BACHELOR OF COMPUTER APPLICATIONS

SEMESTER *5*

DCA3143

E-COMMERCE

Unit 6

XML and Data Warehousing

Table of Contents

1. INTRODUCTION

XML has grown from being just a way of defining data to allowing the definition of that data to be published in a standardised way. It also allows flexible transformations from one format to another and forms the basis of the various XML components that will enable applications to be built.

As you know, XML was defined as a way of describing data in an open, readable, and structured fashion. XML itself is not a format you can use in your application. XML is a rule set that tells you how to define an XML-based format for your own use.

1.1 Learning Objectives

After studying this unit, you should be able to:

- *Elaborate on the Need for XML*
- *Discuss Business to Business scenario.*
- *Relate Datawarehouse, Relational database and XML.*
- *Discuss the benefits of XML schemas to applications.*
- *Understand XML processors and their enforcement in XML structure.*

2. THE NEED FOR XML

XML is an international data standard, a sort of common language for computing. To explain the reason behind XML's popularity, we can say that it is a technique to structure electronic documents, and it intends to separate presentation, structure, and meaning from the actual content. What makes XML truly powerful is the international acceptance it has received. Many individuals and corporations have strived their hard work to make XML interfaces for databases, programming, office application, mobile phones, and more. As the name implies extendable markup language, you can create your tags or use the tags that have already been created.

So, XML is a set of tools that allow developers to create web pages and much more. XML enables developers to set standards defining the information that should appear in a document and what sequence. In combination with other standards, XML makes it possible to describe the content of a document separately from its formatting. It also makes it easy to reuse that content in other applications or for other presentation environments.

Most significant, XML provides a basic syntax that can be used to share information between different computers, applications, and organisations without passing through many conversion layers.

Though web developers are most involved with XML, database developers, document managers, desktop publishers, programmers, scientists, and other academics are also target audiences of XML. Even developers performing tasks on different types of applications with different interfaces and different data structures can share XML formats and tools for parsing those formats into data structures that applications can use.

3. ADVANTAGES OF XML, SGML, AND HTML

To exchange information use of XML offers many benefits. Let us look at the advantages of XML. Later in this segment, we will also see how XML is advantageous over SGML and HTML.

- XML can store and organise just about any information in a form that is tailored to your needs.
- As an open standard, XML is not tied to the fortunes of any single company nor married to any particular software.
- With Unicode as its standard character set, XML supports a staggering number of writing systems (scripts) and symbols, from Scandinavian runic characters to Chinese Han ideographs.
- XML offers many ways to check the quality of a document, with rules for syntax, internal link checking, comparison to document models, and data typing.
- With its clear, simple syntax and unambiguous structure, XML is easy to read and parse by humans and programs alike.
- XML is easily combined with style sheets to create formatted documents in any style you want. The purity of the information structure does not get in the way of format conversions.

All of these advantages come when the world is ready to move to a new level of connectedness. The volume of information within our reach is staggering, but the limitations of existing technology can make it difficult to access. Businesses are scrambling to make a presence on the web and open the pipes of data exchange but are hampered by incompatibilities with their legacy data systems. The open-source movement has led to an explosion of software development, and a consistent communications interface has become a necessity. XML was designed to handle all these things and is destined to be the grease on the wheels of the information infrastructure.

XML is famous for many features, and a few of them are listed below:

- **Easy Data Exchange:** In XML, data and markup are stored as text that we can configure. If we like, we can use XML editors to create XML documents. Still, if something goes wrong, we can examine and modify documents directly because it's all just text.XML

provides a very efficient way of storing most data. Besides, when we standardise markup languages, many different people can use them.

- **Customising Markup Language:** We can create customised markup languages using XML, representing its extraordinary power. We can create a personalised browser to handle that language.
- **Self-Describing Data:** The data in the XML document is self-describing. We can create our tags in XML so that if we go back to our document years later, we can figure out what's going on.
- **Structured and Integrated Data:** In an XML document, we can specify not only data but the structure of that data can also be specified. When dealing with complex and important data, we can integrate various elements into other elements.
- **Well-Formed XML Documents:** XML documents must follow the syntax rules set up properly. Each element also must nest inside any enclosing elements properly. For example, if in any example of XML, we are opening a tag but not closing the same, it is not a well-formed XML document.
- **Valid XML Document:** An XML document is valid if there is a document type definition (DTD) associated with it and if it complies with that DTD.

Now, let us understand the SGML and the advantages of XML over it.

SGML is one of the first languages to tackle the problem of transferring electronic data as marked-up text. SGML derives most of its power, and therefore its complexity, as a meta-language, describes and defines markup languages. The markup contained in SGML and its derivatives are described within SGML. As we know traditionally, markup languages are made of tags and elements used to describe data, either for presentation or as content. The meta-languages apply those tags and elements and describe the systems and meaning of tags or other annotations. Additionally, markup is expected to provide any system or encoding methods and describe how the encoding is to be interpreted.

Extensively SGML is used for technical documentation, government documents, and the like, but it is far too difficult for practical use on the web. However, SGML subsets and derivatives are perfectly adaptable for Web markups, such as XML, HTML, and XHTML. All these

languages have a solid likeness to each other in syntax and markup conventions. Though XML takes multiples functionality from SGML still, it adds several advantages over SGML.

Some of the advantages of XML over SGML are:

- XML allows well-formed documents to be parsed without a DTD, whereas many SGML implementations require some DTD for processing.
- Syntactically XML are simpler and more permissive compared to SGML.
- XML specifications are also minimal.
- XML was created because a direct implementation of SGML on the Internet was complex.

One of SGML's advantages is that it provides significant flexibility for a diverse community of users by providing a wide array of choices, which resulted in a wide range of syntactical variations for documents.

Now, let us understand how HTML differs from XML.

HTML was also created to fulfil a very different requirement than XML. XML-enabled HTML is known as XHTML, which uses universally defined and accepted elements of HTML4.01. At the core, the differences between HTML and XML are straightforward:

"HTML is a presentation markup language, readable and rendered by almost any modern Web browser, whereas XML is a content markup language, with no inherent, or built-in, presentation elements, only content-definition elements".

The web is a compelling medium for exchanging information, but unfortunately, HTML is not designed to accommodate a wide variety of data types. XML, because of its extensibility and derivation from SGML, is much more suited for data exchange (but without resorting to the complexity of SGML).XML allows elements to be designed as an application-specific process, defined in a schema or DTD, and used over the web in a language that best describes the data itself. By defining the XML elements needed, several otherwise nearly impossible tasks can be accomplished. Again, XML defines the data, and HTML defines the presentation. To present it another way, XML processes data, whereas HTML displays data.

Since XML is intended as a content markup language, you can specify what the actual data is that is contained in the tags. For example, you might want to create a list of books that consists of the title, the author, the subject, the publisher, and more. There aren't any HTML elements you can use to specify that the enclosed text is the author's name. XML doesn't have these limitations. You are relatively free to create the elements you need as long as you have them defined in a DTD or other document (such as a schema) that the reading software or agent can access.

To migrate from HTML to XML is not difficult, but the details are essential. Here are some of the key differences between XML and HTML:

- XML is a content markup language while HTML is a presentation markup language.
- XML allows user-defined elements whereas HTML elements are predefined.
- XML requires validation while in HTML, almost anything goes.
- XML is data-driven whereas HTML is display-driven.
- XML allows data exchange between software applications while HTML is designed for visual presentation.
- XML is strictly defined and interpreted whereas HTML is very loosely interpreted.
- XML elements must be closed while in HTML, empty elements do not need to be closed.

4. B2B SCENARIOS

Interaction is the fundamental nature of commerce, where it is online (e-commerce) or offline—the buying and the selling of the goods resulting from the interaction between the sellers and the buyers. Until the early 20's marketplaces were typically physical structures, but in the 21st century, it is more on the internet, popularly known as e-commerce. However, in the 1970s, with the introduction of computer technology and electronic data interchange, people started buying and selling in the virtual world. That shift continued as technologies became more affordable, and business organisations like Amazon started inserting technologies deeper into their internal and external processes using the internet and the web.

The XML promotes a message-oriented view of e-commerce that isolates business transactions from hardware, software, system architectures, and programming languages.

Some examples of business messages are:

- purchase the order from the buyer of the goods to a seller
- invoice from the seller of the goods to the buyer
- requests to fulfil the payment using debit/credit card
- authorisation of using the card
- status on success or failures of the services

4.1 E-Business System Involved: Delivery, Sales

Whether the marketplace is virtual or physical, it is necessary for the buyers and the sellers to interact. No matter the type of business-to-business commerce you are engaged in, whether it is the buyer side, selling side, exchanging, or content aggregation, a marketplace exists only when it connects a buyer with a seller. Since computing technology is now an integral part of the e-commerce infrastructure, it is also an essential part of marketplace interaction. With e-commerce environments, three dimensions are to establish interaction with people and computers. They are:

- **Computer-to-computer:** It involves interaction between browsers, servers, applications, databases, and so on.

- **People-to-people:** This interaction happens between humans who buy, sell, administer the marketplace and provide customer service support.
- **People-to-computer:** In this scenario, the interaction between users and the systems take place that executes commerce processes.

4.2 Cross-Company Communication: Replacement for EDI

EDI came in around the 1960s. It was created so that the data on large amounts of paper information can be kept in a minimum space, but it was costlier than it should be. For example, for a 7MB HDD in the 1960s, you had to pay 18 Lakh INR. Apart from this, there was one more downside: the data was very specifically formatted, and hence it was impossible to read for someone not sound expertise in the language. The traditional EDI had many disadvantages such as:

- It was a costly technology
- It had implicit structure
- It was intended for machines only
- It also requires special networks to operate

With the modern storage solution, it no longer needs to rely on such premium services of EDI; hence this opened the door for XML to become e-commerce integration.

There are various forms of XML used in cross-company communication, such as:

- cXML is used to facilitate communication between various companies
- RailML is a standard for the railway industry to communicate specifications
- adsML is a B2B e-commerce advertising standard that deals with the communication between newspapers, broadcasters, and ad agencies.

Apart from the above mentioned, there are many other forms of XML, but e-commerce uses cXML primarily. There are various advantages of using XML over EDI, they are:

- it has an explicit structure
- it can easily use the internet
- it is cheaper to implement
- it provides easier validation
- it can open up e-commerce to small and medium-sized businesses

5. THE DOCUMENT AS THE APPLICATION

XML-based markup language is called an XML application. This application is not a usual application, which will use XML like web browser, rather an application of XML to a specific domain such as GedML for genealogy or chemical markup language for chemistry.

Each XML application adheres its own vocabulary and syntax. This is much like our languages, in which we use own grammars and vocabulary adhering certain fundamental rules.

XML documents are used in variety of applications such as web publishing, e-businesses and mostly on portable applications. Apart from that, the XML documents are used in metadata applications to make metadata portable and reusable, in general applications to provide standard method to access information and making it easier for use, store, transmit and display data. It is also widely used in web searching, returning useful web results and automating web tasks.

SELF-ASSESSMENT QUESTIONS – 1

1. Since _____ is now an integral part of the e-commerce infrastructure, it is also an essential part of marketplace interaction.
2. EDI stands for _____.
3. Name any two forms of XML used in cross-company communication.

6. XML AND RELATIONAL DATABASES AND DATAWAREHOUSES

The concept of a Data warehouse (DW) is to build a model by retrieving data from an operational system and making it available to a decision support system. This concept tries to address the flow of large amounts of data and the complex nature of the needs of an Information system. Huge redundancies were required to support multiple levels of the decision support system. In a big organization, the decision support system was independent. Each department used different information, but the data required for this information was common. The process of gathering and refining data from various sources was duplicated for each environment. Over time, the data got changed as new decision support systems emerged.

A data warehouse is a core component of Business intelligence. It is a central store for data from various sources of the organization. DWs store historical as well as current data that are used for generating analytical reports for business users.

In technical terms, a DW is a relational data store designed to retrieve data and analyze data

The latest Structured Query Language (SQL) includes a new part known as SQL/XML. This part introduces an XML datatype and facilities that provide for the composition of XML using data extracted from a relational database and, conversely, for storing data extracted from an XML document in a relational database. The XML data type can be used in the same way as any other data type, i.e., as a data type for a column, as a variable, or as a parameter for a function.

The facilities that enable data to be extracted and represented as XML offer various alternatives—the specimen data stored in the relational database tables. Fig. 1 demonstrates these facilities.

department

name	telephone_number	managed_by
HQ	NULL	NULL

Finance Production	ext 452 ext 664	CX137 CA446

employee

payroll_number	surname	forename	birth_date	start_date	department
AY334	Watson	Jenny	1988-05-25	1994-06-03	Finance
CX137	Rogers	Henry	1972-10-01	1995-01-03	Finance
DJ777	Phillips	Barbara	1974-05-05	1992-09-03	Finance
FJ678	Harrison	Roger	1970-01-12	2005-11-05	Finance

Employee_qualification

payroll_number	name	award_date	expiry_date
CX137 CX137 CX137	Dip FM ACCA First Aid	2005-08 2010-10-20 2012-04-25	NULL NULL 2015-04-24

Fig. 1: Specimen Data for XML Representation Examples

One option is to output the complete contents of a table or a schema into a standard XML structure using a simple mapping. There are two possible approaches to this. The first of these approaches provides a structure that is a valid XML document. Fig. 2 shows the XML obtained when outputting the contents of the employee table using this approach.

<employee>
<row>
<payroll_number>AY334</payroll_number>
<surname>Watson</surname>

<forename>Jenny</forename>
<birth_date>1988-05-25</birth_date>
<start_date>1994-06-03</start_date>
<department>Finance</department>
</row>
<row>
<payroll_number>CX137<payroll_number>
<surname>Rogers</surname>
<forename>Henry</forename>
<birth_date>1972-10-01</birth_date>
<start_date>1995-01-03</department>
</row>
<row>
<payroll_numberDJ137</payroll_number>
<surname>Phillips</surname>
<forename>Barbara</forename>
<birth_date>1974-05-05</birth_date>
<start_date>1992-09-03</start_date>
<department>Finance</department>
</row>
<row>
<payroll_number>FJ678</payroll_number>
<surname>Harrison</surname>
<forename>Roger</forename>
<birth_date>1970-01-12</birth_date>
<start_date>2005-11-05</start_date>
<department>Finance</department>
</row>
</employee>

Fig. 2: The employee Table represented as a Valid XML Document

The XML shown in Fig. 2 is a valid XML document because the XML is in the form of a true hierarchy, having a single root element. In this case, the root element is called 'employee', and the name is automatically derived from the name of the table. There are a series of elements, called 'row', for each row in the table. There is an element for each column in the table within each' row' element, with the element name also being automatically derived from the column names.

The alternative approach provides XML elements without the root element. Fig. 3 shows the XML obtained when outputting the contents of the employee table using this alternative approach.

<employee>
<payroll_number>AY334</payroll_number>
<surname>Watson</surname>
<forename>Jenny</forename>
<birth_date>1988-05-25</birth_date>
<start_date>1994-06-03</start_date>
<department>Finance</department>
</employee>
<employee>
<payroll_number>CX137<payroll_number>
<surname>Rogers</surname>
<forename>Henry</forename>
<birth_date>1972-10-01</birth_date>
<start_date>1995-01-03</department>
</employee>
<employee>
<payroll_numberDJ137</payroll_number>
<surname>Phillips</surname>
<forename>Barbara</forename>
<birth_date>1974-05-05</birth_date>
<start_date>1992-09-03</start_date>
<department>Finance</department>

```
</employee>
<employee>
<payroll_number>FJ678</payroll_number>
<surname>Harrison</surname>
<forename>Roger</forename>
<birth_date>1970-01-12</birth_date>
<start_date>2005-11-05</start_date>
<department>Finance</department>
</employee>
```

Fig. 3: The employee Table Represented as XML without a root Element

The XML shown in Fig. 3 has a series of elements, each named after the table name, with one element for each row of the table, but it lacks a root element. As before, within each row element, there is an element for each column in the table, with the element name automatically derived from the column names. This output could be made into a valid XML document by simply adding a root element. It could be done automatically by the database management system.

It is also possible to create an XML document that contains data selected from one or more tables. This is achieved by using special SQL functions, known as XML publishing functions.

XML documents and relational databases represent and structure data in very different ways. This draws an invisible border between the two environments. Getting these data platforms to cooperate efficiently can be as challenging as negotiating a treaty between two very different cultures, as shown in Fig. 4.

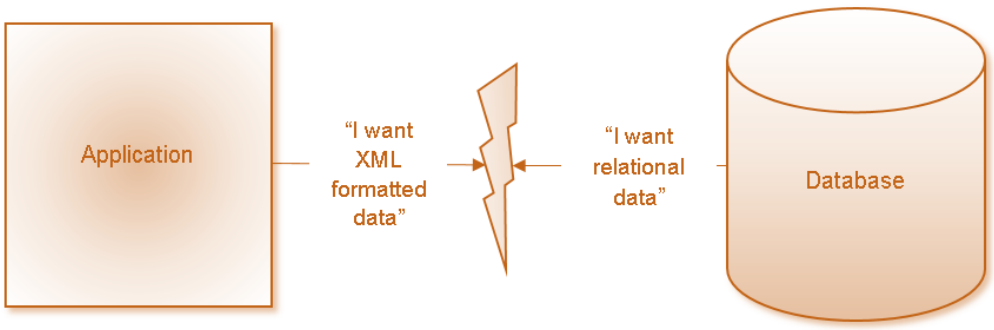

Fig. 4: Application Components and Databases have Different Data Format Preferences

Let us compare XML and relational databases. The essential feature of any data management system is its ability to store information securely. This is where relational databases provide an unparalleled set of features, barely comparable to XML's simple file format. XML documents and relational databases approach the representation of data from different ends of the spectrum. XML documents introduce a cohesive, structured hierarchy, whereas RDBMSs provide a more flexible relational model.

These two platforms face significant integration challenges because XML document hierarchies are difficult to recreate within relational databases, and relational data models are challenging to represent within XML documents. Table 1 shows a data representation comparison.

Table 1: Data Representation Comparisons

	Databases	**XML**
Data Model	Relational data model, consisting of tabular data entities (tables), with rows and columns.	Hierarchical data model composed of document structures with element and attribute nodes.
Data Types	A wide variety of data types typically are provided, including support for binary data.	XSD schemas are equipped with a comparable set of data types.
Data Element Relationships	Column definitions can interrelate within and between tables, according to DDL rules.	References can be explicitly or intrinsically defined between elements.

Just as there are several alternative approaches to extracting data from a relational database and publishing it as XML, many approaches can be used to take data from an XML document and place it in a relational database.

7. BENEFITS OF XML SCHEMAS TO APPLICATIONS

XML schema is also known as XML Schema definition (XSD). It is pretty similar to the database schema that describes the data in a database. Likewise, it is also used to describe and validate the structure and the content of XML data. It defines the data types, attributes, and elements. The schema elements support Namespaces.

The purpose of an XML schema is to define and describe the legal building blocks of an XML document:

- The data types for elements and attributes
- Default or say fixed values for elements and attributes
- The order and the number of child elements
- The elements and attributes that may appear in the document

There are numerous benefits of XML schema. Some of them are:

- XML schema supports Data types
- XML schema use XML syntax
- It secures Data communication

Let's explore them one by one.

XML schema supports data types

It is one of the notable features of XML schemas that it supports data types, which benefits into:

- It makes it easier to define data facets
- Data patterns are easier to define
- Data conversion between different data types become easy
- It is easier to validate the data correctness and
- Describing allowable document content is also easier

XML schema use XML syntax

Another significant benefit of XML schemas is that they are written in XML, which benefits into:

- No need to learn a new programming language

- XML editor can be used to edit Schema files
- XML parser of your choice can be used to parse Schema files
- Using XSLT, you can transform your Schema
- Using XML DOM, You can manipulate your Schema

We know that XML schemas are written in XML; hence it is extensible. With an extensible schema definition, we can:

- Derive own data types from the standard data types
- Reference various XML schema in the same XML document
- Reuse our Schema in other schemas

It secures data communication

It is essential for both sender and receiver that the data must be received the same as it was sent. Using XML schemas, the sender can describe the data so that the receiver can easily understand it.

STUDY NOTE

A simple element is an XML element that contains only text without any elements and attributes. It may have a default value or fixed value specified. The default value is automatically assigned if nothing is specified. Other one is fixed value which is also automatically specified and it cannot specify another value.

For example, A date "04-05-2021" in some countries, be interpreted as 4th May and in other countries as of 5th April. However, the XML date datatype puts the date like this:
<data type=”date”>2021-05-04</date>

Since the XML data type "date" requires the format "YYYY-MM-DD", it ensures a mutual understanding of the content.

SELF-ASSESSMENT QUESTIONS – 2

4. XML documents and relational databases represent and structure data in very different ways. (True/False)
5. XML documents introduce a cohesive, structured _____model, whereas RDBMSs provide a more flexible _____ model.
6. Generating XML from relational data is also called _____or the composition of XML documents.
7. _____ is an ANSI and ISO standard that supports using XML in the context of a relational database management system.
8. _____is used to construct a sequence of XML elements.
9. _____ is used to aggregate XML elements.

8. XML PROCESSORS ENFORCING STRUCTURE

The process of reading XML documents by a software program and acting accordingly is called processing the XML. The program that reads and processes those XML documents is called an XML processor. The primary function of an XML processor is to read the XML file and convert it into an in-memory structure so that the rest of the program can access it. i.e., it converts the XML document into an internal representation for other programs to use. It is called a parser which is an essential component of every XML processing program.

The XML processors are classified into two categories, depending on whether they check XML documents for validation. They are:

a. Validating
b. Non-validating

It is necessary for the processor that discovers a validity error to report it; otherwise, it can continue with standard processing. For example, some of the validating and non-validating processors are depicted in Fig. 6.

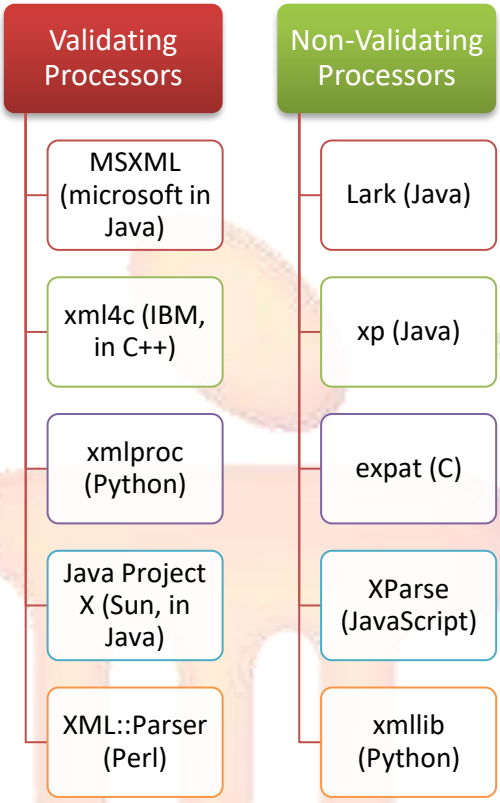

Fig. 6: Examples of Validating and Non-validating Processors

XML's structured text can be processed in several ways, such as programs can look at XML as text, as a stream of events, as a serialisation of some other structure, or as a tree. There are lots of tools supporting all these operations.

a. Treating XML as Text

We know that fundamentally XML documents are text, i.e., both markup and content are represented as text in XML documents. Simple text-editing tools can be used for XML document inspection, creation, and modification.

However, XML has some limitations for the programs that try to process XML documents as text documents. It is quite easy for the XML documents to use basic text-editing tools for regular expressions, but it becomes complex as namespaces, entity processing, and attribute defaulting are added to documents. Using these features becomes problematic when the documents are purely treated as text.

Text-based processing can be performed in addition to other XML processing. Parsing and reserialising XML documents after other processing doesn't always result in the desired results. For example, XSLT will replace entity references with entity content.

b. Treating XML as Events

When an XML parser reads a document, it goes from the beginning of the document to the end. However, it may pause to fetch the external resources for DTD or any other external entity. The list and structures of events become complex as features such as whitespaces, attributes, comments, processing instructions, etc. The "Events" are the content of an element, the start of an element, and the end of an element. For example, given the document:
<name><given>Andrew</given><family>Warner</family></name>

an event-based parser might report events such as:

```
startElement:name
startElement:given
content: Andrew
endElement:given
startElement:family
content:Warner
endElement:family
endElement:name
```

However, event-based parsers only need to keep track of only limited information. They only require understanding the DTD's content if the documents are using them. In addition, they also need to maintain context stacks for namespace declarations and element names, i.e., they are not required to keep a complete record of the document as they parse it, which results in minimisation of the memory needed for the parse. Despite its difficulty, event-based parsers are very useful for various tasks.

c. Treating XML as Tree Models

XML documents also describe tree structures because of the requirements for well-formedness. As we have already studied, an XML document typically contains elements that contain texts, attributes, and elements, and these may also contain texts, attributes, and elements. Apart from that, declarations, comments, and processing instructions also enrich the XML documents.

Working with a tree model of an XML document does not much differ from working with a text-based document. Also, it is easy to move around well-formed portions of a document or modify it as the entire document is always available.

The tree model of documents also has some drawbacks. They use more spaces in the memory, typically multiplying the original document's size. It also requires additional processing for navigating the documents after the parse. Both described issues make it challenging to scale and share applications as they rely on the tree models. Though, Tree-models are still appropriate to use where small numbers of documents are used.

8.1 Channels

The channel definition format (CDF) was an XML application developed by Microsoft for defining channels. Channels allows websites to automatically notify the end users of about any critical information changing. This is also known as push or webcasting method. Internet Explorer was the only browser to adopt CDF.

CDF files are basically XML document which is separate but linked to HTML documents on the site. Every CDF document defines parameter for connection between the readers and the site's content. The data are transferred through push (sending notification) or pull (user choose the pages to load) in the browser to get the update information.

Here are the steps to establish a channel:

- Decide the content to be included in the channel
- Write channel definition file which will identify the content
- Now, link the CDF to home page of website.

SELF-ASSESSMENT QUESTIONS – 3

10. The process of reading XML documents by a software program and acting accordingly is called processing the XML. (True/False)
11. The primary function of an _____ is to read the XML file and convert it into an in-memory structure so that the rest of the program can access it.

9. RELATIONSHIP BETWEEN DATA WAREHOUSE AND XML

Data Integration:

XML can be used as an intermediary format to facilitate data integration between disparate systems and the data warehouse. When data from various sources is in different formats, organizations may use XML to standardize the data representation before loading it into the data warehouse. This allows for easier data transformation and consolidation during the ETL process.

Data Exchange:

XML can serve as a common data interchange format between the data warehouse and external systems, such as applications, partners, or customers. It enables seamless data exchange, ensuring compatibility and interoperability between different systems.

Metadata Management:

XML can be used to define metadata in the data warehouse. Metadata provides information about the structure and meaning of the data, such as table and column names, data types, relationships, and data definitions. Using XML to store metadata allows for flexibility and extensibility in describing complex data structures within the data warehouse.

Data Presentation:

XML can be utilized in the presentation layer of a data warehouse for generating customized reports and dashboards. By exporting query results in XML format, applications and end-users can easily interpret and display the data according to their requirements.

It's important to note that while XML has been widely used in the past for data interchange, newer data integration and exchange technologies, such as JSON (JavaScript Object Notation) and RESTful APIs, have gained popularity due to their simplicity and efficiency. However, XML may still be used in legacy systems or specific domains where it is required or preferred.

10. SUMMARY

- With the emergence of computer technologies, e-businesses have overgrown.
- EDI was a premium solution for data exchange, which is replaced by various forms of XML.
- The XML data type can be used in the same way as any other data type. Such as, as a data type for a column or as variable or as a parameter for a function
- XML documents and relational databases represent and structure data in very different ways.
- There are several ways to publish XML. In this unit, you have studied the following types:
  - Use SQL/XML
  - Use DB2 XML Extender
  - Write your own program
- XML schema is also known as XML schema definition (XSD). The purpose of XSD is to define and describe the legal building blocks of an XML document.
- The primary function of the XML processor is to read and process the XML documents. XML processors are either validating or non-validating.
- XML structured text can be processed in several ways, such as it can be treated as text, events or Tree Models.

11. GLOSSARY

- **e-commerce:** Doing business online either via app or web portal is commonly known as e-commerce, such as Amazon.com is a e-commerce website.
- **Schema:** In terms of database, a schema is nothing but the design which consists three parts, physical, logical and view schema.
- **Dynamic Web:** Dynamic web or dynamic website changes its information depending on the time zone, viewing device, the time of the day etc.
- **ANSI:** American National Standards Institute, it creates standards in various fields including computer programming.
- **ISO:** International Organization for Standardization, it is UK based organisation that also deals with the standards.
- **CLOB:** Character Large Object, is a data type just like int and char used by various DBMS including DB2 and oracle.

12. TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1. Discuss B2B scenarios in the E-business system.

Q2. Explain comparison between XML and Relational database.

LONG ANSWER QUESTIONS

Q1. Discuss the benefits of XML schemas to applications.

Q2. What is an XML processor? Explain how XML's structured texts are processed in several ways.

13. ANSWERS

SELF ASSESSMENT QUESTIONS

1. Computing technology
2. Electronic Data Interchange
3. cXML, RailML
4. True
5. Hierarchy, relational
6. XML publishing
7. SQL/XML
8. XMLFOREST
9. XMLAGG
10. True
11. XML processor

TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Answer 1: In the 1970s, with the introduction of computer technology and electronic data interchange, people started buying and selling in the virtual world. That shift continued as technologies became more affordable, and business organisations like Amazon started inserting technologies deeper into their internal and external processes using the internet and the web.

For more details, refer to section 5.2.

Answer 2: XML documents and relational databases approach the representation of data from different ends of the spectrum. XML documents introduce a cohesive, structured hierarchy, whereas RDBMSs provide a more flexible relational model.

For more details, refer to section 5.4.

LONG ANSWER QUESTIONS

Answer 1: There are numerous benefits of XML schema. Some of them are:

- XML schema supports Data types
- XML schema use XML syntax
- It secures Data communication

For more details, refer to section 5.6.

Answer 2: The program that reads and processes those XML documents is called an XML processor. The primary function of an XML processor is to read the XML file and convert it into an in-memory structure so that the rest of the program can access it. i.e., it converts the XML document into an internal representation for other programs to use.
For more details, refer to section 5.7.

14. SUGGESTED BOOKS AND E-REFERENCES

BOOKS:

- Jamsa, K. A., King, K., & Anderson, S. F. (2002). *HTML & Web design: Tips & techniques.* New York, NY: McGraw-Hill/Osborne.
- Harold, E. R., & Harold, E. R. (2004). *XML 1.1 bible.* Indianapolis, IN: Wiley Pub.
- Hampton, K. (2004). *XML Publishing with AxKit.* Sebastopol, CA: OReilly Media.

REFERENCES:

- The B2B XML Service. (n.d.). Retrieved from *https://www.elia.be/en/customers/customer-tools-and-extranet/the-b2b-xml-service*
- (n.d.). Retrieved from *https://www.w3.org/XML/RDB.html*
- (n.d.). Retrieved from *https://www.ibm.com/docs/en/db2/11.5?topic=overview-comparison-xml-relational-models*
- 10 XML Schema Definition. (n.d.). Retrieved from *https://docs.oracle.com/cd/B14099_19/integrate.1012/b14069/xsd.htm*
- Java & XML, 2nd Edition. (n.d.). Retrieved from *https://docstore.mik.ua/orelly/xml/jxml/ch10_01.htm*
- Features and advantages of XML - javatpoint. (n.d.). Retrieved from *https://www.javatpoint.com/features-and-advantages-of-xml*