Misceláneo de Nuevas Funcionalidades de Oracle Database 12c

Por Francisco Riccio

Introducción

A continuación se detallará una serie de nuevas funcionalidades disponibles en Oracle Database 12c que nos permiten ejecutar tareas cotidianas de una manera más sencilla y óptima al momento de diseñar, mantener y consultar nuestro modelos de base de datos.

Implementación

1. Creación de valores autogenerados a través de IDENTITY

Está opción nos permite tener un campo que tenga un valor autogenerado por cada fila ingresada; en versiones previas a Oracle Database 12c lo implementábamos a través de triggers y secuencias.

Ejemplo del uso de IDENTITY:

```
SQL> create table PERSONA
  2  (
  3   cod number GENERATED AS IDENTITY,
  4   nombre varchar(25),
  5   edad number
  6  );

Table created.
```

Internamente Oracle ha creado una secuencia:

```
SQL> select sequence_name,increment_by from user_sequences;

SEQUENCE_NAME        INCREMENT_BY
-------------------- ------------
ISEQ$$_91407                    1
```

Al realizar operaciones de INSERT no debemos ingresar algún valor sobre el campo autogenerado, ejemplo:

```
SQL> insert into persona (nombre,edad) values ('Francisco',32);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from persona;

       COD NOMBRE                         EDAD
---------- ------------------------- ----------
         1 Francisco                         32
```

Si tratamos de insertar un valor explícitamente al campo generado obtendremos el siguiente error:

```
SQL> insert into persona (cod,nombre,edad) values (2,'Francisco',32);
insert into persona (cod,nombre,edad) values (2,'Francisco',32)
                     *
ERROR at line 1:
ORA-32795: no se puede insertar una columna de identidad siempre generada
```

Podemos también indicar el valor con que comenzará el valor autogenerado y como será su incremento, ejemplo:

```
SQL> create table PERSONA
  2  (
  3   cod number GENERATED AS IDENTITY
  4    (START WITH 10 INCREMENT BY 10 CACHE 1000 NOCYCLE),
  5   nombre varchar(25),
  6   edad number
  7  );

Table created.
```

Aquí estamos definiendo que el primer valor sobre la tabla PERSONA en la columna COD será 10 e incrementará sucesivamente en cada operación de INSERT en 10 en 10 hasta llegar al valor de 1000, después de este valor ya no se podrá realizar más operaciones de inserción.

Nota: Solo una columna por tabla puede tener configurado la opción IDENTITY.

2. Aparecer y Ocultar columnas de una tabla

A partir de la versión Oracle Database 12c es posible ocultar una columna y luego volverla a presentar con la finalidad de asegurarnos que un campo no estará disponible a los usuarios durante un tiempo que veamos pertinente. También lo podemos utilizar para desplegar un nuevo aplicativo que explícitamente consultará las nuevas columnas invisibles creadas en la tabla en pro de no generar disrupción en el servicio a las aplicaciones ya existentes.

Mientras una columna está en estado invisible puede ser alterado mediante operaciones DML o consultado de manera explícita.

Sintaxis: ALTERT ABLE <NOMBRE_TABLA> MODIFY <NOMBRE_COLUMNA> INVISIBLE|VISIBLE

A continuación se presenta un ejemplo donde la columna NOMBRE de la tabla PERSONA se colocará en estado invisible, se realizarán operaciones DML sobre la tabla y por último se regresará al estado visible donde veremos la información modificada.

```
SQL> alter table persona modify nombre invisible;

Table altered.

SQL> insert into persona (nombre,edad) values ('Francisco',32);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from persona;

       COD       EDAD
---------- ----------
        10         32

SQL> update persona set nombre='Francisco Riccio' where cod=10;

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from persona;

       COD       EDAD
---------- ----------
        10         32

SQL> alter table persona modify nombre visible;

Table altered.

SQL> select * from persona;

       COD       EDAD NOMBRE
---------- ---------- ------------------------
        10         32 Francisco Riccio

SQL>
```

Nota 1: Si una columna se encuentra en estado INVISBLE puede ser aún consultada si se coloca explícitamente el nombre de la columna en la sentencia SELECT, ejemplo:

```
SQL> alter table persona modify nombre invisible;

Table altered.

SQL> select nombre from persona;

NOMBRE
------------------------
Francisco Riccio
```

Nota 2: Esto no aplica a tablas externas, temporales, cluster y campos cuyo tipo de dato fue creado por el usuario.

Nota 3: Cuando una columna es colocada VISIBLE automáticamente se corre a la última posición de las columnas de la tabla.

3. Creación de Índices sobre campos ya previamente indexados.

Oracle Database 12c nos da la posibilidad de crear nuevos índices donde las columnas que conforman este nuevo índices ya han sido previamente indexadas.

El requisito para poder realizar esta labor es que el índice a crear sea de diferente tipo al índice que ya tiene los campos indexados.

Los tipos de índices son: BTREE, BITMAP, BTREE CLUSTER, HASH CLUSTER, GLOBAL & LOCAL, REVERSE, DOMAIN e Índices basados en Funciones.

La utilidad a esta nueva funcionalidad es que ahora podemos crear diferentes índices a los mismos campos indexados que serán utilizados por diferentes aplicaciones dependiendo de su naturaleza en diferentes momentos. Por ejemplo, una aplicación data warehouse se beneficiará del uso de índices BITMAP mientras una aplicación transaccional se beneficiará de un índice BTREE, donde ambos índices comparten los mismos campos.

Lo mismo sucedería si creamos un índice de tipo GLOBAL y otro LOCAL para diferentes objetivos que utilizarían las aplicaciones.

Ejemplo:

```
SQL> create index IDX_EDAD_PERSONA_01 on PERSONA(nombre);

Index created.

SQL> alter index IDX_EDAD_PERSONA_01 invisible;

Index altered.

SQL> create index IDX_EDAD_PERSONA_02 on PERSONA(nombre) reverse;

Index created.
```

Como se puede apreciar tenemos dos índices con las mismas columnas como definición de indexación, pero solo un índice es visible y el resto estará invisible.

Si colocamos en estado visible el índice IDX_EDAD_PERSONA_01, conseguiremos el siguiente error:

```
SQL> alter index IDX_EDAD_PERSONA_01 visible;
alter index IDX_EDAD_PERSONA_01 visible
*
ERROR at line 1:
ORA-14147: Hay un ϕndice VISIBLE existente definido en el mismo juego de
columnas.
```

4. Introducción a la cláusula FETCH FIRST

A continuación se presenta la información registrada en la tabla PERSONA.

```
SQL> select * from persona;

       COD        EDAD NOMBRE
---------- ---------- ----------
        10         32 Francisco
        20        103 Matilde
        30         30 Pedro
        40         31 Cesar
        50         32 Martin
```

Como requerimiento se le solicita devolver las 2 personas más jóvenes de la tabla PERSONA.

En las versiones previas para realizar dicho requerimiento se tendría que construir una consulta similar a esta:

```
SQL> select nombre
  2  from
  3   (select nombre
  4    from persona
  5    order by edad
  6   )
  7  where rownum<=2;

NOMBRE
----------------------
Pedro
Cesar
```

Esta nueva versión de Oracle Database nos ofrece la cláusula FETCH FIRST para simplificar el query que cumpliría con el requerimiento.

Ejemplo:

```
SQL> select nombre
  2  from persona
  3  order by edad
  4  FETCH FIRST 2 ROWS ONLY;

NOMBRE
------------------------
Pedro
Cesar
```

Si nos piden traer la tercera y cuarta persona más joven, podemos simplificarlo de la siguiente manera:

```
SQL> select nombre
  2  from persona
  3  order by edad
  4  OFFSET 2 ROWS FETCH NEXT 2 ROWS ONLY;

NOMBRE
------------------------
Francisco
Martin
```

En este caso le indicamos a Oracle que ordene la tabla por el campo edad y se ubique en la fila número 3 (OFFSET 2, las filas están numeradas a partir de la posición 0) y traiga 2 filas a partir de esa posición.

Por último si requerimos traer el 75% de las filas de la tabla PERSONA podemos ejecutar las siguiente consulta:

```
SQL> select nombre
  2  from persona
  3  order by edad
  4  fetch first 75 percent rows with ties;

NOMBRE
------------------------
Pedro
Cesar
Martin
Francisco
```

La tabla cuenta con 5 filas por lo cual el 75% sería 3.75 ≈ 4, el cual es el número correcto que devuelve el query con la opción FETCH FIRST # PERCENT ROWS WITH TIES.

Conclusiones

Se ha presentado algunos beneficios que están disponibles en esta nueva versión de Oracle Database; dando énfasis a nuevas opciones que tenemos en el diseño y mantenimiento de nuestros modelos de base de datos y sobre todo la manera como consultamos su información de manera óptima.

Publicado por Ing. Francisco Riccio. Es un IT Oracle Specialist e instructor de cursos oficiales de certificación Oracle. Está reconocido por Oracle como un Oracle ACE y certificado en productos de Oracle Application / Oracle Database / Oracle Developer.

e-mail: francisco@friccio.com

web: www.friccio.com