

Prácticas de dependencias PL/SQL 12c-18c avanzado

1. Introducción

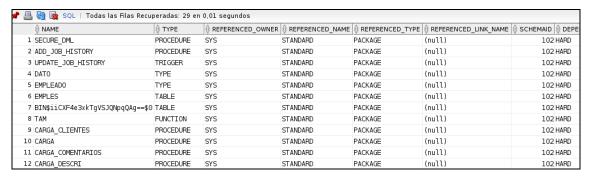
- Las tablas que vamos a usar durante el curso pertenecen al usuario HR.
- Básicamente usaremos las siguientes:
 - o EMPLOYEES: tabla de empleados
 - o DEPARTMENTS: contiene lo departamentos
 - o REGIONS: contiene las regiones
 - o JOBS: contiene los tipos de trabajos
 - o COUNTRIES: contiene los países
- También usaremos el usuario SYSTEM para varias de las prácticas
- ¡¡¡Cualquier duda que tengas no dudes en pedir ayuda¡¡¡¡



2. INTRODUCCIÓN A DEPENDENCIAS

 Determinar los componentes que dependen del paquete "STANDARD" del usuario "SYS"

SELECT * FROM USER_DEPENDENCIES WHERE REFERENCED_NAME='STANDARD' AND REFERENCED_OWNER='SYS'



 Comprobamos si existen objetos que estén inválidos. (los resultados pueden ser muy distintos en vuestra Base de datos).

SELECT * FROM USER_OBJECTS WHERE STATUS='INVALID';

	∜ OBJECT_NAME	∜ SUBOBJECT_NAME	∜ OBJECT_ID ∜ DATA	OBJECT_ID	OBJECT_TYPE	CREATED	∜ LAST_DDL_TIME	₹ TIMESTAMP	∜ STATUS	∜ TE
1	CARGA_CLIENTES	(null)	76330	(null)	PROCEDURE	08/06/19	09/06/19	2019-06-09:17:29:09	INVALID	N
2	CARGA	(null)	76343	(null)	PROCEDURE	08/06/19	08/06/19	2019-06-08:23:16:17	INVALID	N
3	CARGA_COMENTARIOS	(null)	76362	(null)	PROCEDURE	08/06/19	09/06/19	2019-06-09:19:41:41	INVALID	N
4	HELLO_WORLD	(null)	80077	(null)	PROCEDURE	15/07/19	15/07/19	2019-07-15:23:08:18	INVALID	N

3. DEPENDENCIAS DIRECTAS

- Vamos a comprobar la dependencia directa
- Creamos una tabla basada en la tabla LOCATIONS

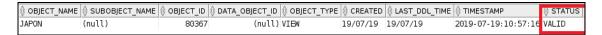
CREATE TABLE LOCALIDADES AS SELECT * FROM LOCATIONS;

Creamos una vista basada en la tabla anterior

CREATE VIEW JAPON AS SELECT * FROM LOCALIDADES WHERE COUNTRY ID='JP'

Comprobamos que su estado es válido

SELECT * FROM USER_OBJECTS WHERE OBJECT_NAME='JAPON';



Ahora modificamos la tabla original, cambiando una de sus columnas

ALTER TABLE LOCALIDADES MODIFY CITY VARCHAR2(100);



• Comprobamos de nuevo el estado de la vista

SELECT * FROM USER_OBJECTS WHERE OBJECT_NAME='JAPON';

	⊕ SUBOBJECT_NAME		DATA_OBJECT_ID			\$ LAST_DDL_TIME		∯ STATUS
JAPON	(null)	80367	(null)	VIEW	19/07/19	19/07/19	2019-07-19:10:57:16	INVALID

- Para que vuelva a estar válida solo es necesario ejecutarla, ya que eso comprueba la validez de la operación y si es correcto la compila
- La otra opción (por ejemplo si la vista es muy pesada y no queremos ejecutarla), es compilarla

```
ALTER VIEW JAPON COMPILE;

SELECT * FROM USER_OBJECTS WHERE OBJECT_NAME='JAPON';
```

4. DEPENDENCIAS INDIRECTAS

 Vamos ahora a crear una función que utilice la vista para comprobar la dependencia indirecta, por ejemplo que devuelva el número de filas de la vista

```
CREATE OR REPLACE FUNCTION NUM_JAPON
RETURN NUMBER
IS
    X NUMBER;
BEGIN
    SELECT COUNT(*) INTO X FROM JAPON;
    RETURN X;
END;
/
```

• Comprobar si el estado de la función es válido

SELECT * FROM USER OBJECTS WHERE OBJECT NAME='NUM JAPON';

⊕ OBJECT_NAME	SUBOBJECT_NAME	⊕ OBJECT_ID ⊕ DATA	_OBJECT_ID \$\text{OBJECT_TYPE}		\$ LAST_DDL_TIME	∯ TIMESTAMP	∯ STATUS
NUM_JAPON	(null)	80371	(null) FUNCTION	19/07/19	19/07/19	2019-07-19:19:32:51	VALID

• Ahora modificamos de nuevo la tabla, para ver el resultado en la vista y en el procedimiento

ALTER TABLE LOCALIDADES MODIFY CITY VARCHAR2(50);

SELECT * FROM USER_OBJECTS WHERE OBJECT_NAME LIKE '%JAPON%';

OBJECT_NAME		# OBJECT_ID # DATA	A_OBJECT_ID	⊕ OBJECT_TYPE		∜ LAST_DDL_TIME		# STATUS
IAPON	(null)	80367	(null)	VIEW	19/07/19	19/07/19	2019-07-19:19:28:07	INVALID
IUM_JAPON	(null)	80371	(null)	FUNCTION	19/07/19	19/07/19	2019-07-19:19:32:51	INVALID

Compilamos la función

ALTER FUNCTION NUM_JAPON COMPILE;



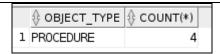
• Comprobamos el estado de la vista y el procedimiento

SELECT * FROM USER_OBJECTS WHERE OBJECT_NAME LIKE '%JAPON%';

OBJECT_NAME	\$\text{\$ SUBOBJECT_NAME}		DATA_OBJECT_ID			\$ LAST_DDL_TIME	∯ TIMESTAMP	∜ STATUS
JAPON	(null)	80367	(null)	VIEW	19/07/19	19/07/19	2019-07-19:19:45:51	VALID
NUM_JAPON	(null)	80371	(null)	FUNCTION	19/07/19	19/07/19	2019-07-19:19:45:51	VALID

- Podemos comprobar que los dos están válidos. ¿por qué? Es debido a que al compilar la función, Oracle también compila la vista de la que depende, para comprobar que está todo OK.
- Nos conectamos ahora a una sesión como usuario SYSTEM
- Averiguamos el número y tipo de los objetos inválidos en toda la Base de Datos (el resultado puede ser distintos en vuestra Base de datos). En este caso, consultamos la tabla DBA OBJECTS.

SELECT OBJECT_TYPE,COUNT(*) FROM DBA_OBJECTS WHERE STATUS='INVALID' GROUP BY OBJECT_TYPE;



5. VER JERARQUÍA DE DEPENDENCIAS

 Averiguar los objetos referenciados por la tabla LOCALIDADES que hemos creado anteriormente

SELECT * FROM DBA_DEPENDENCIES WHERE REFERENCED_NAME='LOCALIDADES';



- Vemos que solo aparece la vista, pero nO la función NUM_JAPON, ya que depende de forma indirecta a través de la vista.
- Con la siguiente vista (a través de la cláusula jerárquica START WITH)
 podemos poner un elemento y saber todos los objetos de los que depende. Por
 ejemplo, si queremos saber los objetos de los que depende la función
 NUM_JAPON:

```
select
| lpad (' ', 2 * (level - 1)) || to_char (level, '999') as "NIVEL",
| owner || '.' || name || ' (' || type || ')' as "OBJETO",
| referenced_owner || '.' || referenced_name || ' (' || referenced_type || ')' as "OBJETO
| REFERENCIADO"
| from
| dba_dependencies
| start with
| owner = 'HR'
| and
| name = 'NUM_JAPON'
| connect by prior
| referenced_owner = owner
```



```
and prior
  referenced_name = name
and prior
  referenced_type = type
;
```

NIVEL	∳ ОВЈЕТО	⊕ OBJETO REFERENCIADO
1	HR.NUM_JAPON (FUNCTION)	HR.JAPON (VIEW)
2	HR.JAPON (VIEW)	HR.LOCALIDADES (TABLE)
1	HR.NUM_JAPON (FUNCTION)	SYS.STANDARD (PACKAGE)
1	HR.NUM_JAPON (FUNCTION)	SYS.SYS_STUB_FOR_PURITY_ANALYSIS (PACKAGE)

- Podemos comprobar que primero depende la vista y luego de la tabla. También tiene dependencias de objetos de SYS, al ser una función.
- De todas formas, si no somos SYSTEM, no podemos consultar la DBA_DEPENDENCIES. Por tanto, podemos modificar la vista para que funcione con el usuario HR y la vista USER_DEPENDENCIES. Hay que eliminar cualquier referencia a la columna OWNER.

```
select
| lpad (' ', 2 * (level - 1)) || to_char (level, '999') as "NIVEL",
| name || ' (' || type || ')' as "OBJETO",
| referenced_owner || '.' || referenced_name || ' (' || referenced_type || ')' as "OBJETO
| REFERENCIADO"
| from
| user_dependencies
| start with
| name = 'NUM_JAPON'
| connect by prior
| referenced_name = name
| and prior
| referenced_type = type
```

6. UTLDTREE.SQL

- Siguiendo el vídeo del curso, cargamos la utilidad UTLDTREE.SQL del directorio "rdbms/admin" de Oracle. Lo hacemos dentro del usuario HR
- Una vez terminado ejecutamos el procedimiento "DEPTREE_FILL" con la tabla LOCALIDADES y vemos el resultado con la vista "DEPTREE"

```
EXECUTE deptree_fill('TABLE','HR','LOCALIDADES');

SELECT * FROM DEPTREE;
```

NESTED_LEVEL	∯ TYPE		NAME	∜ SEQ#
0	TABLE	HR	LOCALIDADES	0
1	VIEW	HR	JAPON	201
2	FUNCTION	HR	NUM_JAPON	202

7. DMBS_UTILITY

Vamos a ver ahora como usar el paquete DBMS_UTILITY



• En primer lugar, comprobamos las dependencias de LOCALIDADES

SET SERVEROUTPUT ON

EXECUTE DBMS_UTILITY.GET_DEPENDENCY('TABLE','HR','LOCALIDADES');

DEPENDENCIES ON HR.LOCALIDADES

*TABLE HR.LOCALIDADES()

- * VIEW HR.JAPON()
- * FUNCTION HR.NUM JAPON()

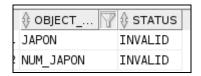
Procedimiento PL/SQL terminado correctamente.

• Modificamos ahora la tabla para que invalide sus objetos dependientes

ALTER TABLE LOCALIDADES MODIFY CITY VARCHAR2(100);

• Comprobamos el estado de los objetos dependientes

SELECT OBJECT_NAME, STATUS FROM USER_OBJECTS WHERE OBJECT NAME LIKE '%JAPON%';



• Ahora, validamos el procedimiento

EXECUTE DBMS_UTILITY.VALIDATE('HR','NUM_JAPON',1);

 Volvemos a preguntar por el estado. Vemos que también ha validado la vista, ya que ha tenido que acceder a ella para validar el procedimiento

SELECT OBJECT_NAME,STATUS FROM USER_OBJECTS WHERE OBJECT_NAME LIKE '%JAPON%';

