CONSULTAS XQUERY

Una consulta en XQuery es una expresión que lee datos de uno o más documentos en XML y devuelve como resultado otra secuencia de datos en XML.

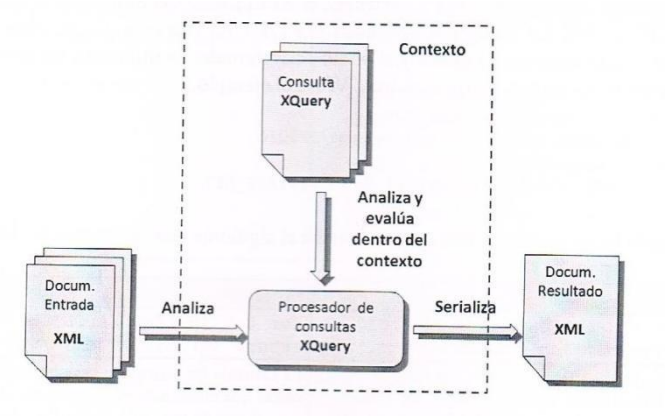


Ilustración 1 Procesamiento de un consulta XQuery

XQuery contiene a XPath, es decir toda expresión de consulta en XPath es válida y devuelve el mismo resultado en XQuery. XQuery nos va a permitir:

x Seleccionar información basada en un criterio específico. x Buscar información en un documento o conjunto de documentos. x Unir datos desde múltiples documentos o colecciones de documentos.
x Organizar, agrupar y resumir datos. x Transformar y reestructurar datos XML en otro vocabulario o estructura. x Desempeñar cálculos aritméticos sobre números y fechas. x Manipular cadenas de caracteres a formato de texto.

En XQuery las consultas siguen la norma FLWOR (leído como flower), corresponde a las siglas de For, Let, Where, Order y Return. Permite a diferencia de Xpat manipular, transformar y organizar los resultados de las consultas. La sintaxis general de una estructura FLWOR es: **for** <variable> in <expresión XPath> **let** <variables vinculadas> **where** <condición XPath> **order by** <expresión> **return** <expresión de salida>

x **For**: se usa para seleccionar nodos y almacenarlos en una variable, similar a la cláusula FROM de SQL. Dentro del **for** escribimos una expresión XPath que seleccionará los nodos. Si se especifica más de una variable en el for se actúa como producto cartesiano. Las variables comienzan con **$**. Las consultas XQuery deben llevar obligatoriamente una orden **Return**, donde indicaremos los que queremos que nos devuelva la consulta.

Ejemplos:

XQUERY	**XPATH**
Consulta que devuelva los elementos EMP_ROW	
`for $emp in /EMPLEADOS/EMP_ROW return $emp`	`/EMPLEADOS/EMP_ROW`
Consulta que devuelva los apellidos de los empleados	

`for $emp in /EMPLEADOS/EMP_ROW/APELLIDO return $emp for $emp in /EMPLEADOS/EMP_ROW return $emp/APELLIDO`	`/EMPLEADOS/EMP_ROW/APELLIDO`

x **Let**: permite que se asignen valores resultantes de expresiones XPath a variables para simplificar la representación. Se pueden poner varias líneas let una por cada variable o separar las variables por comas.

Ejemplo: Se crean 2 variables, el APELLIDO del empleado se guarda en $nom, y el OFICIO en $ofi. La salida sale ordenada por OFICIO y se crea $ofi. La salida sale ordenada por OFICIO y se crea una etiqueta <APE_OFI></APE_OFI> que incluye el nombre y el oficio concatenado. Se utilizarán las llaves en el return para añadir el contenido de las variables.

XQUERY
`for $emp in /EMPLEADOS/EMP_ROW let $nom := $emp/APELLIDO, $ofi := $emp/OFICIO order by $emp/OFICIO return <APE_OFI>{concat($nom, ', ', $ofi)}</APE_OFI>`

La cláusula **let** se puede utilizar sin for. Ejemplo:

SIN FOR	**CON FOR**
`let $ofi := /EMPLEADOS/EMP_ROW/OFICIO return <OFICIOS>{$ofi}</OFICIOS>`	`for $emp in /EMPLEADOS/EMP_ROW let $ofi := $emp/OFICIO return <OFICIOS>{$ofi}</OFICIOS> o for $ofi in /EMPLEADOS/EMP_ROW/OFICIO return <OFICIOS>{$ofi}</OFICIOS>`
La cláusula let vincula la variable *$ofi* con todo el resultado de la expresión. En este caso vincula todos los oficios creando un elemento <OFICIOS> con todos los oficios.	La cláusula for vincula la variable *$ofi* con cada nodo oficio que encuentre en la colección de datos, creando un elemento por cada oficio. Por eso aparece la etiqueta <OFICIOS> para cada oficio.

x **Where**: filtra los elementos eliminando todos los valores que no cumplan las condiciones dadas.

x **Order by**: ordena los datos según el criterio dado.

x **Return**: construye el resultado de la consulta en XML, se pueden añadir etiquetas XML a la salida, si añadimos etiquetas los datos a visualizar los encerramos entre llaves {}. Además en el return se pueden añadir condiciones usando if-then-else y así tener más versatilidad en la salida.

Hay que tener en cuenta que la cláusula else es obligatoria y debe aparecer siempre en la expresión condicional, se debe a que toda expresión XQuery debe devolver un valor. Si no existe ningún valor a devolver al no cumplirse la cláusula if, devolvemos una secuencia vacía con else ().

Ejemplo: devuelve los departamentos de tipo A encerrados en una etiqueta.

XQUERY

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

Se utiliza la función data() para extraer el contenido en texto de los elementos. También se utilizará data() para extraer el contenido de los atributos.

Ejemplo

XPath	
Correcta	**Errónea**
`//empleado/data(@salario)` Devuelve los salarios	`//empleado/@salario` Salario no es un nodo

Dentro de las asignaciones **let** en las consultas XQuery podemos utilizar expresiones del tipo:

`let $var := //empleado/@salario` esto no da error, pero si queremos extraer los datos pondremos: `let $var := data(//empleado/@salario)` o `return data($var)`

Ejemplos:

Consulta XQuery	**Resultado**
Devuelve los nombres de los empleados, los que son directores entre las etiquetas <DIRECTOR> </DIRECTOR> y los que no lo son entre las etiquetas <EMPLE> </EMPLE>	

`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>`
	<EMPLE>ALONSO</EMPLE>
	<EMPLE>ARROYO</EMPLE>
	<DIRECTOR>CEREZO</DIRECTOR>
	<EMPLE>FERNANDEZ</EMPLE>
	<EMPLE>GIL</EMPLE>
	<DIRECTOR>JIMENEZ</DIRECTOR>
	……..

Devuelve los nodos DEP_ROW de un documento ubicado en una carpeta del disco duro	
`for $de in doc('file:///C:\Users\Usuario\Documents\CSDAMultiplataforma\AccesoDatosActual\BDeXist\ColeccionPruebas\departamentos.xml')/departamentos/DEP_ROW return $de`	
Obtiene los nombres de empleados de los departamentos de tipo ‘A’ que son profesores	

```
for $profesores in /colegio/departamento[@tipo='A']/empleado let
$profesor := $profesores/nombre, $puesto := $profesores/puesto
where $puesto = 'Profesor'
return $profesor
```

```
<nombre>Alicia Martín</nombre>
<nombre>Mª Jesús Ramos</nombre>
<nombre>Pedro Paniagua</nombre>
```

Devuelve el nombre del departamento encerrado entre las etiquetas <tipoA></tipoA>, si es del tipo A y <tipoB></tipoB>, si no lo es.

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

```
<tipoA>Informática</tipoA>
<tipoA>Matemáticas</tipoA>
<tipoB>Análisis</tipoB>
```

Devuelve el nombre de los profesores encerrados entre las etiquetas <tipoA></tipoA>, si pertenece a un departamento del tipo A y <tipoB></tipoB>, si no lo es

```
for $depar in /colegio/departamento let
$tip := $depar/@tipo
return if($tip = 'A')
          then <tipoA>{$depar/empleado/nombre}</tipoA>
          else <tipoB>{$depar/empleado/nombre}</tipoB>
```

```
<tipoA>
    <nombre>Juan Parra</nombre>
    <nombre>Alicia Martín</nombre>
</tipoA>
<tipoA>
    <nombre>Juan Parra</nombre>
    <nombre>Mª Jesús Ramos</nombre>
    <nombre>Pedro Paniagua</nombre>
    <nombre>Antonia González</nombre>
</tipoA>
<tipoB>
    <nombre>Laura Ruiz</nombre>
    <nombre>Mario García</nombre> </tipoB>
```

LA ETIQUETA ES POR DEPARTAMENTO.

```
for $emp in /colegio/departamento/empleado
let $tip := //$emp/../data(@tipo) return
if($tip = "A")
          then <tipoA>{data($emp/nombre)}</tipoA>
          else <tipoB>{data($emp/nombre)}</tipoB>
```

```
<tipoA>Juan Parra</tipoA>
<tipoA>Alicia Martín</tipoA>
<tipoA>Juan Parra</tipoA>
<tipoA>Mª Jesús Ramos</tipoA>
<tipoA>Pedro Paniagua</tipoA>
<tipoA>Antonia González</tipoA>
<tipoB>Laura Ruiz</tipoB>
<tipoB>Mario García</tipoB>
```

LA ETIQUETA ES POR PROFESOR.

Devuelve los nombres de los departamentos y el número de empleados que tiene encerrados entre etiquetas.

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

```
<depart>Informática<emple>2</emple>
</depart>
<depart>Matemáticas<emple>4</emple>
</depart>
<depart>Análisis<emple>2</emple>
</depart>
```

Obtiene los nombres de departamento, los empleados que tiene y la media del salario entre etiquetas.

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

```
<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>
```

OPERADORES Y FUNCIONES MÁS COMUNES EN XQUERY

Las funciones y operadores soportados por XQuery prácticamente son los mismos que los soportados por Xpath. Soporta operadores y funciones matemáticas, de cadenas, para el tratamiento de expresiones regulares, comparaciones de fechas y horas, manipulación de nodos XML, manipulación de secuencias, comprobación y conversión de tipos y lógica booleana. Los operadores y funciones más comunes son:

Matemáticos: +, -, *, div (se utiliza div en lugar de la /), idiv (es la división entera), mod.

Comparación: =, !=, <, >, <=, >=, not().

Secuencia: *union* (|), **intersect**, *except*.

Redondeo: *floor(), ceiling(), round()*.

Funciones de agrupación: *count(), min(), max(), sum(), avg()*.

Funciones de cadena: *concat(), string-length(), starts-with(), ends-with(), substring(), upper-case(), lower-case(), string()*.

Uso general: *distinct-values()* extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados. *empty()* devuelve cierto cuando la expresión entre paréntesis está vacía. Y *exists()* devuelve cierto cuando una secuencia contiene al menos un valor.

Los **comentarios** en XQuery van encerrados entre caras sonrientes: (:Esto es un comentario:).

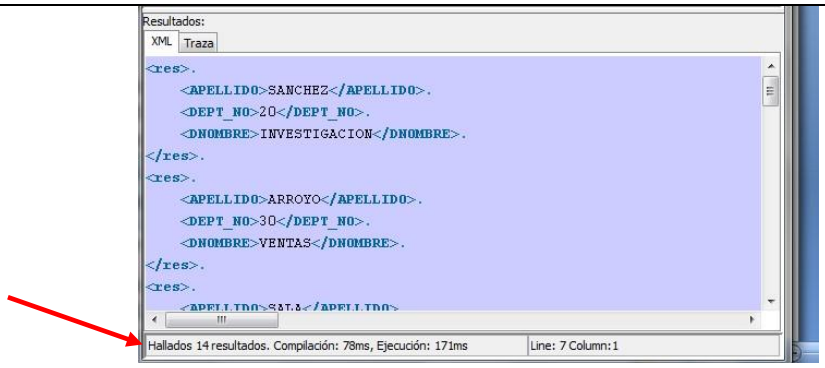
Ejemplos:

Consulta XQuery	**Resultado**
Obtener los oficios que empiezan por P	
`for $ofi in /EMPLEADOS/EMP_ROW/OFICIO where starts-with(data($ofi), 'P') return $ofi`	<OFICIO>PRESIDENTE</OFICIO>
Obtener los nombres de los oficios y el número de empleados de cada uno ellos	
`for $ofi in distinct-values(/EMPLEADOS/EMP_ROW/OFICIO) let $con := count(/EMPLEADOS/EMP_ROW[OFICIO = $ofi]) return concat($ofi, ' - ',$con)`	EMPLEADO - 4 VENDEDOR - 4 DIRECTOR - 3 ANALISTA - 2 PRESIDENTE - 1
Obtener el número de empleados de cada departamento y la media del salario redondeada.	
`for $depar in distinct-values(/EMPLEADOS/EMP_ROW/DEPT_NO) let $con := count(/EMPLEADOS/EMP_ROW[DEPT_NO = $depar]) let $sala := round(avg(/EMPLEADOS/EMP_ROW[DEPT_NO = $depar]/SALARIO)) return concat('Departamento: ', $depar, ' Núm. empleados: ', $con, ' Media salario: ', $sala)`	Departamento: 20 Núm. empleados: 5 Media salario: 2274 Departamento: 30 Núm. empleados: 6 Media salario: 1736 Departamento: 10 Núm. empleados: 3 Media salario: 2892

CONSULTAS COMPLEJAS CON XQUERY

Las consultas XQuery nos permiten trabajar con varios documentos xml para extraer su información, podremos incluir tantas sentencias *for* como necesitemos, incluso dentro del *return*. Además podremos añadir, borrar e incluso modificar elementos, bien generando un documento *xml* nuevo o utilizando las sentencias de actualización de eXist.

Joins de documentos

Consulta XQuery
Visualizar por cada empleado del documento *empleados.xml*, su apellido, su número de departamento y el nombre del departamento que se encuentra en el documento *departamentos.xml*
`for $emp in (/EMPLEADOS/EMP_ROW) let $ape := $emp/APELLIDO let $dep := $emp/DEPT_NO`
`let $dnom := (/departamentos/DEP_ROW[DEPT_NO = $dep]/DNOMBRE) return <res>{$ape, $dep, $dnom}</res>`
Se pueden eliminar los paréntesis y obtendremos el mismo resultado `for $emp in /EMPLEADOS/EMP_ROW let $ape := $emp/APELLIDO let $dep := $emp/DEPT_NO let $dnom := /departamentos/DEP_ROW[DEPT_NO = $dep]/DNOMBRE return <res>{$ape, $dep, $dnom}</res>`
Se pueden utilizar varios for para extraer los distintos valores `for $emp in /EMPLEADOS/EMP_ROW let $ape := $emp/APELLIDO let $dep := $emp/DEPT_NO for $departamentos in /departamentos/DEP_ROW[DEPT_NO = $dep] let $dnom := $departamentos/DNOMBRE return <res>{$ape, $dep, $dnom}</res>`

Utilizando los documentos *departamentos.xml* y *empleados.xml*, obtener por cada departamento, el nombre de departamento, el número de empleados y la media de salario.
Se puede no utilizar el segundo for y escribir siempre la ruta `for $departamentos in /departamentos/DEP_ROW let $dnum := $departamentos/DEPT_NO let $con := count(/EMPLEADOS/EMP_ROW[DEPT_NO = $dnum]/EMP_NO) let $media := avg(/EMPLEADOS/EMP_ROW[DEPT_NO = $dnum]/SALARIO) return <resul>{$departamentos/DNOMBRE} <numemp>{$con}</numemp> <medsalario>{$media}</medsalario></resul>`

```
<resul>
  <DNOMBRE>CONTABILIDAD</DNOMBRE>
  <numemp>3</numemp>
  <medsalario>2892</medsalario>
</resul>
<resul>
  <DNOMBRE>INVESTIGACION</DNOMBRE>
  <numemp>5</numemp>
  <medsalario>2274</medsalario>
</resul>
```

```
<resul>
  <DNOMBRE>VENTAS</DNOMBRE>
  <numemp>6</numemp>
  <medsalario>1736</medsalario>
</resul>
<resul>
  <DNOMBRE>PRODUCCION</DNOMBRE>
  <numemp>0</numemp>
  <medsalario/>
</resul>
```

Convertir la salida de la consulta anterior, de manera que la media del salario y el número de empleados sean atributos de cada departamento. Creamos la salida como una concatenación de los datos a obtener

```
for $departamentos in /departamentos/DEP_ROW
let $dnum := $departamentos/DEPT_NO
let $con := count(/EMPLEADOS/EMP_ROW[DEPT_NO = $dnum]/EMP_NO) let
$media := avg(/EMPLEADOS/EMP_ROW[DEPT_NO = $dnum]/SALARIO) return
concat('<departamento media salario = "', round($media), '"
numero empleados = "', $con, '">',
data($departamentos/DNOMBRE),'</departamento>')
```

```
<departamento media salario = "2892" numero
empleados = "3">CONTABILIDAD</departamento>
<departamento media salario = "2274" numero
empleados = "5">INVESTIGACION</departamento>
```

```
<departamento media salario = "1736"
numero empleados =
"6">VENTAS</departamento>
<departamento media salario = "" numero
empleados =
"0">PRODUCCION</departamento>
```

Utilizando los documentos *departamentos.xml* y *empleados.xml*, obtener por cada departamento, el nombre del empleado que más gana.

```
for $dep in /departamentos/DEP_ROW
let $codep := $dep/DEPT_NO let
$nbdep := $dep/DNOMBRE
let $salmax := max(/EMPLEADOS/EMP_ROW[DEPT_NO = $codep]/SALARIO)
let $nbemp := /EMPLEADOS/EMP_ROW[DEPT_NO = $codep and SALARIO =
$salmax]/APELLIDO
return <depar>
     {data($nbdep)}<salmax>{data($salmax)}</salmax><emple>{data($nb
emp)}</emple></depar>
```

```
<depar>CONTABILIDAD<salmax>4100</salmax>
  <emple>REY</emple>
</depar>
<depar>INVESTIGACION<salmax>3000</salmax>
  <emple>GIL FERNANDEZ</emple>
</depar>
```

```
<depar>VENTAS<salmax>3005</salmax>
  <emple>NEGRO</emple>
</depar>
<depar>PRODUCCION<salmax/>
  <emple/>
</depar>
```

Utilización de varios for

La utilización de varios *for* es muy útil para consultas en documentos XML anidados y también cuando utilizamos varios documentos unidos por una cláusula ***where*** como una combinación de tablas en SQL.

Consulta XQuery

Por cada departamento del documento *colegio.xml* obtener el número de empleados que hay en cada puesto de trabajo. Utilizaremos un *for* para obtener los nodos departamento y el segundo *for* para obtener los distintos puestos de cada departamento.

```
for $dep in /colegio/departamento
for $pue in distinct-values($dep/empleado/puesto)
let $con := count(/colegio/departamento/empleado[puesto = $pue])
return
<depart>{data($dep/nombre)}<puesto>{data($pue)}</puesto><profesore
s>{$con}</profesores></depart>
```

```
<depart>Informática<puesto>Asociado</puesto>
  <profesores>3</profesores>
</depart>
<depart>Informática<puesto>Profesor</puesto>
  <profesores>3</profesores>
</depart>
<depart>Matemáticas<puesto>Técnico</puesto>
  <profesores>1</profesores>
</depart>
```

```
<depart>Matemáticas<puesto>Profesor</puesto>
  <profesores>3</profesores>
</depart>
<depart>Matemáticas<puesto>Tutor</puesto>
  <profesores>1</profesores>
</depart>
<depart>Análisis<puesto>Asociado</puesto>
  <profesores>3</profesores>
</depart>
```

Por cada departamento del documento *colegio.xml* obtener el salario máximo y el empleado que tiene dicho salario. El primer for obtiene los nodos departamento y el segundo for los empleados de cada departamento. Para sacar el máximo en la salida preguntamos si el salario es el máximo..

```
for $dep in /colegio/departamento
for $emp in $dep/empleado let
$nbemple := $emp/nombre let
$sal := $emp/@salario
return if($sal = $dep/max(empleado/@salario))
     then
<depart>{data($dep/nombre)}<salmaximo>{data($sal)}</salmaximo
>         <empleado>{data($nbemple)}</empleado></depart>
else ()
```

También se pueden poder los dos for en la misma línea de la siguiente manera

```
for $dep in /colegio/departamento, $empleados in $dep/empleado
```

```
<depart>Informática<salmaximo>2300</salmaximo>
  <empleado>Alicia Martín</empleado>
</depart>
<depart>Matemáticas<salmaximo>2500</salmaximo>
  <empleado>Antonia González</empleado>
</depart>
<depart>Análisis<salmaximo>2200</salmaximo>
  <empleado>Mario García</empleado>
</depart>
```

Obtener el salario máximo mostrando además el nombre del empleado y el departamento al que pertenece.

```
let $salmax := max( /colegio/departamento/empleado/@salario)
for $dep in /colegio/departamento let
$dnom := $dep/nombre
let $nom := /colegio/departamento/empleado[@salario =
$salmax]/nombre
return if($dep/empleado/@salario = $salmax)
          then
<depart>{data($dnom)}<salmax>{data($salmax)}</salmax><empleado>{da
ta($nom)}</empleado></depart>
          else ()
```

```
<depart>Matemáticas<salmax>2500</salmax>
  <empleado>Antonia González</empleado>
</depart>
```

Por cada puesto del documento *colegio.xml* obtener el empleado con salario máximo y dicho salario. El primer *for* obtiene los distintos puestos de trabajo y el segundo *for* obtiene los empleados que tienen ese puesto de trabajo. En el *if* se pregunta si el salario del empleado es igual al salario máximo de los empleados del oficio del primer *for*.

```
for $pue in distinct-values(/colegio/departamento/empleado/puesto)
for $emp in /colegio/departamento/empleado[puesto = $pue]
let $sal := $emp/@salario let
$nom := $emp/nombre return
if($sal =
max(/colegio/departamento/empleado[puesto=$pue]/@salario))
     then
<puesto>{data($pue)}<maxsalario>{data($sal)}</maxsalario><empleado
>{data($nom)}</empleado></puesto>
          else ()
```

También podemos resolver la consulta utilizando un solo for, de la siguiente manera:

```
for $emp in
/colegio/departamento/empleado let $pue
:= $emp/puesto let $sal := $emp/@salario
let $nom := $emp/nombre order by $pue
return if($sal =
max(/colegio/departamento/empleado[puesto=$pue]/@salario))
     then
<puesto>{data($pue)}<maxsalario>{data($sal)}</maxsalario><empleado
>{data($nom)}</empleado></puesto>
          else ()
```

```
<puesto>Asociado<maxsalario>2200</maxsalario>
  <empleado>Mario García</empleado>
</puesto>
<puesto>Profesor<maxsalario>2300</maxsalario>
  <empleado>Alicia Martín</empleado>
</puesto>
<puesto>Profesor<maxsalario>2300</maxsalario>
  <empleado>Pedro Paniagua</empleado>
```

```
</puesto>
<puesto>Tutor<maxsalario>2500</maxsalario>
  <empleado>Antonia González</empleado>
</puesto>
<puesto>Técnico<maxsalario>1900</maxsalario>
  <empleado>Juan Parra</empleado> </puesto>
```

Visualizar por cada empleado del documento *empleados.xml*, su apellido, su número de departamento y el nombre del departamento que se encuentra en el documento *departamentos.xml*. Utilizando dos *for* y *where*

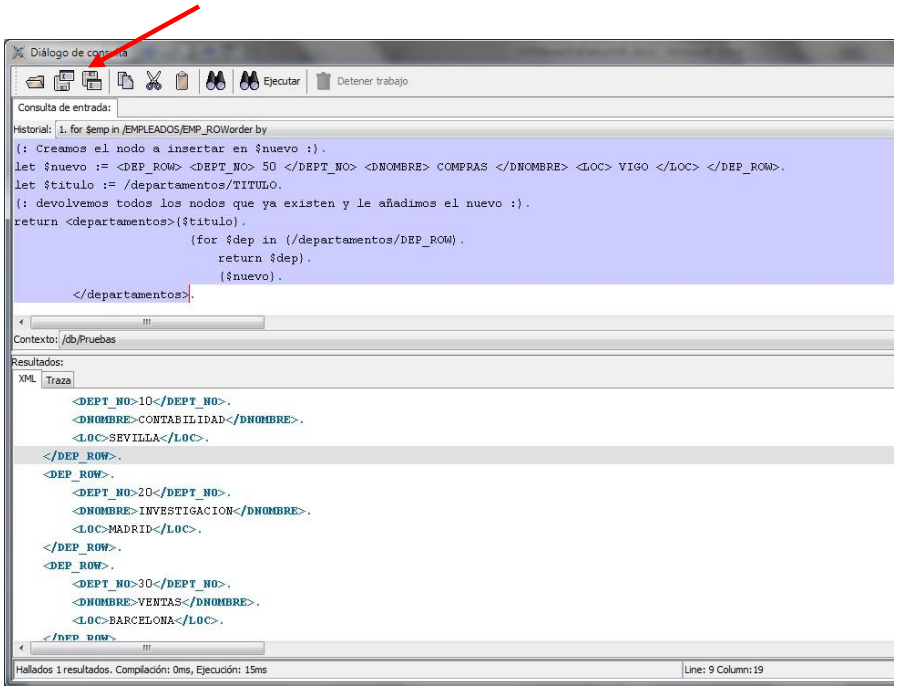
```
for $emp in (/EMPLEADOS/EMP_ROW)
for $dep in /departamentos/DEP_ROW
let $ape := $emp/APELLIDO let
$edep := $emp/DEPT_NO
where data($edep) = data($dep/DEPT_NO)   return
<res>{$ape, $edep}{$dep/DNOMBRE}</res>
```

Altas, bajas y modificaciones de nodos en documentos XML

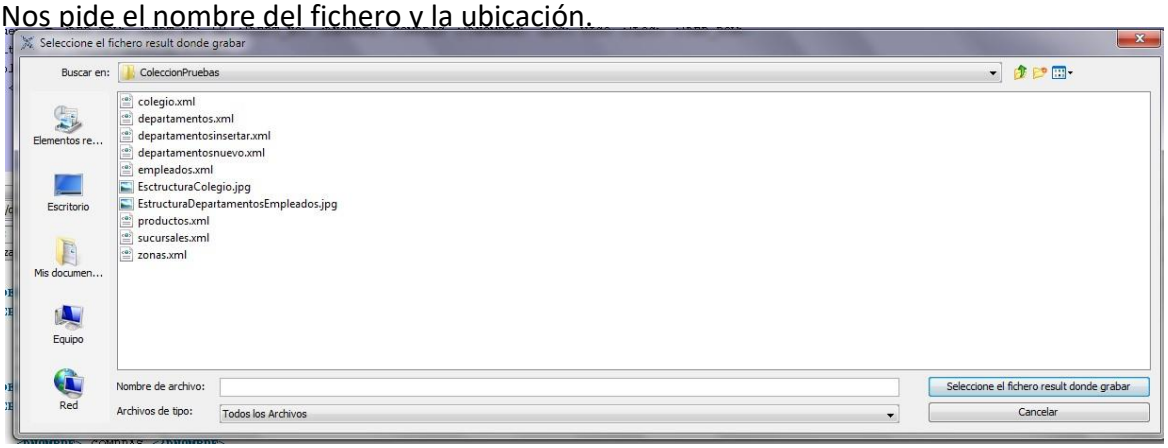
x **Altas**: Esta consulta va a generar una salida en la que se van a incluir todos los nodos del documento departamentos.xml, más uno nuevo a insertar.

```
(: Creamos el nodo a insertar en $nuevo :)
let $nuevo := <DEP_ROW> <DEPT_NO> 50 </DEPT_NO> <DNOMBRE>
COMPRAS </DNOMBRE> <LOC> VIGO </LOC> </DEP_ROW>
let $titulo := /departamentos/TITULO
(: devolvemos todos los nodos que ya existen y le añadimos el nuevo :)
return <departamentos>{$titulo}
        {for $dep in (/departamentos/DEP_ROW)              return $dep}
                {$nuevo}
</departamentos
```

En la variable *$nuevo* añadimos el nodo a insertar, en *$titulo* guardamos el nodo TITULO del documento *departamentos.xml*. En el *return* creamos una etiqueta llamada *<departamentos>* e incluimos el título, todos los nodos DEP_ROW de *departamentos.xml* (los obtenemos en el *for*) y el nodo nuevo. Si se desea guardar la salida generada se pulsa el botón **Grabar resultados en fichero** del **Diálogo de Consulta**.



Nos pide el nombre del fichero y la ubicación.



x **Eliminar**: Se va a eliminar el nodo cuyo número de departamento es el 10. Utilizamos el documento departamentos.xml. Se trata de generar una consulta que obtenga todos los nodos del documento

menos el nodo con ese número de departamento. Utilizaremos where para no seleccionar el departamento 10.

```
(: obtenemos todos los DEP_ROW salvo el que tenga DEPT_NO = 10
:)
let $titulo := /departamentos/TITULO return
<departamentos>{$titulo}
               {for $dep in (/departamentos/DEP_ROW)
               where data($dep/DEPT_NO) != 10
                                    return $dep}
          </departamentos>
```

x **Actualizar**: En la siguiente consulta se va a actualizar el departamento cuyo número de departamento es el 20. Vamos a cambiar la localidad (LOC) por GRANADA.

```
(: modificar la localidad del DEPT_NO = 20 :)
let $titulo := /departamentos/TITULO (:
recuperamos el nombre del DEPT_NO = 20 :)
let $dnom := /departamentos/DEP_ROW[DEPT_NO =
20]/DNOMBRE/text()
```

```
(: creamos un nodo con los nuevos valores :) let
$modif :=
<DEP_ROW><DEPT_NO>20</DEPT_NO><DNOMBRE>{$dnom}</DNOMBRE><LOC>GRA
NADA</LOC></DEP_ROW>
return <departamentos>{$titulo}
               {for $dep in (/departamentos/DEP_ROW)
               return if($dep/DEPT_NO/text()= 20)
                                    then $modif
               else $dep
          }
          </departamentos>
```

Lo que hacemos es recuperar los datos del nodo a modificar, para posteriormente insertar ese nodo modificado, sacamos el nombre del departamento, el número y la localidad no se recuperan porque no se necesitan. Después de crea el nodo *$modif* con todas las etiquetas; de forma que cuando se van generando las etiquetas con los departamentos al llegar al departamento a modificar (20) se devuelve el nodo *$modif*, y si no es el 20 se visualiza *$dep*.

Actualizar los salarios de los empleados del departamento 10, se les sube a todos la cantidad de 200.

```
(: aumentar 200 € el salario de los empleados del departamento
20 :)
for $emp in /EMPLEADOS/EMP_ROW
let $nemp := $emp/EMP_NO, $ape := $emp/APELLIDO, $ofi :=
$emp/OFICIO, $dir := $emp/DIR
let $fec := $emp/FECHA_ALT, $dep := $emp/DEPT_NO, $sal :=
$emp/SALARIO
let $salnu := number($emp/SALARIO/text())+200
return if($dep/text() = 20)
                then <EMP_ROW>{$nemp, $ape, $ofi, $dir,
$fec}<SALARIO>{$salnu}</SALARIO>{$dep}</EMP_ROW>
                else $emp
```

Lo que hacemos es recuperar todos los nodos de empleados a variables, de forma que si el empleado es del departamento 10, se crea un nuevo nodo <EMP_ROW> con los datos de sus nodos y el salario actualizado. Si el empleado no es del departamento 10 se devuelve el nodo EMP_ROW leído.

Igual que la consulta anterior, pero la salida es al documento empleados.xml con el salario actualizado

```
let $titulo := /EMPLEADOS/TITULO return
<EMPLEADOS>
          {$titulo}
          {
            for $emp in /EMPLEADOS/EMP_ROW
               let $nemp := $emp/EMP_NO, $ape := $emp/APELLIDO,
$ofi := $emp/OFICIO, $dir := $emp/DIR
               let $fec := $emp/FECHA_ALT, $dep := $emp/DEPT_NO,
$sal := $emp/SALARIO
               let $salnu := number($emp/SALARIO/text())+200
               return
                    if($dep/text() = 10)
                         then <EMP_ROW>{$nemp, $ape, $ofi, $dir,
$fec}<SALARIO>{$salnu}</SALARIO>{$dep}</EMP_ROW>
               else $emp
          }
     </EMPLEADOS>
```

Sentencias de actualización eXist

Estas sentencias permiten hacer altas, bajas y modificaciones de nodos y elementos en documentos XML. Se pueden usar las sentencias de actualización en cualquier punto pero si se utiliza en la cláusula RETURN de una sentencia FLWOR, el efecto de la actualización es inmediato. Todas las sentencias de actualización comienzan con la palabra UPDATE y a continuación la instrucción.

Las instrucciones son:

x **insert**: se utiliza para insertar nodos. El lugar de inserción se especifica:

③ **into**: el contenido se añade como último hijo de los nodos especificados.

③ **following**: el contenido se añade inmediatamente después de los nodos especificados. ③ **precedind**: el contenido se añade antes de los nodos especificados El formato es:

update insert ELEMENTO into EXPRESION update

insert ELEMENTO following EXPRESION update

insert ELEMENTO preceding EXPRESION

Ejemplos: en el fichero *zonas.xml* colección *Pruebas*

Inserta una zona en *zonas.xml* en la última posición

```
(: insertar una zona en la última posición :) update
insert <zona>
     <cod_zona>50</cod_zona>
     <nombre>Galicia</nombre>
     <director>Jose Pereira</director>
    </zona>
    into /zonas
```

Inserta una cuenta en el documento *sucursales .xml* del tipo PENSIONES a la SUC1.

```
(: insertar una cuenta en el documento sucursales.xml del
tipo PENSIONES a la sucursal SUCI :) update insert
     <cuenta tipo = "PENSIONES">
          <nombre>Alberto Pérez</nombre>
     <numero>30311478</numero>
     <aportacion>5000</aportacion>
     </cuenta>
into /sucursales/sucursal[@codigo="SUC1"]
```

Inserta en el documento departamentos de la BD los nodos DEP_ROW del documento externo departamentosinsertar.xml

```
(: Inserta en el documento departamentos de la BD los nodos DEP_ROW
del documento externo departamentosinsertar.xml  :) for $dep in
doc('file://C:\Users\Usuario\Documents\CSDAMultiplataforma\AccesoDa
tosActual\BDeXist\ColeccionPruebas\departamentosinsertar.xml')/depa
rtamentosinsertar/DEP_ROW
return update insert $dep into /departamentos
```

x **replace**: sustituye el nodo especificado en NODO con VALOR_NUEVO. NODO debe devolver un único ítem: si es u elemento, VALOR_NUEVO debe ser también un elemento. Si es un nodo de texto o atributo su valor será actualizado con la concatenación de todos los valores de VALOR_NUEVO.

Formato: **update replace NODO with VALOR_NUEVO**

Cambia la etiqueta director de la zona 50 y su contenido, en el documento *zonas.xml*.

```
update replace
/zonas/zona[cod_zona = 50] /director with
<directora>Pilar Manteiga</directora>
```

Cambia el nodo completo DEP_ROW del departamento 10, por los nuevos datos y las etiquetas que escribamos

```
update replace /departamentos/DEP_ROW[DEPT_NO = 10]
with <DEP_ROW>
     <DEPT_NO>10</DEPT_NO>
     <DNOMBRE>Nuevo 10</DNOMBRE>
     <LOC>Nueva Localidad</LOC>
</DEP_ROW>
```

x **value**: actualiza el valor del nodo especificado en NODO con VALOR_NUEVO. Si NODO es un nodo de texto ó atributo su valor será actualizado con la concatenación de todos los valores de VALOR_NUEVO.

Formato: **`update value NODO with VALOR_NUEVO`**

Cambia el apellido del empleado 7369 del documento *empleados.xml*

```
update value /EMPLEADOS/EMP_ROW[EMP_NO = 7369]/APELLIDO with
'Alberto García Pereira'
```

Cambia el atributo tipo de la primera cuenta de la sucursal SUC3, del documento *sucursales.xml*

```
update value /sucursales/sucursal [@codigo =
'SUC3']/cuenta[1]/@tipo with
'NUEVOTIPO'
```

Cambia el salario, aumentando en 200, de los empleados del departamento 10, del documento *empleados.xml*

```
for $emp in /EMPLEADOS/EMP_ROW[DEPT_NO =
10] let $sal := $emp/SALARIO return update
value $emp/SALARIO            with
data($sal)+200
```

x **delete**: elimina los nodos indicados en la expresión.

Formato: **`update delete expresión`**

Elimina la zona con código 50, en el documento *zonas.xml*

```
update delete /zonas/zona[cod_zona = 50]
```

x **rename**: renombra los nodos devueltos en NODO (debe devolver una relación de nodos o atributos) por el NUEVO_NOMBRE

Formato: **`update rename NODO as NUEVO_NOMBRE`**

Cambia el nombre del nodo EMP_ROW del documento *empleados.xml*

```
update rename /EMPLEADOS/EMP_ROW as 'fila_emple'
```