

PRÁCTICA 05.

Objetivo: Uso del lenguaje SQL/XML para construir instancias XML de datos relacionales. Definición de tablas y columnas de tipo XMLType. Uso de funciones de manipulación sobre el tipo de dato XMLType.

Sesiones: 1 (2h) 18/5

Ejercicio:

Información sobre las instrucciones SQL/XML en Oracle

<http://www.mcs.csueastbay.edu/support/oracle/doc/10.2/appdev.102/b14259/xdm13gen.htm#sthref1494>

1. Utilizando el esquema de la BD "restaurante" se van a ejecutar las siguientes consultas para crear instancias XML. El objetivo es comprobar el funcionamiento de las funciones SQL/XML y de las incorporadas por Oracle.

1.1. Crea una instancia XML con el nombre, apellido y dni de cada empleado.

```
SELECT XMLELEMENT ("Emp",
    XMLELEMENT ("Nombre", e.nombre ),
    XMLELEMENT ("Apellido", e.apellido),
    XMLELEMENT("DNI", e.dni ) )
FROM empleado e;
```

1.2. Crea la instancia xml anterior, poniendo el DNI como atributo en vez de como elemento.

```
SELECT XMLELEMENT ("Emp", XMLATTRIBUTES(e.dni ),
    XMLELEMENT ("Nombre", e.nombre ),
    XMLELEMENT ("Apellido", e.apellido) )
FROM empleado e;
```

1.3. Crea una instancia XML para cada restaurante con el nombre y primer apellido del director concatenado en un solo elemento.

```
SELECT XMLELEMENT("Restaurante",
    XMLELEMENT("Código",r.id_res),
    XMLELEMENT("Director", e.nombre ||' '|| e.apellido))
FROM restaurante r, empleado e
where r.director=e.id_emple;
```

1.4. Crea una instancia XML para cada restaurante indicando el nº de reservas que tiene.

```

SELECT XMLELEMENT ("Restaurante",
    XMLELEMENT ("Código", r.id_res ),
    XMLELEMENT("Num_reservas",(select count(*) from reserva where
    reserva.id_res= r.id_res)))
FROM restaurante r;

```

- 1.5. Se ejecuta la consulta anterior pero utilizando XMLFOREST en vez de XMLELEMENT. Recordad que XMLFOREST no permite incluir atributos

```

SELECT XMLFOREST ( r.id_res AS "Código" ,
    (select count(*) from reserva where
    reserva.id_res= r.id_res) as Num_Reservas)
FROM restaurante r;

```

- 1.6. Crea una instancia XML que muestre todos los empleados de la cadena hotelera

```

SELECT XMLAGG (
    XMLELEMENT ("NombreEmpleado", nombre ||' '|| apellido || ' ' || apellido2 ))
FROM empleado;

```

- 1.7. Crea una instancia XML para cada ciudad con los restaurantes que tiene en ella.

```

SELECT
    XMLELEMENT("Resultado",
        XMLELEMENT("Ciudad",r.ciudad),
        XMLELEMENT("Restaurantes", XMLAGG (
            XMLELEMENT ("código", r.id_res )
            ORDER BY r.id_res)))
FROM restaurante r
group by r.ciudad

```

- 1.8. Ahora se obtiene la consulta anterior pero construyendo instancias XML bien formadas

```

SELECT SYS_XMLGen(
    XMLELEMENT("Resultado",
        XMLELEMENT("Ciudad",r.ciudad),
        XMLELEMENT("Restaurantes", XMLAGG (
            XMLELEMENT ("código", r.id_res )
            ORDER BY r.id_res))))
FROM restaurante r GROUP BY r.ciudad

```

- 1.9. Ahora haremos uso de XMLCOLATTVAL para construir instancias XML sin utilizar XMLELEMENT

```
SELECT XMLCOLATTVAL(e.nombre, e.apellido, e.dni as "identificador" )
FROM empleado e;
```

- 1.10. Si nuestro objetivo es crear un solo documento XML en vez de tantas instancias como filas devuelva la consulta, haremos uso de SYS_XMLAGG

```
SELECT SYS_XMLAGG (XMLCOLATTVAL(e.nombre, e.apellido, e.dni as
"identificador" )) FROM empleado e;
```

2. Creación de tabla con columna XMLType y uso del lenguaje de manipulación.

- 2.1. Se creará la tabla siguiente:

```
create table empleado_cv
(id_emple number(6,0) primary key,
cv xmltype)
```

- 2.2. Y se insertarán las instancias XML

```
insert into empleado_cv values (1004,
xmltype(
'<DatosPersonales>
<Nombre>Juan</Nombre>
<Apellido1>Fernandez</Apellido1>
<Apellido2>Miguel</Apellido2>
<FechaNac>01-01-2000</FechaNac>
<Direccion caracter="Trabajo">
<Calle>c/ San Juan 23</Calle>
<Localidad>Leganes</Localidad>
<CodigoPostal>28120</CodigoPostal>
<Pais>España</Pais>
</Direccion>
<Direccion caracter="Particular">
<Calle>c/ San Juan 25</Calle>
<Localidad>Vallecas</Localidad>
<CodigoPostal>28320</CodigoPostal>
<Pais>España</Pais>
</Direccion>
</DatosPersonales>'))
```

```
insert into empleado_cv values (1010,
xmltype(
'<DatosPersonales>
<Nombre>Nerea</Nombre>
<Apellido1>García</Apellido1>
<Apellido2>Maode</Apellido2>
<FechaNac>01-01-1989</FechaNac>
<Direccion caracter="Trabajo">
<Calle>c/ Azucenas 22</Calle>
<Localidad>Madrid</Localidad>
<CodigoPostal>28080</CodigoPostal>
<Pais>España</Pais>
</Direccion>
</DatosPersonales>'))
```

2.3. A continuación utilizaremos las instrucciones de manipulación:

2.3.1. Obtendremos la fecha de nacimiento de este empleado

```
select extract(cv,
              '/DatosPersonales/FechaNac') from empleado_cv;
```

2.3.2. Si en vez del nodo se quiere extraer su valor

```
select extractvalue(cv,
                    '/DatosPersonales/FechaNac') from empleado_cv;
```

2.3.3. Buscamos cuántos empleados son de Leganés

```
Select count(*) from empleado_cv
where existsNode(cv, '/DatosPersonales/Direccion[Localidad="Leganes"]') = 1;
```

2.3.4. Buscamos el nombre y la dirección de trabajo de los empleados

```
select extractvalue(cv, '/DatosPersonales/Nombre'),
       extract(cv, '/DatosPersonales/Direccion[@character="Trabajo"]')
from empleado_cv;
```

2.3.5. Buscamos las direcciones de los empleados de España. Ver la diferencia en el resultado al usar EXTRACT

```
select direcciones.getstringval() from empleado_cv e,
       table(XMLSequence(extract(e.cv, '/DatosPersonales/Direccion'))) direcciones
where existsNode( e.cv, '/DatosPersonales/Direccion[Pais="Espana"]') = 1;
```

```
select extract( e.cv, '/DatosPersonales/Direccion[Pais="Espana"]')
from empleado_cv e
```

2.3.6. La función XMLQuery solo funciona en versión de Oracle 10g, no en Express

```
Select XMLQuery(
        'for $i in /DatosPersonales
        return $i'
        PASSING cv RETURNING CONTENT) Personas
FROM empleado_cv;
```

2.3.7. Actualizamos la fecha de nacimiento del empleado 1004

```
UPDATE empleado_cv
SET cv = updateXML(cv,'/DatosPersonales/FechaNac/text()', '10-02-1970')
WHERE id_emple=1004;
```

2.3.8. Borrarnos de este mismo empleado el nodo direccion

```
UPDATE empleado_cv
SET cv = deleteXML(cv,'/DatosPersonales/Direccion')
WHERE id_emple=1004;
```

```
select * from empleado_cv --comprobamos el borrado
```

3. Registro de schema XML y creación de tabla basado en él.

3.1. Se registrará el schema cv.xsd

```
CREATE DIRECTORY xmldir AS 'c:/temp';
```

```
BEGIN
```

```
DBMS_XMLSCHEMA.registerSchema(
  SCHEMAURL => 'http://xmlns.oracle.com/xdb/cv.xsd',
  SCHEMADOC => bfilename('XMLDIR','cv.xsd'),
  CSID => nls_charset_id('AL32UTF8'));
```

```
END;
```

3.2. A continuación se analizará el schema XML registrado y se observarán los tipos de objeto creados.

3.3. Se insertará el fichero cv.xml y se consultará que se ha insertado en la tabla CV creada

```
Insert into CV values
```

```
(XMLType(bfilename('XMLDIR', 'cv.xml'),
          nls_charset_id('AL32UTF8')));
```

```
select (object_value).getstringval() from CV
```

3.4. Se realizarán las siguientes consultas con la función XMLQuery

```
Select XMLQuery(  
    'for $i in /CV/DatosPersonales  
    return $i'  
    PASSING object_value RETURNING CONTENT) DatosPersonales  
FROM CV;
```

```
Select XMLQuery(  
    'for $i in /CV  
    where $i/DatosPersonales/Apellido1 = "Zorrilla"  
    return <Detalle>  
        {$i/DatosPersonales/Direccion/Calle}  
        {$i/DatosPersonales/Direccion/Localidad}  
        <Español>{if ($i/DatosPersonales/Direccion/Pais = "Espana")  
            then "Si"  
            else "No"}  
        </Español>  
        </Detalle>'  
    PASSING object_value RETURNING CONTENT) DatosPersonales  
FROM CV;
```