el nombre de los departamentos de tipo B y con 2 o más empleados. Es lo mismo que poner `/universidad/departamento[@tipo="B" and count(empleado)>=2]/nombre/text()`.

5.3.4 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, en la Figura 5.15 se ve el procesamiento básico de una consulta XQUERY. XQuery contiene a XPath, toda expresión de consulta en XPath es válida y devuelve el mismo resultado en XQuery. Xquery nos va a permitir:

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

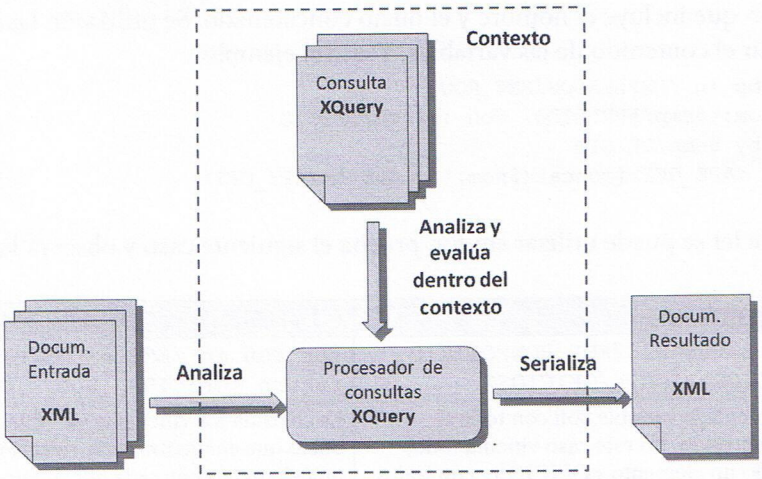

Figura 5.15. Procesamiento de una consulta XQuery.

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 esta:

```
for <variable> in <expresión XPath>
let <variables vinculadas>
where <condición XPath>
order by <expresión>
return <expresión de salida>
```

- **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 lo que queremos que nos devuelva la consulta. Por ejemplo estas consultas devuelve la primera los elementos EMP_ROW y la segunda los apellidos de los empleados. Unas escritas en XQuery y las otras en XPath:

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

- **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.

  En el siguiente 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 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. Véase el ejemplo:

```
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, prueba el siguiente caso y observa la diferencia:

SIN FOR	CON FOR
`let $ofi := /EMPLEADOS/EMP_ROW/OFICIO return <OFICIOS>{$ofi}</OFICIOS>`	`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 ellos.	La cláusula for vincula la variable $ofi con cada nodo oficio que encuentre en la colección de datos, creando una elemento por cada oficio. Por eso aparece la etiqueta <OFICIOS> para cada oficio.

- **Where:** filtra los elementos, eliminando todos los valores que no cumplan las condiciones dadas.
- **Order by:** ordena los datos según el criterio dado.
- **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 ***condicionales 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 ().

El siguiente ejemplo devuelve los departamentos de tipo A encerrados en una etiqueta:

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

Utilizaremos la función **data()** para extraer el contenido en texto de los elementos. También se utiliza data() para extraer el contenido de los atributos p.ej. esta consulta XPath **`//empleado/@salario`** es errónea, pues salario no es un nodo, pero esta otra consulta **`data(//empleado/@salario)`** devuelve los salarios.

Dentro de las asignaciones **`let`** en las consultas XQuery podremos 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:

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

`for $de in doc('file:///D:/XML /prue-`
`baxquery/NUEVOS_DEP.xml')/NUEVOS_DEP/`
`DEP_ROW return $de`
	Devuelve los nodos DEP_ROW de un documento ubicado en una carpeta del disco duro.

`for $prof in /universidad/departa-`
`mento[@tipo='A']/empleado`
`let $profe:= $prof/nombre,`
`$puesto:= $prof/puesto`
`where $puesto='Profesor'`
`return $profe`
	Obtiene los nombres de empleados de los departamentos de tipo A, cuyo puesto es Profesor. Esto hace lo mismo:
	
	`for $prof in /universidad/departa-`
	`mento[@tipo='A']/empleado`
	`where $prof/puesto='Profesor'`
	`return $prof/nombre`
	
	El resultado es:
	
	<nombre>Alicia Martín</nombre>
	<nombre>Mª Jesús Ramos</nombre>
	<nombre>Pedro Paniagua</nombre>

`for $dep in /universidad/departamento`
`return if ($dep/@tipo='A')`
`then <tipoA>{data($dep/nombre)}</tipoA>`
`else <tipoB>{data($dep/nombre)}</tipoB>`
	Devuelve el nombre de departamento encerrado entre las etiquetas <tipoA></tipoA>, si es del tipo = A, y <tipoB></tipoB>, si no lo es.
	
	`<tipoA>Informática</tipoA>`
	`<tipoA>Matemáticas</tipoA>`
	`<tipoB>Análisis</tipoB>`

`for $dep in /universidad/departamento`
`let $nom:= $dep/empleado`
`return <depart>{data($dep/nombre)}`
`<emple>{count($nom)}</emple></depart>`
	Obtiene los nombres de departamento y los empleados que tiene entre etiquetas:
	
	`<depart>Informática<emple>2</emple></depart>`
	`<depart>Matemáticas<emple>4</emple></depart>`
	`<depart>Análisis<emple>2</emple></depart>`

`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>`
	Obtiene los nombres de departamento, los empleados que tiene y la media del salario entre etiquetas:
	
	`<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>`

5.3.5 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 se muestran en la siguiente tabla.

- 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(), avg(), sum().
- 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 elemento.
- Los comentarios en XQuery van encerrados entre caras sonrientes: **(:** Esto es un comentario **:)**

Ejemplos utilizando el documento EMPLEADOS.xml:

- Los nombres de oficio que empiezan por P.

```
for $ofi in /EMPLEADOS/EMP_ROW/OFICIO
where starts-with(data($ofi),'P')
return $ofi
```

SALIDA:
```
<OFICIO>PRESIDENTE</OFICIO>
```

- Obtiene los nombres de oficio y los empleados de cada oficio. Utiliza la función distinct-values para devolver los distintos oficios.

```
for $ofi in distinct-values(/EMPLEADOS/EMP_ROW/OFICIO)
let $cu:=count(/EMPLEADOS/EMP_ROW[OFICIO =$ofi])
return concat($ofi,' = ',$cu)
```

SALIDA:
```
EMPLEADO = 4
VENDEDOR = 4
DIRECTOR = 3
ANALISTA = 2
PRESIDENTE = 1
```

- Obtiene el número de empleados que tiene cada departamento y la media de salario redondeada:

```
for $dep in distinct-values(/EMPLEADOS/EMP_ROW/DEPT_NO)
let $cu:=count(/EMPLEADOS/EMP_ROW[DEPT_NO =$dep])
let $sala:= round(avg(/EMPLEADOS/EMP_ROW[DEPT_NO =$dep]/SALARIO))
return concat('Departamento: ',$dep,'. Num emples = ',$cu, '. Media salario =
',$sala)
```

SALIDA:
```
Departamento: 20. Num emples = 5. Media salario = 2274
Departamento: 30. Num emples = 6. Media salario = 1736
Departamento: 10. Num emples = 3. Media salario = 2892
```

Si se desea devolver el resultado entre etiquetas pondremos en el return (p.ej):

```
return <depart><cod>{$dep}</cod><emples>{$cu}</emples><medsal>{$sala}</medsal></
depart>
```

```
<depart><cod>20</cod><emples>5</emples><medsal>2274</medsal></depart>
<depart><cod>30</cod><emples>6</emples><medsal>1736</medsal></depart>
<depart><cod>10</cod><emples>3</emples><medsal>2892</medsal></depart>
```

Actividad 3: Utilizando el documento ***productos.xml.*** Realiza las siguientes consultas XQuery:

- Obtén por cada zona el número de productos que tiene.
- Obtén la denominación de los productos entres las etiquetas <zona10></zona10> si son del código de zona 10, <zona20></zona20> si son de la zona 20, <zona30></zona30> si son de la 30 y <zona40></zona40> si son de la 40.
- Obtén por cada zona la denominación del o de los productos más caros.
- Obtén la denominación de los productos contenida entre las etiquetas <placa></placa> para los productos en cuya denominación aparece la palabra *Placa Base*, <memoria></memoria>, para los que contienen a la palabra *Memoria* <micro></micro>, para los que contienen la palabra *Micro* y <otros></otros> para el resto de productos.

Utilizando el documento ***sucursales.xml.*** Realiza las siguientes consultas XQuery:

- Devuelve el código de sucursal y el número de cuentas que tiene de tipo AHORRO y de tipo PENSIONES
- Devuelve por cada sucursal el código de sucursal, el director, la población, la suma del total debe y la suma del total haber de sus cuentas.
- Devuelve el nombre de los directores, el código de sucursal y la población de las sucursales con más de 3 cuentas.
- Devuelve por cada sucursal , el código de sucursal y los datos de las cuentas con más saldo debe.
- Devuelve la cuenta del tipo PENSIONES que ha hecho más aportación.

5.3.6 CONSULTAS COMPLEJAS CON XQUERY

Dentro de las consultas XQuery podremos trabajar con varios documentos xml para extraer su información, podremos incluir tantas sentencias for como se deseen, 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. A continuación se muestran ejemplos de diversa complejidad:

- **Joins de documentos.**

  - 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 $emple:= $emp/APELLIDO
let $dep:= $emp/DEPT_NO
let $dnom:= (/departamentos/DEP_ROW[DEPT_NO =$dep]/DNOMBRE)
return <res>{$emple, $dep, $dnom} </res>
```