5.5.2. Lenguajes de consultas XPath y XQuery

Ambos son estándares para acceder y obtener datos desde documentos XML, estos lenguajes tienen en cuenta que la información en los documentos está semiestructurada o jerarquizada como árbol.

XPath, es el lenguaje de rutas de XML, se utiliza para navegar dentro de la estructura jerárquica de un XML.

XQuery es a XML lo mismo que SQL es a las bases de datos relacionales, es decir, es un lenguaje de consulta diseñado para consultar documentos XML. Abarca desde archivos XML hasta bases de datos relacionales con funciones de conversión de registros a XML. XQuery contiene a XPath, toda expresión de consulta en XPath es válida en XQuery, pero XQuery permite mucho más.

5.5.2.1. Expresiones XPath

XPath es un lenguaje que permite seleccionar nodos de un documento XML y calcular valores a partir de su contenido. Existen varias versiones de XPath aprobadas por el W3C, aunque la versión más utilizada sigue siendo la versión 1.

La forma en que XPath selecciona partes del documento XML se basa en la representación arbórea que se genera del documento. A la hora de recorrer un árbol XML podemos encontrarnos con los siguientes tipos de nodos:

- ***nodo raíz,*** es la raíz del árbol, se representa por /.
- ***nodos elemento,*** cualquier elemento de un documento XML, son las etiquetas del árbol.
- ***nodos texto,*** los caracteres que están entre las etiquetas.
- ***nodos atributo,*** son como propiedades añadidas a los nodos elemento, se representan con @.
- ***nodos comentario,*** las etiquetas de comentario.
- ***nodos espacio de nombres,*** contienen espacios de nombres.
- ***nodos instrucción de proceso***, contienen instrucciones de proceso, van entre las etiquetas <? ..... ?>.

Por ejemplo. Dado este documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<universidad>
<espacio xmlns="http://www.misitio.com"
          xmlns:prueba="http://www.misitio.com/pruebas"  />
  <!-- DEPARTAMENTO 1 -->
  <departamento telefono="112233" tipo="A">
   <codigo>IFC1</codigo>
   <nombre>Informática</nombre>
  </departamento>
  <!-- DEPARTAMENTO 2 -->
  <departamento telefono="990033" tipo="A">
    <codigo>MAT1</codigo>
    <nombre>Matemáticas</nombre>
  </departamento>
  <!-- DEPARTAMENTO 3 -->
  <departamento telefono="880833" tipo="B">
   <codigo>MAT2</codigo>
   <nombre>Análisis</nombre>
```

```
</departamento>
universidad>
```

Nos encontramos los siguientes tipos de nodos:

TIPO NODO	
emento	<universidad>, <departamento>, <codigo>, <nombre>
xto	IFC1, Informática, MAT1, Matemáticas, MAT2, Análisis
ibuto	telefono="112233" tipo="A" , telefono="990033" tipo="A", telefono="880833" tipo="B"
mentario	<!-- DEPARTAMENTO 1 -->, <!-- DEPARTAMENTO 2 --> <!-- DEPARTAMENTO 3 -->
pacio de nombres	<espacio xmlns="http://www.misitio.com" xmlns:prueba=http://www.misitio.com/pruebas />
trucción de proceso	<?xml version="1.0" encoding="ISO-8859-1"?>

s test sobre los tipos de nodos pueden ser:

- Nombre del nodo, para seleccionar un nodo concreto, ej.: /universidad
- ***prefix:****,* para seleccionar nodos con un espacio de nombres determinado.
- ***text(),*** selecciona el contenido del elemento, es decir, el texto, ej.: //nombre/text().
- ***node(),***selecciona todos los nodos , los elementos y el texto, ej.: /universidad/node().
- ***processing-instruction(),*** selecciona nodos que son instrucciones de proceso.
- ***comment(),*** selecciona los nodos de tipo comentario, /universidad/comment().

La sintaxis básica de XPath es similar a la del direccionamiento de ficheros. Utiliza ***scriptores de ruta o de camino*** que sirven para seleccionar los nodos o elementos que se cuentran en cierta ruta en el documento. Cada descriptor de ruta o paso de búsqueda puede a su z dividirse en tres partes:

- ***eje:*** indica el nodo o los nodos en los que se realiza la búsqueda.
- ***nodo de comprobación:*** especifica el nodo o los nodos seleccionados dentro del eje.
- ***predicado:*** permite restringir los nodos de comprobación. Los predicados se escriben entre corchetes.

Las expresiones XPath se pueden escribir utilizando una sintaxis abreviada, fácil de leer, o a sintaxis más completa en la que aparecen los nombres de los ejes (AXIS), más compleja. Por mplo, estas dos expresiones devuelven los departamentos con más de 3 empleados, la primera la forma abreviada y la segunda es la completa:

```
/universidad/departamento[count(empleado)>3]
/child::universidad/child::departamento[count(child::empleado)>3]
```

En este tema estudiaremos la ***sintaxis abreviada***, así pues, los descriptores se forman nplemente nombrando la etiqueta separada por / (hay que poner el nombre de la etiqueta tal al está en el documento XML, recuerda que hace distinción entre mayúsculas y minúsculas).

Si el descriptor comienza con / se supone que es una **ruta desde la raíz**. Para seguir una ruta dicaremos los distintos nodos de paso: /paso1/paso2/paso3… Si las rutas comienzan con / son **tas absolutas**, en caso contrario serán relativas.

Si el descriptor comienza con // se supone que la ruta descrita puede comenzar en cualquier parte de la colección.

EJEMPLOS XPATH UTILIZANDO UNA SINTAXIS ABREVIADA:

A partir de la colección ***ColeccionPruebas***, que contiene los documentos *departamentos.xml* y *empleados.xml*, cuyas estructuras se muestras en la Figura 5.13. Realiza desde el diálogo de consultas las siguientes consultas:

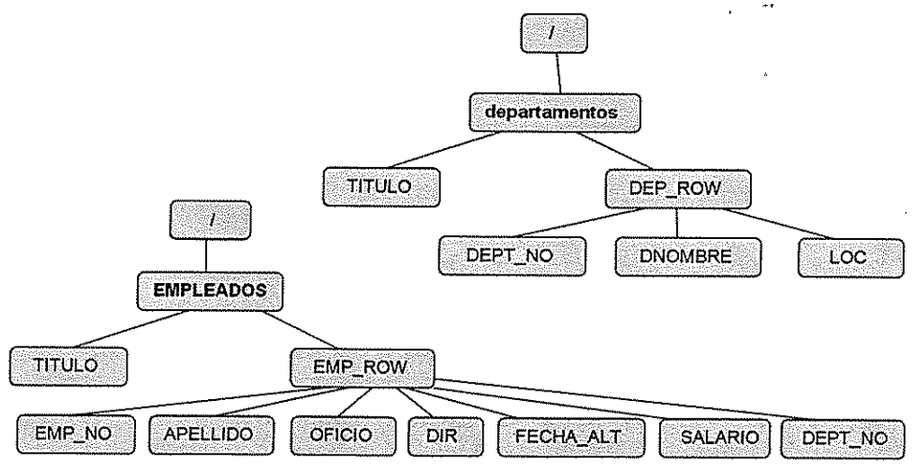

Figura 5.13. Estructuras de *departamentos.xml* y *empleados.xml*.

- /, si ejecutamos esta orden en el (***Diálogo de Consulta***) y se está dentro del contexto **/db/ColeccionPruebas** devuelve todos los datos de los departamentos y los empleados, es decir, incluye todas las etiquetas que cuelgan del nodo *departamentos* y del nodo *EMPLEADOS,* que están dentro de la colección *Prueba*. Si se ejecuta desde XQuery Sandbox, se obtiene otro resultado, pues el contexto no es el mismo.
- /departamentos, devuelve todos los datos de los departamentos, es decir, incluye todas las etiquetas que cuelgan del nodo raíz *departamentos*.
- /departamentos/DEP_ROW, devuelve todas las etiquetas dentro de cada DEP_ROW.
- /departamentos/DEP_ROW/**node(),** devuelve todas las etiquetas dentro de cada DEP_ROW, no incluye DEP_ROW.
- /departamentos/DEP_ROW/DNOMBRE, devuelve los nombres de departamentos de cada DEP_ROW, entre etiquetas.
- /departamentos/DEP_ROW/DNOMBRE/**text(),** devuelve los nombres de departamentos de cada DEP_ROW, ya sin las etiquetas.
- //LOC/text(), devuelve todas las localidades, de toda la colección (//), solo hay 4.
- //DEPT_NO, devuelve todos los números de departamentos, entre etiquetas. Observa que en este caso devuelve 23 filas en lugar de 4, es porque recoge todos los elementos DEPT_NO de la colección, y recuerda que en la colección también hemos incluido el documento *empleados.xml*, que tiene 14 empleados con su DEPT_NO, y *departamentosnuevo.xml* con 5 DEPT_NO.

- ***El operador*** * se usa para nombrar a cualquier etiqueta, se usa como comodín. Por ejemplo:

  – El descriptor /*/DEPT_NO selecciona las etiquetas DEPT_NO que se encuentran a 1 nivel de profundidad desde la raíz, en este caso ninguna.

  – /*/*/DEPT_NO selecciona las etiquetas DEPT_NO que se encuentran a dos niveles de profundidad desde la raíz, en este caso 23.

  – /departamentos/* selecciona las etiquetas que van dentro de la etiqueta departamentos y sus subetiquetas.

  – /* ¿Qué salida produce este descriptor?, depende del contexto en el que se ejecute. Se mostrarán todas las etiquetas de todos los documentos, es decir, se mostrarán todos los documentos de la colección.

- ***Condiciones de selección.*** Se utilizarán los corchetes para seleccionar elementos concretos, en las condiciones se pueden usar los comparadores: ***<, >, <=, >=, =, != , or, and y not*** (or, and y not deben escribirse en minúscula). Se utilizará el separador | para unir varias rutas. Ejemplos:

  – /EMPLEADOS/EMP_ROW[DEPT_NO=10], selecciona todos los elementos o nodos (etiquetas) dentro de EMP_ROW de los empleados del DEPT_NO 10.

  – /EMPLEADOS/EMP_ROW/APELLIDO|/EMPLEADOS/EMP_ROW/DEPT_NO selecciona los nodos APELLIDO y DEPT_NO de los empleados.

  – /EMPLEADOS/EMP_ROW[DEPT_NO=10]/APELLIDO/text(), selecciona los apellidos de los empleados del DEPT_NO=10.

  – /EMPLEADOS/EMP_ROW[not(DEPT_NO = 10)], selecciona todos los empleados (etiquetas) que NO son del DEPT_NO igual a 10.

  – /EMPLEADOS/EMP_ROW[not(OFICIO = 'ANALISTA')]/APELLIDO/text(), selecciona los APELLIDOS de los empleados que NO son ANALISTAS.

  – /EMPLEADOS/EMP_ROW[DEPT_NO=10]/APELLIDO | /EMPLEADOS/EMP_ROW[DEPT_NO=10]/OFICIO, selecciona el APELLIDO y el OFICIO de los empleados del DEPT_NO=10.

  – //*[DEPT_NO=10]/DNOMBRE/text(), /departamentos/DEP_ROW[DEPT_NO=10]/DNOMBRE/text(), estas dos consultas devuelven el nombre del departamento 10.

  – //*[OFICIO="EMPLEADO"]//EMP_ROW, devuelve los empleados con OFICIO "EMPLEADO", por cada empleado devuelve todos sus elementos. Busca en cualquier parte de la colección //.

  – Esto devuelve lo mismo: /EMPLEADOS/EMP_ROW[OFICIO="EMPLEADO"]//EMP_ROW.

  – /EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=10], devuelve los datos de los empleados con SALARIO mayor de 1300 y del departamento 10.

  – /EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=20]/APELLIDO | /EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=20]/OFICIO, devuelve el APELLIDO y el OFICIO de los empleados con SALARIO mayor de 1300 y del departamento 20. Se utiliza el separador | para unir las dos rutas.

- ***Utilización de funciones y expresiones matemáticas.***

  – Un número dentro de los corchetes representa la posición del elemento en el conjunto seleccionado. Ejemplos:

  /EMPLEADOS/EMP_ROW[1], devuelve todos los datos del primer empleado.

  /EMPLEADOS/EMP_ROW[5]/APELLIDO/text(), devuelve el APELLIDO del quinto empleado.

  – La función **last()** selecciona el último elemento del conjunto seleccionado. Ejemplos:

  /EMPLEADOS/EMP_ROW[last()],selecciona todos los datos del último empleado.

  /EMPLEADOS/EMP_ROW[last()-1]/APELLIDO/text(),devuelve el APELLIDO del penúltimo empleado.

  – Lo función **position()** devuelve un número igual a la posición del elemento actual.

  /EMPLEADOS/EMP_ROW[position()=3], obtiene los elementos del empleado que ocupa la posición 3.

  /EMPLEADOS/EMP_ROW[position()=3]/APELLIDO, selecciona el apellido del los elementos cuya posición es menor de 3, es decir, devuelve los apellidos del primer y segundo empleado.

  – La función **count()** cuenta el número de elementos seleccionados. Ejemplos:

  /EMPLEADOS/count(EMP_ROW), devuelve el número de empleados.

  /EMPLEADOS/count(EMP_ROW[DEPT_NO=10]), cuenta el nº de empleados del departamento 10.

  /EMPLEADOS/count(EMP_ROW[OFICIO="EMPLEADO" and SALARIO>1300]), cuenta el nº de empleados con oficio EMPLEADO y SALARIO mayor de 1300.

  //*[count(*)=3], devuelve elementos que tienen 3 hijos.

  //*[count(DEP_ROW)=4], devuelve los elementos que contienen 4 hijos DEP_ROW, devolverá la etiqueta departamentos y todas las subetiquetas.

  – La función **sum()** devuelve la suma del elemento seleccionado. Ejemplos:

  sum(/EMPLEADOS/EMP_ROW/SALARIO), devuelve la suma del SALARIO.

  Si la etiqueta a sumar la considera *string* **hay que convertirla a número utilizando la función number.**

  sum(/EMPLEADOS/EMP_ROW[DEPT_NO=20]/SALARIO), devuelve la suma de SALARIO de los empleados del DEPT_NO 20.

  – Función **max()** devuelve el máximo, **min()** devuelve el mínimo y **avg()** devuelve la media del elemento seleccionado. Ejemplos:

  max(/EMPLEADOS/EMP_ROW/SALARIO), devuelve el salario máximo.

  min(/EMPLEADOS/EMP_ROW/SALARIO), devuelve el salario mínimo.

  min(/EMPLEADOS/EMP_ROW[OFICIO="ANALISTA"]/SALARIO), devuelve el salario mínimo de los empleados con OFICIO ANALISTA..

  avg(/EMPLEADOS/EMP_ROW/SALARIO), devuelve la media del salario.

avg(/EMPLEADOS/EMP_ROW[DEPT_NO=20]/SALARIO), devuelve la media del salario de los empleados del departamento 20.

– La función **name()** devuelve el nombre del elemento seleccionado. Ejemplos:

//*[name()='APELLIDO'], devuelve todos los apellidos, entre sus etiquetas.

count(//*[name()='APELLIDO']), cuenta las etiquetas con nombre APELLIDO.

– La función **concat(cad1, cad2, ...)** concatena las cadenas. Ejemplos:

/EMPLEADOS/EMP_ROW[DEPT_NO=10]/concat(APELLIDO, " - ",OFICIO)

Devuelve el apellido y el oficio concatenados de los empleados del departamento 10

/EMPLEADOS/EMP_ROW/concat(APELLIDO, " - ", OFICIO, " - ", SALARIO)

Devuelve la concatenación de apellido, oficio y salario de los empleados.

– La función **starts-with(cad1, cad2)** es verdadera cuando la cadena cad1 tiene como prefijo a la cadena cad2. Ejemplos:

/EMPLEADOS/EMP_ROW[starts-with(APELLIDO,'A')], obtiene los elementos de los empleados cuyo APELLIDO empieza por 'A'.

/EMPLEADOS/EMP_ROW[starts-with(OFICIO,'A')]/concat(APELLIDO," - ",OFICIO)

obtiene APELLIDO y NOMBRE concatenados de los empleados cuyo OFICIO empieza por 'A'.

– La función **contains(cad1, cad2)** es verdadera cuando la cadena cad1 contiene a la cadena cad2.

/EMPLEADOS/EMP_ROW[contains(OFICIO,'OR')]/OFICIO, devuelve los oficios que contienen la sílaba 'OR' .

/EMPLEADOS/EMP_ROW [contains(APELLIDO,'A')]/APELLIDO, devuelve los apellidos que contienen una 'A'.

– La función **string-length(argumento)** devuelve el número de caracteres de su argumento.

/EMPLEADOS/EMP_ROW/concat(APELLIDO,' = ', string-length(APELLIDO)), devuelve concatenados el apellido con su número de caracteres.

/EMPLEADOS/EMP_ROW[string-length(APELLIDO)<4], devuelve los datos de los empleados cuyo APELLIDO tiene menos de 4 caracteres.

– Operador matemático **div( )** realiza divisiones en punto flotante.

/EMPLEADOS/EMP_ROW/concat(APELLIDO,' , ',SALARIO,' - ',SALARIO div 12), devuelve los datos concatenados de APELLIDO, SALARIO y el salario dividido por 12.

sum(/EMPLEADOS/EMP_ROW/SALARIO) div count(/EMPLEADOS/EMP_ROW), devuelve la suma de salarios dividido por el contador de empleados.

– Operador matemático **mod( )** calcula el resto de la división

/EMPLEADOS/EMP_ROW/concat(APELLIDO,' , ',SALARIO,' - ',SALARIO mod 12), devuelve los datos concatenados de APELLIDO, SALARIO y el resto de dividir el SALARIO por 12.

/EMPLEADOS/EMP_ROW[(SALARIO mod 12)=4], devuelve los datos de los empleados cuyo resto de dividir el SALARIO entre 12 sea igual a 4.

- ***Otras funciones.***
  - ***data(expresión XPath),*** devuelve el texto de los nodos de la expresión sin las etiquetas.
  - ***number(argumento),*** para convertir a número el argumento, que puede ser cadena, booleano o un nodo.
  - ***abs(num),*** devuelve el valor absoluto del número.
  - ***ceiling(num),*** devuelve el entero más pequeño mayor o igual que la expresión numérica especificada.
  - ***floor(num)***, devuelve el entero más grande que sea menor o igual que la expresión numérica especificada.
  - ***round(num),*** redondea el valor de la expresión numérica.
  - ***string(argumento),*** convierte el argumento en cadena.
  - ***compare(exp1,exp2),*** compara las dos expresiones, devuelve 0 si son iguales, 1 si exp1>exp2, y -1 si exp1<exp2.
  - ***substring(cadena,comienzo,num),*** extrae de la *cadena*, desde la posición indicada en *comienzo* el número de caracteres indicado en *num*.
  - ***substring(cadena,comienzo),*** extrae de la *cadena*, los caracteres desde la posición indicada por *comienzo*, hasta el final.
  - ***lower-case(cadena),*** convierte a minúscula la *cadena*.
  - ***upper-case(cadena),*** convierte a mayúscula la *cadena*.
  - ***translate(cadena1,caract1,caract2),*** reemplaza dentro de *cadena1*, los caracteres que se expresan en *caract1*, por los correspondientes que aparecen en *caract2*, uno por uno.
  - ***ends-with(cadena1,cadena2),*** devuelve true si la *cadena1* termina en *cadena2*.
  - ***year-from-date(fecha),*** devuelve el año de la fecha, el formato de fecha es AÑO-MES-DIA.
  - ***month-from-date(fecha)***, devuelve el mes de la fecha.
  - ***day-from-date(fecha),*** devuelve el día de la fecha.

Puedes encontrar más funciones en la URL: http://www.w3schools.com/xsl/xsl_functions.asp

ACTIVIDAD 5.1

Dado el documento ***productos.xml*** que está dentro de la colección ***ColeccionPruebas***, con información de datos de productos, y cuya estructura es la siguiente:

```
<produc>
    <cod_prod>xxxxxx</cod_prod>
    <denominacion>xxxxxxxxxxxx</denominacion>
    <precio>xxxx</precio>
    <stock_actual>xxx</stock_actual>
    <stock_minimo>xxxx</stock_minimo>
    <cod_zona>xxxx</cod_zona>
</produc>
```

Realiza las siguientes consultas XPath:

- Obtén los nodos denominación y precio de todos los productos.
- Obtén los nodos de los productos que sean placas base.

- Obtén los nodos de los productos con precio mayor de 60 € y de la zona 20.
- Obtén el número de productos que sean memorias y de la zona 10.
- Obtén la media de precio de los micros.
- Obtén los datos de los productos cuyo stock mínimo sea mayor que su stock actual.
- Obtén el nombre de producto y el precio de aquellos cuyo stock mínimo sea mayor que su stock actual y sean de la zona 40.
- Obtén el producto más caro.
- Obtén el producto más barato de la zona 20.
- Obtén el producto más caro de la zona 10.

5.5.2.2. Nodos atributos XPath

Un nodo puede tener tantos atributos como se desee, y para cada uno se le creará un nodo atributo. Los nodos atributo NO se consideran como hijos, sino más bien como etiquetas añadidas al nodo elemento. Cada nodo atributo consta de un nombre, un valor (que es siempre una cadena) y un posible "espacio de nombres".

Partimos del documento ***universidad.xml,*** que se encuentra dentro de la *ColeccionPruebas* subida en los anteriores apartados. El documento está formado por los nodos atributo: teléfono y tipo, que pertenecen al elemento departamento y salario que pertenece al elemento empleado. El documento es el siguiente:

```
<universidad>
 <departamento telefono="112233"
tipo="A">
  <codigo>IFC1</codigo>
  <nombre>Informática</nombre>
  <empleado salario="2000">
    <puesto>Asociado</puesto>
    <nombre>Juan Parra</nombre>
  </empleado>
  <empleado salario="2300">
   <puesto>Profesor</puesto>
   <nombre>Alicia Martín</nombre>
  </empleado>
 </departamento>

 <departamento telefono="990033"
tipo="A">
  <codigo>MAT1</codigo>
  <nombre>Matemáticas</nombre>
  <empleado salario="1900">
    <puesto>Técnico</puesto>
    <nombre>Ana García</nombre>
  </empleado>
  <empleado salario="2100">
     <puesto>Profesor</puesto>
  <nombre>Mª Jesús Ramos</nombre>
  </empleado>
  <empleado salario="2300">
     <puesto>Profesor</puesto>
     <nombre>Pedro Paniagua</nombre>
  </empleado>
  <empleado salario="2500">
    <puesto>Tutor</puesto>
    <nombre>Antonia González</nombre>
  </empleado>
 </departamento>

 <departamento telefono="880833"
tipo="B">
  <codigo>MAT2</codigo>
  <nombre>Análisis</nombre>
  <empleado salario="1900">
     <puesto>Asociado</puesto>
     <nombre>Laura Ruiz</nombre>
  </empleado>
  <empleado salario="2200">
     <puesto>Asociado</puesto>
     <nombre>Mario García</nombre>
  </empleado>
 </departamento>

</universidad >
```

El árbol del documento se muestra en la Figura 5.14., los atributos se representan con elipses.

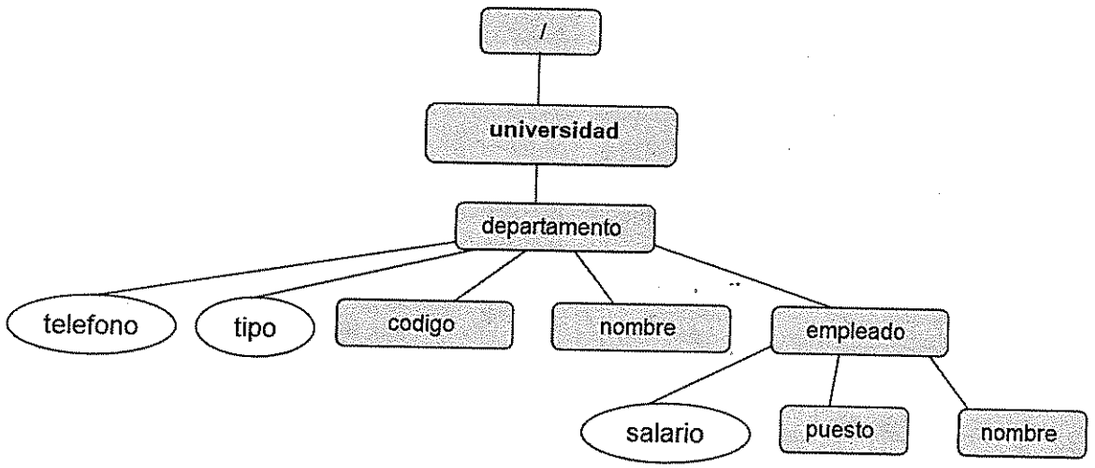

Figura 5.14. Árbol del documento *universidad.xml*

Para referirnos a los atributos de los elementos se usa @ antes del nombre, por ejemplo, ***@telefono, @tipo, @salario***. En un descriptor de ruta los atributos se nombran como si fueran etiquetas hijo pero anteponiendo @.

Ejemplos de consultas utilizando nodos atributo:

– /universidad/departamento[@tipo], se obtienen los datos de los departamentos que tengan el atributo tipo. Si ponemos data(/universidad/departamento[@tipo]), nos devuelve los datos sin las etiquetas.

– /universidad/departamento/empleado[@salario], se obtienen los datos de los empleados que tengan el atributo salario.

– /universidad/departamento[@telefono="990033"], se obtienen los datos del departamento cuyo teléfono es 990033. Si ponemos data(/universidad/departamento[@telefono="990033"]), devuelve lo mismo, pero sin las etiquetas de los elementos.

– /universidad/departamento[@telefono="990033"]/nombre/text(), se obtienen el nombre de departamento cuyo teléfono es 990033.

– //departamento[@tipo='B'], se obtienen los datos de los departamentos cuyo tipo es B.

– /universidad/departamento[@tipo="A"]/empleado, se obtienen los datos de los empleados de los departamentos del tipo A.

– /universidad/departamento/empleado[@salario>"2100"], se obtienen los datos de los empleados cuyo salario es mayor de 2100.

– /universidad/departamento/empleado[@salario>"2100"]/nombre/text(), se obtienen los nombres de los empleados cuyo salario es mayor de 2100.

– /universidad/departamento/empleado[@salario>"2100"] /concat(nombre,' ', @salario), se obtienen los datos concatenados del nombre de empleado y su salario, de los empleados cuyo salario es mayor de 2100.

– /universidad/departamento[@tipo="A"]/count(empleado), devuelve el número de empleados que hay en los departamentos de tipo=A.

– /universidad/departamento[@tipo="A"]/concat(nombre,' ',count(empleado)), devuelve por cada departamento del tipo=A, la concatenación de su nombre y el número de empleados.

– /universidad/departamento/concat(nombre,' ',count(empleado)), devuelve el número de empleados por cada departamento

– sum(//empleado/@salario), devuelve la suma total del salario de todos los empleados, hace lo mismo que esto: sum(/universidad/departamento/empleado/@salario)

– /universidad/departamento/concat(nombre,' Total=', sum(empleado/@salario)), obtiene por cada departamento la concatenación de su nombre y el total salario.

– min(//empleado/@salario), devuelve el salario mínimo de todos los empleados.

– /universidad/departamento/concat(nombre,' Minimo=', min(empleado/@salario))

/universidad/departamento/concat(nombre,' Máximo=', max(empleado/@salario))

La primera obtiene por cada departamento la concatenación de su nombre y el mínimo salario, y la segunda el máximo salario.

– /universidad/departamento/concat(nombre,' Media=', avg(empleado/@salario)), obtiene por cada departamento la concatenación de su nombre y la media de salario.

– /universidad/departamento[count(empleado)>3], obtiene los datos de departamentos con más de 3 empleados. /universidad/departamento[count(empleado)>3]/nombre/text(), en este caso devuelve el nombre de los departamentos con más de 3 empleados.

– /universidad/departamento[@tipo="A" and count(empleado)>2]/nombre/text(), devuelve el nombre de los departamentos de tipo A y con más de 2 empleados.

ACTIVIDAD 5.2

Dado el documento ***sucursales.xml*** que se encuentra dentro de la colección ***ColeccionPruebas***. Este documento contiene los datos de las sucursales de un banco. Por cada sucursal tenemos el teléfono, el código, el director de la sucursal, la población y las cuentas de la sucursal. Y por cada cuenta tenemos el tipo de cuenta AHORRO o PENSIONES, el nombre de la cuenta, el número, el saldo haber y el saldo debe. Estos datos son:

```
<sucursales>
  <sucursal telefono="xxxxxxxx" codigo="xxxx">
      <director>xxxxxxxxxxxxxxx</director>
            <poblacion>xxxxxxxxxx</poblacion>
            <cuenta tipo="xxxxxxxxx">
                  <nombre>xxxx</nombre>
                  <numero>xxxx</numero>
                  <saldohaber>xxxxxx</saldohaber>
                  <saldodebe>xxxxx</saldodebe>
            </cuenta>
            . . . . . .
  </sucursal>
  . . . . . .
</sucursales>
```

Realiza las siguientes consultas XPath:

- Obtener los datos de las cuentas bancarias cuyo tipo sea AHORRO.

- Obtener por cada sucursal la concatenación de su código, y el número de cuentas del tipo AHORRO que tiene.
- Obtener las cuentas de tipo PENSIONES de la sucursal con código SUC3.
- Obtener por cada sucursal la concatenación de los datos, código sucursal, director, y total saldo haber.
- Obtener todos los elementos de las sucursales con más de 3 cuentas.
- Obtener todos los elementos de las sucursales con más de 3 cuentas del tipo AHORRO.
- Obtener los nodos del director y la población de las sucursales con más de 3 cuentas.
- Obtener el número de sucursales cuya población sea Madrid.
- Obtener por cada sucursal, su código y la suma de las aportaciones de las cuentas del tipo PENSIONES.
- Obtener los nodos número de cuenta, nombre de cuenta y el saldo haber de las cuentas con saldo haber mayor de 10000.
- Obtener por cada sucursal con más de 3 cuentas del tipo AHORRO, su código y la suma del saldo debe de esas cuentas.

5.5.2.3. Axis XPath

Un AXIS o eje, especifica la dirección que se va a evaluar, es decir, si nos vamos a mover hacia arriba en la jerarquía o hacia abajo, si va a incluir el nodo actual o no, es decir, define un conjunto de nodos relativo al nodo actual. Los nombres de los ejes son los siguientes:

Nombre de Axis	Resultado
ancestor	Selecciona los antepasados (padres, abuelos, etc.) del nodo actual
ancestor-or-self	Selecciona los antepasados (padres, abuelos, etc.) del nodo actual y el nodo actual en sí
attribute	Selecciona los atributos del nodo actual
child	Selecciona los hijos del nodo actual
descendant	Selecciona los descendientes (hijos, nietos, etc.) del nodo actual
descendant-or-self	Selecciona los descendientes (hijos, nietos, etc.) del nodo actual y el nodo actual en sí
following	Selecciona todo el documento después de la etiqueta de cierre del nodo actual
following-sibling	Selecciona todos los hermanos que siguen al nodo actual
parent	Selecciona el padre del nodo actual
self	Selecciona el nodo actual

La sintaxis para utilizar ejes es la siguiente: ***Nombre_de_eje::nombre_nodo[expresión]***

En la ventana del ***Diálogo de consulta*** del *Cliente de Administración de eXist*, se puede ver en la parte de *Resultados*, y dentro de la pestaña *Trace*, la traza de las consultas, y en la traza podemos observar los ejes utilizados. Véase Figura 5.15.

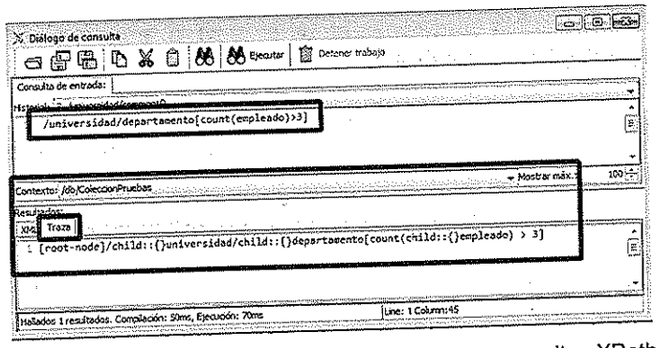

Figura 5.15. Ejes en la traza de ejecución de las consultas XPath.

Ejemplos con Axis XPath :

– /universidad/child::*, es lo mismo que /child::universidad/child::element(). Devuelve todos los hijos de universidad, es decir, los nodos de los departamentos.

– /universidad/departamento/descendant::*, devuelve los descendientes del nodo departamento, esto hace lo mismo:

/child::universidad/child::departamento/descendant::element()

– /universidad/departamento/descendant::empleado, devuelve los nodos empleado descendientes de los nodos departamento.

– /universidad/descendant::nombre, devuelve todos los elementos nombre descendientes de universidad, tanto nombres de departamentos como de empleados. Si ponemos esto nos devuelve el texto del nombre: data(/universidad/descendant::nombre).

– /universidad/departamento/following-sibling::*, selecciona todos los hermanos de departamento a partir del primero, siguiendo el orden en el documento.

Si ponemos /universidad/departamento[2]/following-sibling::*, selecciona todos los hermanos de departamento a partir del segundo.

– //empleado/following-sibling::node(), selecciona todos los hermanos de los elementos empleado que encuentre en el contexto.

En este caso //empleado/following-sibling::empleado[@salario>2100], selecciona todos los hermanos de los elementos empleado que tienen el salario >2100.

– //empleado[nombre="Ana García"]/following-sibling::*, selecciona los nodos hermanos de Ana García.

– //empleado[nombre="Ana García"]/following-sibling::empleado/nombre/text(), selecciona los nombres de los empleados hermanos de Ana García.

– //empleado[nombre="Ana García"]/following-sibling:: empleado[puesto="Profesor"]/nombre/text(), selecciona los nombres de los empleados hermanos de Ana García que son profesores.

– //empleado/parent::departamento/nombre, selecciona el nombre de los padres de los elementos empleado.

– //empleado[nombre="Ana García"]/parent::departamento/nombre, selecciona el nombre del padre de la empleada Ana García.

– /descendant::departamento[1], selecciona los descendientes del departamento que ocupa la posición 3 en el documento.

– /child::universidad/child::departamento[count(child::empleado)>3], es lo mismo que /universidad/departamento[count(empleado)>3], obtiene los departamentos con más de 3 empleados.

– /child::universidad/child::departamento/child::nombre, obtiene las etiquetas con los nombres de los departamentos. Es lo mismo que /universidad/departamento/nombre, y que data(/universidad/departamento/nombre).

– /child::universidad/child::departamento/child::nombre/child::text(), obtiene los nombres de los departamentos.

– /child::universidad/child::departamento[attribute::tipo = "B"] [count(child::empleado) >= 2]/child::nombre/child::text() , devuelve el nombre de los departamentos de tipo B y con 2 o más empleados. Es lo mismo que poner /universidad/departamento[@tipo="B" and count(empleado)>=2]/nombre/text().