

# XPath

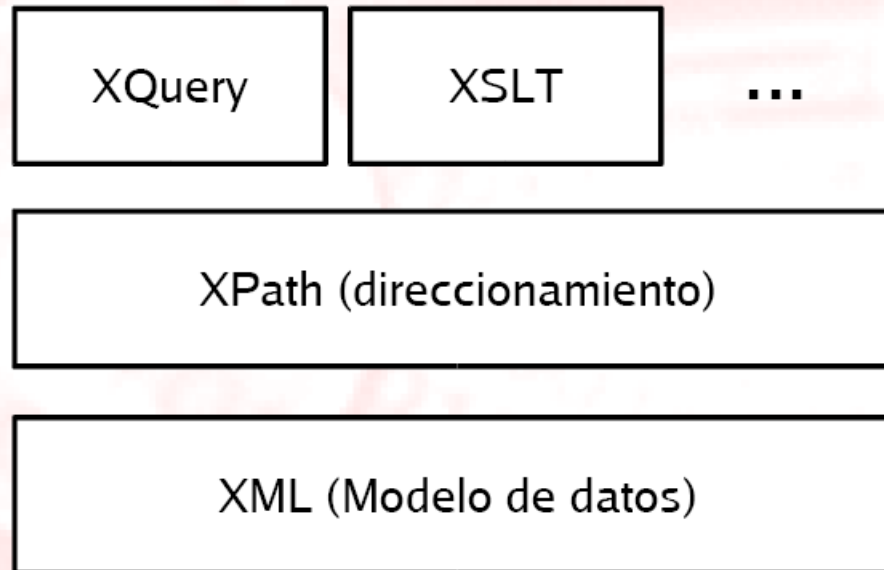
# XPATH

- ▶ Base para otras tecnologías
  - XQuery
  - XSLT
  - XPointer
- ▶ Lenguaje “básico” para buscar en XML

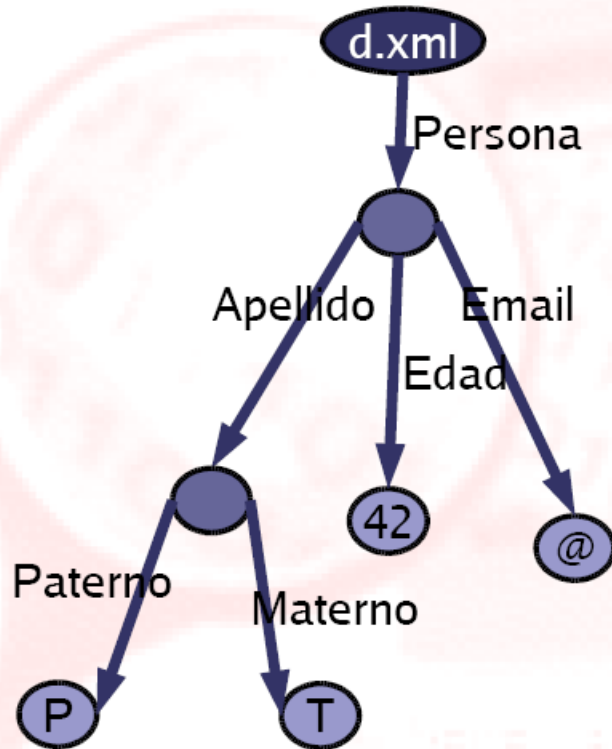
Tan importante para  
documentos estructurados como es SQL  
para BD relacionales

# Tecnologías interdependientes

## Tecnologías interdependientes



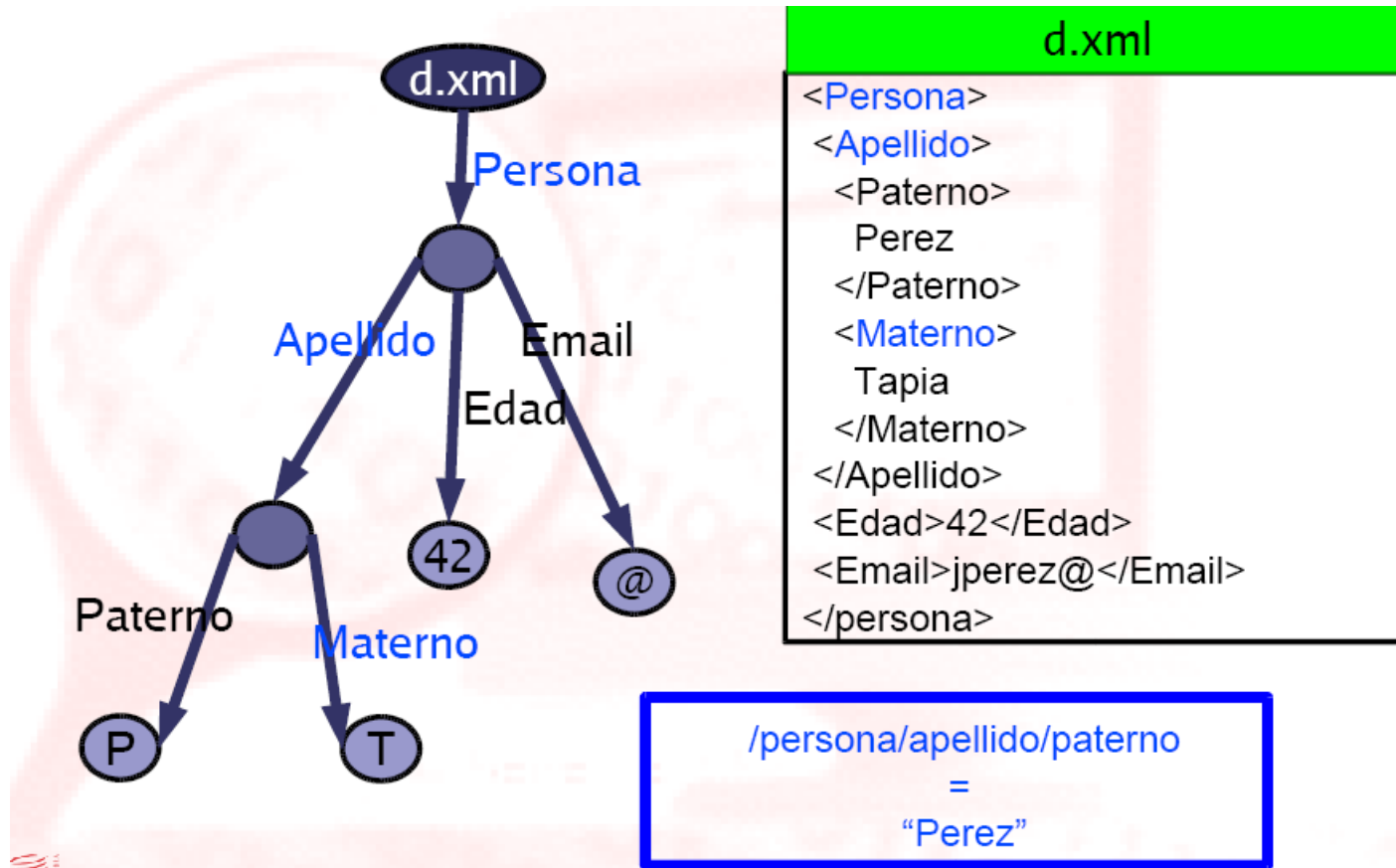
# Un documento como un árbol



d.xml

```
<Persona>
  <Apellido>
    <Paterno>
      Perez
    </Paterno>
    <Materno>
      Tapia
    </Materno>
  </Apellido>
  <Edad>42</Edad>
  <Email>jperez@</Email>
</persona>
```

# Una ruta en el árbol



# Objetivo de XPath

- ▶ Identificar elementos
  - A una profundidad arbitraria
  - En base al conjunto de nodos en el camino
- ▶ Importante: siempre hay un nodo de contexto

# Expresiones XPath

- ▶ Tienen la forma
  - nodo1 /nodo2 /.../nodoN
  - En el ejemplo: persona/apellido/materno
- ▶ Describen un camino (*path*)
- ▶ Resultado:
  - Un conjunto de nodos
  - String, número o boolean
- ▶ El resultado podría no ser un doc. XML

# Parecido a sist. de archivos

Ficheros y directorios	Nodos dentro de nodos
Respecto a directorio actual	Respecto a nodo actual o de contexto
* = cualquier cosa	* = cualquier nodo
Un fichero por ruta	Uno o varios nodos por ruta



# Expresiones XPath

- ▶ Falta: cómo seleccionar entre múltiples posibilidades
  - Para esto se usan predicados entre corchetes [ ... ]
- ▶ Falta: cómo buscar en múltiples documentos
  - No pueden realizar “*joins*”

# Ventajas

- ▶ Compacto, eficiente
  - “Encontrar en la lista de autores el apellido de un autor que tenga el atributo tipo con el valor clásico”
  - `autores/autor[@tipo='clasico']/apellido`
- ▶ Funciones básicas
  - Strings, números

# Rutas “child” (nodos hijo)

doc.xml

```
<noticia>
  <titulo>Título</titulo>
  <fuente>upi</fuente>
  <cuerpo fecha="hoy">
    <reportero cod="3">
      Juan
    </reportero>
    <p>Párrafo
      <b>uno</b></p>
    <p>Párrafo dos</p>
  </cuerpo>
</noticia>
```

- child::fuente
  - <fuente>upi</fuente>
- child::\*
  - <titulo>...</cuerpo>
- child::text()
  - (nada)
- child::cuerpo/child::reportero
  - <reportero>Juan</reportero>
- child::cuerpo/child::p/child::text()
  - Párrafo
- Abreviado: “child::p” es igual a “p”

# “parent”, “ancestor”, “self”

## “parent”, “ancestor”, “self”

doc.xml

```
<noticia>
<titulo>Título</titulo>
<fuente>upi</fuente>
<cuerpo fecha="hoy">
  <reportero cod="3">
    Juan
  </reportero>
  <p>Párrafo
    <b>uno</b></p>
  <p>Párrafo dos</p>
</cuerpo>
</noticia>
```

♦ parent::cuerpo/parent::noticia

<noticia>...</noticia>

♦ Abreviado: “parent::\*” es igual a “..”

♦ ../../fuente

<fuente>upi</fuente>

♦ ancestor::noticia/titulo

<titulo>Título</titulo>

♦ self::reportero

<reportero>Juan</reportero>

♦ Abreviado: “self::\*” es igual a “.”

# “attribute”, “descendant”, “root”

doc.xml

```
<noticia>
  <titulo>Título</titulo>
  <fuente>upi</fuente>
  <cuerpo fecha="hoy">
    <reportero cod="3">
      Juan
    </reportero>
    <p>Párrafo
      <b>uno</b></p>
    <p>Párrafo dos</p>
  </cuerpo>
</noticia>
```

♦ attribute::fecha

♦ hoy

♦ Abreviado: “attribute::x” es igual a “@x”

♦ reportero/@cod

♦ 3

♦ descendant::b|

♦ <b>uno</b>

♦ Abreviado: “descendant::b” igual a “./b”

♦ ./noticia

♦ <noticia>...</noticia>

♦ p/b/text()

uno

# “preceding”, “following”

## doc.xml

```
<noticia>
  <titulo>Título</titulo>
  <fuente>upi</fuente>
  <cuerpo fecha="hoy">
    <reportero cod="3">
      Juan
    </reportero>
    <p>Párrafo
      <b>uno</b></p>
    <p>Párrafo dos</p>
  </cuerpo>
</noticia>
```

- preceding::\*
  - <reportero cod="3">Juan</reportero>
- following::p
  - <p>Párrafo dos</p>

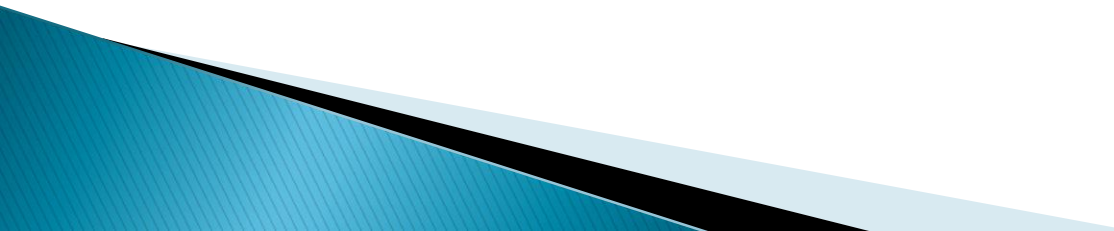
# Predicados (condiciones)

doc.xml

```
<noticia>
<titulo>Título</titulo>
<fuente>upi</fuente>
< cuerpo fecha="hoy">
  <reportero cod="3">
    Juan
  </reportero>
  <p>Párrafo
    <b>uno</b></p>
  <p>Párrafo dos</p>
</cuerpo>
</noticia>
```

- ♦ `reportero[@cod=2]`
  - ♦ (nada)
- ♦ `p[position()=1]`
  - ♦ `<p>Párrafo <b>uno</b></p>`
- ♦ `p[position()=last()]`
  - ♦ `<p>Párrafo dos</p>`
- ♦ `p[child::b]` ó `p[b]`
  - ♦ `<p>Párrafo <b>uno</b></p>`
- ♦ `reportero[.='Juan']`
  - `<reportero cod="3"> Juan`  
`<reportero />`
- ♦ Se puede combinar ... `"/libro/capitulo  
[position()=3]/seccion[position()=2]"`

# Resumen de sintaxis abreviada 1 / 2

- ▶ X – hijo elemento “X”
  - ▶ \* – todos los hijos elemento
  - ▶ text() – todos los hijos texto
  - ▶ @Y – atributo “Y”
  - ▶ X[1] – primer hijo “X”
  - ▶ X[last()] – último hijo “X”
  - ▶ \*/X – nietos “X”
  - ▶ X//Y – descendientes “Y” de hijo “X”
- 



# Resumen de sintaxis abreviada 2 / 2

- ▶ //Y – descendientes “Y” de la RAIZ
- ▶ .. – padre
- ▶ //X[1][@Y='Z'] – primeros hijos X con atributo Y='Z'

# Buscar por contenido

libros.xml

```
<libros>
  <libro>
    <titulo>XXX</titulo>
    <año>1890</año>
  </libro>
  <libro>
    <titulo>YYY</titulo>
    <año>1950</año>
  </libro>
  <libro>
    <año>1830</año>
  </libro>
</libros>
```

- libro[titulo = 'XXX']/año
  - 1890
- libro[not(titulo = 'XXX')]
  - 1950

# Buscar con funciones

libros.xml

```
<libros>
  <libro>
    <titulo>XXX</titulo>
    <año>1890</año>
  </libro>
  <libro>
    <titulo>YYY</titulo>
    <año>1950</año>
  </libro>
  <libro>
    <año>1830</año>
  </libro>
</libros>
```

- **concat**(libro[1]/titulo, libro[2]/año)
  - XXX1950
- libro[**starts-with**(titulo,'X')]/año
  - 1890
- libro[**contains**(año,9)]/año
  - 1950

# Funciones básicas

libros.xml

```
<libros>
  <libro>
    <titulo>XXX</titulo>
    <año>1890</año>
  </libro>
  <libro>
    <titulo>YYY</titulo>
    <año>1950</año>
  </libro>
  <libro>
    <año>1830</año>
  </libro>
</libros>
```

- libro[position()=last()]/año
  - 1830
- count(libro)
  - 3
- libro[count(titulo)=0]/año
  - 1830
- count(libro/titulo)
  - 2

# Funciones sobre nodos

- ▶ **id()**. Selecciona elementos por su identificador, de la forma:
  - Conjunto\_nodos  $\leq$  id(valor)..
- ▶ **last()**. Devuelve la posición del último nodo de la lista procesada, de la forma:
  - Numero  $\leq$  last()
- ▶ **name()**. Devuelve el nombre de un nodo específico de la forma:
  - Cadena  $\leq$  name(nodo)
- ▶ **position()**. Devuelve la posición en la lista de nodos, del nodo que se esté procesando en cada momento, de la forma:
  - numero=position().

# Ejemplo position()

- ▶ con el uso de la función position() es posible seleccionar:
  - el segundo capítulo de libro mediante el uso de la:  
`//capitulo[position()=2]`.
  - todos los capítulo menos el último:  
`//capitulo[not(position()=last())]`.
  - el penúltimo capítulo: `//capitulo[last()-1]`.

# Funciones de strings

- ▶ Contatenar cadenas:
  - `concat('El',' ','XML')` devuelve: `'EL XML'`
- ▶ Búsqueda de caracteres
  - `substringafter('axbyc','x')='byc'`
  - `substringbefore('axbyc','x')='a'`
- ▶ Los índices empiezan desde '1', típico de los estándares de XML
  - `substring("abcde", 2, 3) = "bcd"`
  - `stringlength("tres") = 4`
- ▶ Traducir caracteres string,fuente,destino
- ▶ `translate("BAR","ABC","abc") = "baR"`

# Funciones de strings

## ▶ Booleana:

- `contains('XML','X')` devuelve: `true`
- `starts-with('XML','X')` devuelve: `true`

## ▶ Transformaciones:

- `string(314)` devuelve: `'314'`



# Sobre números

- ▶ Redondeo superior:
  - `ceiling(3.14)` devuelve: 4.
- ▶ Redondeo inferior
  - `floor(3.14)` devuelve: 3
- ▶ Transformación:
  - `number('100')` es 100.
- ▶ Redondeo:
  - `round(3.14)` devuelve: 3

# Ejemplo:

## XQuery requiere XPath

```
<books-with-prices>
  { FOR $a in document("A/bib.xml")//book,
    $b in document("B/reviews.xml")//entry
    WHERE $b/title = $a/title
    RETURN
      <book-with-prices>
        { $b/title }
        <price-A>
          { $a/price/text() }
        </price-A>
        <price-B>
          { $b/price/text() }
        </price-B>
      </book-with-prices>
    }
</books-with-prices>
```

# Ejemplo:

## XSLT requiere XPath

```
<xsl:template match="/doc">
  <otherdoc>
    <xsl:for-each select="item">
      <otheritem>
        <xsl:value-of select="sub-item/*"/>
      </otheritem>
    </xsl:for-each>
  </otherdoc>
</xsl:template>
</xsl:stylesheet>
```

# XPATH

- ▶ **XPath 1.0** Recomendación del 16 de Noviembre de 1999.
  - <http://www.w3.org/TR/xpath/>
- ▶ **XPath 2.0** Recomendación del 23 de Enero de 2007.  
Última revisión 14 de Diciembre de 2010.
  - <http://www.w3.org/TR/xpath20/>