented in this manner in order that you can parse the search string by the module known as XML::Parser.

Now in constructing a query, the tags are listed first that are to be searched in the document. For example, let us consider the search string as below:

```
<BOOK>
    <AUTHOR></AUTHOR>
</BOOK>
```

From the above example, you can see that this search string is searching for the AUTHOR tag which is nested inside a BOOK tag. Make note that till now no information has been chosen for retrieval. Now we will actually take some information, as shown in the example below:

```
<BOOK>
     <AUTHOR>$author</AUTHOR>
</BOOK>
```

Here, $author is the variable name which will take the information that it looks for inside this tag. This will make this information obtainable to be used in the CONSTRUCT section of the query. As you have observed, the variable names begin with a dollar sign ($). This is because dollar sign is called for by the specification. In case of Perl, this signifies that if the query is contained in double quotes, you can avoid the dollar sign.

Now suppose we want to search for books that are non-fiction. This is shown as below:

```
<BOOK TYPE='non-fiction'>
<AUTHOR>$author</AUTHOR>
</BOOK>
```

This can also be expressed as a regular expression, as shown below:

```
<BOOK TYPE='non-.*'>
<AUTHOR>$author</AUTHOR>
</BOOK>
```

Here this module varies from the specification. As defined in the specification, the regular expression ability only permits for a subset of the ability obtainable in a Perl regular expression. By means of this module, the complete range of the syntax of regular expression has been made obtainable. This signifies that there is a need to escape things like periods(.), parenthesis (), and brackets ([]). All non tag matched are considered as case insensitive.

Now suppose that apart from matching the TYPE, we also want to retrieve the value. This is shown in the following example:

```
<BOOK TYPE='non-.* AS_ELEMENT $type'>
<AUTHOR>$author</AUTHOR>
</BOOK>
```

Here, the keyword named as AS_ELEMENT permits you to save the corresponding value to make use of it afterwards in the CONSTRUCT section of the query.

ORDER-BY: The ORDER-BY clause provides permission to arrange the data retrieved in the variables. Numerous variables can be specified, and also DESCENDING can be specified for a reverse sort. This clause is not necessary. For example:

```
ORDER-BY $type, $author DESCENDING
```

IN: The IN clause is a necessary clause that provides the file name of the XML file. It can be a single file name included in quotes.

2. CONSTRUCT

The CONSTRUCT section provides permission to state a pattern for output. The pattern will compare each and every character from the first space after the word CONSTRUCT to the end of the XML-QL query. For example:

```
$ql = '(where clause...) CONSTRUCT Type: $type Author: $author';
```

The repetition of this construct is done for each match discovered and returned as a single string.

2.2 Lorel

Lorel is considered as an early query language used for semi structured data. Lorel language makes use of the OEM (Object Exchange Model) as the data model for semi structured data. Lorel is utilised to expand OQL (Object Query Language) for the procedure of querying elements. This is done by depending on coercion at various levels to hold back the powerful typing of OQL. Lorel is also used to extend OQL with path expressions. This is done in order that user can state the patterns that are corresponding to actual paths in referred data.

One of the advantages of Lorel language is its easy syntax which makes users to understand it more clearly.

The drawback of Lorel language is that it is dependent on OQL parser. Also it comprises of limited functionalities.

2.3 Quilt

Quilt is considered as a functional language where you can represent a query as expression. Quilt expressions include seven major forms. They are:

- path expressions
- element constructors
- FLWR expressions
- expressions with operator and functions
- conditional expressions
- quantifiers
- variable bindings

Apart from join operations, quilt also includes nested expressions. Thus, it mainly includes a subquery inside a single query. For the progress of XQuery (discussed later), important traits of Quilts were utilised.

Now we have shown the syntax as below which is considered as the top- level structure of a Quilt query. In the existing version, the grammar is incomplete and is still being developed

as the language progresses. In the syntax shown in Figure 13.1 below, we have shown the terminal symbols in angular brackets and we have not further specified their lexical structure.

Query	::=	Function_defn* Expression
Function_defn	::=	'FUNCTION' <Function_name> '(' <Variable>* ')' '{' Expression '}'
Expression	::=	<Variable> | <Constant> <Function_name> '(' ExpressionList ')' Expression Operator Expression <XPathExpression> | ElementConstructor | FLWR_Expression | '(' Expression ')'
ExpressionList	::=	Expression | Expression '.' Expression
ElementConstructor	::=	StartTag ExpressionList Enbibitemag
StartTag	::=	'<' <String> Attribute* '>'
Enbibitemag	::=	'<' <String> '>'
Attribute	::=	<String> '=' Expression | Expression
FLWR_Expression	::=	(FOR_clause | LET_clause)+ WHERE_clause? RETURN_clause
FOR_clause	::=	'FOR' FOR_binding (, FOR_binding)*
FOR_binding	::=	<Variable> '|N|' 'DISTINCT'? Expression
LET_clause LET_binding	::= ::=	'LET' LET_binding (, LET_binding)* <Variable> ':=' 'DISTINCT'? Expression
WHERE_clause	::=	'WHERE' Expression
RETURN_clause	::=	'RETURN' Expression
		SORTBY_clause?
SORTBY_clause	::=	'SORTBY' '(' ExpressionList
		')' ('ASCENDING' | 'DESCENDING')?
Operator	::=	'<' | '<=' '>' | '>=' | '=' | '|=| |'+'|...

Fig 13.1: Quilt Query Syntax

You can begin a Quilt query by defining one or more functions that we use in the body of the query. The body of the query is just an expression. A significant type of query relies on a "FLWR_expression". This expression is constructed from FOR, LET, WHERE, and RETURN clauses.

However a query may also rely on other kinds of expressions. These other kinds of expressions may be an XPath expression or an element constructor. We use element constructors for producing new elements of output that enclose data calculated by the query.

Make note that XPath Expression is considered as a terminal symbol in this grammar. This symbol symbolises an expression depending on the shortened syntax of XPath. This is improved by using the following operators which are taken from XQL:

1. ***BEFORE and AFTER operators:*** Let us consider "a BEFORE b". It assesses to the group of nodes in 'a' that takes place previous to some node in 'b'. Now consider "a AFTER b". It assesses the group of nodes in 'a' that takes place after some node in 'b'.
2. ***INTERSECT operator:*** Let us consider "a INTERSECT b". It assesses to the group of nodes in 'a' that are also located in 'b'. This is considered as an improvement to XPath, which includes an operator for set union but not for intersection.
3. ***EXCEPT operator:*** Let us consider "a EXCEPT b". It assesses to the set of nodes in 'a' that are also not located in 'b'.

Make note that till now no definite standard function library is developed for Quilt. The advantage of using Quilt is that it provides strong functionalities.

The drawback is that it does not provide any support for textual resemblance.

2.4 XQL

XQL is another XML query language which makes use of path expressions. Thus the main constructs of XQL matches directly to the main structures of XML. Because of this nature, XQL is considered to be very much associated to XPath. In case of XQL, document nodes have an essential role. Nodes comprise identity. Also, the identity, query results series, and associations of containment are maintained by the nodes. The nodes emerge from various different sources. But, the process of bringing nodes to the query is not specified by XQL. Joins and some functions are also supported by XQL.

XQL is generally used to find and sort the text and elements in the document of an Extensible Markup Language (XML). We use XML files to send collections of data among computers on the Web.

XQL offers a device which is used for searching and/or choosing out particular items in the collection of data. It relies on the pattern syntax which is utilised in the Extensible Stylesheet Language (XSL) and is projected as an extension to it.

A declarative manner in which particular elements are specified for processing is performed by the XSL pattern language. Simple directory notation is used by it. For example, book/author signifies: Choose every author element in every book element in a specific context.

The capability to utilise Boolean logic, to sort out elements, to index into a an elements collection are added to this directory outline details by XQL. By means of XQL, you can write a program to look for repositories of XML files. This is done to offer hypertext links to particular elements, and for further applications.

XQL query includes shorter expressions which are easy to use. However, semantics may not be very instinctive.

2.5 XQuery

XQuery is an XML query language which is generated for XML data sources. W3C working group have produced XQuery and now it is in approval status. XQuery is considered as an industry standard query language.

Declarative access to XML data is provided by XQuery similar to as SQL performs for relational data. We have shown XQuery and other associated XML standards in the Figure 13.2. You can see from the Figure 13.2 that both XPath and XQuery make use of XPath as the basis and utilise XPath expressions.

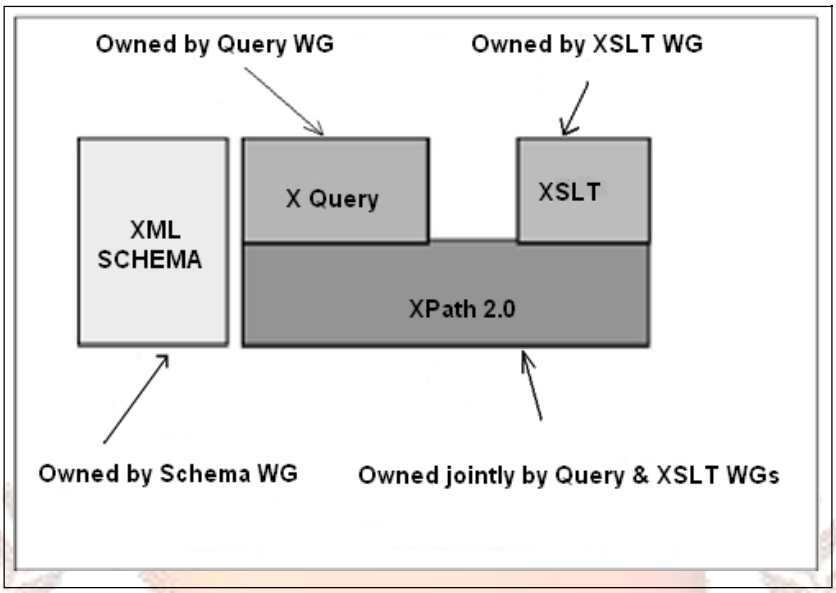

Fig 13.2: XQuery and Related Standards

XQuery comprises of two parts which are discussed below:

- The prolog, which is non-compulsory, comprises of declarations that describe the query's execution environment.
- The query body comprises of an expression that offers the query's result. The XQuery's input and output are XDM instances at all times.

The XQuery's body comprises of any or all of the following expression:

- Literals and variables
- Path expressions
- Predicates
- If ..then..else
- Constructors
- Comparisons
- FLWOR expressions

Now we will discuss FLWOR expressions as below.

FLWOR expressions

FLWOR expressions are considered as a main section of the XQuery language. The full form of FLWOR is FOR LET WHERE ORDER BY and RETURN.

Frequently the FLWOR expression is matched with the SELECT FROM WHERE ORDER BY statement in SQL.

We have shown an example of a simple FLWOR expression in an XQuery as below:

XQuery

for $x in db2-fn:xmlcolumn('ITEM.DESCRIPTION') let $p := $x/Item/description/name

where $x/Item/description/price < 3

return <cheap_items> {$p} </cheap_items>

The “for” clause iterate via a series and connects a variable to items in the series, separately. $x is the binding to the series of XML documents accumulated in a column named DESCRIPTION. Throughout every iteration of the “for” clause, $x keeps one item from the group of XML documents XQuery variable $p. For example, if the XML document associated with $x comprises of four product names, all the product names will turn out to be section of the sequence and will be associated to $p.

The “where” clause is considered as analogous to the where clause in SQL and sorts the outcome group depending on the specified predicate. In the specified example, the where clause chooses the items whose price is less than “3”.

The XDM instance that is to be returned is specified by the “return” clause. You can return the XDM instance when it comes out from the query assessment or by grouping with further elements created in the return clause. An example of this can be seen above.

We have given below some more examples of XQuery FLWOR:

1. XQuery for $i in (1 to 3) return $i

OR

2. XQuery for $x in db2-fn:xmlcolumn(‘ITEM.DESCRIPTION’)//text() Execution Result

1

1

2

3

All the text nodes are returned from the above query from all the XML documents accumulated in DESCRIPTION column of the ITEM table.

SELF-ASSESSMENT QUESTIONS – 1

1. The syntax of XML-QL comprises of two parts, that is a WHERE clause and a _____ clause.
2. Which XML query language is used to extend OQL with path expressions?
   a) Quilt
   b) Lorel
   c) Xquery
   d) XML-QL
3. Quilt query language does not include a subquery inside a single query. (True/ False)
4. Which query language is used to find and sort the elements and text in an Extensible Markup Language (XML) document?

Activity 1

What are the major forms included in Quilt? Also illustrate how to construct Quilt query.

3. APPROACHES FOR XML QUERY LANGUAGES

The function of a query processor is to take out the query's high level abstraction and its procedural assessment into a group of low-level operations. Make note that SQL query is transformed at logical access model, similar to SQL processor.

Then it is transformed at the logical access before accessing the physical storage model. We have shown the different levels of abstraction in XML query processing as compared to SQL abstraction levels in the Table 13.1 shown as below.

Table 13.1: Levels of Abstraction in XML Query Processing as Compared to SQL Abstraction Levels

Level of Abstraction	XDBS	RDBS
Language model	XQuery	SQL
Logical access model	XML query algebra	Relational algebra
Physical access model	Physical XML query algebra	Phyiscal DB-operators
Storage model	XTC, natix, shredded documents, etc	Record-oriented DB-interface

From the above table, XDBS indicates XML database management system and RDBS indicates Relational Database Management System.

The language model is intended to fulfil the demands which are represented in the language ability to carry out search functionality and document-order knowledge and so document-centric features and afterwards the data- centric features which are related to influential choice and conversion.

Then the semantic processing is supposed to be able to examine the query and convert it into a worldwide demonstration to be used during subsequent steps for optimisation.

Algebraic and non-algebraic procedure is implemented by logical access model to optimise the internal demonstration of the query. Non-algebraic optimisation is used to diminish

intermediary results by reforming the query and performing most choosy operations rapidly.

Algebraic optimisation will convert into a more optimised expression in a semantics-preserving way.

Physical access model is associated to system specific concern. Here, every logical algebra operator will be divided into matching physical operators.

Lastly, for optimised query processing, suitable storage model should be organised to diminish I/O prices, CPU prices, storage costs for intermediary outcomes, and communication prices.

Storage models that are used at present include LOBs (Large Objects), some XML-to-relational mappings, or native storage formats such as Niagara and Timber. The relational XML data model and native storage model draw more attentions specified by different proposals for relevant overlying query processors.

A variety of XML query processors have been projected for more efficient query processing. Based on the levels of abstraction, the query processors are divided into the following approaches according to their storage models.

These approaches are Query Processing for Relational Structure and Query Processing on Storage Schema which are discussed as below.

3.1 Query Processing for Relational Structure

When performing query processing for relational structure, XML document or information associated to XML document is accumulated in relational database.

This is because relational database executes improved indexing as compared to simple index formation. RDBMS engine will carry out the query processing by transforming XQuery into SQL, executing the SQL query and making the result of XML serialisable.

You can classify the relational storage methods for XML documents into three groups. They are:

- no XML schema method,

- XML schema method, and
- user defined method.

When no method is there, then relational representation should be derived from the data. After the appearance of schema, relational schema will be produced which comprises relationship between root element and all the sub-elements.

Relational scheme is divided into

- scheme-oblivious and
- scheme conscious method.

A predetermined plan is preserved by Scheme-oblivious method. On the contrary, scheme-conscious approach generates a relational schema depending on DTD/schema of the XML first. It has been observed that schema-oblivious method can execute better than schema conscious method.

A query processor BEA/XQRL executes relational scheme by means of XQuery. Query compiler parses and optimises the query. For extracting the query, XDBC interface acts as an interface among front-end application and query processor.

Then compiler will produce query plan so as to optimise the query. XML data symbolises a stream and XML parser parses it as input. The function and operator libraries included in runtime operators will process the flow and offer output depending on the query plan.

Now we will show the overview of BEA in Figure 13.3.

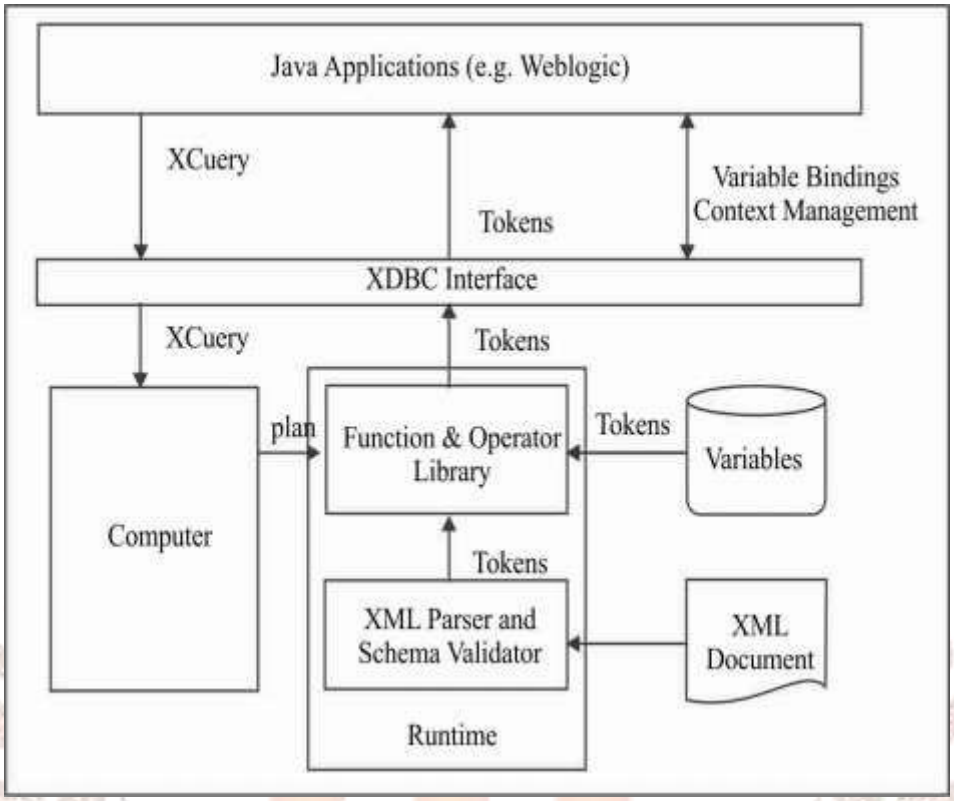

Fig 13.3: Overview of BEA

3.2 Query Processing on Storage Schema

In this approach, elements of XML are allocated label. The reason behind labelling is to generate exclusive identifiers that will be functional for query processing.

Many labelling methods are there which consider trade-off among space possession, contents of information, and appropriateness to updates. Region-based labelling method is most often utilised. The purpose of this method is to label elements to show nesting.

The labelling method for simple nesting is shown in Figure 13.4. The final label indicates status for the node, that is start, end, and level.

```
<a>Unnested content and <b>nested one</b></a>

1. Label <a> with [1,2] and b with [2,1]
2. Add nesting level
   Label <a> with [1,2,1]
   Label <b> with [2,1,2]
```

Fig 13.4: Region-based Labelling Scheme

ORDPATHS is also a labelling method which is executed in SQL server. This method labels every node by a series of integer numbers. As the name suggests, Order, depth, parent, and ancestor-descendant relationships are there in this method.

Then the XML document will be accumulated as persistent trees. If disk is utilised as storage means, XML nodes will be divided between disk and page. Node demonstration is optimised depending on fixed page size.

You can attain the proficient query processing in storage by means of stack- based algorithms such as StackTreeDesc and holistic twig joins. StackTreeDesc algorithm makes use of stack structure to store label of parent elements.

While reaching the path to destination child node, information from stack is united with child label and returned as consequences in descendant order. Then, for the next operation, stack is emptied. Alternatively, holistic twig joins points in an effort to evade constructing mediator consequences when matching twig patterns.

SELF-ASSESSMENT QUESTIONS – 2

6. When performing query processing for storage schema, XML document or information associated to XML document is accumulated in relational database. (True/ False)
7. Which of the following methods is used to preserve a fixed schema by capturing the tree structure of XML documents?
   a) scheme-oblivious method
   b) scheme conscious method
   c) XML schema method
   d) user defined method

4. XML DATABASE MANAGEMENT SYSTEMS

Now we know that XML will turn out to be a major portion of DBMS progress, but still even today we cannot define XML database. At the present time, there are two main approaches which are used for accumulating and recovering XML-based content. They are:

1. A relational or object-oriented database to accumulate XML data and middleware (incorporated or intermediary) to carry out data transfers among the database and XML document.
2. An application XML server that generates XML depending on an initial query of some type (for example, an e-commerce platform for structuring distributed system applications that make use of XML for data transfer). Usually, we refer these as content management systems.

The common point that is to be noted with the above approaches is that XML offers a bridge between structured and unstructured data in databases classes.

The relational database group's concentration on XML as a different data format for relational & object relational data processing devices directs to the growth of query languages like XMLQL, Lorel, and XML Query Language (XQL).

These languages are influenced in the direction of the data-centric outlook of XML, which needs data to be completely structured but unordered. Fundamentally, XML is used to publish structured data in a platform and application in an independent way.

Therefore XML just offers "syntactic sugaring" to the basic relational data. While this data-centric utilisation of XML is suitable, it does not binds the full control of XML.

The document-centric utilisation of XML depends not only on the data depiction expressed via markups, however, it also depends on the component ordering of data. Various systems relating to the document centric outlook make use of XPath.

Xpath is a language designed to recognise parts of XML documents, to query the data of XML. In recent times, XQuery is published by W3C, which unites XML's data and document-centric features, as a candidate standard query language for XML.

Therefore, existing XML management systems can be divided into XML- oriented databases and native XML databases.

XML-oriented databases are typically relational and enclose model- or pattern driven extensions for transporting data to and from XML documents and are usually intended for data-centric documents.

We have discussed below the major differences among XML-oriented databases and native XML databases:

- A native XML database is used to maintain physical structure. An XML- oriented database can do so also, but practice shows a diverse story.
- Native XML databases can accumulate data without schema. We could make use of techniques to recognise structure in unprocessed documents to be accumulated in XML-oriented systems. These types of techniques are relatively limited.
- XPath, DOM, or comparable XML-associated APIs are required to access data in native XML systems. Alternatively, XML-oriented systems provide direct access to the data via open-standard APIs, like open- database connectivity (ODBC).

Example: XTRON is considered as one of the XML data management system by means of an RDBMS. XTRON data management system uses the relational engine without modification. XTRON utilises the information of schema if it is obtainable and accumulates XML data over identical relational tables whether DTD exists or not.

SELF-ASSESSMENT QUESTIONS – 3

8. Which language is designed to recognise parts of XML documents, to query the data of XML?
   a) Xpath
   b) Xquery
   c) XQL
   d) Quilt
9. Many systems that subscribe to the document centric outlook make use of XPath. (True/ False)

Activity 2

What is the function of a query processor BEA? Illustrate with diagram.

5. SUMMARY

Let us recapitulate the important points discussed in this unit;

- XML represents a tree-like structure which is instinctive, human understandable and simple to recognise.
- XML-QL is an XML query language which is used for the process of querying, constructing, converting and incorporating the data of XML.
- Lorel is considered as an early query language used for semi structured data. Lorel language makes use of the OEM (Object Exchange Model) as the data model for semi structured data.
- Quilt is considered as a functional language where you can represent a query as expression. Quilt expressions include seven major forms: path expressions, element constructors, FLWR expressions, expressions with operator and functions, conditional expressions, quantifiers, and variable bindings.
- XQL is another XML query language which makes use of path expressions. XQL is generally used to find and sort the elements (data fields) and text in an Extensible Markup Language (XML) document.
- XQuery is an XML query language which is generated for XML data sources. W3C working group have produced XQuery and now it is in approval status.
- FLWOR expressions are considered as a main section of the XQuery language. The full form of FLWOR is FOR LET WHERE ORDER BY and RETURN.
- The function of a query processor is to take out the query's high level abstraction and its procedural assessment into a group of low-level operations.
- When performing query processing fir relational structure, XML document or information associated to XML document is accumulated in relational database.

- In the approach of Query Processing on Storage Schema, elements of XML are allocated label. The reason behind labelling is to generate exclusive identifiers that will be functional for query processing.
- Existing XML management systems are divided into XML-enabled databases and native XML databases.

6. GLOSSARY

- ***BEA/XQRL:*** A query processor BEA/XQRL executes relational scheme by means of XQuery. Query compiler parses and optimises the query.
- ***FLWOR:*** FLWOR expressions are considered as a main section of the XQuery language which signifies FOR LET WHERE ORDER BY and RETURN.
- ***Lorel:*** It is considered as an early query language used for for semi structured data.
- ***ORDPATHS:*** It is a labelling method which is executed in SQL server. This method labels every node by a series of integer numbers. As the name suggests, Order, depth, parent, and ancestor-descendant relationships are there in this method.
- ***Query processor:*** The function of a query processor is to take out the query's high level abstraction and its procedural assessment into a group of low-level operations.
- ***Quilt:*** It is considered as a functional language where you can represent a query as expression.
- ***XML-QL:*** It is an XML query language which is used for the process of querying, constructing, converting and incorporating the data of XML.
- ***XQL:*** It is generally used to find and sort the elements (data fields) and text in an Extensible Markup Language (XML) document.
- ***XQuery:*** It is an XML query language which is generated for XML data sources.

7. TERMINAL QUESTIONS

1. What is XML-QL? Discuss its syntax with examples.
2. Illustrate the concept of XQuery. Also discuss FLWOR expressions with example.
3. What are the different approaches for XML Query Languages? Explain.
4. Differentiate between XML oriented databases and native XML databases.
5. What are the advantages and disadvantages of Lorel query language? Discuss.

8. ANSWERS

Self-Assessment Questions

1. CONSTRUCT
2. b) Lorel
3. False
4. XQL
5. XQuery
6. False
7. a) Scheme-oblivious
8. a) Xpath
9. True

Terminal Questions

1. XML-QL is an XML query language which is used for the process of querying, constructing, converting and incorporating the data of XML. Refer Section 2 for more details.
2. XQuery is an XML query language which is generated for XML data sources. W3C working group have produced XQuery and now it is in approval status. Refer Section 2 for more details..
3. Based on the levels of abstraction, the query processors are divided into the following approaches according to their storage models. They are relational processing and native storage processing. Refer Section 3 for more details..
4. XML-oriented databases are typically relational and enclose model- or pattern driven extensions for transporting data to and from XML documents and are usually intended for data-centric documents. Refer Section 4 for more details..

5. One of the advantages of Lorel language is its easy syntax which makes users to understand it more clearly. The drawback of Lorel language is that it is dependent on OQL parser. Refer Section 2 for more details.

9. REFERENCES

References:

- Jiang, Z. (2008) *On XML Query Processing*, Southern Illinois University at Carbondale.
- Hunter, D. (2007) *Beginning XML*, 4th Edition, John Wiley & Sons.

E-references

- http://www3.uji.es/~aramburu/curdoc/StoreXML.pdf, 13-04-12
- http://ils.unc.edu/MSpapers/2634.pdf, 13-04-12