

**Batch: B-4 Roll No.: 16010422234 Name: Chandana Ramesh Galgali**

**Experiment No.: 04**

**Aim:** Demonstrate the working of XML as Database

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resource needed:** Notepad++, Web Browser

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Theory:**

XML can be used to store and manage data in a way resembling a database. There are two main approaches to this:

1. Storing XML in Relational Databases (XML-enabled databases):

* Structure: Imagine tables in a relational database, but each cell can hold an entire XML document instead of just simple text.
* Queries: Use extensions to SQL like XQuery or SQL/XML to navigate and extract data from the nested XML structures within the cells.

Example: A library database might store book information (title, author, etc.) in one table, with each book's chapters and paragraphs stored as XML in a separate column. XQuery could then be used to find chapters containing specific keywords.

2. Native XML Databases:

* Structure: Data is stored directly in its native XML format, not shoehorned into tables and rows.
* Storage: Optimized data structures handle the hierarchical nature of XML documents more efficiently than relational databases.
* Queries: Specialized languages like XQuery and XPath are used to navigate and extract data based on the XML structure itself.

Example: A scientific data archive might store complex research results as XML documents with equations, figures, and annotations. XQuery could then be used to retrieve all experiments involving a specific compound.

Data: A list of employees stored in XML format: XML

<?xml version="1.0"?>

<employees>

<employee id="1">

<name>John Doe</name>

<department>Marketing</department>

<salary>50000</salary>

</employee>

<employee id="2">

<name>Jane Smith</name>

<department>IT</department>

<salary>60000</salary>

</employee>

</employees>

3. XML-enabled database:

Store the XML document in a single cell of a table named "employee\_data".

Use XQuery to query the data, like finding all employees with salaries over 55000: SQL

SELECT employee\_data.extract(value(//employee/salary[text() > 55000]), '/employees') FROM employee\_data;

4. Native XML database:

Store the XML document directly in the database.

Use XQuery directly to navigate the structure and find employees from the IT department: XML

for $emp in /employees/employee where $emp/department = "IT" return $emp

Benefits of using XML as a database:

* Flexibility: XML can handle diverse data structures and semi-structured data well.
* Integration: Easily exchange data with other systems that use XML.
* Self-describing: XML tags provide context and meaning to the data.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Activity:**

**1. Create a small XML database**

**2. Query your XML database**

**3. Transform your XML data** 

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Results: (Program printout with output)**

**Code:**

**<!-- xml file -->**

**<?xml version="1.0" encoding="UTF-8"?>**

**<?xml-stylesheet type="text/xsl" href="perfumestyle.xsl"?>**

**<perfumeShop>**

**<perfume id="1">**

**<name>Chanel No. 5</name>**

**<brand>Chanel</brand>**

**<price>100</price>**

**<quantity>20</quantity>**

**</perfume>**

**<perfume id="2">**

**<name>Flowerbomb</name>**

**<brand>Viktor and Rolf</brand>**

**<price>120</price>**

**<quantity>15</quantity>**

**</perfume>**

**<perfume id="3">**

**<name>Black Opium</name>**

**<brand>Yves Saint Laurent</brand>**

**<price>90</price>**

**<quantity>25</quantity>**

**</perfume>**

**</perfumeShop>**

**<!-- xsl file -->**

**<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">**

**<xsl:template match="/">**

**<html>**

**<head>**

**<title>Scent Haven Perfume Inventory</title>**

**</head>**

**<body>**

**<h1>Perfume Inventory</h1>**

**<table border="1">**

**<tr>**

**<th>Name</th>**

**<th>Brand</th>**

**<th>Price ($)</th>**

**<th>Quantity</th>**

**</tr>**

**<xsl:for-each select="/perfumeShop/perfume">**

**<tr>**

**<td><xsl:value-of select="name"/></td>**

**<td><xsl:value-of select="brand"/></td>**

**<td><xsl:value-of select="price"/></td>**

**<td><xsl:value-of select="quantity"/></td>**

**</tr>**

**</xsl:for-each>**

**</table>**

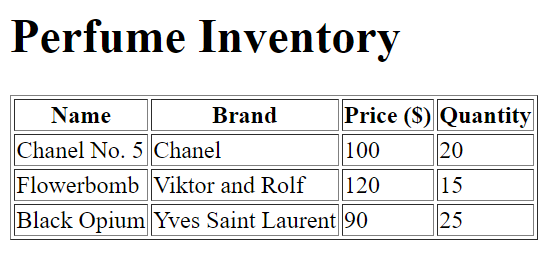
**</body>**

**</html>**

**</xsl:template>**

**</xsl:stylesheet>**

**Output:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Questions:**

**1. What is the difference between an element and an attribute in XML?**

**Ans:** Element: An element is a fundamental building block of an XML document, representing a structure or entity within the document. It consists of a start tag, optional content, and an end tag. Elements can contain other elements, text, or both.

Attribute: An attribute provides additional information about an element. Attributes are name-value pairs that are attached to the opening tag of an element. They provide metadata or characteristics for the element they belong to. Attributes do not contain content like elements; instead, they describe properties of the element.

**2. What is the role of a DTD in XML?**

**Ans:** DTD (Document Type Definition): A DTD is a formal specification that defines the structure, content, and valid elements and attributes within an XML document. It serves as a contract or blueprint for the XML document, outlining the rules and constraints that the document must adhere to. The key roles of a DTD include:

* Defining Document Structure: A DTD specifies the elements and their hierarchy allowed in the XML document.
* Validating Document Content: It provides rules for the content of elements, such as which elements are required, which are optional, and what data types are allowed.
* Enforcing Consistency: By defining a standard structure, a DTD ensures consistency across XML documents of the same type.
* Facilitating Interoperability: DTDs enable different systems to exchange XML documents while ensuring compatibility and adherence to a common structure.

**3. How can you comment on a section of code within an XML document?**

**Ans:** In XML, you can use XML comments to add notes, explanations, or annotations within the document. XML comments start with <!-- and end with -->. Anything between these delimiters is considered a comment and is ignored by XML parsers when processing the document. Here's an example:

<!-- This is a comment in XML -->

<root>

<!-- This is another comment -->

<element>Content</element>

</root>

Comments can span multiple lines and can be placed anywhere within the XML document where text content is allowed, including within elements, between elements, or at the document level.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Outcomes: Create Web pages using HTML 5 and CSS.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conclusion: (Conclusion to be based on the outcomes achieved)**

The experiment successfully demonstrated XML's efficacy as a database solution, offering flexibility, efficiency, and interoperability in managing structured and semi-structured data. While XML may not replace traditional relational databases in all scenarios, its unique features make it a valuable option for certain use cases, particularly those requiring flexibility and compatibility across diverse systems and applications.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**References:**

**Books/ Journals/ Websites:**

* “Web technologies: Black Book”, Dreamtech Publications
* [http://www.w3schools.com](http://www.w3schools.com/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**