



# XQuery

# XQuery



Query Working Group, grupo creado en W3C en 1999 se encarga de elaborar lenguaje Xquery

- Xquery 1.0: Estándar de facto desde Enero de 2007.
  - ✦ <http://www.w3.org/TR/xquery/>
- Inspirado en lenguaje de consultas SQL
  - ✦ No tiene sintaxis XML
- Lenguaje declarativo y fuertemente tipado
- Actualmente hay una recomendación de especificaciones 3.0 de Octubre de 2013.
  - ✦ <http://www.w3.org/TR/xquery-30/>

# XQuery



- **Funcionamiento de una consulta**
  - Entrada: Datos XML
  - Salida: Datos XML

NOTA: Puede verse como una alternativa a XSLT
- **Usa XPath como lenguaje de base**
  - XQuery puede verse como una extensión de XPath
  - Añade capacidades para:
    - ✦ Acceso a fuentes de información
    - ✦ Creación de documentos XML
    - ✦ Expresiones FLOWR

# Ejemplo XQuery



Entrada (alumnos.xml)

```
<alumnos>
  <alumno dni="93940">
    <nombre>Jose</nombre>
    <apells>Bernardo</apells>
    <nota>7</nota>
  </alumno>
  <alumno dni="93940">
    <nombre>Juan </nombre>
    <apells>López</apells>
    <nota>4</nota>
  </alumno>
</alumnos>
```

Consulta (XQuery)

```
for $a in doc("alumnos.xml")//alumno
  where $a/nota > 5
return
  <aprobado>{
    $a/@dni,
    $a/nota
  }</aprobado>
```

Resultado

```
<aprobado dni="93940">
  <nota>7</nota>
</aprobado>
```

# Lenguaje XQuery



- Lenguaje sensible mayúsculas
- Comentarios (: .... :)
- Lenguaje basado en expresiones:
- Expresiones básicas:
  - Números: 5.6
  - Cadenas: “Hola”
  - Constructores: date(“2007-6-30”)
  - Operadores: (2 + 4) \* 5
  - Secuencias: (1, 2, 3) (1 to 3)
  - Variables: \$inicio
  - Invocación de funciones: substring(“Abracadabra”,1,4)
- Función data()
  - Para obtener el contenido de un elemento o atributo

# Lenguaje XQuery



- Acceso a los datos de entrada

`fn:doc(URI)` devuelve el nodo raíz del documento accesible a través de la URI

Ejemplo: `fn:doc("alumnos.xml")`

`fn:collection(URI)` devuelve una secuencia de nodos a partir de una URI

Puede utilizarse para acceder a bases de datos XML

# Xquery

## Estructura de las consultas

- Consulta XQuery: Prólogo + Expresión
- Prólogo: declaraciones de espacios de nombres, de funciones, etc.
- Expresión: Consulta propiamente dicha

**Prólogo**


```
declare function ...  
declare namespace...  
...
```

**Expresión de  
la consulta**

```
for ...  
let ...  
where ....  
order by ...  
return ...
```

# Xquery

## Prólogo



- Contiene declaraciones que marcan el entorno de la consulta.

Ejemplo: versión, espacios de nombres, módulos, base-uri, etc.

```
xquery version="1.0" encoding="utf-8";
```

```
declare namespace a="http://www.alumnos.org/";
```

También pueden declararse variables y funciones



# Lenguaje Xquery

## Expresiones XPath



- XQuery admite expresiones basadas en XPath para seleccionar nodos
- Ejemplo:  
Obtener todos los alumnos aprobados

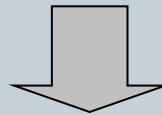
```
doc("alumnos.xml")//alumno[nota > 5]
```

# Lenguaje Xquery

## Creación de nodos

- XQuery facilita la creación directa de nodos
- Ejemplo:

```
<curso fecha="2007">  
  <p>Nota de primer alumno: </p>  
  <p> { doc("alumnos.xml")//alumno[1]/nota } </p>  
</curso>
```



**Salida**

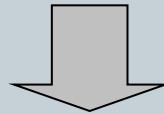
```
<curso fecha="2007">  
  <p>Notas del primer alumno</p>  
  <p>  
    <nota>7</nota>  
  </p>  
</curso>
```

# Lenguaje Xquery

## Creación de nodos mediante constructores

- Sintaxis alternativa

```
element curso {  
  attribute fecha { 2007},  
  element p {"Notas del primer alumno"},  
  element p { doc("alumnos.xml")//alumno[1]/nota }  
}
```



**Salida**

```
<curso fecha="2007">  
  <p>Notas del primer alumno</p>  
  <p>  
    <nota>7</nota>  
  </p>  
</curso>
```

# Xquery

## Expresiones FLWOR



- FLWOR (**F**or **L**et **W**here **O**rders-by **R**eturn)

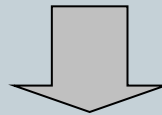
<b>for</b>	Genera secuencias enlazando variables
<b>let</b>	Asociar valores a variables
<b>where</b>	Filtra resultados según una condición
<b>order by</b>	Ordena resultados
<b>return</b>	Generado valores en la salida

# Xquery

## Cláusula FOR

- Permite iterar sobre una secuencia de valores, ligando una variable a cada valor de la secuencia y evaluando una expresión por cada valor de la variable

```
for $n in (1 to 4)  
return $n * $n
```



**Salida**

**1 4 9 16**

# Xquery

## Cláusula FOR



- Se utiliza normalmente para asignar un conjunto de nodos:

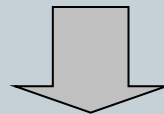
```
for $a in doc("alumnos.xml")//alumno  
return $a/nombre
```

# XQuery

## Cláusula Let

- Let permite asociar un valor a una variable

```
for $m in (1 to 3)
let $n := (1 to 3)
return <valor><m>{$m}</m><n>{$n}</n></valor>
```



**Salida**

```
<valor> <m>1</m><n>1 2 3</n></valor>
<valor> <m>2</m><n>1 2 3</n></valor>
<valor> <m>3</m><n>1 2 3</n></valor>
```

# XQuery

## Cláusula Where

- Permite filtrar los valores que se generarán en la salida
- Contiene una expresión que si se cumple, entonces genera el valor en la salida

**Ejemplo: Obtener nombres de alumnos cuya nota está entre 6 y 8**

```
for $a in doc("alumnos.xml")//alumno  
where $a/nota > 6 and $a/nota < 8  
return $a/nombre
```



# XQuery

## Cláusula Order by



- Indica el criterio de ordenación

```
for $a in doc("alumnos.xml")//alumno  
order by $a/apellidos ascending  
return $a/nombre
```

```
for $a in doc("alumnos.xml")//alumno  
order by number($a/@dni) ascending  
return $a/nombre
```

# XQuery

## Cláusula return

- Se ejecuta una vez por cada tupla obtenida en la cláusula where
- Los resultados son concatenados
- Se suelen incluir constructores de elementos

```
for $a in doc("alumnos.xml")//alumno  
order by number($a/@dni) ascending  
return  
  <alumno>  
    {string($a/nombre), string($a/apells)}  
  </alumno>
```

# XQuery

## Variable posicional at

- En una expresión for permite ligar una variable a la posición del elemento en la expresión

```
for $a at $i in doc("alumnos.xml")//alumno  
return  
  <alumno numero="{ $i }">  
    { $a/nombre }  
  </alumno>
```

# XQuery

## Condicionales

### **if condición then expr else expr**

- La parte else es obligatoria. Si no se desea añadir nada más al resultado puede ponerse else ()

```
for $a in doc("alumnos.xml")//alumno
return
  <alumno>
    {$a/nombre}
    <nota>
      {if ($a/nota > 5) then "Aprobado"
       else "Suspenso" }
    </nota>
  </alumno>
```

# XQuery

## Cuantificadores

- some comprueba si se cumple una condición para algún valor de la secuencia
- every comprueba si se cumple para todos los valores

```
some $a in doc("alumnos.xml")//alumno  
satisfies $a/nota > 5
```

```
every $a in doc("alumnos.xml")//alumno  
satisfies $a/nota > 5
```

```
Nota: xs:decimal($a/nota) > 5
```

# XQuery

## Operadores



- Comparación de valores: **eq, ne, lt, le, gt, ge**
- Comparaciones generales: **=, !=, >=, <, >, >=**
- Comparación de nodos: **is, is not**
- Comparación de posición de nodos: **<<**
- Lógicos: **and, or, not**
- Secuencias de nodos: **union, intersect, except**
- Aritméticos: **+, -, \*, div, idiv, mod**

# XQuery

## Funciones predefinidas



- Entrada: doc, collection
- Agregadas: sum, avg, count, max, min
- Cadenas: string-length, substring, upper-case, lowercase, concat, string, starts-with, ends-with,...
- Funciones generales: distinct-values, empty, exists,...

# XQuery

## Funciones definidas por el usuario

- Es posible declarar nuevas funciones

Admite definiciones recursivas

```
declare function local:factorial($x) {  
    if ($x = 0) then 1  
    else $x * local:factorial($x - 1)  
};  
  
for $a in doc("alumnos.xml")//alumno  
return  
    <factNota>{local:factorial($a/nota)}</factNota>
```



# XQueryX



- XQueryX es una sintaxis XML de XQuery
  - Pensado para procesamiento automatizado
  - No está pensado para edición manual

# Xquery Herramientas

- Saxon: Motor Xquery y XSLT escrito en Java
  - <http://saxon.sourceforge.net/>
- 4 paquetes:
  - Saxon-HE (Home edition). Open source. Contiene implementación parcial de XSLT y XQuery
  - Saxon-PE, Saxon-EE y Saxon-CE (comerciales).
- Línea de comandos

```
> java net.sf.saxon.Query consulta.xq
```