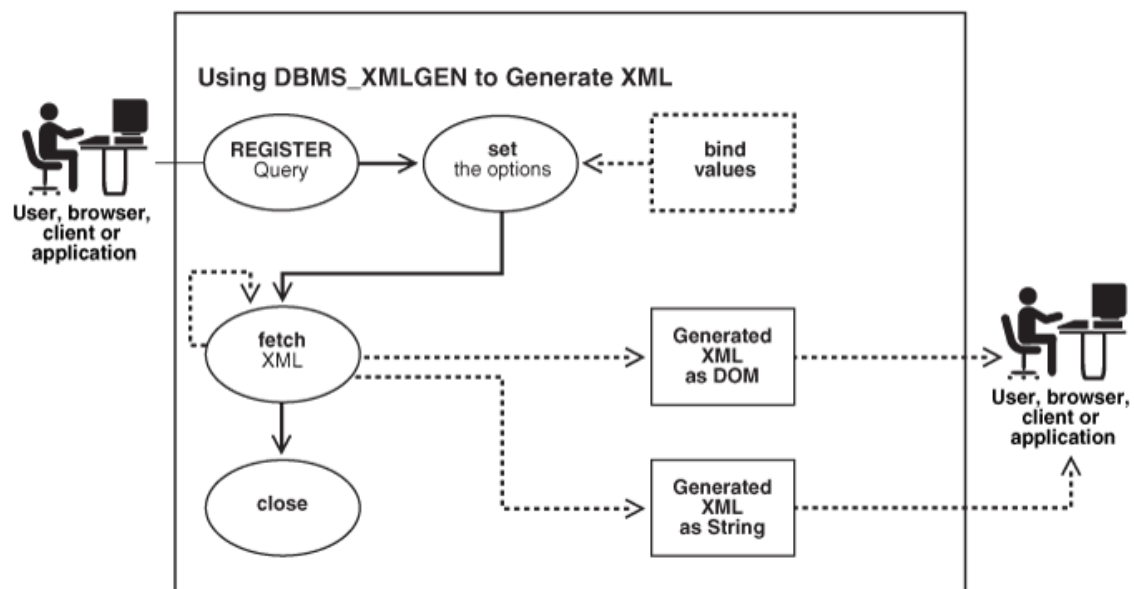


Generación de datos XML usando DBMS_XMLGEN





Usando el paquete PL/SQL DBMS_XMLGEN

El paquete PL/SQL **DBMS_XMLGEN** crea documentos XML a partir de los resultados de una consulta SQL. Recupera un documento XML como un valor CLOB o XMLType.

Los pasos son los siguientes:

1. Obtener el contexto del paquete proporcionando una consulta SQL y llamando a la función PL/SQL **newContext**.
2. Pasar el contexto a todos los procedimientos o funciones del paquete para configurar las distintas opciones. Por ejemplo, para establecer el nombre del elemento ROW, use **setRowTag(ctx)**, donde ctx es el contexto obtenido de la llamada newContext anterior.
3. Obtener el resultado XML, utilizando la función PL/SQL **getXML** o **getXMLType**. Podemos establecer el número máximo de filas que se recuperarán mediante el procedimiento PL/SQL **setMaxRows**, pudiendo llamar a cualquiera de estas funciones repetidamente, recuperando hasta el número máximo de filas para cada llamada. Estas funciones devuelven datos XML (como un valor CLOB y como una instancia de XMLType, respectivamente), a menos que no se recuperen filas. En ese caso, estas funciones devuelven NULL. Para determinar cuántas filas se recuperaron, use la función PL/SQL **getNumRowsProcessed**.
4. Puede restablecer la consulta para comenzar de nuevo y repetir el paso 3.
5. Llame al procedimiento de PL/SQL **closeContext** para liberar los recursos asignados previamente.



Junto con una consulta SQL, el método PL/SQL **DBMS_XMLGEN.getXML()** normalmente devuelve un resultado similar al siguiente, como un valor CLOB:

```
<?xml version="1.0"?>
<ROWSET>
  <ROW>
    <EMPLOYEE_ID>100</EMPLOYEE_ID>
    <FIRST_NAME>Steven</FIRST_NAME>
    <LAST_NAME>King</LAST_NAME>
    <EMAIL>SKING</EMAIL>
    <PHONE_NUMBER>515.123.4567</PHONE_NUMBER>
    <HIRE_DATE>17-JUN-87</HIRE_DATE>
    <JOB_ID>AD_PRES</JOB_ID>
    <SALARY>24000</SALARY>
    <DEPARTMENT_ID>90</DEPARTMENT_ID>
  </ROW>
  <ROW>
    <EMPLOYEE_ID>101</EMPLOYEE_ID>
    <FIRST_NAME>Neena</FIRST_NAME>
    <LAST_NAME>Kochhar</LAST_NAME>
    <EMAIL>NKOCHHAR</EMAIL>
    <PHONE_NUMBER>515.123.4568</PHONE_NUMBER>
    <HIRE_DATE>21-SEP-89</HIRE_DATE>
    <JOB_ID>AD_VP</JOB_ID>
    <SALARY>17000</SALARY>
    <MANAGER_ID>100</MANAGER_ID>
    <DEPARTMENT_ID>90</DEPARTMENT_ID>
  </ROW>
</ROWSET>
```

La asignación predeterminada entre los datos de la BD y datos XML es la siguiente:

- Cada fila devuelta por la consulta SQL se asigna a un elemento XML con el nombre de elemento predeterminado ROW.
- Cada columna devuelta por la consulta SQL se asigna a un elemento hijo del elemento ROW.
- Todo el resultado está envuelto en un elemento ROWSET.

Los nombres de elementos ROW y ROWSET se pueden reemplazar con los nombres que elija, utilizando los procedimientos DBMS_XMLGEN **setRowTagName** y **setRowSetTagName**, respectivamente.



Functions y Procedures del paquete DBMS_XMLGEN

```
SUBTYPE ctxHandle IS NUMBER
```

El identificador de contexto utilizado por todas las funciones.

```
DECLARE  
  qryCtx DBMS_XMLGEN.ctxHandle;
```

```
newContext(  
  queryString IN VARCHAR2)
```

Devuelve un nuevo contexto.

Parámetro: queryString (IN): la cadena que contiene la consulta, cuyo resultado debe convertirse a XML.

Devoluciones: Identificador de contexto. Llamar a esta función primero para obtener un identificador que pueda usar en getXML y otras funciones para recuperar el XML del resultado.

```
setRowTag(ctx IN ctxHandle,  
          rowTag IN VARCHAR2);
```

Parámetros:

ctx (IN): el identificador de contexto obtenido de la llamada newContext.

rowTag (IN): el nombre del elemento FILA. Un valor NULL para rowTag indica que no desea que el elemento ROW esté presente.

Llame a este procedimiento para establecer el nombre del elemento ROW, si no desea que aparezca el nombre ROW predeterminado. También puede establecer rowTag en NULL para suprimir el elemento ROW en sí.



```
setRowSetTag(ctx IN ctxHandle,  
             rowSetTag IN VARCHAR2);
```

Parámetros:

ctx (IN): el identificador de contexto obtenido de la llamada newContext.

rowSetTag (IN): el nombre del elemento raíz del documento que se utilizará en la salida. Un valor NULL para rowSetTag indica que no desea que el elemento ROWSET esté presente.

Llame a este procedimiento para establecer el nombre del elemento raíz del documento, si no desea que se utilice el nombre predeterminado ROWSET. Puede establecer rowSetTag en NULL para suprimir la impresión del elemento raíz del documento.

```
getXML(ctx IN ctxHandle,  
       dtdOrSchema IN number:= NONE)  
RETURN clob;
```

Parámetros:

ctx (IN): el identificador de contexto obtenido al llamar a newContext.

dtdOrSchema (IN): si se debe generar un esquema DTD o XML. Este parámetro no es compatible.

Devoluciones: Un CLOB temporal que contiene el resultado..

```
closeContext(ctx IN ctxHandle);
```

Parámetro: ctx (IN) - el controlador de contexto para cerrar.

Cierra todos los recursos asociados con este identificador. Después de esto, no puede usar el identificador para ninguna otra llamada de función DBMS_XMLGEN.



```
setNullHandling(ctx IN ctxHandle,  
                flag IN NUMBER);
```

El valor flag que vamos a usar es la constante EMPTY_TAG: = 2.

Esto establece, por ejemplo, <foo />.

Ejemplo 1

Vamos a trabajar con una BD que contiene las tablas **EMPLE** y **DEPART** cuya descripción y contenido tenemos a continuación:

DEPART					
Table	Data	Indexes	Model	Constraints	Grants
Query Count Rows Insert Row					
EDIT	DEPT_NO	DNOMBRE	LOC		
	10	CONTABILIDAD	SEVILLA		
	20	INVESTIGACION	MADRID		
	30	VENTAS	BARCELONA		
	40	PRODUCCION	BILBAO		
				row(s) 1 - 4 of 4	

EMPLE					
Table	Data	Indexes	Model	Constraints	Grants
Add Column Modify Column Rename Column Drop Column Rename Copy Drop T					
Column Name	Data Type	Nullable	Default	Primary Key	
EMP_NO	NUMBER(4,0)	No	-	1	
APELLIDO	VARCHAR2(10)	Yes	-	-	
OFICIO	VARCHAR2(10)	Yes	-	-	
DIR	NUMBER(4,0)	Yes	-	-	
FECHA_ALT	DATE	Yes	-	-	
SALARIO	NUMBER(10,0)	Yes	-	-	
COMISION	NUMBER(10,0)	Yes	-	-	
DEPT_NO	NUMBER(2,0)	No	-	-	
					1 - 8



EMPLE

Query Count Rows Insert Row

EDIT	EMP_NO	APELLIDO	OFICIO	DIR	FECHA_ALT	SALARIO	COMISION	DEPT_NO
	7839	REY	PRESIDENTE	-	11/17/1981	650000	-	10
	7566	JIMENEZ	DIRECTOR	7839	04/02/1981	386750	-	20
	7698	NEGRO	DIRECTOR	7839	05/01/1981	370500	-	30
	7782	CEREZO	DIRECTOR	7839	06/09/1981	318500	-	10
	7788	GIL	ANALISTA	7566	11/09/1981	390000	-	20
	7902	FERNANDEZ	ANALISTA	7566	12/03/1981	390000	-	20
	7499	ARROYO	VENDEDOR	7698	02/20/1980	208000	39000	30
	7521	SALA	VENDEDOR	7698	02/22/1981	162500	65000	30
	7654	MARTIN	VENDEDOR	7698	09/29/1981	162500	182000	30
	7844	TOVAR	VENDEDOR	7698	09/08/1981	195000	0	30
	7900	JIMENO	EMPLEADO	7698	12/03/1981	123500	-	30
	7369	SANCHEZ	EMPLEADO	7902	12/17/1980	104000	-	20
	7876	ALONSO	EMPLEADO	7788	09/23/1981	143000	-	20
	7934	MUNOZ	EMPLEADO	7782	01/23/1982	169000	-	10

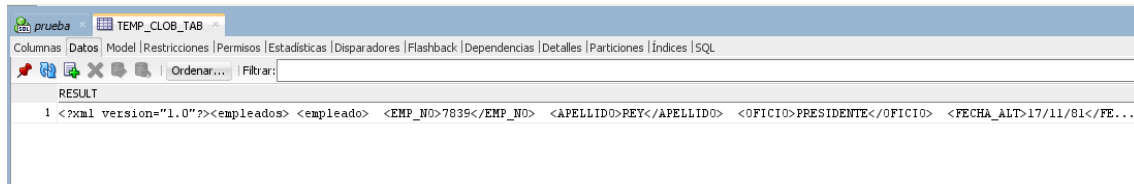
row(s) 1 - 14 of 14

Vamos a generar un XML con todos los datos de los empleados de la tabla EMPLE:

```
1  -- Primer ejemplo ---
2
3  DECLARE
4      qryCtx DBMS_XMLGEN.ctxHandle;
5      result CLOB;
6  BEGIN
7      qryCtx := DBMS_XMLGEN.newContext(
8          'SELECT * FROM emple');
9
10     -- Set the name of the document root element. The default name is ROWSET
11     DBMS_XMLGEN.setRowSetTag(qryCtx, 'empleados');
12
13     -- Set the row header to be EMPLOYEE
14     DBMS_XMLGEN.setRowTag(qryCtx, 'empleado');
15
16     -- Get the result
17     result := DBMS_XMLGEN.getXML(qryCtx);
18     INSERT INTO temp_clob_tab VALUES(result);
19
20     --Close context
21     DBMS_XMLGEN.closeContext(qryCtx);
22 END;
```



El contenido de la tabla **temp_clob_tab**:



RESULT
1 <?xml version="1.0"?><empleados> <empleado> <EMP_NO>7839</EMP_NO> <APELLIDO>REY</APELLIDO> <OFICIO>PRESIDENTE</OFICIO> <FECHA_ALT>17/11/81</FE...

Y si exportamos ese contenido a **xml**:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <empleados>
3 <empleado>
4   <EMP_NO>7839</EMP_NO>
5   <APELLIDO>REY</APELLIDO>
6   <OFICIO>PRESIDENTE</OFICIO>
7   <FECHA_ALT>17/11/81</FECHA_ALT>
8   <SALARIO>65000</SALARIO>
9   <DEPT_NO>10</DEPT_NO>
10 </empleado>
11 <empleado>
12   <EMP_NO>7566</EMP_NO>
13   <APELLIDO>JIMENEZ</APELLIDO>
14   <OFICIO>DIRECTOR</OFICIO>
15   <DIR>7839</DIR>
16   <FECHA_ALT>02/04/81</FECHA_ALT>
17   <SALARIO>386750</SALARIO>
18   <DEPT_NO>20</DEPT_NO>
19 </empleado>
20 <empleado>
21   <EMP_NO>7698</EMP_NO>
22   <APELLIDO>NEGRO</APELLIDO>
23   <OFICIO>DIRECTOR</OFICIO>
24   <DIR>7839</DIR>
25   <FECHA_ALT>01/05/81</FECHA_ALT>
26   <SALARIO>370500</SALARIO>
27   <DEPT_NO>30</DEPT_NO>
28 </empleado>
29 <empleado>
30   <EMP_NO>7782</EMP_NO>
31   <APELLIDO>CEREZO</APELLIDO>
32   <OFICIO>DIRECTOR</OFICIO>
33   <DIR>7839</DIR>
34   <FECHA_ALT>09/06/81</FECHA_ALT>
35   <SALARIO>318500</SALARIO>
36   <DEPT_NO>10</DEPT_NO>
37 </empleado>
```




Ahora vamos a seguir completando el ejemplo realizando los siguientes pasos:

Paso 1

Conseguir que, aquellos valores nulos de la tabla EMPLE (por ejemplo, dir y comisión) que no aparecen en el XML, aparezcan con elementos vacíos.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <EMPLEADOS>
3 <EMPLEADO>
4 <EMP_NO>7839</EMP_NO>
5 <APELLIDO>REY</APELLIDO>
6 <OFICIO>PRESIDENTE</OFICIO>
7 <DIR/>
8 <FECHA_ALT>17/11/81</FECHA_ALT>
9 <SALARIO>650000</SALARIO>
10 <COMISION/>
11 <DEPT_NO>10</DEPT_NO>
12 </EMPLEADO>
13 <EMPLEADO>
14 <EMP_NO>7566</EMP_NO>
15 <APELLIDO>JIMENEZ</APELLIDO>
16 <OFICIO>DIRECTOR</OFICIO>
17 <DIR>7839</DIR>
18 <FECHA_ALT>02/04/81</FECHA_ALT>
19 <SALARIO>386750</SALARIO>
20 <COMISION/>
21 <DEPT_NO>20</DEPT_NO>
22 </EMPLEADO>
```

Paso 2

Obtener XML con datos de ambas tablas:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <EMPLEADOS>
3 <empleado>
4 <EMP_NO>7839</EMP_NO>
5 <APELLIDO>REY</APELLIDO>
6 <DEPT_NO>10</DEPT_NO>
7 <DNOMBRE>CONTABILIDAD</DNOMBRE>
8 </empleado>
9 <empleado>
10 <EMP_NO>7566</EMP_NO>
11 <APELLIDO>JIMENEZ</APELLIDO>
12 <DEPT_NO>20</DEPT_NO>
13 <DNOMBRE>INVESTIGACION</DNOMBRE>
14 </empleado>
15 <empleado>
16 <EMP_NO>7698</EMP_NO>
17 <APELLIDO>NEGRO</APELLIDO>
18 <DEPT_NO>30</DEPT_NO>
19 <DNOMBRE>VENTAS</DNOMBRE>
20 </empleado>
21 <empleado>
22 <EMP_NO>7782</EMP_NO>
23 <APELLIDO>CEREZO</APELLIDO>
24 <DEPT_NO>10</DEPT_NO>
25 <DNOMBRE>CONTABILIDAD</DNOMBRE>
26 </empleado>
```


La palabra clave **MULTISET** de Oracle SQL trata a los empleados que trabajan en el departamento como una lista, y **CAST** asigna al tipo apropiado.

Se crea una instancia de departamento utilizando el constructor **dept_tipo**, y las rutinas DBMS_XMLGEN crean los datos XML para la instancia de objeto.

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <ROWSET>
3 <ROW>
4 <DEPARTAMENTO-dept_no="10">
5 | <DNOMBRE>CONTABILIDAD</DNOMBRE>
6 <EMPLIST>
7 <EMP_TIPO-emp_no="7839">
8 | <APELLIDO>REY</APELLIDO>
9 </EMP_TIPO>
10 <EMP_TIPO-emp_no="7782">
11 | <APELLIDO>CEREZO</APELLIDO>
12 </EMP_TIPO>
13 <EMP_TIPO-emp_no="7934">
14 | <APELLIDO>MUNOZ</APELLIDO>
15 </EMP_TIPO>
16 </EMPLIST>
17 </DEPARTAMENTO>
18 </ROW>
19 <ROW>
20 <DEPARTAMENTO-dept_no="20">
21 | <DNOMBRE>INVESTIGACION</DNOMBRE>
22 <EMPLIST>
23 <EMP_TIPO-emp_no="7566">
24 | <APELLIDO>JIMENEZ</APELLIDO>
25 </EMP_TIPO>
26 <EMP_TIPO-emp_no="7788">
27 | <APELLIDO>GIL</APELLIDO>
28 </EMP_TIPO>
29 <EMP_TIPO-emp_no="7902">
30 | <APELLIDO>FERNANDEZ</APELLIDO>
31 </EMP_TIPO>
32 <EMP_TIPO-emp_no="7369">
33 | <APELLIDO>SANCHEZ</APELLIDO>
34 </EMP_TIPO>
35 <EMP_TIPO-emp_no="7876">
36 | <APELLIDO>ALONSO</APELLIDO>
37 </EMP_TIPO>
38 </EMPLIST>
39 </DEPARTAMENTO>
```