Grado en Ingeniería Informática

Grado en Matemática Computacional

Universitat Jaume I

Tecnologías para la Integración de Información

Ejercicios del tema 3: Tecnologías XML y Datos Semiestructurados

---

Para prepararte de cara al examen de teoría debes realizar estos ejercicios. Si quieres puedes entregar las soluciones para que sean corregidas. Si tienes dudas puedes utilizar las tutorías presenciales o virtuales.

Fecha Máxima de Entrega Recomendada: 8/4/2019

Ejercicio 1

Este documento XML no está bien formado, identifica todas las causas y explica por qué la expresión es incorrecta.

```
<?xml version="1.0" encoding="UTF-8"?>
<account number=123456789>
      <balance><dollar>100,000.00</dollar>
      <!-- Gold Customer -->
      <depositor>
            <firstname>John</firstname>
            <lastname>Doe</lastname>
      </depositor>
</account>
<account number="234567890" number="2">
      <balance><dollar>-200.00</balance></dollar>
      Account is overdrawn!
      <depositor>
            <firstname>Jane</firstName>
            <lastname>Doe</lastName>
      </depositor>
</account>
```

Ejercicio 2

Diseña un DTD que se corresponda con el siguiente esquema XSchema e identifica los datos que se pierden por no poder ser expresados en un DTD.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="lectureCatalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="lecture" type="lectureType" minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="lectureType">
    <xs:sequence>
      <xs:choice>
        <xs:element name="professor">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="assistantProfessor">
          <xs:complexType>
            <xs:attribute name="name" type="xs:string"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>

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

      <xs:element name="event" type="eventType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="eventType">
    <xs:sequence>
      <xs:element name="dayOfWeek" type="weekdayType"/>
      <xs:element name="startingTime" type="xs:time"/>
      <xs:element name="endTime" type="xs:time"/>
      <xs:element name="lectureRoom" type="roomType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="weekdayType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Monday"/>
      <xs:enumeration value="Tuesday"/>
      <xs:enumeration value="Wednesday"/>
      <xs:enumeration value="Thursday"/>
      <xs:enumeration value="Friday"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="roomType">
    <xs:sequence>
      <xs:element name="building" type="xs:string"/>
      <xs:element name="roomNumber" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Ejercicio 3

En la figura 3.7 del tema 3 vemos un ejemplo de cómo utilizar XSchema para definir un tipo XML que sirva para representar el esquema y los datos de una base de datos relacional. Supongamos que en la base de datos tenemos una tabla de empleados que se relaciona con una tabla de departamentos y otra de proyectos. En la figura vemos cómo definir el esquema de la tabla de empleados en XSchema con su clave primaria, su clave candidata y su clave ajena (observa cómo se ha especificado que cada empleado almacena como clave ajena la clave primaria del departamento al que pertenece). En la figura solamente se ha incluido la definición de la tabla de empleados, faltaría completar la definición del resto de las tablas del esquema de la base de datos (departamentos y proyectos). Terminar la definición del documento XSchema para esta base de datos con un elemento para la tabla departamentos y un elemento para la tabla de proyectos. Pon las columnas típicas de los departamentos y los proyectos con sus claves primarias más las claves ajenas necesarias **para especificar que desde un departamento se referencia al empleado que lo dirige y desde cada proyecto se referencia al empleado que lo supervisa y al departamento que lo ejecuta**.

Ejercicio 4

Observar el siguiente fragmento XML:

```
<BAR>
  <ADDR>101 Maple St.</ADDR>
  <PHONE>555-1212</PHONE>
  <PHONE>555-4567</PHONE>
</BAR>
```

Indica cuál de las siguientes declaraciones de elemento podría ser la del elemento <BAR> en un DTD.

a) <!ELEMENT BAR (ADDR+, PHONE*)>
b) <!ELEMENT BAR (PHONE, ADDR, PHONE)>
c) <!ELEMENT BAR (ADDR, NAME+, PHONE*)>
d) <!ELEMENT BAR (NAME, ADDR, PHONE*)>

Ejercicio 5

Observar el siguiente DTD:

```
<?xml version="1.0"?>
<!DOCTYPE A [
  <!ELEMENT A (B+, C)>
  <!ELEMENT B (#PCDATA)>
  <!ELEMENT C (B?, D)>
  <!ELEMENT D (#PCDATA)>
]>
```

Indica cuál de los siguientes documentos es válido para el DTD anterior:

a) <A> <B>hola</B><B>hola</B> <C> <B>hola</B> <B>hola</B> <D>adios</D> </C> </A>
b) <A> <B>hola</B> <B>hola</B> <C> <B>hola</B> <D>adios</D> </A>
c) <A> <B>hola</B> <B>hola</B> <C> <B>hola</B> <D>adios</D> </C> </A>
d) <A> <B>hola</B> <B>hola</B> <C> <D>adios</D> <B>adios</B> </C> </A>

Ejercicio 6

Observar el siguiente fragmento XML:

```
<EMP name = "Kermit">
  <ADDR>123 Sesame St.</ADDR>
  <PHONE type = "cell">555-1212</PHONE>
</EMP>
```

Indica cuál de las siguientes declaraciones de atributo **no podría** pertenecer a un DTD para el que el documento anterior sea válido.

a) <!ATTLIST PHONE type CDATA #REQUIRED>
b) <!ATTLIST PHONE owner CDATA #IMPLIED>
c) <!ATTLIST EMP name ID #IMPLIED>
d) <!ATTLIST EMP ssNo CDATA #REQUIRED>

Ejercicio 7

Escribe un documento XML bien formado y que cumpla las siguientes condiciones:

- El elemento raíz se llama *tasklist*.
- El elemento raíz tiene tres elementos hijo denominados *task*.
- Cada elemento *task* tiene un atributo denominado *name*.
- Los atributos *name* de las tres tareas tienen los valores: *eat*, *drink* y *play*.

Indica cuál de las siguientes opciones cumpe las condiciones anteriores:

a)
```
<?xml version='1.0' encoding='UTF-16'?>
  <tasklist>
    <task name='eat'>
    <task name='drink'>
    <task name='play'>
  </tasklist>
```

b)
```
<?xml version='1.0' encoding='UTF-8'?>
  <tasklist>
    <task name='eat'></task>
    <task name='drink'></task>
    <task name='play'></task>
  </tasklist>
```

c)
```
<?xml version="1.0" encoding="UTF-8"?>
  <tasklist>
    <task name="eat">
    <task name="drink">
    <task name="play">
  </tasklist>
```

d)
```
<?xml version="1.0" encoding="UTF-16"?>
  <tasklist>
    <task name="eat">
    <task name="drink">
    <task name="play">
  </tasklist>
```

Ejercicio 8

Observa el siguiente XSchema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="person">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="fname" type="xs:string"/>
    <xs:element name="initial" type="xs:string" minOccurs="0"/>
    <xs:element name="lname" type="xs:string"/>
    <xs:element name="address" type="xs:string" maxOccurs="2"/>
    <xs:choice>
     <xs:element name="major" type="xs:string"/>
     <xs:element name="minor" type="xs:string"/>
    </xs:choice>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Indica cuáles de los siguientes documentos XML son válidos con respecto al esquema anterior.

a)
```
<person>
  <fname>John</fname>
  <initial>Q</initial>
  <lname>Public</lname>
  <address>123 Public Avenue, Seattle, WA
           98001</address>
  <major>Computer Science</major>
</person>
```

b)
```
<person>
  <lname>Public</lname>
  <fname>John</fname>
  <initial>Q</initial>
  <title>Sr.</title>
  <address>123 Public Avenue, Seattle, WA
           98001</address>
  <major>Computer Science</major>
</person>
```

c)
```
<person>
  <fname>John</fname>
  <initial>Q.</initial>
  <lname>Public</lname>
  <address>123 Public Avenue</address>
  <address>Seattle</address>
  <address>WA 98001</address>
  <major>Computer Science</major>
</person>
```

d)
```
<person>
  <fname>John</fname>
  <initial>Q.</initial>
  <lname>Public</lname>
  <address>123 Public Avenue</address>
  <address>Seattle</address>
  <address>WA 98001</address>
  <minor>History</minor>
</person>
```

Ejercicio 9

Utilizando el documento de las traspas de clase que puedes encontrar en el aula virtual, resolver las siguientes expresiones XPath:

a) Seleccionar el atributo country que poseen los elementos CD.

b) Seleccionar todos los elementos hijo de los CD's que posean el atributo country igual a UK.

c) Seleccionar todos los artistas del catálogo.

d) Seleccionar todos los elementos descendientes de CD.

e) Seleccionar los item que se encuentran entre las posiciones 5 y 10.

f) Seleccionar el texto de todos los elementos title.

g) Seleccionar el texto de cada elemento descendiente de cada elemento CD.

h) Seleccionar el title de todos los CD que tengan una componente tracks.

i) Seleccionar el segundo CD del catálogo.

j) Seleccionar todos los elementos descendientes del catálogo que tengan el atributo country.

k) Visualizar el valor del atributo country del paso anterior.

l) Seleccionar el title de todos los item que no tengan un atributo price.

Ejercicio 10

Utilizando el siguiente documento resolver las siguientes expresiones XPath:

a) Título de las comedias

b) Actores que trabajaron con Nicolas Cage en alguna película.

c) Productores de alguna película en 1992.

d) Título de las películas que tuvieron al menos tres actores.

e) Título de las películas cuyo productor se apellida Wood.

```
<?xml version="1.0" standalone="no" encoding="utf-8"?>
<movies>
        <movie type="mistery" rating="PG-13" review="5" year="1992">
                <title>Raising Arizona</title>
                <writer>Ethan Coen</writer>
                <writer>Joel Coen</writer>
                <producer>
                        <name> Ethan </name>
                        <surname> Coen </surname>
                </producer>
                <director>Joel XXX</director>
                <actor>Holly Coen</actor>
                <actor>Nicolas Cage</actor>
                <actor>John Goodman</actor>
                <comments>A classic one-of-a-kind screwball love story.</comments>
        </movie>
```

```
	<movie type="comedy" rating="R" review="5" year="1992">
		<title>Midnight Run</title>
		<writer>George Gallo</writer>
		<producer>
			<name> Natalie </name>
			<surname> Wood </surname>
		</producer>
		<director>Joel Coen</director>
		<actor>Nicolas Cage</actor>
		<actor>Robert De Niro</actor>
		<comments>The quintessential road comedy.</comments>
	</movie>
	<movie type="comedy" rating="R" review="5" year="1992">
		<title>The Usual Suspects</title>
		<writer>Christopher McQuarrie</writer>
		<producer>Bryan Wood</producer>
		<producer>Michael McDonnell</producer>
		<director>Bryan Singer</director>
		<actor>Stephen Baldwin</actor>
		<actor>Gabriel Byrne</actor>
		<actor>Benicio Del Toro</actor>
		<actor>Chazz Palminteri</actor>
		<actor>Kevin Pollak</actor>
		<actor>Kevin Spacey</actor>
		<comments>A crime mystery with incredibly intricate plot twists.</comments>
	</movie>
	<movie type="sci-fi" rating="PG-13" review="4" year="1992">
		<title>The Abyss</title>
		<writer>James Cameron</writer>
		<producer>Gale Anne Hurd</producer>
		<director>James Cameron</director>
		<actor>Ed Harris</actor>
		<actor>Mary Elizabeth Mastrantonio</actor>
		<comments>A very engaging underwater odyssey.</comments>
	</movie>
</movies>
```

Ejercicio 11

Considerar los siguientes documentos XML: emp.xml y permisos.xml, junto con las siguientes 10 consultas XQuery que se ejecutan sobre ellos. Para cada consulta indicar cuál es la respuesta que devuelve su ejecución (porción de texto XML u otro tipo de datos). Debéis pensar la respuesta sin necesidad de utilizar ningún procesador de consultas XQuery. Para cada consulta también hay que poner un posible enunciado indicando qué datos recupera la consulta y qué condiciones cumplen estos datos. ¿Se te ocurren versiones alternativas para algunas de las consultas?

emp.xml

```
<?xml version="1.0"encoding="utf8" standalone="yes"?>

<empleados>
 <emp id="1">
  <nombre>Juan Bueno</nombre>
  <email>juan</email>
  <ssn>111-11-1111</ssn>
  <jefeEmail>david</jefeEmail>
  <fechaDeInicio>
  1994-11-16</fechaDeInicio>
 </emp>

 <emp id="2">
  <nombre>David Puertas</nombre>
  <email>david</email>
  <ssn>222-22-2222</ssn>
  <jefeEmail>marcos</jefeEmail>
  <fechaDeInicio>
   1992-12-24</fechaDeInicio>
 </emp>

 <emp id="3">
  <nombre>Ana Casas</nombre>
  <email>ana</email>
  <ssn>333-33-3333</ssn>
  <jefeEmail>marcos</jefeEmail>
  <fechaDeInicio>
  1993-08-19</fechaDeInicio>
 </emp>

 <emp id="4">
  <nombre>Celia Verte</nombre>
  <email>celia</email>
  <ssn>444-44-4444</ssn>
  <jefeEmail>ana</jefeEmail>
  <fechaDeInicio>
  1996-07-11</fechaDeInicio>
 </emp>

 <emp id="5">
  <nombre>Marcos Ruiz</nombre>
  <email>marcos</email>
  <ssn>555-55-5555</ssn>
  <jefeEmail></jefeEmail>
  <fechaDeInicio>
   1990-01-01</fechaDeInicio>
 </emp>

</empleados>
```

permisos.xml

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<permisos>

 <peticion id="T1" empID_="2"
  fechaIni="2003-01-02" fechaFin="2003-01-10"
  horas="40" estado="aprobado" codigoAusencia="vacaciones" />

 <peticion id="T2" empID_="4"
  fechaIni="2003-01-10" fechaFin="2003-02-10"
  horas="180" estado="aprobado" codigoAusencia="vacaciones" />

 <peticion id="T3" empID_="1"
  fechaIni="2003-01-15" fechaFin="2003-01-31"
  horas="80" estado="denegado" codigoAusencia="vacaciones" />

 <peticion id="T4" empID_="1"
  fechaIni="2003-02-01" fechaFin="2003-02-01"
  horas="8" estado="aprobado" codigoAusencia="enfermedad" />

 <peticion id="T5" empID_="3"
  fechaIni="2003-02-12" fechaFin="2003-02-12"
  horas="8" estado="aprobado" codigoAusencia="enfermedad" />

 <peticion id="T6" empID_="1"
  fechaIni="2003-03-01" fechaFin="2003-03-05" horas="30"
  estado="aprobado" codigoAusencia="vacaciones" />

 <peticion id="T7" empID_="4"
  fechaIni="2003-03-12" fechaFin="2003-03-15" horas="10"
  estado="aprobado" codigoAusencia="sinPago" />

</permisos>
```

1.	doc("emp.xml")/empleados/emp[jefeEmail='marcos']
2.	doc("emp.xml")/empleados/emp[jefeEmail='' or count(jefeEmail)<1]
3.	doc("emp.xml")/empleados/emp[upper-case(substring(nombre, 1, 1)) = 'J']
4.	data(doc("emp.xml")/empleados/emp[nombre='Ana Casas']/fechaDeInicio)
5.	sum(doc("permisos.xml")/permisos/peticion[@empID_=4 and @estado="aprobado"]/@horas)
6.	doc("permisos.xml")/permisos/peticion[@codigoAusencia="sinPago" and @estado="aprobado"]
7.	for $req in doc("permisos.xml")/permisos/peticion order by xs:double($req/@horas) descending return $req
8.	for $emp in doc("emp.xml")/empleados/emp, $req in doc("permisos.xml")/permisos/peticion[@empID_ = $emp/@id] where $req/@estado = 'denegado' return data($emp/nombre)

9.	<Informe>
	{
	for $emp in doc("emp.xml")/empleados/emp
	let $req := doc("permisos.xml")/permisos/peticion[@empID_ = $emp/@id and @estado="aprobado"]
	order by $emp/nombre ascending
	return
	<permisosDetalles
	empID="{$emp/@id}" empNombre="{$emp/nombre}"
	totalHoras="{sum($req/@horas)}"/>
	}
	</Informe>

10.	declare function local:count-peticiones() as item()*
	{
	for $emp in doc("emp.xml")/empleados/emp
	let $req := doc("permisos.xml")/permisos/peticion[@empID_ = $emp/@id]
	return count($req)
	};
	for $emp in doc("emp.xml")/empleados/emp
	let $req := doc("permisos.xml")/permisos/peticion[@empID_ = $emp/@id]
	where count($req) = max(local:count-peticiones())
	return data($emp/nombre)

Ejercicio 12

Considera el siguiente documento XML:

```
<peliculas>
    <pelicula fechaEstreno="19/12/2007">
        <titulo>Los Otros</titulo>
        <director>Alejandro Amenabar</director>
    </pelicula>
    <pelicula fechaEstreno="14/06/1990">
        <titulo>No es país para pobres</titulo>
        <director>Ethan Coen</director>
        <director>Joel Coen</director>
    </pelicula>
    <pelicula fechaEstreno="19/12/2007">
        <titulo>Harry Potter</titulo>
        <director>Peter Jackson</director>
    </pelicula>
</peliculas>
```

¿Qué secuencias XML devuelven las siguientes consultas?

a)
```
for $b in doc("peliculas.xml")//pelicula/director
return <directores>{$b}</directores>
```

b)
```
let $b := doc("pelicula.xml")//pelicula/director
return <directores>{$b}</directores>
```

c)
```
<directores>
  {for $b in doc("peliculas.xml")//pelicula/director
  return
       if ($b = "Alejandro Amenabar")
          then <espanyol>{$b/data()}</espanyol>
          else <otropais>{$b/data()}</otropais>}
</directores>
```

d)
```
let $b:= doc("pelicula.xml")//pelicula[@fechaEstreno="19/12/2007"]/director
return <directores>{$b}</directores>
```

Ejercicio 13

Haz una consulta XQuery que devuelva los siguientes resultados:

a)
```
<peliculas>
<pelicula>
   <titulo>Los Otros</titulo>
   <fechaEstreno>19/12/2007</fechaEstreno>
</pelicula>
<pelicula>
   <titulo>No es país para pobres</titulo>
   <fechaEstreno>14/06/1990</fechaEstreno>
</pelicula>
<pelicula>
   <titulo>Harry Potter</titulo>
   <fechaEstreno>19/12/2007</fechaEstreno>
</pelicula>
</peliculas>
```

b)
```
<pelicula titulo="Los Otros">
   <fechaEstreno>19/12/2007</fechaEstreno>
</pelicula>
<pelicula titulo="No es país para pobres">
   <fechaEstreno>14/06/1990</fechaEstreno>
   <num_directores>2</num_directores>
</pelicula>
<pelicula titulo="Harry Potter">
   <fechaEstreno>19/12/2007</fechaEstreno>
</pelicula>
```

Ejercicio 14

Observa las siguientes tres consultas escritas en SQL/XML que se ejecutan sobre una base de datos relacional con información de ordenes de pedidos de clientes (hay una tabla de ordenes y otra de líneas de pedido). Para cada consulta da un ejemplo de documento XML que podría obtenerse tras su ejecución.

1.

```
SELECT XMLELEMENT (
        NAME "order",
        XMLATTRIBUTES (o.oid AS "id"),
        XMLELEMENT (
                NAME "signdate",
                o.contractdate
                ),
        XMLELEMENT (
                NAME "amount",
                (SELECT SUM(ol.quantity)
                FROM orderLines ol
                WHERE ol.oid = o.oid)
                )
        )
FROM orders o
WHERE o.status = 'open';
```

2.

```
SELECT XMLELEMENT (
                NAME "order",
                XMLFOREST (
                        o.oid AS "id",
                        o.name AS "name",
                        o.city AS "city"
                        )
        )
FROM orders o
WHERE o.status = 'open';
```

3.

```
SELECT XMLELEMENT(
        NAME "order",
        XMLATTRIBUTES(o.oid AS "id"),
        XMLAGG(
                XMLELEMENT(
                        NAME "item",
                        XMLATTRIBUTES(
                                ol.listnbr AS "listnbr"
                                ),
                        XMLFOREST(
                                ol.name AS "name",
                                ol.quantity AS "quantity"
                )
        )
        ORDER BY ol.listnbr
        )
        )
FROM orders o, orderLines ol
WHERE o.oid = ol.oid
GROUP BY o.oid;
```

Ejercicio 15

Considera las siguientes dos tablas:

```
CREATE TABLE PRODUCT
        ( PID       VARCHAR(10) NOT NULL,
          NAME      VARCHAR(128),
          PRICE     NUMBER(30,2),
          PROMOPRICE  NUMBER (30,2),
          PROMOSTART  DATE,
          PROMOEND    DATE,
          PRIMARY KEY (PID) );
```

PID	NAME	PRICE	PROMO PRICE	PROMO START	PROMO END
100-100-01	Basic 22 inch	9.99	7.25	11/19/2004	12/19/2004
100-101-01	Deluxe 24 inch	19.99	15.99	12/18/2005	02/28/2006
100-103-01	Super Deluxe 26 inch	49.99	39.99	12/22/2005	02/22/2006
100-201-01	Ice Scraper	3.99	-	-	-

```
CREATE TABLE INVENTORY
        ( PID       VARCHAR(10) NOT NULL,
          QUANTITY    INTEGER,
          LOCATION    VARCHAR(128),
          PRIMARY KEY (PID) );
```

PID	QUANTITY	LOCATION
100-100-01	5	-
100-101-01	25	Store
100-103-01	55	Store
100-201-01	99	Warehouse

a) Indica qué valor XML devuelve la siguiente consulta:

```
SELECT XMLELEMENT (NAME "newElem",
                   XMLATTRIBUTES (PID AS "prodID"),
                   XMLFOREST (QUANTITY AS "quantity", LOCATION AS "loc"))
FROM INVENTORY;
```

b) Haz una consulta que devuelva el siguiente valor XML:

```
<allProducts>
        <item>Basic 22 inch</item>
        <item>Deluxe 24 inch</item>
        <item>Super Deluxe 26 inch</item>
        <item>Ice Scraper</item>
</allProducts>
```

c) Haz una consulta que devuelva el siguiente valor XML:

```
<saleProducts>
        <prod id="100-100-01">
                <name>Basic 22 inch</name>
                <numInStock>5</numInStock>
        </prod>
        <prod id="100-101-01">
                <name>Deluxe 24 inch</name>
                <numInStock>25</numInStock>
        </prod>
        <prod id="100-103-01">
                <name>Super Deluxe 26 inch</name>
                <numInStock>55</numInStock>
        </prod>
        <prod id="100-201-01">
                <name>Ice Scraper</name>
                <numInStock>99</numInStock>
        </prod>
</saleProducts>
```

Ejercicio 16

Considera la siguiente tabla:

```
CREATE TABLE EMP (DOC XMLType);
```

Esta tabla contiene las dos filas siguientes:

DOC

```
<dept bldg="101">
        <employee id="901">
                <name>
                        <first>John</first>
                        <last>Doe</last>
                </name>
                 <office>344</office>
                 <salary currency="USD">55000</salary>
        </employee>
        <employee id="902">
                    <name>
                        <first>Peter</first>
                        <last>Pan</last>
                 </name>
                 <office>216</office>
                 <phone>905-416-5004</phone>
        </employee>
</dept>
```

```
<dept bldg="114">
    <employee id="903">
                <name>
                        <first>Mary</first>
                        <last>Jones</last>
                </name>
                <office>415</office>
                <phone>905-403-6112</phone>
                <phone>647-504-4546</phone>
                <salary currency="USD">64000</salary>
    </employee>
</dept>
```

a) Indica cuál es el resultado de la siguiente consulta:

```
SELECT X.*
FROM EMP,
        XMLTABLE ('$d/dept/employee' PASSING EMP.DOC AS d
        COLUMNS
           "empID" INTEGER PATH '@id',
           "firstname" VARCHAR(20) PATH 'name/first',
           "lastname" VARCHAR(25) PATH 'name/last') AS X;
```

b) Escribe una consulta que devuelva el siguiente resultado:

EMPID	FIRSTNAME	LASTNAME	SALARY
901	John	Doe	55000
902	Peter	Pan	-
903	Mary	Jones	64000

c) Indica cuál es el resultado de la siguiente consulta:

```
SELECT X.*
FROM EMP,
        XMLTABLE ('$d/dept/employee/phone' PASSING DOC AS d
        COLUMNS
       "firstname" VARCHAR(20) PATH '../name/first',
       "lastname" VARCHAR(25) PATH '../name/last',
       "phone" VARCHAR(12) PATH '.') AS X
UNION
SELECT Y.*, CAST(NULL AS VARCHAR(12))
FROM EMP,
        XMLTABLE ('$d/dept/employee[not(phone)]' PASSING DOC AS d
        COLUMNS
        "firstname" VARCHAR(20) PATH 'name/first',
        "lastname" VARCHAR(25) PATH 'name/last') AS Y;
```