# Managing XML Documents in PostgreSQL

1) Creating XML document, document type , definition and XML schema.

2) Using a relational database to store the XML document as text.

3) Using a relational database to store the XML document as data elements.

4) Creating or publishing customized XML document from pre-existing relational database.

5) Extracting XML documents from relational database.

6) XML quering.

**Step 1: Create an XML Document**

Create a sample XML document that contains employee data.

**XML Document 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>
```

**Output:**

An XML document with employee data.

**Step 2: Create a Document Type Definition (DTD)**

Define the structure of the XML document using a DTD.

**DTD Code:**

```
<!ELEMENT employees (employee+)>
<!ELEMENT employee (id, name, role)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT role (#PCDATA)>
```

**Output:**

A DTD that specifies the allowed structure and elements of the XML document.

**Step 3: Create an XML Schema**

Define the structure of the XML document using an XML Schema.

**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 Schema that defines the structure, types, and constraints of the XML document.

**Step 4: Create a Table to Store XML Documents as Text**

Create a table in PostgreSQL to store the XML document.

**SQL Code**

```
CREATE TABLE xml_documents (
    id SERIAL PRIMARY KEY,
    doc_name TEXT,
    xml_content TEXT
);
```

**Output:**

A table named `xml_documents` with columns `id`, `doc_name`, and `xml_content`.

**Step 5: Insert XML Document into the Table**

Insert the XML document into the `xml_documents` table.

**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:**

The XML document is inserted into the `xml_documents` table.

**Step 6: Storing XML Document as Data Elements**

Create a table to store individual XML data elements in a relational format.

**SQL Code:**

```
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name TEXT,
    role TEXT
);
```

**Output:**

A table named `employees` is created with columns `id`, `name`, and `role`.

**Step 7: Extract Data from XML and Insert into the Employees Table**

Extract data from the XML stored in the `xml_documents` table and insert it into the `employees` table.

**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:**

XML data is inserted into the `employees` table.

**Step 8: Check Inserted Data**

Verify that the data has been inserted correctly into the `employees` table.

SQL Code:

```
SELECT * FROM employees;
```

**Output:**

| id | name | role |
|---|---|---|
| 101 | John Doe | Manager |
| 102 | Jane Smith | Developer |

**Step 9: Create Customized XML Document**

Generate a customized XML document from the data stored in the `employees` table.**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:**

```
<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>
```

**Step 10: Extract XML Document**

Retrieve the XML document stored in the `xml_documents` table.

**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>

**Step 11: Querying XML Data**

Query specific elements from 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:**

employee_names

---

{John Doe, Jane Smith}

---