**Xpath-xquery**

Hay que descargar en casa basex . aplicación gratuita

Para iniciar los ficheros: DatabaseàOpen&ManageàOpen(abajo izquierda de la ventana emergente).

Es un servicio para hacer consultas y modificaciones de documentos xml.

Diferencia entre mayúsculas y minúsculas

Cuando escribimos una barra (/) en las consultas vamos accediendo a los diferentes niveles de xml, es decir, es como una jerarquía de carpetas, por cada / pasamos a un nivel interior. Esto te interesa cuando quieres hacer una consulta cuyos datos quieres que pertenezcan a ese campo determinado antes de pasar al siguiente.

Por ejemplo, buscar los valores máximos de cada categoría.

Para hacer varias consultas simultaneas e independientes entre sí podemos usar la coma(,) o la barra vertical (|).

Para buscar por atributo debemos poner un @antes del nombre del atributo y su valor

Ej: //cuenta[@tipo="AHORRO"]

La doble barra (//) realiza una búsqueda recursiva, se salta el ir yendo de nivel a nivel y va a directamente buscando en todos los elementos. Te muestra todo, aunque esté a diferentes niveles, incluso si está en diferentes bibliotecas. Es como un atajo

El operador .. es para subir al padre

El operador . es para referirse al actual

Para hacer filtrados en la búsqueda se utilizan los []. Para seleccionar el valor de la búsqueda. Es como poner condicionales en la búsqueda. Ej: dentro de las suscursales la que cumplan la condición de tener tipo ahorro

Operadores de filtrado con los caracteres siguientes: >,<,!=(distinto), >=,<=, and, or.

(: texto :) para escribir comentarios

Funciones exists-xpath

* **Data**

**Te da en dato del nodo seleccionado**

* **Number**: Permite sacar valor de tipo texto y convertirlo a numérico.

concat(string(333), 'hola')

//produc/concat(concat('Articulo:',denominacion),concat('€',precio))

* **String**: convierte un numero a cadena de texto.

substring(cadena,posicion)

substring(//produc[cod\_prod=1010]/denominacion,4) -> Un parámetro

substring(//EMP\_ROW[EMP\_NO='7369']/OFICIO,1,4) -> Dos parámetros

* **Upper-case/Lower-case**: Para mostrar el resultado en mayúsculas o en minúsculas .

upper-case(cadena)

//produc/upper-case(denominacion)

Primer en mayúsculas y resto en minúsculas:

concat(substring(//EMP\_ROW[EMP\_NO='7369']/APELLIDO,1,1),lower-case(substring(//EMP\_ROW[EMP\_NO='7369']/APELLIDO,2)))

Visualizar la denominación de un producto junto con el precio multiplicado 1.21

//produc/concat(string(denominacion),precio\*1.21)

//produc/concat(denominacion,' ',number(precio\*1.21))

* **Contains:**devuelve un verdadero si contiene una cadena de texto concreta

//produc[contains(denominacion,'i5')]

//EMP\_ROW[contains(APELLIDO,'A')]

* **Starts-with:** devuelve true o false si empieza por una determinada cadena de texto
* **Concat:** para unir cadenas de texto ()

FUNCIONES DE AGRUPACIÓN siempre acompañado de paréntesis **nombre(…)**

**Avg**= significa average (media aritmética)count

Ej: avg (//produc]cod\_zona=”20”]/precio[

//produc[precio>avg(//precio)]/concat('articulo: ',denominacion)

La consulta es sacar los listados de precio superior a la media y su campo articulo con la palabra denominación.

**Count:** para contar el numero de entradas que cumplen las condiciones. Siempre da numero entero positivo

count(//produc[precio>avg(//precio)]/concat('Articulo: ',denominacion))

Mostrar la cantidad de artículos de aquellos productos que superen la media

-funcion max min: muestra la denominación del producto más caro o más barato

- Distinct values: mostrar los valores no repetidos dentro de un conjunto.

distinct-values(//product/concat (código: cod\_zona))

-Sum: suma de los valores numéricos de los elementos

Ejercicio:

Mostrar la denominación de los productos que están bajo mínimos, es decir , su stock actual es inferior al mínimo.

//produc[number(stock\_actual)<number(stock\_minimo)]/concat("producto:",denominacion)

//produc[number(stock\_actual)-number(stock\_minimo)<0]/concat("producto:",denominacion)

Consultas cruzadas

Se trata de cualquier consulta que se ejecuta sobre todos los xml que hay en la misma colección , de eso nos podemos aprovechar para hacer subconsultas.

Zonas xml y productos xml

//zona[cod\_zona=//produc[stock\_actual=max(//stock\_actual)]/cod\_zona]/director

Ej: de la zona 10 quiero saber su nombre y la suma de todos los stocks actual y mínimo de esa zona

//zona[cod\_zona=10]sum stock\_actual,[min stock\_actual]

concat(sum(//produc[cod\_zona=10]/stock\_actual),"-", sum(//produc[cod\_zona=10]/stock\_minimo),"-",//zona[cod\_zona=10]/nombre)

mostrar el director de zona, en donde se venden productos de tarjeta o del tipo memoria

//zona director

//zona[cod\_zona=distinct-values(//produc[contains(denominacion,'Memoria') || contains(denominacion,'Tarjeta') ]/cod\_zona)]/director

Denominación del producto (de cualquier zona) cuyo precio supera la media de los precios de la zona 20

//denominacion (precio>avg(precio[cod\_zona=20]))

//produc[precio>avg(//produc[cod\_zona=20]/precio)]/denominación

Xpath

for <variable> in <expresión XPath>

let <variables vinculadas>

where <condición XPath>

order by <expresión>

return <expresión de salida>

Es sql para xml. Sirve para hacer consultas dentro de un texto plano

for $productos in //produc

let $stock\_real:=($productos/stock\_actual)-($productos/stock\_minimo)

where $productos/cod\_zona=10

order by $productos/precio

return <stock\_real>{$stock\_real}</stock\_real>

Siempre sigue el mismo protocolo for- let-where- order by- return

For es un bucle donde se busca una variable (que defines tú y que es el equivalente a i en los bucles for de programacion), ahí defines la ruta a seguir para tener esa variable.

Ejemplo si tenemos una carpeta llamada alumnos dentro de la carpeta asignatura dentro de la carpeta curso, para llamar a la variable alumnos tendríamos que escribir:

For $alumnos in //curso/asignatura/alumnos

Let es para poner el resto de variables que van a intervenir en la operación. En este apartado siempre se pone := en vez de in.

Para el mismo supuesto de antes sería

Let $alumnos := //curso/asignatura/alumnos

Pero si por ejemplo queremos hacer algo más complicado que nombrar una variable, tenemos que tener en cuenta algunas cosas:

1.-nombrar la variable (con dólar)

2.- operador morsa(:=)

3.-funcionas (sum, count, concate, etc. Estas siempre abren parentesis)

4.-si queremos buscar dentro de un atributo tenemos que poner antes de su nombre@= al valor que queremos que tenga ese atributo encerrado entre comilla porque es una referencia textual

5.-si queremos buscar algo en específico, esto tiene que ir entre corchetes[].

6. Si queremos añadir varias clausulas se añaden con comas

Where la condición que tiene que cumplir las variables

Order by , el orden en el que aperecera en pantalla

Y return lo que va a aparecer por pantalla

Para definir una variable en x-query se pone antes un símbolo de dólar ($), es un atajo. Al nombrarlo tienes que poner la ruta real de esa variable (y no volvela a poner más)

Si ponemos un if para declarar una condición, a continuación, hay que poner un then para definir que va a apasar en caso de que se cumple la anterior condición

El concat solo es posible si se une un elemento con otro elemento. Si intento unir un elemento con otros 4, nos dará error. Aunque si puedes hacer un elemento con otro elemento, con otro elemento. Pero cada elemento debe de ser una unidad no puede ser una consulta que de como resultado un grupo de más de un elemento.

Mostrar los apellidos y el puesto de cada empleado

for $emp in /EMPLEADOS/EMP\_ROW

let $nom:=$emp/APELLIDO, $ofi:=$emp/OFICIO

order by $emp/OFICIO

return <APEOFI>{concat($nom,' ',$ofi)}</APEOFI>

En xpath sería:

/EMPLEADOS/EMP\_ROW/concat("<APEOFI>",APELLIDO," ",OFICIO,"</APEOFI>")

Sobre la consulta anterior, experimentaremos con la parte del return

return <APEOFI>{concat($nom,' ',$ofi)}</APEOFI>

return $nom

return data($nom)

En el return se puede meter una sentencia fi como muestra el siguiente ejemplo

for $dep in /universidad/departamento

return if ($dep/@tipo='A')

then <tipoA>{data($dep/nombre)}</tipoA>

else ()

(:where $productos/cod\_zona=10:)

order by $productos/precio

return if ($productos/cod\_zona=10)

then <stock\_real>{$stock\_real}</stock\_real

for $dep in /universidad/departamento

return if ($dep/@tipo='A' )

then <tipoA>{data($dep/nombre)}</tipoA>

else if ($dep/@tipo='B')

then <tipoB>{data($dep/nombre)}</tipoB>

else ()

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>

Para la consulta, mostrar los nombres de los profesores del departamento A, el resultado sería;

for $prof in /universidad/departamento[@tipo='A']/empleado

let $profe:=$prof/nombre, $puesto:=$prof/puesto

(:where $puesto='Profesor'

return $profe:)

return if ($puesto='Profesor')

then $profe

Mostrar el nombre de los departamentos que son del tipo A y tipo B en dos nodos diferenciados: <tipoA> <tipoB> el resultado sería:

for $dep in /universidad/departamento

return if ($dep/@tipo='A')

then <tipoA>{data($dep/nombre)}</tipoA>

else <tipoB>{data($dep/nombre)}</tipoB>

sería:

for $dep in /universidad/departamento

let $nom:=$dep/empleado

return <depart>{data($dep/nombre)} </depart> |

<emple>{count($nom)}</emple>

Mostrar el nombre de departamento y número de empleados que tiene por cada uno de ellos con el siguiente formato:

<depart>Informática</depart>

<emple>2</emple>

<depart>Matemáticas</depart>

<emple>4</emple>

for $dept in /universidad/departamento

let $nom:=$dept/nombre, $num:=count($dept/empleado)

return (<depart>{data($nom)}</depart>, <emple>{data($num)}</emple>)

for $dep in //departamento

let $nom:=$dep/empleado

return <departamento>

<depart>{data($dep/nombre)} </depart>

<emple>{count($nom)}</emple>

</departamento>

Consultas avanzadas

Para sacar el apellido del empleado el numero de departamento y su nombre.

for $emp in (/EMPLEADOS/EMP\_ROW)

let $emple:= $emp/APELLIDO, $dep:= $emp/DEPT\_NO, $dnom:= (/departamentos/DEP\_ROW[DEPT\_NO =$dep]/DNOMBRE)

return <res>{$emple, $dep, $dnom} </res>

Para sacar el nombre del departamento , la suma de los salarios de los empleados y el total de empleados de ese departamento

for $dep in /departamentos/DEP\_ROW

let $d:=$dep/DEPT\_NO, $tot:=sum(/EMPLEADOS/EMP\_ROW[DEPT\_NO=$d]/SALARIO), $cu:=count(/EMPLEADOS/EMP\_ROW[DEPT\_NO=$d])

return <resul>{$dep/DNOMBRE}<sumsalario>{$tot}</sumsalario><totemple>{$cu}</totemple></resul>

Convertir la salida de la consulta anterior, de manera que el total salario, y el total empleados, sean atributos de cada departamento:

for $dep in /departamentos/DEP\_ROW

let $d:=$dep/DEPT\_NO, $tot:=sum(/EMPLEADOS/EMP\_ROW[DEPT\_NO=$d]/SALARIO), $cu:=count(/EMPLEADOS/EMP\_ROW[DEPT\_NO=$d])

return <resul>{$dep/DNOMBRE}<sumsalario>{$tot}</sumsalario><totemple>{$cu}</totemple></resul>

for $emp in //EMP\_ROW

let $d:=$emp/DEPT\_NO, $nom:=$emp/APELLIDO, $sal:=$emp/SALARIO, $ndep:=(/ /DEP\_ROW[DEPT\_NO=$d]/DNOMBRE), $salmax:= max(//EMP\_ROW[DEPT\_NO=$d]/SALARIO)

return

if ($sal=$salmax)

then

<depart>{data($ndep)}<salmax>{data($sal)}</salmax><emple>{data($nom)}</emple></depart>

else ()

for $dep in //DEP\_ROW

let $num\_dep:= $dep/DEPT\_NO

let $dnom:= $dep/DNOMBRE

let $numEmp:= max(//EMP\_ROW[DEPT\_NO=$num\_dep]/SALARIO)

let $nomEmp:= //EMP\_ROW[DEPT\_NO=$num\_dep and SALARIO=$numEmp]/APELLIDO

return <dato>{$dnom,$nomEmp,$numEmp}</dato

¿Cuántos empleados por cada puesto en cada departamento?

|  |  |
| --- | --- |
| for $dep in /universidad/departamento  for $pue in distinct-values($dep/empleado/puesto)  let $cu:=count($dep/empleado[puesto=$pue])  return <departamentos><depart>{data($dep/nombre)}  </depart><puesto>{data($pue)}</puesto><profes>  {$cu}</profes></departamentos> | primer bucle for (declarando primera variable)  segundo bucle for(declarando variable) |

Para interpretar la consulta como una tabla con columnas dentro de html

|  |  |
| --- | --- |
| <HTML>  <BODY>  <table>  {  for $dept in (//DEP\_ROW)  let $nodept:=$dept/DEPT\_NO , $nomdep:=$dept/DNOMBRE  for $emple in (//EMP\_ROW[DEPT\_NO=$nodept])  let $empleado:=$emple/APELLIDO  return <tr><td>{data($nodept)}</td><td>{data  ($nomdep)}</td><td>{data($empleado)}</td></tr>  }  </table>  </BODY>  </HTML> | Por cada bucle de departamento  Recuperas número de departamento y nombre  Haces el nexo de unión para que no te haga el producto cartesiano  Y en el return me lo muestras organizándolo por etiquetas de tabla  Cierralo las etiquetas de tabla |

Para consultar los empleados cuyo sueldo sea el superior dentro de su departamento

|  |  |
| --- | --- |
| for $dept in (//DEP\_ROW)  let $nodept:= $dept/DEPT\_NO  let $nomdep:= $dept/DNOMBRE  let $emplecaro:=//EMP\_ROW[SALARIO=max(//EMP\_ROW  [DEPT\_NO=$nodept]/SALARIO) and DEPT\_NO=$nodept]/APELLIDO  return  <res>  <dept>{data($nomdep)}</dept>  <nodept>{data($nodept)}</nodept>  <empleados>  {for $apellido in $emplecaro  return concat($apellido,' , ')  }  </empleados>  </res> | Defines las variables  Condiciones a cumplir  Por cada empleado con el sueldo mas alto encadenas su apellido y punto |

Para consultar los empleados cuyo sueldo es mayor a la media de su departamento

|  |  |
| --- | --- |
| for $depar in distinct-values(//DEPT\_NO)  let $emple:= //EMP\_ROW[DEPT\_NO=$depar]  let $nom:= data(//DEP\_ROW[DEPT\_NO=$depar]/DNOMBRE)  let $media:= avg($emple/SALARIO)  let $emplemed:= $emple[SALARIO>=$media]  return <emp><depart>{$nom}</depart>  <empleados>{for $apellido in $emplemed/APELLIDO  return concat($apellido,' ')}</empleados>  </emp> |  |

**Modificaciones de xml**

update insert ELEMENTO into EXPRESION (dentro del nodo al final)

update insert ELEMENTO following EXPRESION (inserta un nodo justo en el nodo posterior)

update insert ELEMENTO preceding EXPRESION (inserta un nodo justo en el nodo anterior)

update insert attribute ELEMENTO into EXPRESION

ejemplo:

|  |  |
| --- | --- |
| update insert  <zona><codzona>50</codzona>  <nombre>Madrid-OESTE </nombre>  <director>Alicia Ramos Martin</director> </zona> into /zonas |  |
| update insert  <zona><codzona>50</codzona>  <nombre>Madrid-OESTE </nombre>  <director>Alicia Ramos Martin</director> </zona> preceding //zona[cod\_zona=40] |  |
| update insert  <cuenta tipo="PENSIONES"><nombre>Alberto Morales  </nombre><numero>30302900</numero>  <aportacion>5000</aportacion></cuenta> into /sucursales/sucursal[@codigo="SUC1"] |  |
|  |  |

Replace: reemplazar un nodo=> update replace NODO with VALOR NUEVO

|  |  |
| --- | --- |
| Ej.:  update replace  /zonas/zona[codzona=50]/director  with <directora>Pilar Martin Ramos</directora> |  |
| update replace /departamentos/DEP\_ROW[DEPT\_NO=10]  with <DEP\_ROW><DEPT\_NO>10</DEPT\_NO><DNOMBRE>NUEVO10</DNOMBRE> <LOC>TALAVERA</LOC></DEP\_ROW> |  |
|  |  |
|  |  |

**update value NODO with 'VALOR NUEVO'**

|  |  |
| --- | --- |
| update value /sucursales/sucursal[@codigo='SUC3']/cuenta[1]/@tipo  with 'NUEVOTIPO' |  |
| update value /sucursales/sucursal[@codigo='SUC3']/cuenta[@tipo="PENSIONES"]/aportacion  with 5555 |  |
| for $em in /EMPLEADOS/EMP\_ROW[DEPT\_NO=10]  let $sal:=$em/SALARIO  return update value $em/SALARIO with data($sal)+200 |  |
| for $em in /EMPLEADOS/EMP\_ROW[DEPT\_NO=10]  let $sal:=$em/SALARIO  return update value $sal with $sal+200 |  |
| for $em in /EMPLEADOS/EMP\_ROW[DEPT\_NO=10]  let $sal:=$em/SALARIO/text()  return update value $em/SALARIO with $sal+200 |  |

Delete

|  |  |
| --- | --- |
| update delete //zona[codzona=50]  Otro ejemplo:  for $dep in /zonas/zona/cod\_zona  return  update delete $dep |  |
|  |  |
|  |  |
|  |  |
|  |  |

Para modificar los datos de manera segura creando un punto de control al que podamos volver, seleccionamos todo el texto control alt, control c para crearlo

Control a, control v para pegar ese punto de control.

rename

|  |  |
| --- | --- |
| update rename /EMPLEADOS/EMP\_ROW as 'fila\_emple' |  |
|  |  |
|  |  |
|  |  |
|  |  |

Actividad: Inserta <nuevosal> en /EMPLEADOS/EMP\_ROW[DEPT\_NO=20]

for $empleado in /EMPLEADOS/EMP\_ROW [DEPT\_NO=20]

let $empleado := $empleado/SALARIO

return update insert<nuevosalario>{$salario\*1.1}</nuevosalario> into $empleado

para borrar ese nodo

update delete /nuevosalario