Chapter 2
Background and History

2.1 Introduction of Database and History of Database

Database technology is having a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, low, education, and library science. A database management system (DBMS) is a collection of interrelated data and a set of programs to access those data. Database systems are designed to manage large information. Management of data involves both defining structure for storage of information and providing mechanism for manipulation of information.

The origins of the database go back in 1960, for ‘Apollo’ moon-landing project [4]. At that time, there was no system available that would manage the vast amount of information that the project would generate. The first database product GUAM (Generalized Update Access Method) was developed on the concept of hierarchical tree which is nothing but the first data model i.e. hierarchical database management system. In the mid 1960, IBM developed first commercial DBMS is known as IMS (Information Management System). It is still main hierarchical DBMS used by most large mainframe installations. In the mid-1960s, another significant development was IDS (Integrated Data Store) from General Electronic [3]. The project developed new type of database system known as network DBMS which represents more complex relationships. In 1970, E.F. Codd developed first relational data model which is considered as second generation of DBMS [2]. In response to the increasing complexity of database applications, two new systems have emerged: the Object Oriented DBMS (OODBMS) and the Object-Relational DBMS (ORDBMS). This evolution represents third-generation DBMSs. In 1990, onwards emerging of web-based application in computer, E-commerce etc hence the fourth generation DBMS is used. The data is represented by semi-structure data model i.e. by XML. Today to mange search engine data and Face book data, the new term “big data” is used. DBMS evolutions are listed in table 2.1 below:-
<table>
<thead>
<tr>
<th>Years</th>
<th>Database characteristics</th>
<th>Commercial Product available</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>Navigational Database System</td>
<td>SABRE that was used by IBM to help American Airline manage its reservation</td>
</tr>
<tr>
<td></td>
<td>User need to know the physical structure of the database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network model and Hierarchical model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The first Data Systems language (CODASYL) was developed</td>
<td>Information Management System the First DBMS product developed by IBM</td>
</tr>
<tr>
<td></td>
<td>DBTG (Data Base Task Group) was formed in 1967.</td>
<td>IDS (Integrated Data Store) the first Network model developed by General Electronics.</td>
</tr>
<tr>
<td>1970</td>
<td>E F code introduced new relational model</td>
<td>INGRES was developed at the University of California-Berkeley</td>
</tr>
<tr>
<td></td>
<td>New QUEL - query language was introduced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A new database model called Entity-Relationship, or ER, was proposed by P. Chen in 1976</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The development of structural query language called SQL, which has since become standard query language for relational DBMS</td>
<td>System R was developed at IBM in San Jose and led IBM's SQL/DS &amp; DB2, ORACLE, HP's Allbase, and Tandem's Non-Stop SQL. DB2 became one of the first DBMS (Database Management System) product based on the relational model (Oppel, 2011).</td>
</tr>
<tr>
<td></td>
<td>Various commercial products DB2 and SQL/DS, Oracle were developed.</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>Network and hierarchical models lost their charm with no further development of these systems, but some legacy systems are still in use today</td>
<td>RIM, RBASE 5000, PARADOX, OS/2 Database Manager, Dbase III, IV (later FoxBASE, and Visual FoxPRO), and Watcom SQL. All these systems were introduced during the1980’s and were based on the relational model.</td>
</tr>
<tr>
<td></td>
<td>OODBMS System was developed</td>
<td></td>
</tr>
<tr>
<td>Early 1990</td>
<td>Client Server Model development</td>
<td>World Wide Web, ASP, Front Page, JDBC, EJB, Apache and MySQL etc development of technology</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ORDBMS --- bridge the gap between relational database and the Object-Oriented model</td>
<td>Traditional RDBMS product focus on the effective management of data</td>
<td></td>
</tr>
<tr>
<td>Parallel Database products are in market</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Late 1990</td>
<td>Explosive growth of World Wide Web, the database system had to support very high transaction-processing rates as well as very high reliability and 24 * 7 availability</td>
<td>Database system support Web interfaces to data.</td>
</tr>
<tr>
<td>In 2000</td>
<td>Major development which took place during 1997 was the introduction of the Extensible Markup Language (XML) (DuCharme, 2012).</td>
<td>Growth of open source database system - MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>New query language XQuery was introduced for XML</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Databases have become more sophisticated and have features such as triggers, cascading update, and delete. This prevented updaters from creating inconsistencies between the tables.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Databases also developed a simplified procedural language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specialized techniques -- column stored system, highly parallel database system designed for analysis of very large database.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specialized Distributed database system have been built to handle the requirement of very large Web site such as Amazon, Facebook, Google, Yahoo!, Microsoft etc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Work on management and analysis of streaming data such as stock market ticker data or computer network monitoring data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data mining techniques are widely used. Applications are Web-based produce recommendation, automatic placement of relevant advertisements on Web pages.</td>
<td></td>
</tr>
<tr>
<td>21st Century</td>
<td>Data is used in large science databases like the genome project, the geological, national</td>
<td>Sybase/SQL Servers’ transact SQL, Oracle’s PL/SQL, and</td>
</tr>
</tbody>
</table>
security, and space exploration data PostgresSQL.

Huge terabyte systems are also appearing and will require novel means of handling and analyzing data.

New concept Data mining, data warehousing, and data marts was introduced for handling terabyte data

**Carlo Strozzi introduced NoSQL** - It is lightweight Open-Source relational database that did not expose the standard SQL interface

In 2011, DB developers started working on **UnQL (Unstructured Query Language)**, a specification for a query language for NoSQL databases

UnQL is claimed to be a superset of SQL within which SQL is a very constrained type of UnQL for which the queries always return the same fields, same number, names and types.

UnQL does not cover the data definition language (DDL) SQL statements like CREATE TABLE or CREATE INDEX.

major internet companies, such as Google, Amazon, Twitter, and Facebook used NoSQL concept for data management

The database evolution study suggests that from year 2000 the growth of internet application is increased. The need of database is not restricted to the database experts. Hence a user friendly, portable, flexible database as well as query language is required for manipulation of data information. The XML serves all these needs. It acts as database as well as transforms the data from one application to another application. It is a textual data format with strong support via Unicode for the languages of the world. Thus XML is defined [24] as a general-purpose markup language that is used to describe both data and a corresponding structure in machine-human readable format. It is a simplified descendent version of Standard Generalized Mark-up Language [21] which was originally designed in 1960 to facilitate information-sharing among homogeneous systems, XML initially appeared in 1996 and in February 1998 it was accepted [24] by the World Wide Web Consortium (W3C) as the standard medium for exchanging the data over the web.
2.2 XML History

XML is emerging as a de facto standard for information exchange over internet. It is derived from SGML and HTML. The current infrastructure is available to deal with HTML content and the same infrastructure can be re-used to work with XML; hence most of the companies can easily integrate their business document. For example, General Motors [25], Jet Blue Airline use XML in their business.

Apart from these characteristics, the XML has additional characteristics and these are: - high flexibility, platform independency, portability, simplicity, and usability. It is a human language which is conversable and readable by the people who had no formal introduction to XML. Flexibility means programmer or user can create his own tags; which are not limited to standard tags; which are not predefined by programmer. XML is vendor independent and system independent also. If a program is using vendor dependent tags, there are limitations such as the browsers and other programs associated with it. In such a case, these tags need to be approved of by the concerned authorities. Even the users have to get accustomed to the usage. This is a time consuming process. This limitation is overcome by XML. It is fully compatible with applications like JAVA, C etc and it can be combined with any other application which is capable of processing XML, irrespective of the platform. Due to all these characteristics, today XML is in use in most of the businesses.

XML was emerged from SGML [24]. In 1980, SGML (Standard Generalized Markup Language) was defined by ISO 8879. SGML has been the standard, vendor-independent way to maintain repositories of structured documentation for more than decade, but it was not well suited to serve documents over the web. In 1990 HTML came into picture. It is simple and has predefined tag semantics and tag sets, but unable to represent structural relationship and search utility. These drawbacks are overcome by XML. It provides a facility to define tags and structural relationships between them. There is no predefining meaning of tag set and preconceive semantics. It will either define by the applications that process them or by style sheet this enhances the functionality of the internet. Following fig 2.1 summarizes XML development.
The XML file is simple text file which is composed of element node, attribute node and comment node.

2.3 XML Data Representation Components:

The XML element is a piece of text that is enclosed within a start tag (<-- /) and end tag (/ -- >) It represents the object or entity. Element node is simple element, complex element and mixed element. An XML element can be associated with one or more XML attributes. The attribute node represents the characteristics of element node. There are two major issues to be considered when using attributes. An attribute cannot be repeated more than once within a single element definition and its value part must be enclosed in double-quotes as a string regardless its data-type. Three special-purposes attributes namely “id”, “idref” and “idrefs” are used to support references between different parts of an XML documents. The benefit of this layout (1) minimizing the XML document size by preventing data redundancy (2) reducing the cost of updating such reference data by ensuring that update will take place at one position in the XML document. Comment node represents comments of the XML document. It can be found
anywhere inside XML documents and they are ignored by most of the API that manipulates XML documents. The XML document structure is mapped in fig 2.2 as follows:-

![Fig 2.2 XML File structure](image)

Other XML components are DTD (Documents Type Definition) and XML schema. They are used for validating XML document.

This XML file is converted into predefined data representation format which is called as XML data model. XML data is represented into different data model, some of the important data models are: (1) XPath (2) DOM (3) XML infoset (4) edge label tree (5) node label tree (6) OEM etc. This model has an important role in query processing. The desired data must be retrieved as per the requirement.

2.4 XML- Query Languages:--

XML supports two types of queries: IR Style and Database Style of query. Database style of queries is used on ‘Data-Centric’ application.

The Database queries are broadly classified into containment queries, order queries and content-based queries [1]. The containment queries retrieve ‘Parent-Child’, ‘Ancestor-Descendent’ relationships. The order sensitive queries are forward and backward navigation queries. They are following, following-sibling, preceding, preceding-sibling, child axis, self axis queries. The content-based queries are associated with element or attribute node content .These query check the attribute and element content and retrieve it.
These queries are further classified into: tree structure queries and starting node queries. The tree structure queries return a small tree. They are further classified into simple path query which corresponds to a chain-path and branch path expressions – which corresponds to a small tree, called twig. The starting node queries are further divided into “total matching queries and partial matching queries. The total matching queries start from the root of the document where as partial matching queries start from some internal node.

IR-Style of queries is used to query text-dense XML repository whose value elements are involving long text. These types of queries do not work well in common database query standards. These queries are classified into DB+IR Style and IR-Style of queries. DB+IR queries enhance database-style XML queries such as XPath and XQuery queries. For IR-Style of queries, a full-fledged retrieval technique is required, for example ‘keyword’ search is used for it. Some of the query language supports IR Style of query, or Database Style of query or both.

XML has a rich set of query languages. Milestone of XML languages [14] are listed on the following tables2.2:-

<table>
<thead>
<tr>
<th>Query Language</th>
<th>Language Type</th>
<th>Input Model</th>
<th>Class of Query</th>
<th>Public Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lorel [49]</td>
<td>Declarative</td>
<td>OEM</td>
<td>Path expressions within OQL</td>
<td>1997</td>
</tr>
<tr>
<td>XQL [50]</td>
<td>Functional</td>
<td>Node-Labeled Tree Data Model</td>
<td>XQL based on path expressions</td>
<td>1999</td>
</tr>
<tr>
<td>Quilt [51]</td>
<td>Functional</td>
<td>Edge - Labeled Tree Data Model</td>
<td>Quilt Expression</td>
<td>2000</td>
</tr>
<tr>
<td>XPath [16]</td>
<td>Functional</td>
<td>XPath</td>
<td>Axis query and predicate query</td>
<td>2002</td>
</tr>
</tbody>
</table>
XPath and XQuery [5] are two major languages for database-style of queries and originally developed and recommended by the W3C Consortium [21]. XPath [22] is a basic query language that selects the nodes from XML documents from the root path. By using XPath query, the XML tree is traverse in different direction. XQuery [23] language is more expressive than XPath. An XQuery include axis traversal of XML tree as well as For-Let-Where-Return (FLWR) clauses, which can be nested and composed with full generality [7] means each clause includes a sub-XQuery.

The XPath is used for traversal of a complete XML file in different directions like forward, backward direction. There is lots of work is performed on the query optimization of the XML file. XQuery not only traverse a complete tree but also performed selective operation on it. It performs almost all type of query operation of XML database. Much expressive powered of XQuery increased the optimization and evaluation complexity. In literature different XQuery optimizer techniques are proposed by tree algebra for XML (TAX) [8] generalized tree pattern (GTP) [27], and tree logical class (TLC) [17] etc. For our research we select XQuery language and all different types of query, especially FLOWR clause which may return a single node, twig pattern and all order node i.e. following, following-sibling, preceding, preceding-sibling etc.

Another reason of selection of XQuery Language, it satisfies the need of all community i.e. “Document Community”, “Database Community” and “Programming Community”.

- Document-Centric applications often require text search, XQuery supports operations on document order and axis expressions, which are used to navigate in the document and to access the context of particular document fragments.
- Data-Centric applications often require very efficient selection, retrieval, and transformation of small fragments of data stored in massively large databases. XQuery incorporates features from query languages for relational databases (SQL) and Object-Oriented databases (OQL).

XQuery is a purely functional language which supports user defined functions and recursion.

The different feature of XQuery Language as per specification of [W3C3, W3CS4] is:-

- It is an XML query language with some programming language features and SQL-like semantics.
- It is developed from XPath data model.
- It has all functionalities, libraries and capabilities of XPath2.0 (i.e. XPath 2.0 is a subset of XQuery)
- It is supported by most commercial RDBMS such as IBM, Oracle and Microsoft SQL-Server.
- In addition to XPath2.0 capabilities, XQuery supports FLWOR [W3CS4] expressions (FLWOR is an acronym for "For, Let, Where, Order by, Return")
- XQuery1.0, a W3C Recommendation on 23/Jan/2007, is the latest version of XQuery.

Hence the researcher used the XQuery language for the research the detail of XML query processing is discusses as following:

2.5 XML Query Processing :-

As per W3C [8] XML processes are divided into two steps: - “External Query is processing “and “Internal Query processing “[5]. The fig 2.3 provides the abstract view of XML query processing.
**External Query Processing:** - In these steps, the XML document is converted into data model. This data model is used for query processing. It is a two step process:-

**Step I:** - **Data Model Generation:** - XML file is parsed. Different tokens are generated and are validated against elements and attributes declarations and target namespace. It generates information set called as PSVI (Post-Schema Validation Infoset). From PVSI XML-data model is generated.

**Step II:** - **In scope schema define:** - XDM i.e. XML Schema Definition Model is used for defining schema definition. It defines all schema used by XML file. The file is divided into three parts: - in-scope schema types, in-scope element definition and in-scope-attribute declaration. ‘In-scope schema type’ includes definition of all types of definitions found in imported schema. ‘In-scope element declaration’ includes all element declarations found in imported schema. ‘In-scope attribute declaration’ defines all the attributes found in imported schema.

**Internal Query Processing:** - In this step XQuery is processed against XML file which is stored into XML Data model. Following steps are used for query processing:-

1. **Parsing:** - The query is parsed and checked for grammatical error, i.e. syntax error. If no error is found, an internal operation tree of the parsed query is created.

2. **Normalization or OP-Tree Generation:** - In this process, complex query is rewritten into simple query expression i.e. converting the query expression into core language. If the query is not well-typed, static type errors are generated. For instance, a comparison between an integer value and a string value might be detected as a type error during the static type analysis.

For instance, the following [expression/query]

```xml
for $i in (1, 2),
   $j in (3, 4)
Return
   Element pair { ($i,$j)
```

is normalized to the Core expression

```xml
For $i in (1, 2) return
for $j in (3, 4) return
   element pair { ($i,$j)
```
3. **Static type analysis**: This is an optional step. Static type analysis checks whether each query is well-typed, and if so, determines its static type. Static type analysis is defined only for query Core language. Static type analysis works by recursive application of the static typing rules over a given expression.

4. **Dynamic Processing**: This step is divided into two sub steps.
   - **Dynamic Context Processing**: The dynamic semantics of query depends on the dynamic input. Dynamic input defines processing environment.
   - **Dynamic Evaluation**: In this phase, dynamic value of a query is computed. All dynamic values are provided in this phase. It may be intermediate result of the operation, or reading the value of variable at runtime, dynamic variable. This Dynamic evaluation may result in a value OR a dynamic error, which may be a non-type error or a type error.

In XML Query Processes, important task is XML Data Model generation i.e. Organizing XML data file in such a way that desired data can be accessed easily.

XML is a hierarchical database and semi-structure database. It doesn’t have fixed tuple structures for storing a record in the database. The data in XML is tagged with an infinite set of user-defined tags that come in pairs and can be nested to arbitrary levels [26]. Therefore, storing data into proper table structure is a challenge. The XML data should be organized in such a way that it should maintain ‘Sibling’, ‘P-C’ and ‘A-D’ relationship. Once the data is stored into a file, then indexing must be created for fast accessing data which will provide a good query performance result. In literature, different approaches are proposed for XML indexing and query processing they are:

2.6 **XML Query Processing Approaches**:

Indexing plays an important role in any database to retrieve specific data without scanning a complete file. Well known Indexing structure (hash tables and B-structure as well as multidimensional index structures) has been developed in RDBMS as well as OOAD databases [20]. In XML, index creation is a difficult process because data is hierarchical or semi-structured. In literature, different types of indices are created on it. For example, structure index (index on XML document structure), Value index – index on node value and Path index (index
on tree path). In the following section, we describe XML indexes and query processing approaches.

XML is a text file. For effective query processing, researchers have developed different types of storage of the XML file. Broadly these storages are classified into [9] 1) Relational Storage. 2) Native Storage and 3) Navigation Storage.

**A. RDBMS Approach:-**

XML is a hierarchical, recursive and ordered database. Relational database provides week support for hierarchical data. Therefore, it is difficult to convert XML data into relational format. In relational conversion, XML to RDBMS, schema defines a set of table for fixed or frequent structure; sometime additional ordering column is added for supporting order data. It avoids storage of label and factorizing storage of values for good query performance.

Once data is shredded into tables, full power of relational engine such as indexing and query capabilities can be used to manage, and query data [8]. There are two approaches to design database schemas for XML documents: Structural mapping and Model Mapping.

(a) In Structural mapping approach, relational schema is defined for each XML structure, a relation or a class. It is created for each element type in the XML documents.

(b) In Model mapping, a fixed database schema is defined for XML document which supports dynamic operations such as insertion, updates and deletion effectively. This approach is suitable for dynamic XML document. In this approach, fixed databases schema is used to store the structure of all XML documents.

**B. Native Approach:-**

Native Approach of XML database model integrates specific techniques for storing the data, and accessing XML documents. These techniques include new storage methods, indexing structure, and join algorithm for query processing. The Native Database Architecture is divided into two types: Text-Based Native XML Database and Model-Based Native XML Database.

(a) **Text-Based Native XML Database:** - XML file is stored as a text file and indexed on element content or element path. The file is stored using ‘BLOB’ or ‘CLOB’ data type. Text-Based Native XML Database architecture speeds up the data retrieval process, because the data is stored into continuous bytes on the disk. Therefore, single index lookup retrieves the desired result.
(b) **Model-based Native XML database**: - XML document builds an internal object model. This object can be stored into Relational database or Object Oriented database. The objects are interpreted according to database style. For example, XML file is read from DOM object. If it is to be stored into rational database, it has to be stored into different tables such as Elements table, Attributes table, PCDATA tables, Entities tables and Entity-References tables. If it is stored into OODBMS, each XML node is interpreted as an object, and it is stored into ORDBMS format. This format is being used by Oracle and DB2 since 2007, after introducing new data type `.XML’, which stores XML files in hierarchical format.

The advantage of this is a new XML storage technique is listed below:-

1. Native XML database has ability to confirm an XML document against different XML Schema; but in RDBMS, a document is always confirmed against a schema. One as to one relationship is maintained in RDBMS.
2. Native XML database supports XQuery language.
3. XQuery expression contains different ‘Conditional Expressions’ and user-defined functions, which require the confirmation of the documents against the schema. This schema may have multiple versions. This is possible only in Native XML Storage.
4. Data retrieval is faster than RDBMS because Native Database stores the entire XML document physically together, and uses physical pointer between the parts of this document. This allows document to be retrieved either without join or with physical join. This join is always faster than the logical join which is used by the RDBMS system.

**C. Navigation Approach:-**

In this approach, XML data arrives in the form of data streams, without any associated indexes, and the entire XML data tree has to be traversed node by node in the depth-first order, which preserves the original XML document order. The key idea of this approach is the use of run-time stack data structure for query evaluation. XML file is read character by character and its tokens are stored into a stack. When an opening tag is encountered, the stack-top element is transformed to new state. These newly generated states are collected into a new stack. After calculation on these states they are then pushed into the stack. When a closing tag is encountered, the stack-top element is simply popped out of the stack.
2.7 XML Indexes Approaches:

Indexes prebuilt on XML data can facilitate XML query processing by locating data quickly. Index is built on traversing paths with the element set or on the value of elements and attributes. Broadly, XML indices are classified into three types: Structural Summary Index, Structural Join Index, and Sequence Index. Detail discussion is as below:

a. **Structural Summary Index:** - In ‘Structural Summary index’, indices are built on XML document paths. Therefore simple path query i.e. identification of P-C relationship can be solved by this index. The Structural Summary Index includes associated XML document path, element set and the paths emerging from these elements. The Structural Summary Index is implemented as tree structure data type in which each node represents a tag name and its associated path starting from root node till the concerned node. These indexes are further classified as:

a. **Selective Index:** - From the literature study these indexes are created on the attribute or element value. They are further classified into: Simple index, Full-text.

b. **Simple index:** - This index returns the set of elements and attributes that satisfy a certain condition. The condition might be the content associated with an element or attribute (value-based index), or their tag names (name index). Developers typically build this index using relational index technologies, such as B-tree, Order index.

b. **Full-text indexes:** - This index returns the set of elements that satisfy a certain condition over their textual content. Full-text index typically rely on inverted index, which associates each word with its position in its host document. Developers can implement a full-text index as a B-tree. The location scheme is used for building it, and can answer full-text queries easily.

c. **Structural Join Index:** - Structural Join Index, index attributes and elements with a particular name. In general, developers use these indexes in the context of join-based processing in which the query processor first determines elements that match query tree nodes. Optimization of XML queries is a key issue in Structural join and researchers have proposed various techniques for solving it. Two major techniques are: i) Merge Join Algorithm. ii) Stack-based Algorithm. According to literature review, this type of index is very important for queries evaluation because they process both DB and IR Style of queries. This index is classified in three main groups: Node Index, Path Encoding and Sequence based index.
1. **Node Index**: The researchers have identified two effective approaches for storing and accessing XML document: XML-to-RDBMS mapping approach, and Native database approach. The first category uses the mature relational database management system to store and manipulate the XML data and XML queries into SQL. Once data is converted into relational tables, then it takes advantage of existing indexing, recovery mechanism and concurrency mechanism. The main concern of such mapping approaches is to find the best relational representation for the XML which preserves the whole XML semantics, while minimizing the cost of the mapping process in terms of processing-time and storage-space. The main crisis with this mapping is document order semantics and unique identification of a record but it is solved by Node Indexing Schema. In this schema, a unique code is assigned to each node in the XML tree and preserves the nested hierarchical relationship and structural relationship. It is structure-orientation approach which reflects the document hierarchy in the index structure. These types of indices are good for answering content based queries. These indices are created on XML data storage. According to Weigel et al. [8], a labeling schema for a document tree D is a decentralized structure summary of a specific set of tree relationships in D.

The main goal of the existing node-labeling solution is: a) assigning a unique code for each node in the XML tree. b) Preserving the nested hierarchical relationship (structural) relationship during XML updates, c) Minimizing the re-labeling cost (including the processing time and I/O accessibility) in the case of data updates, and d) reducing the required storage to store generated code [18]. The label of the node suggests the position of the node in tree, and relationship with other nodes.

The main problem of this approach is updating cost is high because when new node is inserted or deleted from the tree, re-labeling is required i.e. each node is assigned a new label again. Another problem is ‘Label length’. As XML tree grows depth wise, the node-label length is increased and after 5th level it is impossible to read. [15]. Most of the research tried to reduce re-labeling. But these new techniques fail to support specific requirement of processing hierarchical, semi-structured data. Node Label indexes can process ‘Total Match Query’ but they are insufficient for processing ‘Partial Matching’ and ‘Content-based’ query.
2. Path Encoding:-

The Path encoding schema stores the entire path summary information. The XML distinct paths from root node to any arbitrary node in the document are stored. The queries are executed by assessing this path summary information instead of executing it on the structure of XML document. This indexing technique is ideal for evaluating simple path query instead of evaluating multiple path queries. Because it involved multiple join operation for the execution.

3. Sequence-based Indexes:-

Sequence based indexes convert both the XML query and XML documents into sequences and use the well-established sequences matching techniques to obtain query answers (e.g. [12] [13]). The good feature of sequence-based indexing techniques is that they use the entire query tree as one unit, thus the expensive join operations are avoided during the query evaluation.

In this approach, XML documents and branching queries are represented as sequence, and subsequent matching provides the query answers. These approaches don’t support queries with selection condition over nodes. The Sequence approach has the following limitations: 1) it supports only the ordered twig queries. 2) It may involve a large number of index investigation and index-probe operations might result in high random disk I/O costs. 3) It might repeatedly visit many data nodes unnecessarily. The pictorial representation of all above approaches is listed in fig 2.4
The detail discussion of XML indexing and XML storage are discussed into next chapter.

**Conclusion and Summary:**
This chapter discusses the progression of DBMS. Then it discusses the preliminary information of XML and its storage and indexing technique. We also discuss the query processing technique and XML indexing approach. The detail literature review and limitations and challenges are discussed in the next chapter.
References:

1. Barbara Catania and Anna Maddalena, Athena Vakali "XML Document Indexes: A Classification " SEPTEMBER • OCTOBER 2005 Published by the IEEE Computer Society 1089-7801/05/$20.00 © 2005 IEEE IEEE INTERNET COMPUTING


9. Ioana Manolescu-Goujot INRIA Futurs - LRI - PCRI, France Gemo group XML query processing: storage and query model interplay


11. Jonathan RobieXML Processing and Data Integration with Xquery Published by the IEEE Computer Society 089-7801/07 2007 IEEE


