/ -> 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 sirbe 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)