**DIA 1**

/ -> de nivel en nivel

// -> buscar lo que sea, recursivamente

//departamentos/DEP\_ROW/DEPT\_NO/LOC = //LOC

Para hacer comentarios con (\* ... \*)

/.. -> sacar los nodos padre

/. -> sacar el nodo actuar

//LOC/.. = //departamentos/DEP\_ROW/DEPT\_NO/.

FILTRADO

El filtrado se hace entre corchetes

Y para comparar, añadir varios filtrados utilizaremos los caracteres que normalmente usamos en otros lenguajes(sql):

=,>,<,!=,>=,<=, and, or o ||

//DEP\_ROW[DEPT\_NO>20]

//DEP\_ROW[DEPT\_NO<40]

//DEP\_ROW[DEPT\_NO>20 and DEPT\_NO<40]

Ejercicio ejemplo

(:Saca el numero de los departamentos que estan en Sevilla:)

//DEP\_ROW[LOC="SEVILLA"]/DEPT\_NO

(:Saca el numero de los departamentos que estan en Sevilla y Bilbao:)

//DEP\_ROW[LOC="SEVILLA" or LOC="BILBAO"]/DEPT\_NO

Para hacer referencia a una etiqueta es con @(nombre del atributo)

//libro[@anyo=1994]/titulo

El @ tambien nos sirve para sacar para sacar el atributo

//libro/@anyo

Otros

count: nos sirve para contar el numero de datos

count(//libro[precio>50.00])

number: nos permite pasar de valor cadena a numerico

number('33') div 3

concat: para pasar de valor numerico a cadena

concat(string(333), 'hola')

substring: para sacar parte de las cadena

//substring(titulo,1,3)

upper-case: para pasar todos los caracteres a mayusculas

//upper-case(titulo)

lower-case: hace lo mismo que el upper-case, pero a minusculas

//lower-case(titulo)

contains: para sacar los que tienen parte de lo que le pidamos, eso si devolvera true y false segun lo tenga o no

//contains(titulo, "TCP")

Ejercicio ejemplo

(:Sacar el nombre y apellidos del autor del libro que tenga TCP:)

//libro[contains(titulo, "TCP")]/concat(nombre," ", apellido)

**DIA 2**

La **función sum** sirve para sumar elementos

sum(//salario)

La **funcion count** sirve para contar elementos

count(//salario)

La **funcion avg** sirve para sacar la media

avg(//salario)

La funcion max sirve coger el salarario maximo

max(//salario)

La funcion min sirve coger el salarario minimo

min(//salario)

La funcion distinct-values sirve para que no se repitan valores

distinct-values(//produc/concat('La denominacion es ', denominacion, 'y su precio es de ', precio))

ejercicios

Mostrar la denominación de los productos cuyo stock\_minimo sea igual o superior al actual.

**//produc[number(stock\_minimo >= stock\_actual)]**

**Mostrar la denominación (sin etiquetas XML), de aquellos productos de las zonas 10, 20 o 30 cuyo precio supera los 120€.**

**//produc[number(cod\_zona)>=10 and number (cod\_zona)<=30 and number(precio) > 120]/concat('' , denominacion)**

Mostrar la denominación de productos cuyo precio supera la media de todos precios

**//produc[number(precio) > avg(//precio)]/denominación**

Mostrar la denominación de producto con la siguiente sintaxis: "El producto más caro es: " , para aquel producto que más vale.

**//produc [precio=max(//precio)]/concat('El producto mas caro es ', denominacion)**

- 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)]/denominación**

**DIA 3**

El corchete es como ‘cuyo precio es…’

De la zona 10 saber la suma y el stock mínimo y stock actual

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

**Mostrar el director de zona , en donde se venden productos de tarjeta o del tipo memoria.**

//zona[contains(cod\_zona, distinct-values(//produc[contains(denominacion, "Tarjeta") or contains(denominacion, "Memoria")]/cod\_zona))]/director

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

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

Sacar los nodos atributos es con @

//departamento[@telefono =112233 or @telefono =990033]

Sacar el max y min del salario

concat('Max: ',max(//empleado/@salario),' Min: ',min(//empleado/@salario))

la función data solamente te saca solo los datos

**DIA 4 – XQUERY**

Sintaxis de xquery

for <variable> in <expresión XPath>

let <variables vinculadas>

where <condición XPath>

order by <expresión>

return <expresión de salida>

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>

for $emp in /EMPLEADOS/EMP\_ROW

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

order by $emp/OFICIO

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

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

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

return $nom

return data($nom)

return concat($nom,' ',$ofi)

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

for $dep in /universidad/departamento

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

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

else ()

for $dep in //departamento[@tipo='A']

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

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>

Mostrar los nombres de profesores del departamento A.

for $dep in /universidad/departamento

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

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

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

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

<tipoB>