**3.Managing XML Documents in PostgreSQL**

**3) a) Creating XML document, document type , definition and XML schema.**

**Aim:**

To create an XML document, define its structure using DTD, and validate it using XML Schema.

**Procedure:**

1. Create an XML Document: Write an XML document using a text editor and save it with the .xml extension.
2. Define DTD: Define the document structure in a .dtd file.
3. Validate with XML Schema (XSD): Create an XML Schema file (.xsd) to validate the XML document.

**XML Code:**

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

<!DOCTYPE employees SYSTEM "employees.dtd">

<employees>

<employee>

<id>101</id>

<name>John Doe</name>

<role>Manager</role>

</employee>

<employee>

<id>102</id>

<name>Jane Smith</name>

<role>Developer</role>

</employee>

</employees>

**DTD Code:**

<!ELEMENT employees (employee+)>

<!ELEMENT employee (id, name, role)>

<!ELEMENT id (#PCDATA)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT role (#PCDATA)>

**XML Schema Code:**

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="employees">

<xs:complexType>

<xs:sequence>

<xs:element name="employee" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="id" type="xs:int"/>

<xs:element name="name" type="xs:string"/>

<xs:element name="role" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

**Output:**

An XML document validated successfully against DTD and XSD.

**Result:**

The XML document is structured and validated using DTD & XML.

**3) b) Storing the XML Document as Text in PostgreSQL**

**Aim:**

To store an XML document as text in PostgreSQL.

**Procedure:**

1. Create a table to store the XML document as text.
2. Insert the XML document into the table.

**SQL Code:**

CREATE TABLE xml\_documents (

id SERIAL PRIMARY KEY,

doc\_name TEXT,

xml\_content TEXT

);

**Output:**

TABLE CREATED.

**SQL Code:**

INSERT INTO xml\_documents (doc\_name, xml\_content)

VALUES ('Employee Data',

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

<employees>

<employee>

<id>101</id>

<name>John Doe</name>

<role>Manager</role>

</employee>

<employee>

<id>102</id>

<name>Jane Smith</name>

<role>Developer</role>

</employee>

</employees>

);

**Output:**

INSERT 0 1.

**Result:**  
 The XML document is successfully stored as text in PostgreSQL.

**3) c) Storing XML Document as Data Elements**

**Aim:**

To store XML data as individual elements in PostgreSQL.

**Procedure:**

1. Create a table with columns for individual elements.
2. Parse the XML and insert its elements as rows in the table.

**SQL** **Code:**

CREATE TABLE employees (

id INT PRIMARY KEY,

name TEXT,

role TEXT

);

**Output:**

TABLE CREATED.

**SQL Code:**

INSERT INTO employees (id, name, role)

SELECT (xpath('/employees/employee/id/text()',xml\_content))[1]::TEXT::INT AS id,

(xpath('/employees/employee/name/text()', xml\_content))[1]::TEXT AS name,

(xpath('/employees/employee/role/text()', xml\_content))[1]::TEXT AS role

FROM xml\_documents

WHERE doc\_name = 'Employee Data';

**Output:**

INSERT 0 1

**Result:**

XML elements are successfully stored as relational data.

**3) d) Publishing a Customized XML Document from Relational Data**

**Aim:**

To create a customized XML document from relational data.

**Procedure:**

1. Use the xmlelement() function to generate an XML document.

**SQL Code:**

SELECT xmlelement(name employees,

xmlagg(

xmlelement(name employee,

xmlelement(name id, id),

xmlelement(name name, name),

xmlelement(name role, role)

)

)

)

FROM employees;

**Output:**

Xml\_data

---------------------------------------------------------------------------

<employees><employee><id>101</id><name>John Doe</name><role>Manager</role>

</employee><employee><id>102</id><name>Jane Smith</name><role>Developer

</role></employee></employees>

(1 row)

<employees><employee><id>101</id><name>JohnDoe</name><role>Manager</role>

</employee><employee><id>102</id><name>JaneSmith</name><role>Developer</role></employee></employees>

(1 row)

**Result:**

Relational data is successfully converted to a structured XML document.

**3) e) Extracting XML Documents from a Relational Database**

**Aim:**

To extract stored XML documents from PostgreSQL.

**Procedure:**

1. Retrieve the stored XML data using SQL queries.

**SQL Code:**

SELECT xml\_content FROM xml\_documents WHERE doc\_name = 'Employee Data';

**Output:**

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

<employees>

<employee>

<id>101</id>

<name>John Doe</name>

<role>Manager</role>

</employee>

<employee>

<id>102</id>

<name>Jane Smith</name>

<role>Developer</role>

</employee>

</employees>

(1 row)

**Result:**

Stored XML document extracted successfully.

**3) f) Querying XML Data**

**Aim:**

To perform XPath-based queries on XML data in PostgreSQL.

**Procedure:**

1. Use the xpath() function to query nodes in the XML document.

**SQL Code:**

SELECT xpath('/employees/employee/name/text()', xml\_content) AS employee\_names

FROM xml\_documents

WHERE doc\_name = 'Employee Data';

**Output:**

xpath

---------------------------

{John Doe, Jane Smith}

(1 row)

**Result:**

The XML document is successfully queried using XPath.