Problemas - XPath 1

Ejercicios de preparación

1. Clica en el dashboard de eXistDB el apartado “Collections”. Crea una nueva colección de documentos que llamarás “nueva”
2. Añade a la colección nueva todos los documentos contenidos en el archivo “ColecciónPruebas”
3. Clica en la app eXide. Desplázate por los directorios de la izquierda hasta localizar la colección “nueva”
4. Clica en algún documento para visualizarlo
5. Crea en la pestaña new XQuery una primera consulta:

```
doc('db/nueva/departamentos.xml')/departamentos
```

6. Comprueba el resultado de las siguientes consultas
   a. /departamentos → devuelve todos los datos de departamentos (Esta sería la misma consulta del ejercicio anterior pero sin indicar toda la ruta)
   b. /departamentos/DEP_ROW → devuelve todas las etiquetas de cada DEP_ROW
   c. /departamentos/DEP_ROW/DNOMBRE → devuelve nombres de departamentos entre etiquetas
   d. /departamentos/DEP_ROW/DNOMBRE/text() → Lo mismo que antes pero sin etiquetas
   e. //LOC/text() → localidades

NOTA: / se usa para dar rutas absolutas. Si el descriptor comienza con // se supone que la ruta descrita puede comenzar en cualquier parte

Ejercicios

7) Ahora averigua el resultado de las siguientes consultas (utilizaremos el documento 'db/nueva/empleados.xml')

a) **/EMPLEADOS/EMP_ROW[DEPT_NO=10]**
Obtenemos los datos completos de los empleados del departamento 10.

b) **/EMPLEADOS/EMP_ROW/APELLIDO|/EMPLEADOS/EMP_ROW/DEPT_NO**
Mediante este xPath conseguiremos únicamente el campo apellido y departamento de cada empleado de la base de datos.

c) **/EMPLEADOS/EMP_ROW [DEPT_NO=10]/APELLIDO/text()**
Obtendremos únicamente el texto que se encuentra dentro de las etiquetas apellidos de los empleados pertenecientes al departamento 10.

d) **/EMPLEADOS/EMP_ROW [not(OFICIO='ANALISTA')]**
Esta expresión retorna la información de aquellos empleados cuyo oficio no es Analista.

e) **/EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=20]/APELLIDO**
Retornará los datos de los empleados del departamento 20 que tengan un salario mayor de 1300.

f) **/EMPLEADOS/EMP_ROW[1]**
Obtendremos los datos completos del primer empleado de la lista.

8) Investiga en la web las siguientes funciones de XPath y pon algún ejemplo utilizando los documentos departamentos.xml y empleados.xml

a) **last()** → //DEP_ROW[last()]/DNOMBRE
Retorna el nombre del departamento en la última posición.

b) **position()** → //EMP_ROW[position()=6]/DIR
Obtenemos el numero del director del empleado que está en la posición 6.

c) **count()** → count(//DEP_ROW)
Obtenemos el número total de departamentos.

d) **sum(),div(),mod()** → sum(//SALARIO)
Obtenemos el total de todos los salarios de los empleados

e) **max(), min(),avg()** → min(//EMP_ROW[DEPT_NO=20]/SALARIO)
Obtenemos el salario más bajo del departamento 20

f) **concat(cadena1, cadena2,...)** → concat(//EMP_ROW[4]/APELLIDO, " empezó en la empresa el ", //EMP_ROW[4]/FECHA_ALT)
Obtenemos una frase que concatena el apellido del empleado y su fecha de inicio en la empresa.

g) **starts-with (cadena1, cadena2)** → //EMP_ROW[starts-with(APELLIDO, "J")]/SALARIO
Obtenemos el salarios de aquellos empleados cuyo apellido empieza por “J”

h) **contains(cad1,cad2)** → //EMP_ROW[contains(APELLIDO, "T")]/APELLIDO
Conseguimos el oficio de aquellos empleados cuyo apellido contienen la letra “T”.

i) **string-length(argumento)** → //EMP_ROW/APELLIDO[string-length() < 4]
Retorna el apellido de los empleados cuyo apellido tiene menos de 4 letras

9) Resuelve las siguientes consultas:

a) Devuelve el apellido del penúltimo empleado (NOTA: utilizar last())

```
/EMPLEADOS/EMP_ROW[position() = last() -1]/APELLIDO/text()
```

b) Obtén los elementos del empleado que ocupa la posición 3 (position())

```
/EMPLEADOS/EMP_ROW[position() = 3]
```

c) Cuenta el número de empleados del departamento 10

```
/EMPLEADOS/count(EMP_ROW[DEPT_NO = 10])
```

d) Obtén la suma de SALARIO de los empleados del DEPT_NO =20

```
/EMPLEADOS/sum(EMP_ROW[DEPT_NO = 20]/SALARIO)
```

e) Obtén el salario máximo, el mínimo de los empleados con OFICIO=ANALISTA

```
/EMPLEADOS/concat("Salario mínimo: ", min(EMP_ROW[OFICIO = "ANALISTA"]/SALARIO), ", Salario
    máximo: ", max(EMP_ROW[OFICIO = "ANALISTA"]/SALARIO))
```

f) Obtén la media de salario en el DEPT_NO=10

```
/EMPLEADOS/avg(EMP_ROW[DEPT_NO = 10]/SALARIO)
```

g) Devuelve la concatenación de apellido, oficio y salario

```
/EMPLEADOS/EMP_ROW/concat(APELLIDO, " ", OFICIO, " ", SALARIO)
```

h) Obtén los elementos de los empleados cuyo apellido empieza por ‘A’

```
/EMPLEADOS/EMP_ROW[starts-with(APELLIDO, "A")]
```

i) Devuelve los oficios que contienen la sílaba ‘OR’

```
/EMPLEADOS/EMP_ROW[contains(OFICIO, "OR")]/OFICIO
```

j) Obtén los datos de los empleados cuyo apellido tiene menos de 4 caracteres

```
/EMPLEADOS/EMP_ROW/APELLIDO[string-length() < 4]
```

10) Resuelve las siguientes consultas referentes al documento productos.xml. Este documento contiene los datos de los productos de una distribuidora de componentes informáticos. La estructura del documento es:

```
<produc>
    <cod_prod>xxx</cod_prod>
    <denominacion>xxxx</denominacion>
    <precio>xxx</precio>
    <stock_actual>xxx</stock_actual>
    <stock_minimo>xxxx</stock_minimo>
    <cod_zona>xxx</cod_zona>
</produc>
```

a) Obtén la denominación y precio de todos los productos

```
/productos/produc/denominacion | /productos/produc/precio
```

b) Obtén los productos que sean "Placa Base"

```
/productos/produc[contains(denominacion, "Placa Base")]
```

c) Obtén los productos cuyo precio sea mayor que 60€ y de la zona 20

```
/productos/produc[cod_zona = 20 and precio > 60]
```

d) Obtén el número de los productos que sean memorias y de la zona 10

```
/productos/count(produc[contains(denominacion, "Memoria") and cod_zona = 10])
```

e) Obtén la media de los precios de los micros

```
/productos/avg(produc[contains(denominacion, "Micro")]/precio)
```

f) Obtén los datos de los productos cuyo stock mínimo sea mayor que el stock actual (NOTA: usa función number())

```
/productos/produc[number(stock_minimo) > number(stock_actual)]
```

g) Obtén el producto más caro

```
/productos/max(produc/precio)
```

h) Obtén el producto más barato de la zona 20

```
/productos/produc[precio = min(precio) and cod_zona = 20]
```

Problemas - XQuery 1

1. Prueba las siguientes expresiones en eXide y averigua qué devuelven:

1. CÓDIGO

```
for $emp in /EMPLEADOS/EMP_ROW
order by $emp/APELLIDO
return if ($emp/OFICIO='DIRECTOR')
then <DIRECTOR>{$emp/APELLIDO/text()}</DIRECTOR>
else <EMPLE> {data($emp/APELLIDO)} </EMPLE>
```

1. RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
    <EMPLE>ALONSO</EMPLE>
    <EMPLE>ARROYO</EMPLE>
    <DIRECTOR>CEREZO</DIRECTOR>
    <EMPLE>FERNANDEZ</EMPLE>
    <EMPLE>GIL</EMPLE>
    <DIRECTOR>JIMENEZ</DIRECTOR>
    <EMPLE>JIMENO</EMPLE>
    <EMPLE>MARTIN</EMPLE>
    <EMPLE>MUÑOZ</EMPLE>
    <DIRECTOR>NEGRO</DIRECTOR>
    <EMPLE>REY</EMPLE>
    <EMPLE>SALA</EMPLE>
    <EMPLE>SANCHEZ</EMPLE>
    <EMPLE>TOVAR</EMPLE>
```

2. CÓDIGO

```
for $prof in /universidad/departamento[@tipo='A']/empleado
let $profe:=$prof/nombre, $puesto:=$prof/puesto
where $puesto='Profesor'
return $profe
```

2. RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
    <nombre>Alicia Martín</nombre>
    <nombre>Mª Jesús Ramos</nombre>
    <nombre>Pedro Paniagua</nombre>
```

3. CÓDIGO

```
for $dep in /universidad/departamento
```

```
return if ($dep/@tipo='A')
then <tipoA>{data($dep/nombre)}</tipoA>
else <tipoB>{data($dep/nombre)}</tipoB>
```

3. RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
    <tipoA>Informática</tipoA>
    <tipoA>Matemáticas</tipoA>
    <tipoB>Análisis</tipoB>
```

4. CÓDIGO

```
for $dep in /universidad/departamento
let $nom:=$dep/empleado
return
       <depart>{data($dep/nombre)}
           <emple>{count($nom)} </emple>
       </depart>
```

4. RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
    <tipoA>Informática</tipoA>
    <tipoA>Matemáticas</tipoA>
    <tipoB>Análisis</tipoB>
```

5. CÓDIGO

```
for $dep in /universidad/departamento
let $emp:=$dep/empleado
let $sal:=$dep/empleado/@salario
return
       <depart>{data($dep/nombre)}
          <emple>{count($emp)}</emple>
          <medsal>{avg($sal)}</medsal>
       </depart>
```

5. RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
    <depart>Informática<emple>2</emple>
       <medsal>2150</medsal>
    </depart>
    <depart>Matemáticas<emple>4</emple>
       <medsal>2200</medsal>
```

```
    </depart>
    <depart>Análisis<emple>2</emple>
       <medsal>2050</medsal>
    </depart>
```

6. CÓDIGO

```
for $dep in /universidad/departamento
let $emp:=$dep/empleado
let $sal:=$dep/empleado/@salario
let $maxi:=max($dep/empleado/@salario)
let $emplmax:=$dep/empleado[@salario=$maxi]
return
       <depart>{data($dep/nombre)}
          <emple>{count($emp)}</emple>
          <medsal>{avg($sal)}</medsal>
          <salmax>{$maxi}</salmax>
          <emplemax>{$emplmax/nombre/text()} - {data($emplmax/@salario)}</emplemax>
       </depart>
```

6. RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
    <depart>Informática<emple>2</emple>
       <medsal>2150</medsal>
       <salmax>2300</salmax>
       <emplemax>Alicia Martín - 2300</emplemax>
    </depart>
    <depart>Matemáticas<emple>4</emple>
       <medsal>2200</medsal>
       <salmax>2500</salmax>
       <emplemax>Antonia González - 2500</emplemax>
    </depart>
    <depart>Análisis<emple>2</emple>
       <medsal>2050</medsal>
       <salmax>2200</salmax>
       <emplemax>Mario García - 2200</emplemax>
    </depart>
```

Problemas - XQuery 2

1. Resuelve las siguientes consultas utilizando el documento EMPLEADOS.xml

a. Obtén los nombres de oficio que empiezan por P

CÓDIGO

```
for $emple in /EMPLEADOS/EMP_ROW
where (starts-with($emple/OFICIO, "P"))
return $emple/OFICIO
```

RESPUESTA

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

b. Obtén los nombres de oficio y el número de los empleados de cada oficio. Utiliza distinct-values

CÓDIGO

```
for $oficio in distinct-values(/EMPLEADOS/EMP_ROW/OFICIO)
let $num_emple := /EMPLEADOS/EMP_ROW[OFICIO = $oficio]/EMP_NO
return <OFICIO> {$oficio}
               <NUM_EMPLE>{$num_emple}</NUM_EMPLE>
       </OFICIO>
```

RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
<OFICIO>EMPLEADO<EMPLEADOS>
       <EMP_NO>7369</EMP_NO>
       <EMP_NO>7876</EMP_NO>
       <EMP_NO>7900</EMP_NO>
       <EMP_NO>7934</EMP_NO>
   </EMPLEADOS>
</OFICIO>
<OFICIO>VENDEDOR<EMPLEADOS>
       <EMP_NO>7499</EMP_NO>
       <EMP_NO>7521</EMP_NO>
       <EMP_NO>7654</EMP_NO>
       <EMP_NO>7844</EMP_NO>
   </EMPLEADOS>
</OFICIO>
<OFICIO>DIRECTOR<EMPLEADOS>
       <EMP_NO>7566</EMP_NO>
       <EMP_NO>7698</EMP_NO>
       <EMP_NO>7782</EMP_NO>
```

```
   </EMPLEADOS>
</OFICIO>
<OFICIO>ANALISTA<EMPLEADOS>
       <EMP_NO>7788</EMP_NO>
       <EMP_NO>7902</EMP_NO>
   </EMPLEADOS>
</OFICIO>
<OFICIO>PRESIDENTE<EMPLEADOS>
       <EMP_NO>7839</EMP_NO>
   </EMPLEADOS>
</OFICIO>
```

c. Obtén el número de empleados que tiene cada departamento y la media de salario redondeada

CÓDIGO

```
for $dept in distinct-values(/EMPLEADOS/EMP_ROW/DEPT_NO)
let $num_emple := /EMPLEADOS/EMP_ROW[DEPT_NO = $dept]/EMP_NO
let $salario := /EMPLEADOS/EMP_ROW[EMP_NO = $num_emple] /SALARIO
return <DEPART> {$dept}
              <NUM_EMPLE>{count($num_emple)}</NUM_EMPLE>
              <AVG_SALARIO>{avg($salario)}</AVG_SALARIO>
       </DEPART>
```

RESPUESTA

```
<?xml version="1.0" encoding="UTF-8"?>
<DEPART>20<NUM_EMPLE>5</NUM_EMPLE>
   <AVG_SALARIO>2274</AVG_SALARIO>
</DEPART>
<DEPART>30<NUM_EMPLE>6</NUM_EMPLE>
   <AVG_SALARIO>1735.8333333333333</AVG_SALARIO>
</DEPART>
<DEPART>10<NUM_EMPLE>3</NUM_EMPLE>
   <AVG_SALARIO>2891.6666666666665</AVG_SALARIO>
</DEPART>
```

2. Utilizando el documento productos.xml, resuelve con XQuery:

a. Obtén por cada zona el número de productos que tiene

```
let $zona := /productos/produc/cod_zona
let $prodDiferente := distinct-values($zona)
return
       <productos> {
       for $zona in $prodDiferente
       return
       <nombreProd> {
       concat("Zona: ", $zona, " Cantidad: ", count(/productos/produc[cod_zona=$zona]))}
       </nombreProd>
       }
       </productos>
```

b. Obtén la denominación de los productos entre las etiquetas si son del código de zona 10, si son del código de zona 20, etc.

```
for $prod in /productos/produc
return
	if ($prod/cod_zona = 10) then <zona10>{$prod/denominacion/text()}</zona10>
	else if ($prod/cod_zona = 20) then <zona20>{$prod/denominacion/text()}</zona20>
	else if ($prod/cod_zona = 30) then <zona30>{$prod/denominacion/text()}</zona30>
	else <zona40>{$prod/denominacion/text()}</zona40>
```

c. Obtén por cada zona la denominación del o de los productos más caros.

```
let $zona := /productos/produc/cod_zona
let $prodDiferente := distinct-values($zona)
return
	<productos> {
	for $zona in $prodDiferente
		return
		<maxPrecio> {
		concat("Zona: ", $zona, " Precio Maximo: ",
max(/productos/produc[cod_zona=$zona]/precio))
		}
	</maxPrecio>
	}
	</productos>
```

d. Obtén la denominación de los productos contenida entre las etiquetas para los productos en cuya denominación aparece la palabra Placa Base, para los que contienen la palabra Memoria, para los que contienen la palabra Micro y para el resto de productos

```
let $prod := /productos/produc
return
	<productos> {
	for $prod in $prod
		let $denom := $prod/denominacion
	return
	if(contains($denom, "Placa Base")) then
	<placaBase> { data($denom)} </placaBase>
	else if(contains($denom, "Memoria")) then
	<memoria> { data($denom)} </memoria>
	else if(contains($denom, "Micro")) then
	<micro> { data($denom)} </micro>
	else
	<resto> {data($denom)} </resto>
	}
	</productos>
```