

Database Systems Lab(MCSE506P)

EXERCISE 9: XML, XSD, XQUERY

Name : Mahesh Jagtap

Reg No:24MCS1017

Date: 04/10/2024

1. Create three XML documents for:

- a. DEPARTMENT (DEPT_ID, DEPT_NAME).
- b. PROJECT (PROJECT_ID, PROJECT_NAME, DID).
- c. EMPLOYEE (EMP_ID, NAME, GENDER, SALARY, DEPT_ID, PID, DOJ).

Each XML should have 5 nodes at least.

department.xml->

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

```
<DEPARTMENT_MCS1017>
```

```
  <DEPARTMENT>
```

```
    <DEPT_ID>D001</DEPT_ID>
```

```
    <DEPT_NAME>Human Resources</DEPT_NAME>
```

```
  </DEPARTMENT>
```

```
  <DEPARTMENT>
```

```
    <DEPT_ID>D002</DEPT_ID>
```

```
    <DEPT_NAME>Finance</DEPT_NAME>
```

```
  </DEPARTMENT>
```

```
  <DEPARTMENT>
```

```
    <DEPT_ID>D003</DEPT_ID>
```

```
    <DEPT_NAME>IT Support</DEPT_NAME>
```

```
  </DEPARTMENT>
```

```
  <DEPARTMENT>
```

```
    <DEPT_ID>D004</DEPT_ID>
```

```
    <DEPT_NAME>Marketing</DEPT_NAME>
```

```
  </DEPARTMENT>
```

```
  <DEPARTMENT>
```

```
    <DEPT_ID>D005</DEPT_ID>
```

```
    <DEPT_NAME>Research and Development</DEPT_NAME>
```

```
  </DEPARTMENT>
```

```
</DEPARTMENT_MCS1017>
```

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

```
employee.xml->
```

```
<EMPLOYEE_MCS1017>
```

```
  <EMPLOYEE>
```

```
    <EMP_ID>E2000</EMP_ID>
```

```
    <NAME>John Doe</NAME>
```

```
    <GENDER>Male</GENDER>
```

```
    <SALARY>75000</SALARY>
```

```
    <DEPT_ID>D001</DEPT_ID>
```

```
    <PID>1001</PID>
```

```
    <DOJ>2022-05-15</DOJ>
```

```
  </EMPLOYEE>
```

```
  <EMPLOYEE>
```

```
    <EMP_ID>E2001</EMP_ID>
```

```
    <NAME>Jane Smith</NAME>
```

```
    <GENDER>Female</GENDER>
```

```
    <SALARY>65000</SALARY>
```

```
    <DEPT_ID>D002</DEPT_ID>
```

```
    <PID>1002</PID>
```

```
    <DOJ>2023-04-10</DOJ>
```

```
  </EMPLOYEE>
```

```
  <EMPLOYEE>
```

```
    <EMP_ID>E2002</EMP_ID>
```

```
    <NAME>Mike Brown</NAME>
```

```
    <GENDER>Male</GENDER>
```

```
    <SALARY>55000</SALARY>
```

```
    <DEPT_ID>D003</DEPT_ID>
```

```
    <PID>1003</PID>
```

```
    <DOJ>2023-07-20</DOJ>
```

```
  </EMPLOYEE>
```

```
  <EMPLOYEE>
```

```
    <EMP_ID>E2003</EMP_ID>
```

```
    <NAME>Emily Wilson</NAME>
```

```
    <GENDER>Female</GENDER>
```

```
    <SALARY>72000</SALARY>
```

```
    <DEPT_ID>D004</DEPT_ID>
```

```
        <PID>1004</PID>
        <DOJ>2022-09-05</DOJ>
    </EMPLOYEE>
    <EMPLOYEE>
        <EMP_ID>E2004</EMP_ID>
        <NAME>Chris Johnson</NAME>
        <GENDER>Male</GENDER>
        <SALARY>80000</SALARY>
        <DEPT_ID>D005</DEPT_ID>
        <PID>1005</PID>
        <DOJ>2021-12-30</DOJ>
    </EMPLOYEE>
</EMPLOYEE_MCS1017>
```

project.xml

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

<PROJECT_MCS1017>
    <PROJECT>
        <PROJECT_ID>P101</PROJECT_ID>
        <PROJECT_NAME>GITHUB</PROJECT_NAME>
        <DID>1</DID>
    </PROJECT>
    <PROJECT>
        <PROJECT_ID>P102</PROJECT_ID>
        <PROJECT_NAME>CODEMASTER</PROJECT_NAME>
        <DID>2</DID>
    </PROJECT>
    <PROJECT>
        <PROJECT_ID>P103</PROJECT_ID>
        <PROJECT_NAME>VSCODE</PROJECT_NAME>
        <DID>4</DID>

    </PROJECT>
    <PROJECT>
        <PROJECT_ID>P104</PROJECT_ID>
        <PROJECT_NAME>AWS</PROJECT_NAME>
        <DID>3</DID>
```

```

    </PROJECT>
  <PROJECT>
    <PROJECT_ID>P105</PROJECT_ID>
    <PROJECT_NAME>COPILOT</PROJECT_NAME>
    <DID>2</DID>
  </PROJECT>
</PROJECT_MCS1017>

```

2. Create suitable XSD files for the three XML documents and validate the same.

schema.xsd

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="DEPARTMENT_MCS1017">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DEPARTMENT" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DEPT_ID" type="xs:string"/>
              <xs:element name="DEPT_NAME" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="PROJECT_MCS1017">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PROJECT" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="PROJECT_ID" type="xs:string"/>
              <xs:element name="PROJECT_NAME" type="xs:string"/>
              <xs:element name="DID" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="EMPLOYEE_MCS1017">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="EMPLOYEE" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="EMP_ID" type="xs:string"/>
                        <xs:element name="NAME" type="xs:string"/>
                        <xs:element name="GENDER" type="xs:string"/>
                        <xs:element name="SALARY" type="xs:decimal"/>
                        <xs:element name="DEPT_ID" type="xs:string"/>
                        <xs:element name="PID" type="xs:string"/>
                        <xs:element name="DOJ" type="xs:date"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>

```

validate.java

```

import java.io.File;
import javax.xml.XMLConstants;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;

public class XSDValidator {
    public static void main(String[] args) {
        String xsdPath = "C:/Users/Jagta/OneDrive/Desktop/VIT
SEM1/DBMS/LAB ASSIGNMENTS/schema.xsd";

        String[] xmlPaths = {
            "C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB
ASSIGNMENTS/department.xml",

```

```

        "C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB
        ASSIGNMENTS/employee.xml",
        "C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB
        ASSIGNMENTS/project.xml"
    };
    for (String xmlPath : xmlPaths) {
        validateXMLSchema(xsdPath, xmlPath);
    }
}

public static void validateXMLSchema(String xsdPath, String
xmlPath) {
    try {
        SchemaFactory factory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = factory.newSchema(new File(xsdPath));
        Validator validator = schema.newValidator();
        validator.validate(new StreamSource(new
File(xmlPath)));
        System.out.println(xmlPath + " is valid against " +
xsdPath);
    } catch (Exception e) {
        System.out.println(xmlPath + " is not valid. Error: "
+ e.getMessage());
    }
}
}

```

OUTPUT:

```

C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBMS\LAB ASSIGNMENTS>javac XSDValidator.java
C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBMS\LAB ASSIGNMENTS>java XSDValidator
C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB ASSIGNMENTS/department.xml is valid against C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/
DBMS/LAB ASSIGNMENTS/schema.xsd
C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB ASSIGNMENTS/employee.xml is valid against C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DB
MS/LAB ASSIGNMENTS/schema.xsd
C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB ASSIGNMENTS/project.xml is valid against C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBM
S/LAB ASSIGNMENTS/schema.xsd

```

3. Perform the following operations:

a. Retrieve the department name for department id D001.

```
for $dept in //DEPARTMENT[DEPT_ID='D001']
return $dept/DEPT_NAME
```

   1 Result, 38 b	Result	 Result Total Time: 74.7 ms
<code><DEPT_NAME>Human Resources</DEPT_NAME></code>		Result: <ul style="list-style-type: none">- Hit(s): 1 Item- Updated: 0 Items- Printed: 38 b- Read Locking: department- Write Locking: (none)

b. Retrieve the project name and department id of project id P102.

```
for $proj in //PROJECT[PROJECT_ID='P102']
return (concat($proj/PROJECT_NAME, ' - ', $proj/DEPT_ID))
```

   1 Result, 13 b	Result	 Result Total Time: 7.68 ms
CODEMASTER -		Result: <ul style="list-style-type: none">- Hit(s): 1 Item- Updated: 0 Items- Printed: 13 b- Read Locking: (none)- Write Locking: (none)

c. Retrieve the count of male employees.

```
count(//EMPLOYEE[GENDER='Male'])
```

   1 Result, 1 b	Result	 Result Total Time: 9.61 ms
3		Result: <ul style="list-style-type: none">- Hit(s): 1 Item- Updated: 0 Items- Printed: 1 b- Read Locking: employee_mcs1017- Write Locking: (none)

d. Retrieve the names and dept ids of employees who joined after 01-Jan-2020.

```
for $emp in //EMPLOYEE[xs:date(DOJ) > xs:date('2020-01-01')]
return concat($emp/NAME, ' - ', $emp/DEPT_ID)
```

Result	
5 Results, 96 b	Result
John Doe - D001	
Jane Smith - D002	
Mike Brown - D003	
Emily Wilson - D004	
Chris Johnson - D005	

e. Find the average salary of employees belonging to PID P1002.

```
let $salaries := //EMPLOYEE[PID='P1002']/SALARY
return avg($salaries)
```

Result	
1 Result, 5 b	Result
72500	

f. How many male employees work for dept id D001?

```
count(//EMPLOYEE[GENDER='Male' and DEPT_ID='D001'])
```

Result	
1 Result, 1 b	Result
1	

g. List the names of employees whose salary ranges between 60000 to 70000.

```
for $emp in //EMPLOYEE[SALARY >= 60000 and SALARY <= 70000]
return $emp/NAME
```

Result	
1 Result, 23 b	Result
<NAME>Jane Smith</NAME>	

