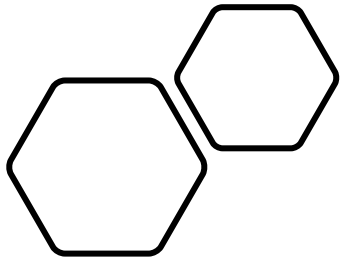
Curso 2023-2024



UD6: Almacenamiento de información

Lenguajes de marcas y sistemas de gestión de información

Almacenamiento de información

- XML es un estándar potente y de amplia aceptación para guardar e intercambiar información. Esto es en gran medida debido a la estandarización por parte del W3C. Características:
- Permite dar formato (estructura) a la información, sin definir la presentación de la misma.
- Se guarda como texto plano, lo que posibilita que sea fácilmente accesible desde cualquier programa.
- Es necesario procesar todo el XML para acceder a un nodo o atributo.

Ambitos de aplicación

Ámbitos de aplicación

- Los documentos y requerimientos de datos XML pueden ser agrupados en dos categorías generales:
  - Sistemas centrados en los datos.
  - Sistemas centrados en los documentos.
- La mayoría de los productos se centran en servir a uno de estos formatos mejor que al otro.

Sistemas centrados en los datos

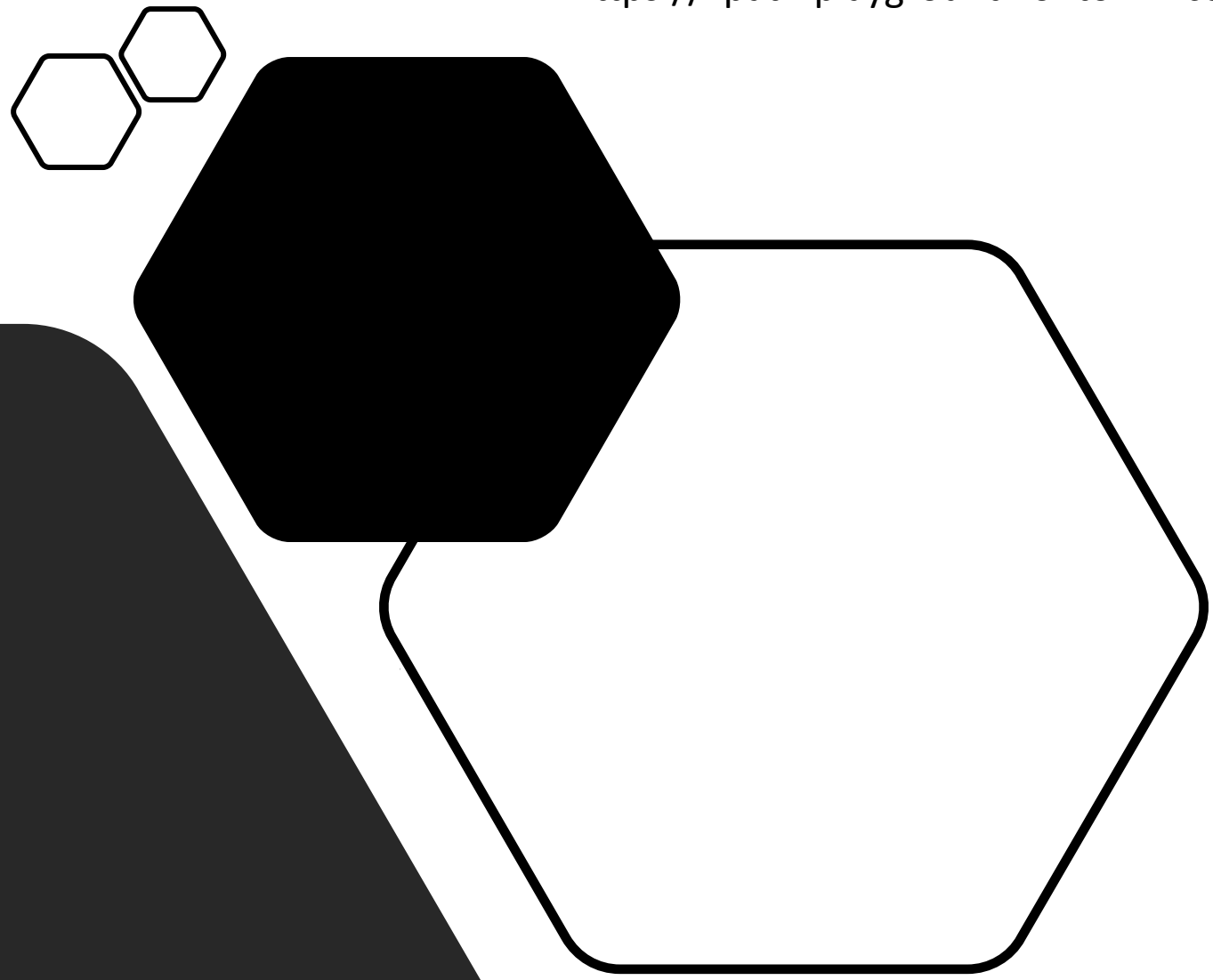
- Los documentos tienen una estructura bien definida y contienen datos que pueden ser actualizados y usados de diferentes formas.
- Es adecuado para ítems (contenidos de periódicos, artículos, publicidad, facturas, compras, etc) y algunos documentos menos estructurados.
- Las bases de datos relacionales tradicionales son mejores para tratar con requerimientos basados en los datos.

Sistemas centrados en los documentos

- Los documentos tienden a ser más impredecibles en tamaño y contenido. Presentan más tipos de datos, de tamaño más variable, con reglas flexibles para campos opcionales y para el propio contenido.

- Los sistemas de almacenamiento XML deben acomodarse eficientemente con ambos tipos de requerimientos de datos, dado que XML está siendo usado en sistemas que administran ambos tipos de datos.

https://xpath.playground.fontoxml.com/



XQuery

XQuery

- Lenguaje de consulta que permite extraer y procesar información de documentos XML o bases de datos XML.
- Tiene algunas similitudes a nivel semántico con SQL, aunque dispone de ciertas capacidades de programación a mayores.
- También comparte similitudes con XPath, además de soportar las mismas funciones y operadores.

XQuery

- Los principales usos de XQuery son tres:
  - Recuperar información a partir de conjuntos de datos XML.
  - Transformar unas estructuras de datos XML en otras que organizan la información de forma diferente.
  - Ofrecer una alternativa a XSLT para realizar transformaciones de datos XML a otro tipo de formatos.
- Algunas de las características más relevantes de su sintaxis son:
  - Las cadenas de caracteres pueden estar rodeadas de comillas simples o dobles.
  - Distingue entre mayúsculas y minúsculas.
  - Las variables comienzan por el símbolo $

XQuery

- Cada expresión XQuery la evalúa un procesador y se ejecuta sobre un documento XML para obtener un conjunto de datos.
- Estas expresiones utilizan XPath para acceder a los distintos contenidos del documento.
- Las cláusulas FLWOR (For, Let, Where, Order by, Return) proporcionan a XQuery toda la potencia necesaria para realizar todo tipo de operaciones de consulta sobre documentos XML.
- Lógicamente, las cláusulas XQuery deben ser ejecutadas sobre un documento previamente referenciado. La referencia se realiza mediante la función **doc**.

XQuery

- Una **consulta** XQuery es una **expresión** que lee una secuencia de datos en XML y devuelve otra secuencia de datos en XML como resultado.
- Características de las expresiones XQuery:
  - En XQuery todo es una expresión que se evalúa a un valor.
  - El valor de una expresión es una secuencia de nodos y/o valores.
  - La mayoría de las expresiones están compuestas por la combinación de expresiones más simples unidas mediante operadores y palabras reservadas.
  - Toda expresión XPath es también una expresión XQuery válida.

Sintaxis XQuery

- Los caracteres **{** y **}** delimitan las expresiones que son evaluadas para **crear un documento nuevo**.
- Los comentarios se escriben entre los caracteres (: y :).
- Se admiten las expresiones condicionales if-then-else con la misma semántica que en los lenguajes de programación tradicionales.
- Las consultas XQuery pueden estar formadas por hasta cinco tipos de cláusulas diferentes, conocidas como cláusulas FLWOR.

Sintaxis XQuery

```
(: Selecciona los títulos de los libros de una biblioteca :)
<ul>
{
  for $x in doc("libros.xml")/libros/libro
  where $x/editorial = "Alamut"
  order by $x/titulo
  return <li>{data($x/titulo)}</li>
}
</ul>
```

XQuery: cláusulas FLWOR

- **for**: para recorrer colecciones. Asocia una variable con cada elemento devuelto por la consulta.
- **let**: para vincular resultados a variables.
- **where**: para incorporar condiciones a las cláusulas for.
- **order by**: para ordenar los resultados.
- **return**: para indicar qué resultados se han de devolver.
- Las cláusulas FLWOR deben ser escritas siempre en minúsculas.
- Sintaxis FLWOR:
  - `{for | let} [where] [order by] return`

Cláusula for

- Asocia una o más variables con cada nodo que encuentre en la colección de datos.
- Es la cláusula que permite establecer el ámbito y los datos a procesar en una consulta XQuery.
- La sintaxis es la siguiente:

```
for $variable in expresión
```

- Siendo *$variable* el nombre de la variable que vamos a utilizar.
- Expresión es la expresión que se va a evaluar. Puede ser cualquier cosa que retorne un valor. En general será una expresión XPath.

```xml
<discos>
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
</discos>
```

```
for $a in /discos
return $a
```


```xml
<discos>
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
</discos>
```

```xml
<discos>
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
</discos>
```

```
for $a in //disco
return $a
```

```xml
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
```

Cláusula let

- Permite asignar un valor a una variable, la cual podremos utilizar posteriormente.
- Su uso permite que las consultas sean más legibles y fáciles de mantener.
- La sintaxis es la siguiente:

```
let $variable := expresión
```

- Siendo *$variable* el nombre de la variable que vamos a utilizar.
- Expresión es la expresión que se va a evaluar para asignarle un valor a la variable.
- A pesar de denominarse variable, es **inmutable**: una vez definida no se podrá modificar.

Cláusula where

- Se usa para filtrar los resultados producidos por las cláusulas for y let y limitarlos a aquellos elementos que cumplen ciertas condiciones especificadas en la consulta.
- La sintaxis es la siguiente:

```
for $variable in expresión
where condición
return resultado
```

- Siendo *$variable* el nombre de la variable que vamos a utilizar.
- Expresión es la expresión que se va a evaluar para asignarle un valor a la variable.
- Condición es la expresión booleana que se evalúa para determinar si se incluye o no un elemento en el resultado.
- Resultado es la información que queremos recuperar.

Cláusula where

- Si usamos where en combinación con for, se evalúa la condición por cada uno de los elementos indicados en el for.
- Si usamos where en combinación con let, se evalúa la condición para un único elemento.

```xml
<discos>
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
</discos>
```

```
for $a in //disco
where $a/@grupo="Daft Punk" and $a/ventas>=3000000
return $a
```

```xml
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
```

Cláusula order by

- Se usa para ordenar los resultados de una consulta en función de ciertos criterios. Ordena el resultado después de aplicar la cláusula **where** (si existe).
- La sintaxis es la siguiente:

```
for $variable in expresión
[where condición]
order by criterio [descending]
return resultado
```

- Siendo *$variable* el nombre de la variable que vamos a utilizar.
- Expresión es la expresión que se va a evaluar para asignarle un valor a la variable.
- Condición es la expresión booleana que se evalúa para determinar si se incluye o no un elemento en el resultado.
- Los criterios son los campos por los que vamos a ordenar los resultados. Por defecto la ordenación se realiza de forma ascendente, pero podemos añadir descending para ordenar de forma descendente.
- Resultado es la información que queremos recuperar.

```xml
<discos>
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
</discos>
```

```
for $a in //disco
where $a/@grupo="Daft Punk" and $a/ventas>=3000000
order by $a/lanzamiento/anho descending
return $a
```

```xml
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
```

```xml
<discos>
    <disco grupo="Daft Punk">
        <titulo>Homework</titulo>
        <ventas>2300000</ventas>
        <lanzamiento>
            <anho>1997</anho>
            <mes>1</mes>
            <dia>20</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Discovery</titulo>
        <ventas>3000000</ventas>
        <lanzamiento>
            <anho>2001</anho>
            <mes>3</mes>
            <dia>13</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Human After All</titulo>
        <ventas>1200000</ventas>
        <lanzamiento>
            <anho>2005</anho>
            <mes>03</mes>
            <dia>14</dia>
        </lanzamiento>
    </disco>
    <disco grupo="Daft Punk">
        <titulo>Random Access Memories</titulo>
        <ventas>5000000</ventas>
        <lanzamiento>
            <anho>2013</anho>
            <mes>05</mes>
            <dia>17</dia>
        </lanzamiento>
    </disco>
</discos>
```

```
for $a in //disco
where $a/@grupo="Daft Punk" and $a/ventas>=3000000
order by $a/lanzamiento/anho descending
return data($a/titulo)
```

```
"Random Access Memories"
"Discovery"
```

```xml
<notas>
    <alumno nombre="Isaac García">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">7</nota>
            <nota categoria="examen">5</nota>
            <nota categoria="proyecto">6</nota>
        </materia>
    </alumno>
    <alumno nombre="María Pérez">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">5</nota>
            <nota categoria="examen">6</nota>
            <nota categoria="proyecto">7</nota>
        </materia>
    </alumno>
    <alumno nombre="Alejandro Vázquez">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">8</nota>
            <nota categoria="examen">5</nota>
            <nota categoria="proyecto">5</nota>
        </materia>
    </alumno>
    <alumno nombre="Isabel Souto">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">5</nota>
            <nota categoria="examen">6</nota>
            <nota categoria="proyecto">8</nota>
        </materia>
    </alumno>
</notas>
```

```
for $alumno in //alumno
order by $alumno/@nombre
return $alumno/@nombre
```

```
nombre= "Alejandro Vázquez"
nombre= "Isaac García"
nombre= "Isabel Souto"
nombre= "María Pérez"
```

```xml
<notas>
    <alumno nombre="Isaac García">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">7</nota>
            <nota categoria="examen">5</nota>
            <nota categoria="proyecto">6</nota>
        </materia>
    </alumno>
    <alumno nombre="María Pérez">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">5</nota>
            <nota categoria="examen">6</nota>
            <nota categoria="proyecto">7</nota>
        </materia>
    </alumno>
    <alumno nombre="Alejandro Vázquez">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">8</nota>
            <nota categoria="examen">5</nota>
            <nota categoria="proyecto">5</nota>
        </materia>
    </alumno>
    <alumno nombre="Isabel Souto">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">5</nota>
            <nota categoria="examen">6</nota>
            <nota categoria="proyecto">8</nota>
        </materia>
    </alumno>
</notas>
```

```
for $alumno in //alumno
order by
$alumno/materia/nota[@categoria="ejercicios"]
return $alumno/@nombre
```

```
nombre= "María Pérez"
nombre= "Isabel Souto"
nombre= "Isaac García"
nombre= "Alejandro Vázquez"
```

```xml
<notas>
    <alumno nombre="Isaac García">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">7</nota>
            <nota categoria="examen">5</nota>
            <nota categoria="proyecto">6</nota>
        </materia>
    </alumno>
    <alumno nombre="María Pérez">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">5</nota>
            <nota categoria="examen">6</nota>
            <nota categoria="proyecto">7</nota>
        </materia>
    </alumno>
    <alumno nombre="Alejandro Vázquez">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">8</nota>
            <nota categoria="examen">5</nota>
            <nota categoria="proyecto">5</nota>
        </materia>
    </alumno>
    <alumno nombre="Isabel Souto">
        <materia nombre="Bases de datos">
            <nota categoria="ejercicios">5</nota>
            <nota categoria="examen">6</nota>
            <nota categoria="proyecto">8</nota>
        </materia>
    </alumno>
</notas>
```

```
for $alumno in //alumno
order by
$alumno/materia/nota[@categoria="ejercicios"]
descending
return $alumno/@nombre
```

```
nombre= "Alejandro Vázquez"
nombre= "Isaac García"
nombre= "Isabel Souto"
nombre= "María Pérez"
```

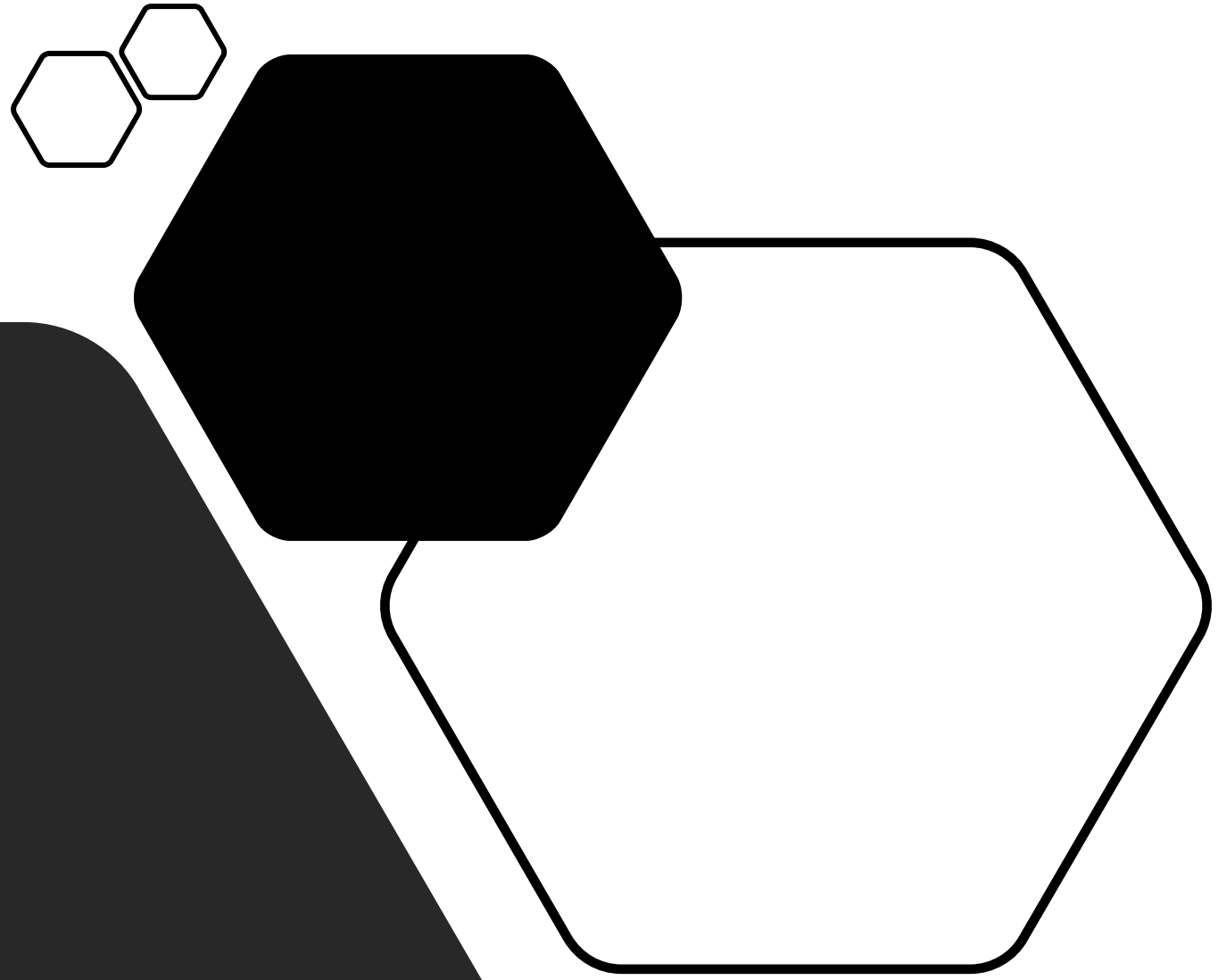
Cláusula return

- Se usa para especificar la información que se debe devolver como resultado de una consulta.
- La sintaxis es la siguiente:

```
for $variable in expresión
return resultado
```

- Siendo *$variable* el nombre de la variable que vamos a utilizar.
- Resultado es la información que queremos recuperar.
- Dentro de resultado podemos usar funciones y expresiones XQuery para realizar cálculos y manipulaciones más avanzadas.

Funciones XQuery



Funciones numéricas

Función	Descripción
floor()	Redondea al valor numérico inferior más próximo.
ceiling()	Redondea al valor numérico superior más próximo.
round()	Redondea al valor más próximo.
count()	Cuenta la cantidad de ítems de una colección.
min()	Devuelve el mínimo de los valores de los nodos dados.
max()	Devuelve el máximo de los valores de los nodos dados.
avg()	Calcula el valor medio de los valores dados.
sum()	Calcula la suma total de una cantidad de ítems dados.

Funciones de cadenas de texto

Función	Descripción
concat()	Devuelve una cadena construida por la unión de dos cadenas dadas.
string-length()	Devuelve la cantidad de caracteres que forman una cadena.
contains()	Determina si una cadena de texto (primer argumento) contiene otra (segundo argumento).
starts-with()	Determina si una cadena empieza por otra dada.
ends-with()	Determina si una cadena termina por otra dada.
upper-case()	Convierte la cadena dada a mayúsculas.
lower-case()	Convierte la cadena dada a minúsculas.

Funciones de uso general

Función	Descripción
empty()	Devuelve true cuando la secuencia dada no contiene ningún elemento.
exists()	Devuelve true cuando una secuencia contiene, al menos, un elemento.
distinct-values()	Extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados.
data()	Devuelve el valor de los elementos que recibe como argumentos, es decir, sin etiquetas.

Operadores XQuery

Operadores aritméticos

Operador	Descripción
+	Suma
-	Resta
*****	Multiplicación
div	División
mod	Módulo

Operadores lógicos

Operador	Descripción
and	Operador “Y” lógico
or	Operador “O” lógico

Operadores de comparación de nodos

Operador	Descripción
is	true si las dos variables comparadas están ligadas al mismo nodo
is not	true si las dos variables comparadas no están ligadas al mismo nodo

Operadores de comparación

Operador	Descripción
=	Igual a
!=	Distinto a
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

Operadores de secuencias de nodos

Operador	Descripción
union	Devuelve una secuencia que contiene todos los nodos que aparecen en alguno de los dos operandos que recibe
intersect	Devuelve una secuencia que contiene todos los nodos que aparecen en los dos operandos que recibe.
except	Devuelve una secuencia que contiene todos los nodos que aparecen en el primer operando que recibe y que no aparecen en el segundo.