

Aplicación práctica de estructuras jerárquicas para la generación de meta datos usando características de Oracle XML DB

Héctor Suárez Barenca

*Nada como la abstracción para desaparecer a la intuición
Shai Simonson*

Exploraremos una relación práctica entre las estructuras jerárquicas y las estructuras relacionales, enfocándonos principalmente en el uso de meta datos aprovechando características de Oracle XML DB. La idea de modelar estructuras de datos con múltiples elementos (incluso anidados), por ejemplo meta datos asociados a tipos de documentos de entrada supone propiedades heterogéneas debido a su naturaleza, como muestra mencionamos que un tipo de documento *fotografía* no se describe de la misma forma en la que se podría describir a un tipo de documento *informe general de avances* debido a que no cuentan con el mismo número y tipo de meta datos.

Introducción

La actividad de modelar estructuras de datos con múltiples elementos (incluso anidados) como meta datos de un objeto, digamos, tipos de documento, supone propiedades heterogéneas debido a su naturaleza. Pensemos por ejemplo en un tipo de documento *fotografía*, es claro que éste no se describe de la misma forma que se podría describir a un tipo de documento *informe general de avances*. Entonces, si en una organización se requiere un sistema que permita almacenar todos

los tipos de documentos que en ésta se procesan, y como es de esperarse, estos tipos de documentos son desconocidos ¿cómo podríamos representar estructuras jerárquicas usando bases de datos relacionales?, ¿cómo le hacemos para crear los meta datos de un objeto cuando no conocemos el número de elementos representan a éste?, ¿cómo buscamos en estos meta datos si el número y los tipos de datos de los elementos que los representan son desconocidos?.

Problemática

Imaginemos que deseamos meta datos de campos de formularios de captura o tipos de documentos, digamos que tenemos dos tipos de documentos que se describen de la siguiente manera:

Fotografía
Tamaño
Colores
Contenido
Fecha en la que fue tomada
Autor o fuente

Oficio
Número
Procedencia
Asunto
Fecha

Figura 1. Meta datos

Se observa que el número de elementos que describen al objeto puede variar, por lo que se espera que a mayor número de tipos de objetos (en este caso tipos de documentos), la cantidad de elementos para cada objeto también crezca y además sea diferente a los ya definidos, aunque habrá casos en los que un elemento de los meta

datos pueda ser usado por uno o más objetos (imaginemos por ejemplo el meta dato *nombre*).

La primera solución que muy probablemente aparecería por nuestras mentes sería la de crear una tabla con el máximo número de columnas permitidas y asociar una columna de la tabla con un elemento de uno solo un objeto, por supuesto que esta solución es poco efectiva y traería consigo algunos problemas asociados a la búsqueda y creación de *n* objetos, dado que estaría limitada por la cantidad de columnas máximas permitidas en la tabla. Otra posible alternativa de solución es definir una tabla de la siguiente forma:

Objeto
Id, primary key
Descripcion, varchar(64)
Contenido
Fecha en la que fue tomada
Autor o fuente

Meta_dato
Id, primary key
Descripcion, varchar(64)

Valor
Id, primary key
Meta_dato_id, foreign key
Valor, varchar(1024)
Instancia, integer
Objeto_id, foreign key

Figura 2. Representación del modelo

Por simplicidad en este momento, supondremos que la tabla *meta_dato* no existe y en su caso consideraremos lo siguiente:

Valor
Id, primary key
Nombre_meta_dato, varchar(64)
Valor, varchar(1024)
Instancia, integer
Objeto_id, foreign key

Figura 3. Tabla simplificada

Teniendo el modelo, probemos poblando las tablas como se muestra a continuación:

Objeto	
Id	Descripción
1	Oficio
2	Acta de nacimiento

Figura 4. Tabla objeto poblada con dos registros

Valor				
Id	Nombre_meta_dato	Valor	Instancia	Objeto_id
1	A	10	1	1
2	B	12	1	1
3	C	15	1	1
4	A	20	2	1
5	B	35	2	1
6	C	12	2	1
7	A	20	3	1
8	B	40	3	1
9	C	12	3	1
10	D	60	1	2

Figura 5. Tabla valor poblada con cuatro registros

Observemos que es posible almacenar m datos para n número de objetos (en este caso tipos de documentos) sin importar el número de elementos meta dato de cada objeto ni el número de registro por cada objeto.

```
select nombre_meta_dato, valor
from valor
where objeto_id=1
      and nombre_meta_dato = 'a' and valor=20
      and ...¿?
group by nombre,valor
```

Como se logra ver, esta solución no nos lleva a mucho, con algunos cambios podremos lograr algo como:

```
select c.instancia
from (
select instancia
from valor
where objeto_id=1
and nombre_meta_dato= 'a' and valor=20
) t, valor c
where objeto_id=1
and c. nombre_meta_dato ='c' and valor=12
and t.instancia=c.instancia;
```

Parecería que esta sería una posible solución pero si observamos con detenimiento podremos notar que es una solución parcial (y poco “elegante”), ¿Qué pasaría si son más campos en *objeto*?, ¿Y si el criterio de búsqueda es por más campos?, las cosas empiezan a complicarse un poco. ¿Cómo atacaremos este problema entonces?.

Afortunadamente Oracle ha introducido desde hace ya varios años una característica muy poderosa a `select xmlelement("documento",`

Respuesta

Teniendo el modelo y los datos, intentemos ahora consultar los registros donde el *objeto* sea un *oficio* y el campo $a=20$ y $c=12$ (recordar que son meta datos). ¿Cómo muy probablemente sería el query?

su base de datos llamada XML DB. En Pocas palabras, Oracle XML DB aprovecha las ventajas de la tecnología de bases de datos relacionales más las ventajas de la tecnología XML.

Primero convertiremos renglones de la tabla *valor* a elementos y atributos XML, de esta forma el resultado será una estructura jerárquica XML, teniendo ésta, estaremos preparados para buscar en ella usando XPATH.

```
xmlattributes( o.descripcion as "nombre", instancia as "instancia" ),
xmlagg(xmlelement( "campo" , xmlattributes(nombre_meta_dato as "nombre"),valor)))
from valor v, objeto o
where objeto_id=1
and v.objeto_id=o.id
group by instancia, o.descripcion
```

Lo que produce:

```
<documento nombre="Oficio" instancia="1">
  <campo nombre="a">10</campo>
  <campo nombre="b">12</campo>
  <campo nombre="c">15</campo>
</documento>
<documento nombre="Oficio" instancia="2">
  <campo nombre="a">20</campo>
  <campo nombre="b">35</campo>
  <campo nombre="c">12</campo>
</documento>
<documento nombre="Oficio" instancia="3">
  <campo nombre="a">20</campo>
  <campo nombre="b">20</campo>
  <campo nombre="c">12</campo>
</documento>
```

Podríamos también generar elementos en lugar de atributos usando el siguiente query:

```
select xmlelement( "documento",
xmlattributes( o.objeto as "nombre", instancia as "instancia" ),
xmlagg(
xmlelement( "campo" ,
xmlelement("nombre",nombre_meta_dato),
xmlelement("valor",valor)) ) )
from objeto o, valor v
where objeto_id=1
and v.objeto_id=o.id
group by instancia
```

Lo que daría como resultado:

```

<documento nombre="Oficio" instancia="1">
  <campo><nombre>a</nombre><valor>10</valor></campo>
  <campo><nombre>b</nombre><valor>12</valor></campo>
  <campo><nombre>c</nombre><valor>15</valor></campo>
</documento>
<documento nombre="Oficio" instancia="2">
  <campo><nombre>a</nombre><valor>20</valor></campo>
  <campo><nombre>b</nombre><valor>35</valor></campo>
  <campo><nombre>c</nombre><valor>12</valor></campo>
</documento>
<documento nombre="Oficio" instancia="3">
  <campo><nombre>a</nombre><valor>20</valor></campo>
  <campo><nombre>b</nombre><valor>20</valor></campo>
  <campo><nombre>c</nombre><valor>12</valor></campo>
</documento>

```

Hecho esto, se podrían efectuar búsquedas usando XPATH.

Conclusiones

El hecho de usar tecnologías como Oracle XML DB también implicará tomar consideraciones sobre el modelo a usar, en el caso aquí expuesto habrá que analizar la viabilidad de la creación de vistas y el impacto de éstas cuando se manejan grandes cantidades de datos (millones de registros por ejemplo); es un hecho que el uso de memoria podrá incrementarse por la conversión masiva de información (datos relacionales a jerárquicos) y su consulta en línea, otra buena idea sería crear objetos basados totalmente en XSD, o tan solo en XML (sin XSDs).

Glosario

Meta datos. Dato que describe información. Información que caracteriza el quién, qué, dónde y cómo, relacionado a una colección de datos.

XML (Extensible Markup Language). Es una forma flexible de crear formatos comunes de información y compartir ambos, el formato y los datos a través de distintos medios como intranets, la red, etc.

Oracle XML DB. Es una característica del manejador de bases de datos Oracle que es 100% compatible con el modelo de datos W3C XML que provee nuevos métodos estándares para acceder, consultar y navegar en documentos XML. En resumen, Oracle XML DB aprovecha las ventajas de la tecnología de bases de datos relacionales más las ventajas de la tecnología XML.

XPATH. Es un lenguaje que describe la forma de localizar y procesar elementos en documentos XML usando una sintaxis de acceso basada una ruta de la estructura lógica o jerárquica de un documento.

XSD (XML Schema Definition). Es una recomendación del consorcio W3C (World Wide Web Consortium), que especifica cómo formalmente describir elementos en un documento.

XML. En general, un esquema es una representación abstracta de las características de un objeto y su relación con otros objetos.

Referencias

Anónimo, Oracle® XML DB Developer's Guide 10g Release 2 (10.2)

Vadim Tropashko, 2005. Nested intervals Tree Encoding with continued fractions

Vadim Tropashko, 2005. Nested intervals with Farey fractions

Héctor Suárez Barenca es el Director del Área de Investigación y Desarrollo de Datateam Consulting, es el encargado de estudiar, conocer e indicar la tecnología que utilizará el área y definir la arquitectura del software desarrollado por la empresa. Cuenta con una amplia experiencia en el desarrollo y definición de arquitectura de sistemas Web en plataforma Java. Estudio Ingeniería Electrónica en el Instituto Politécnico Nacional, ha ganado diferentes premios en el área de programación, es experto en desarrollo Java y manejo de bases de datos.

E-mail: hector.suarez@datateam.com.mx