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] →** Esta expresión retorna la información completa de los usuarios que pertenecen al departamento 10.

   b. **/EMPLEADOS/EMP_ROW/APELLIDO|/EMPLEADOS/EMP_ROW/DEPT_NO** → Con esta expresión obtenemos el apellido y el departamento de cada uno de los empleados.

   c. **/EMPLEADOS/EMP_ROW [DEPT_NO=10]/APELLIDO/text()** → Nos retorna el apellido en formato texto de todos los empleados del departamento 10.

   d. **/EMPLEADOS/EMP_ROW [not(OFICIO=’ANALISTA’)]** → Obtenemos la información de los empleados que no tienen el oficio “Analista”.

e. **/EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=20]/APELLIDO** → Esta expresión nos retorna la información de aquellos empleados que pertenecen al departamento 20 y cobran más de 1300.

f. **/EMPLEADOS/EMP_ROW[1]** → Nos retorna la información 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()**
//EMP_ROW[DEPT_NO=20][last()]/APELLIDO
Esto retorna el apellido del último empleado del departamento 20.

b. **position()** →
//DEP_ROW[position()=3]/DNOMBRE
Retorna el nombre del departamento que se encuentra en la posición 3.

c. **count()** →
count(//EMP_ROW)
Retorna el número total de empleados.

d. **sum(),div(),mod()** →
sum(//SALARIO) div(count(//EMP_NO))
Obtenemos el salario medio mediante el uso de tres operadores distintos, suma de salarios, contar cuántos empleados ahí y dividir el total por el número de empleados.

e. **max(), min(),avg()** →
avg(//SALARIO)
Es equivalente a la expresión anterior. Obtenemos el salario medio

f. **concat(cadena1, cadena2,...)** →
concat(//EMP_ROW[5]/APELLIDO, " tiene el cargo de ", //EMP_ROW[5]/OFICIO)
Construimos una frase que nos muestra el apellido y el cargo del empleado.

g. **starts-with (cadena1, cadena2)** →
//EMP_ROW[starts-with(OFICIO, "P")]
Retorna aquellos empleados cuyo oficio empieza con la letra “P”

h. **contains(cad1,cad2)** →
//DEP_ROW[contains(LOC, "B")]/DNOMBRE
Nos retorna el nombre de los departamentos cuya localidad contiene las letras “BA”

i. **string-length(argumento)** →
//DEP_ROW/LOC[string-length() > 6]
Obtenemos las localidades con más de 6 letras que pertenecen a los departamentos.

9. **Resuelve las siguientes consultas:**

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

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

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

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

   c. **Cuenta el número de empleados del departamento 10**

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

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

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

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

```
concat("Mínimo: ", min(//EMP_ROW[OFICIO = "ANALISTA"]/SALARIO), ", Máximo: ",
max(//EMP_ROW[OFICIO = "ANALISTA"]/SALARIO))
```

   f. **Obtén la media de salario en el DEPT_NO=10**

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

   g. **Devuelve la concatenación de apellido, oficio y salario**

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

   h. **Obtén los elementos de los empleados cuyo apellido empieza por 'A'**

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

   i. **Devuelve los oficios que contienen la sílaba 'OR'**

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

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

```
//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

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

b. Obtén los productos que sean “Placa base”

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

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

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

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

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

e. Obtén la media de los precios de los micros

```
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())

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

g. Obtén el producto más caro

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

h. Obtén el producto más barato de la zona 20

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

Problemas - XQuery 1

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

EXPRESIÓN 1	RESULTADO

`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>`
	`<?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>
```

EXPRESIÓN 2

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

RESULTADO 2

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

EXPRESIÓN 3

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

RESULTADO 3

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

EXPRESIÓN 4

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

RESULTADO 4

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

EXPRESIÓN 5

```
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>
```

RESULTADO 5

```
<?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>
```

EXPRESIÓN 6

```
for $dep in /universidad/departamento
let $emp:=$dep/empleado
let $sal:=$dep/empleado/@salario
```

RESULTADO 6

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

```
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>
```

```
   <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**

EXPRESIÓN	RESULTADO

`for $emp in /EMPLEADOS/EMP_ROW`
`where (starts-with($emp/OFICIO, "P"))`
`return $emp/OFICIO`
	`<?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**

EXPRESIÓN	RESULTADO

`for $emp in distinct-values(/EMPLEADOS/EMP_ROW/OFICIO)`
`let $num_empleado := /EMPLEADOS/EMP_ROW[OFICIO = $emp]/EMP_NO`
`return <OFICIO> {$emp}`
 	
`<NUM_EMPLEADO>{$num_empleado}</NUM_EMPLEADO>`
`</OFICIO>`
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<OFICIO>EMPLEADO<NUM_EMPLEADO>`
	`<EMP_NO>7369</EMP_NO>`
	`<EMP_NO>7876</EMP_NO>`
	`<EMP_NO>7900</EMP_NO>`
	`<EMP_NO>7934</EMP_NO>`
	`</NUM_EMPLEADO>`
	`</OFICIO>`
	`<OFICIO>VENDEDOR<NUM_EMPLEADO>`
	`<EMP_NO>7499</EMP_NO>`
	`<EMP_NO>7521</EMP_NO>`
	`<EMP_NO>7654</EMP_NO>`
	`<EMP_NO>7844</EMP_NO>`
	`</NUM_EMPLEADO>`
	`</OFICIO>`
	`<OFICIO>DIRECTOR<NUM_EMPLEADO>`
	`<EMP_NO>7566</EMP_NO>`
	`<EMP_NO>7698</EMP_NO>`

```
            <EMP_NO>7782</EMP_NO>
        </NUM_EMPLEADO>
    </OFICIO>
    <OFICIO>ANALISTA<NUM_EMPLEADO>
            <EMP_NO>7788</EMP_NO>
            <EMP_NO>7902</EMP_NO>
        </NUM_EMPLEADO>
    </OFICIO>
    <OFICIO>PRESIDENTE<NUM_EMPLEADO>
            <EMP_NO>7839</EMP_NO>
        </NUM_EMPLEADO>
    </OFICIO>
```

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

EXPRESIÓN	RESULTADO

`for $depart in distinct-values(/EMPLEADOS/EMP_ROW/DEPT_NO)`
`let $num_empleado := /EMPLEADOS/EMP_ROW[DEPT_NO = $depart]/EMP_NO`
`let $salario := /EMPLEADOS/EMP_ROW[EMP_NO = $num_empleado] /SALARIO`
`return <DEPART> {$depart}`
 	
`<NUM_EMPLEADO>{count($num_empleado)}</NUM_EMPLEADO>`
 	
`<AVG_SALARIO>{avg($salario)}</AVG_SALARIO>`
`</DEPART>`
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<DEPART>20<NUM_EMPLEADO>5</NUM_EMPLEADO>`
	`<AVG_SALARIO>2274</AVG_SALARIO>`
	`</DEPART>`
	`<DEPART>30<NUM_EMPLEADO>6</NUM_EMPLEADO>`
	`<AVG_SALARIO>1735.8333333333333</AVG_SALARIO>`
	`</DEPART>`
	`<DEPART>10<NUM_EMPLEADO>3</NUM_EMPLEADO>`
	`<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 $productoDistinto := distinct-values($zona)
return
    <productos> {
    for $zona in $productoDistinto
    return
        <nombre-producto> {
        concat("Zona: ", $zona, " Cantidad: ", count(/productos/produc[cod_zona=$zona]))}
        </nombre-producto>
    }
    </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 $producto in /productos/produc
return
    if ($producto/cod_zona = 10) then <zona10>{$producto/denominacion/text()}</zona10>
    else if ($producto/cod_zona = 20) then <zona20>{$producto/denominacion/text()}</zona20>
    else if ($producto/cod_zona = 30) then <zona30>{$producto/denominacion/text()}</zona30>
    else <zona40>{$producto/denominacion/text()}</zona40>
```

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

```
let $zona := /productos/produc/cod_zona
let $productoDistinto := distinct-values($zona)
return
    <productos> {
        for $zona in $productoDistinto
            return
            <producto-mas-caro> {
            concat("Zona: ", $zona, " Precio Max: ",
max(/productos/produc[cod_zona=$zona]/precio))
            }
        </producto-mas-caro>
        }
    </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 $producto := /productos/produc
return
    <productos> {
        for $producto in $producto
            let $prodDenominacion := $producto/denominacion
        return
        if(contains($prodDenominacion, "Placa Base")) then
        <placa-base> { data($prodDenominacion)} </placa-base>
        else if(contains($prodDenominacion, "Memoria")) then
        <memoria> { data($prodDenominacion)} </memoria>
        else if(contains($prodDenominacion, "Micro")) then
        <micro> { data($prodDenominacion)} </micro>
        else
        <resto> {data($prodDenominacion)} </resto>
        }
    </productos>
```