Unidad 5:

SQL
2ª Parte

Tratamiento de Datos

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Manipulación de Datos. Tablas, vistas y Sinónimos

OBJETIVOS

- Manejar con fluidez las órdenes para insertar, modificar y eliminar filas de una tabla
- Utilizar la orden INSERT
- Utilizar la orden UPDATE
- Utilizar la orden DELETE
- Entender el concepto de integridad de datos
- Entender los conceptos de COMMIT y ROLLBACK
- Utilizar órdenes para crear y suprimir vistas y sinónimos
- Descubrir las ventajas de recurrir a los sinónimos
- Utilizar sinónimos y vistas.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Inserción de Datos. INSERT

INSERT

Permite añadir filas de datos a una tabla

Formato:

Insert into TABLA [(col1 [,col2]...)] **values** (valor1 [,valor2]...);

TABLA ➔ **nombre de la tabla**
Col1,... ➔ **columnas donde se van a introducir valores**
Valor1,... ➔ **valores a cargar en las columnas**

PROFESORES (nif,apellidos,especialidad, codigo_centro)

Insert into PROFESORES (apellidos,especialidad,nif)
values ('AITOR MENTA', 'CIENCIAS', 12345678);

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Inserción de Datos. INSERT

INSERT

- Si no se especifican las columnas, se considera, por defecto, todas la columnas
- Cualquier columna que no se encuentre en la lista de columnas recibirá el valor NULL, siempre y cuando no esté definida como NOT NULL, en cuyo caso INSERT fallará
- Los valores deben de coincidir con el tipo de dato definido para cada columna.

PROFESORES (nif,apellidos,especialidad, codigo_centro)

Insert into PROFESORES (apellidos,especialidad,codigo_centro)

values ('AITOR MENTA', 45,'CIENCIAS');

ERROR !!!!!!!!!!!!!!!!!!!!!!!!

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Inserción de Datos. INSERT

INSERT: Consideraciones a tener en cuenta

- Cuando insertamos asegurarse que los datos que se insertan son, al menos, los obligatorios de un registro de esa tabla, es decir:
  - la clave primaria y que su valor no exista ya
  - campos definidos en la tabla como no nulos
  - Campos que son clave ajena de otra tabla. En este caso podría no ser obligatorio en función de cómo está definida la restricción de clave ajena: ON DELETE SET NULL/ON DELETE CASACADE/RESTRICT.

    En el caso de que esté definida la restricción ON DELETE CASCADE / RESTRICT será obligatorio poner un valor que además exista como clave primaria en la tabla a la que hace referencia. → **CONSISTENCIA**

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Inserción de Datos. INSERT

INSERT CON SELECT

Permite añadir a una tabla las filas resultantes de una consulta.

Formato:

insert into TABLA1 [(col1 [,col2]...)]
SELECT {col1 [,col2]...| *} from TABLA2 [cláusulas de SELECT];

Ejemplo:

Insert into EMPLE30
Select emp_no,apellido,oficio,dir,fecha_alt,salario,comision,dept_no
from EMPLE where dept_no=30;

Inserción de Datos. INSERT

INSERT CON SELECT

```
Insert into EMPLE30 select * from EMPLE where dept_no=30;

Insert into NOMBRE (nombre) select apellido from EMPLE where dept_no=20;
//los apellidos tendrán que tener <=longitud que el nombre//

Insert into EMPLE select 1111,'GARCIA','ANALISTA',7566,sysdate,2000,1200,dept_no from
    EMPLE where dept_no=
        (select dept_no from EMPLE
            group by dept_no having count(*) =
                (select max(count(*)) from EMPLE group by dept_no));
```

VALORES FIJOS

Valores que saldrán de la subconsulta

O también

```
Insert into EMPLE
  values (1111,'GARCIA','ANALISTA',7566,sysdate,2000,1200,
        ( select dept_no from EMPLE
            group by dept_no having count(*) =
                (select max(count(*)) from EMPLE group by dept_no)))
```

Valor de colum, que sale de una consulta select

Inserción de Datos. INSERT

INSERT: Formatos

- Si insertamos valores en todos los campos de la tabla (respetar el orden de columnas que tiene la tabla)

  Insert into TABLA **values** (valor1, valor2,……, valor N)

- Si insertamos solo algunos campos de la tabla

  Insert into TABLA (col1, col2, ….,col N) **values** (valor1, valor2,……, valor N)

- Si lo hacemos a través de una consulta

  Insert into TABLA (consulta_select …. que devuelva tantas columnas como tiene la tabla)

- Si lo hacemos a través de una consulta con solo algunas columnas

  Insert into TABLA (col1, col2,…) (consulta_select …. que devuelva tantas columnas como se especifiquen)

Un valor puede calcularse con un select

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Inserción de Datos. INSERT

INSERT CON SELECT. Ejercicios:

- En el esquema U4, comprobar que la tabla EMPLE y DEPART tienen creadas las restricciones de integridad (claves primarias y clave ajena)

EMP_NO	APELLIDOS	OFICIO	FECHA_ALT	SALARIO	DEPT_NO
1111	Tu_nombre	ESTUDIANTE	Fecha_hoy	111	1

- Comprobar si se ha añadido la fila, en caso contrario que error nos da y razonar por qué
- Añadir en DEPART un registro con el DEPT_NO=1, nombre='prueba', y ciudad = 'XXXX'
- Insertar ahora el registro y comprobar si lo ha añadido

Inserción de Datos. INSERT

INSERT CON SELECT. Ejercicios:

- Salir del SQL y seleccionar la opción ROLLBACK antes de abandonar el programa. Volver a entrar y comprobar si existe la fila.
- Realizar los mismos ejercicios anteriores pero seleccionando la opción COMMIT después de insertar.
- Comprobar que pasa si salimos del SQL y volvemos a entrar. ¿están los datos?
- Comprobar en la tabla EMPLE los campos que no pueden ser nulos
- Insertar una fila con unos datos inventados, pero sin el dept_no. ¿qué sucede? ¿Por qué?

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Inserción de Datos. INSERT

INSERT CON SELECT. Ejercicios:

- Insertar en la tabla EMPLE un nuevo usuario con la siguiente información y los datos que faltan se completarán con los mismos datos que tiene el usuario SALA.

EMP_NO	APELLIDOS	FECHA_ALT
2222	QUIROGA	Fecha_hoy

- Insertar en la tabla DEPART el departamento 50 correspondiente al DAI localizado en SOTRONDIO

- Añadir a la tabla EMPLE todos los usuarios de la tabla EMPLE asignándoles como fecha la fecha de hoy y como código de departamento=50.

Modificación de datos. UPDATE

UPDATE

Actualizar valores de una o más columna para una o varias filas

Formato:

***Update** TABLA **set** col1=valor1,col2=valor2,...[**where condición**];*

Ejemplos:

update CENTROS set direccion='C/ Rio,23',num_plazas=295 where cod_centro= 22;

Update CENTROS set direccion='C/ Pilon,22',num_plazas=295; OJO!!!

Actualiza todas las filas a menos que indiquemos el WHERE con la condición que especifíque que filas hay que modificar, si no ponemos WHERE actuará en todas las filas

Modificación de datos. UPDATE

UPDATE con SELECT

Permite incluir una subconsulta en la instrucción UPDATE

Formato:

update *TABLA* ***set*** *col1=valor1, col2=valor2,...*
where *col3 = (****select ....);***

update *TABLA* ***set*** *(col1, col2, col3,...) = (select col1, col2, col3,...)*
where *condición;*

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Modificación de datos. UPDATE

UPDATE con SELECT

```
Update centros set
  (direccion,num_plazas) = (select direccion, num_palzas from
                                        centros where cod_centro= 50)
where cod_centro = 10;
```

```
Update emple set salario=salario/2, comision=0
 where dept_no= (select dept_no from emple
                        group by dept_no having count(*)=
                                (select max(count(*) from emple
                                        group by dept_no));
```

Modificación de datos. UPDATE

UPDATE. Ejercicios

- Modificar el departamento de tu usuario y asignarle el valor 99. Si hay problemas de restricciones buscar una solución alternativa añadiendo la información previa necesaria.
- Duplicar el salario de tu usuario en la tabla EMPLE
- Pasar a minúsculas el apellido de todos los usuarios del departamento 50
- Asignar a tu usuario el doble del salario del usuario que más gana como vendedor del departamento de ventas
- Pasar a mayúsculas los apellidos de todos los usuarios.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Borrado de filas. DELETE

DELETE

Permite eliminar una o varias filas de una tabla.

Formato:

Delete from *TABLA* ***where*** *condición;*

!!!!!NOTA IMPORTANTÍSIMO: sin la cláusula where se borrarán todas las filas.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Borrado de filas. DELETE

DELETE. Ejercicios

- Borrar el empleado creado con tu nombre.
- Borrar los departamentos de la tabla DEPART que tengan mas de 10 empleados. Si no se puede razonar el motivo
- Borrar los empleados que son vendedores del departamento 50
- Borrar todos los usuarios que quedan del departamento 50

ROLLBACK, COMMIT y AUTOCOMMIT

ROLLBACK, COMMIT y AUTOCOMMIT

- **COMMIT**: Permite aplicar de forma definitiva los cambios a la base de datos

  commit;

- **AUTOCOMMIT**: Parámetro que si está activado, permite validar automáticamente las transacciones

  Show autocommit;

  Set autocommit ON/OFF; (valor por omisión OFF)

- **ROLLBACK**: deshacemos los cambios que hemos realizado en las tablas desde el último COMMIT.

  Rollback;

ROLLBACK, COMMIT y AUTOCOMMIT

COMMIT implícitos:

Existen varias órdenes en SQL que fuerzan a que se ejecute un COMMIT sin necesidad de indicarlo

QUIT	**CREATE TABLE**	**DROP TABLE**
EXIT	**CREATE VIEW**	**DROP TABLE**
CONNECT	**GRANT**	**DROP VIEW**
DISCONNECT	**REVOQUE**	**AUDIT**
	ALTER	**NOAUDIT**

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas y Sinónimos

INTRODUCCIÓN

- Creación de vistas (view)
- Sinónimos (synonum)
- Secuencias (sequences)

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

CREACION DE UNA VISTA. USO DE UNA VISTA

- **VISTA:** es una tabla lógica que permite acceder a la información de una o varias tablas
- No contiene información por sí misma, sino que su información está basada en la que contienen otras tablas
- Tiene por tanto la misma estructura que una tabla y se trata de igual forma
- Una vista es una sentencia SQL
- Al borrar una tabla la vista asociada deja de funcionar

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

CREACION DE UNA VISTA. USO DE UNA VISTA

Formato:

CREATE [OR REPLACE] **VIEW** nombre_vista [(columna [,columna])]
AS consulta;
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

CREACION DE UNA VISTA. USO DE UNA VISTA

Formato:

Nombre_vista: nombre de la vista

Columna: nombre de las columnas de la vista, si no se especifican se adoptarán los mismos que los de la consulta

Consulta: determina las columnas y las tablas que aparecerán en la vista

OR REPLACE: crea de nuevo la vista si ya existía.

WITH CHECK OPTION especificar que solo registros accesibles a la vista puedan ser insertados o actualizados.

Constraint -nombre asignado a la restricción

CHECK OPTIONWITH READ ONLY asegurar que no se puedan realizar operaciones DML.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

CREACION DE UNA VISTA. USO DE UNA VISTA

Ejemplo

```
Create view DEP30 as select APELLIDO,OFICIO,SALARIO from
                        EMPLE where DEPT_NO=30;

Create or replace view DEP30 (APE,OFI,SAL) as
                        select APELLIDO,OFICIO,SALARIO
                            from EMPLE where DEPT_NO=30;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

CREACION DE UNA VISTA. USO DE UNA VISTA

Para consultar las vistas creadas se dispone de la **vista** USER_VIEWS:

SELECT * FROM USER_VIEWS;

Si queremos seleccionar solo el nombre de la vista y los campos que la forman, deberemos visualizar solo las columnas de la vista:

select VIEW_NAME, TEXT from USER_VIEWS

VIEW_NAME: muestra el nombre de la vista

TEXT: muestra la sentencia que permite obtener la vista

Vistas (View)

CREACION DE UNA VISTA. USO DE UNA VISTA

Ejemplo:

/* CREAMOS LA VISTA*/

```
create view DEP3O as select apellido,salario,oficio
   from EMPLE where dept_no=30;
```

/* Visualizamos la vista y la consulta de la que se obtiene las vista que tenemos o hemos creado*/

```
SELECT VIEW_NAME,TEXT FROM USER_VIEWS;
```

VIEW_NAME	TEXT
VISTA	select * from emple natural inner join depart
OFICIOS	SELECT distinct oficio FROM emple
DEPARTAMENTOS	SELECT dept_no, dnombre FROM depart
DEP3O	select apellido,salario,oficio from EMPLE where dept_no=30

Vistas (View)

BORRADO DE UNA VISTA

NOTA: Recordar que si se elimina una tabla utilizada para obtener una vista, esta última seguirá existiendo pero quedará inutilizada.

Formato:

```
DROP VIEW nombre_vista;
```

Ejemplo: `DROP VIEW dep30;`

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

OPERACIONES SOBRE VISTAS

Las operaciones a realizar son las mismas que las que se llevan a cabo sobre tablas:

- SELECT
- INSERT
- UPDATE
- DELETE

NOTA: Las operaciones sobre una vista realmente se están realizando sobre la(s) tabla(s)

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

OPERACIONES SOBRE VISTAS

Consideraciones a tener en cuenta:

No se pueden **BORRAR** registros si la vista contiene:

- FUNCIONES DE GRUPO
- LA CLÁUSULA GROUP BY
- LA PALABRA RESERVADA DISTINCT

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

OPERACIONES SOBRE VISTAS

Consideraciones a tener en cuenta:

No se puede **MODIFICAR** datos a través de la vista si ésta contiene:

- FUNCIONES DE GRUPO
- LA CLÁUSULA GROUP BY
- LA PALABRA RESERVADA DISTINCT
- COLUMNAS DEFINIDAS CON EXPRESIONES

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

OPERACIONES SOBRE VISTAS

Consideraciones a tener en cuenta:

No se puede **AGREGAR** datos a través de la vista si ésta contiene:

- FUNCIONES DE GRUPO
- LA CLÁUSULA GROUP BY
- LA PALABRA RESERVADA DISTINCT
- COLUMNAS DEFINIDAS CON EXPRESIONES
- COLUMNAS DEFINIDIAS COMO NOT NULL EN TABLAS BASE

Vistas (View)

Cláusula WITH CHECK OPTION

especificar que solo registros accesibles a lavista puedan ser insertados o actualizados

```
create OR REPLACE view vista as
    select emp_no, apellido, dept_no, dnombre from emple
     natural inner join depart where dnombre='INVESTIGACION'
     with check option

UPDATE Vista SET DNOMBRE ='INDUSTRIAL'
                                    WHERE EMP_NO=7876
```

No actualiza porque la vista solo considera INVESTIGACION y no permite cambiar

Vistas (View)

Cláusula WITH READ ONLY

asegura que no se realicen operaciones DML a través de vistas

```
create OR REPLACE view vista as
    select emp_no, apellido, dept_no, dnombre from emple
     natural inner join depart
     WITH READ ONLY

DELETE FROM VISTA
    WHERE DNOMBRE ='VENTAS'
```

No realiza operaciones DML por
La opción with read only

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

Ejercicios

- A partir de la tabla EMPLE y DEPART (Usuario U4) crear una vista llamada **EMP_DEPT** que contenga las columnas EMP_NO, APELLIDO,DEPT_NO Y DNOMBRE. Asegurarse que no se pueden realizar acciones DML sobre ella. Visualizar las filas de la vista

- Insertar una fila en la tabla EMPLE con los siguientes datos para cada una de las columnas y comprobar la tabla EMPLE y la vista EMP_DEPT

  9999, 'DESDIZ', 'VENDEDOR',7782,fecha_hoy,3333,0, 10

- Insertar una fila en la **vista creada** con los siguientes datos y comprobar que sucede

  2222, 'SUELA', 20, 'INVESTIGACION'

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

Ejercicios

- Borrar de la tabla EMPLE el usuario ‘DESDIZ’. Comprobar la tabla y la vista
- Borrar de la vista EMP_DEPT el usuario de apellido SALA y comprobar que sucede
- Actualizar en la vista EMP_DEPT el apellido SALA y sustituirlo por LASA. Comprobar que sucede

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Vistas (View)

Ejercicios

- Crear una vista llamada **PAGOS** a partir de las filas de la tabla EMPLE, cuyo departamento sea 10. Las columnas de la vistas se denominarán NOMBRE, SAL_MES, SAL_ANNO y DEPT_NO.
  - El NOMBRE se almacenará utilizando la función INITCAP
  - El SAL_MES es el SALARIO
  - El SAL_ANNO es el salario anual

- En la vista PAGOS actualizar el empleado de NOMBRE =‘Cerezo’ y llamarle ‘Manzano’. Comprobar que sucede

- Modificar el SAL_MES para el usuario NOMBRE=‘Cerezo’ y asignarle 3000. Comprobar que sucede

Vistas (View)

Ejercicios

- Insertar un usuario nuevo en la vista con los siguiente valores y comprobar que sucede:

  'PEPE', 1000, 12000, 10

- Crear una vista llamada VMEDIA a partir de las tablas EMPLE y DEPART. La vista estará formada el número de departamento, el nombre del departamento, la media de salario y el máximo salario de cada uno de los departamentos

- Eliminar un departamento cualquiera de la vista y comprobar que sucede. Intentar realizar cualquier otra acción (inserción, actualización) y comprobar que sucede.

Sinónimos

Creación de Sinónimos

Es un nombre que se puede dar a una tabla o una vista, permitiendo de esta forma abreviar a la hora de escribir el acceso a la misma.

Ejemplo: Acceso a tablas de otro usuario

Formato:

CREATE [public] SYNONYM nom_sinonimo FOR [esquema_usr.]nombre_tabla;

Public: hace que el sinónimo esté disponible para todos los usuarios. Solo puede ser usado por el DBA o usuarios con el privilegio CREATE PUBLIC SYNONYM

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Sinónimos

Creación de Sinónimos

Ejemplo:

```
Create synonym DEPARTAMENTOS for DEPART;

Create synonym DEPART for U3.DEPART;
```

El administrador podría crear el siguiente sinónimo público:

```
Create public synonym EMPLEADOS for U3.EMPLE;
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Sinónimos

Borrado de Sinónimos

Formato:

DROP [public] SYNONYM [usuario].nom_sinonimo ;

Al igual que antes, solo el DBA y los usuarios con el privilegio DROP PUBLIC SYNONYM pueden suprimir sinónimos públicos.

Los usuarios con el privilegio DROP ANY SYNONYM pueden borrar los sinónimos de otros usuarios

DROP SYNONYM DEPARTAMENTOS;

DROP PUBLIC SYNONYM EMPLEADOS; (solo DBA o usr con priv.)

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

Sinónimos

Borrado de Sinónimos

Para ver los sinónimos, existe una vista USER_SYNONYMS que permite ver los sinónimos que son propiedad del usuario.

DESC SYS.USER_SYNONYMS;

```
SELECT      SYNONYM_NAME “Nombre Sinónimo”,
            TABLE_OWNER “Propietario”,
            TABLA_NAME “Tabla
            from USER_SYNONYMS;
```

Renombrar Tablas, Vistas, Sinónimos

RENAME

Permite cambiar el nombre a una tabla, vista o un sinónimo.

Oracle invalida todos los objetos que dependan del objeto renombrado, como las vistas, los sinónimos que hacen referencia a la tabla renombrada.

Formato:

RENAME nombre_old TO nombre_new;

Ejemplo:

```
create synonym ALUM from ALUMNOS;        //sinónimo para tabla ALUMNOS
rename ALUMNOS to TALUMNOS;              // al renombrar la tabla, el sinónimo
                                            ALUM deja de funcionar
```

OTROS OBJETOS

SECUENCIAS

- Objeto de base de datos que sirve para generar enteros únicos de forma secuencial
- Útil para generar automáticamente valores para claves primarias
- Es necesario tener el privilegio CREATE ANY SEQUENCE para poder crear secuencias.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

OTROS OBJETOS

SECUENCIAS

```
CREATE SEQUENCE nombre_secuencia
                [ INCREMENT BY entero]
                [START WITH entero]
                [MAXVALUE entero  | NOMAXVALUE]
                [MINVALUE entero  | NOMINVALUE]
                [CYCLE | NOCLYCLE]
                [ORDER | NOORDER]
                [CACHE entero | NOCACHE];
```

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

OTROS OBJETOS

SECUENCIAS

INCREMENT BY entero : incremento de la secuencia de números

START WITH entero : valor inicial de la secuencia

MAXVALUE entero |NOMAXVALUE: valor máximo/para no definir máximo de10^{27}.

MINVALUE entero | NOMINVALUE: valor mínimo/para no definir mínimo de10^{27}

CYCLE | NOCLYCLE: comienza desde el principio al alcanzar MAXVALUE/ no ciclico

ORDER | NOORDER: ORDER garantiza que los números sean generados en el orden solicitado. NOORDER no garantiza ese orden

CACHE entero | NOCACHE : CACHE permite que los número de la secuencia para ser asignados se almacenen en la memoria, haciendo más rápido el acceso. El valor mínimo es 2. NOCACHE impide la asignación previa de números a la secuencia.

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

OTROS OBJETOS

SECUENCIAS

Para acceder a los valores actuales de la secuencia

Nombresecuencia.CURRVAL → devuelve el valor actual de la secuencia
Nombresecuencia.NEXTVAL → devuelve el siguiente valor de la secuencia

Ejemplo:

```
CREATE TABLE frutas (
codigo          NUMBER(2) NOT NULL PRIMARY KEY,
nombre          VARCHAR2(15)
);

CREATE SEQUENCE codigo START WITH 1 INCREMENT BY 1 MAXVALUE 99;
INSERT INTO frutas VALUES (codigo.nextval, 'MANZANAS');
```

OTROS OBJETOS

SECUENCIAS. EJERCICIOS

- Diseñar la tabla USUARIOS con los siguientes campos:

  num_usr → number(1); clave primaria
  Nom_usr → varchar(20);

- Crear una secuencia SECUENCIA_USR, que empiece en el número 1 y su último valor sea 5. El incremento de la secuencia será de dos. En el SQLdeveloer pinchar la pestaña SECUENCE y comprobar la secuencia

- Insertar el nombre USUARIO1 y asignarle como clave primaria el número generado por la secuencia

- Insertar el nombre USUARIO2 y asignarle como clave primaria el siguiente número generado por la secuencia

- Comprobar, utilizando la tabla DUAL, el valor actual que tiene la secuencia

- Ir insertando usuarios sucesivamente USUARIO3,…… y comprobar que sucede cuando se inserta el USUARIO6

TRANSACCIONES

TRANSACCIONES

***TRANSACCION**: una o más sentencias SQL.*

Mientras que sobre una transacción no se haga **COMMIT**, los resultados de ésta pueden deshacerse (**ROLLBACK**). Una vez realizado el COMMIT no se pueden deshacer.

```
update emple set dept_no=99 where dept_no=50;
Delete  from emple where detp_no=99
ROLLBACK;
```

```
update emple set dept_no=99 where dept_no=50;
Delete  from emple where detp_no=99
Insert into emple values (valor1,…….,valorN);
COMMIT;
```

TRANSACCIONES

TRANSACCIONES

Las sentencias de control de transacciones gestionan los cambios que realizan las sentencias DML y las agrupa en transacciones. Estas sentencias te permiten realizar las siguientes acciones:

- Hacer permanentes los cambios producidos por una transacción (***COMMIT***).
- Deshacer los cambios de una transacción (***ROLLBACK***) desde que fue iniciada o desde un punto de restauración (***ROLLBACK TO SAVEPOINT***). Un punto de restauración es un marcador que puedes establecer dentro del contexto de la transacción. *ROLLBACK* finaliza la transacción, pero *ROLLBACK TO SAVEPOINT* no la finaliza.
- Establecer un punto intermedio (***SAVEPOINT***) a partir del cual se podrá deshacer la transacción.
- Indicar propiedades para una transacción (***SET TRANSACTION***).
- Especificar si una restricción de integridad aplazable se comprueba después de cada sentencia DML o cuando se ha realizado el *COMMIT* de la transacción (***SET CONSTRAINT***).

TRANSACCIONES

CAMBIOS PERMANENTES

- Utilizar *COMMIT*, la cual ordena a la base de datos que haga permanentes las acciones incluidas en la transacción.
- Ejecutar una sentencia DDL (*CREATE*, *DROP*, *RENAME*, o *ALTER*). La base de datos ejecuta implícitamente una orden *COMMIT* antes y después de cada sentencia DDL.
- Si el usuario cierra adecuadamente las aplicaciones de gestión de las bases de datos Oracle, se produce un volcado permanente de las cambios efectuados por la transacción.

TRANSACCIONES

DESHACER CAMBIOS

- *ROLLBACK* permite deshacer los cambios efectuados por la transacción actual, dándola además por finalizada.

- Si no se han hecho permanentes los cambios de una transacción, y la aplicación termina incorrectamente, la base de datos de Oracle retorna al estado de la última transacción volcada, deshaciendo los cambios de forma implícita.

TRANSACCIONES

DESHACER CAMBIOS PARCIALMENTE

- Creando en *punto de restauración* (*SAVEPOINT*): marcador intermedio declarado por el usuario en el contexto de una transacción.
- Para establecer un punto de restauración se utiliza la sentencia *SAVEPOINT* con la sintaxis:

  SAVEPOINT nombre_punto_restauración;

- La restauración de los cambios hasta ese punto se hará con un comando con el siguiente formato:

  ROLLBACK TO SAVEPOINT nombre_punto_restauración;

TRANSACCIONES

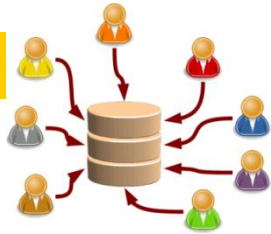
DESHACER CAMBIOS PARCIALMENTE

```
UPDATE T_PEDIDOS SET NOMBRE='jorge' WHERE CODPEDIDO=125;
SAVEPOINT solouno;

 UPDATE T_PEDIDOS SET NOMBRE = 'jorge‘;
SAVEPOINT todos;

SELECT * FROM T_PEDIDOS;

ROLLBACK TO SAVEPOINT solouno;

COMMIT;
```

Solo guardamos hasta la primera modificación

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

CONCURRENCIA A DATOS



PROBLEMAS ASOCIADOS A LA CONCURRENCIA

- **Concurrencia de datos** asegura que los usuarios pueden acceder a los datos al mismo tiempo.

- **Consistencia de datos**: asegura que cada usuario tiene una vista consistente de los datos, incluyendo los cambios visibles realizados por las transacciones del mismo usuario y las transacciones *finalizadas* de otros usuarios.

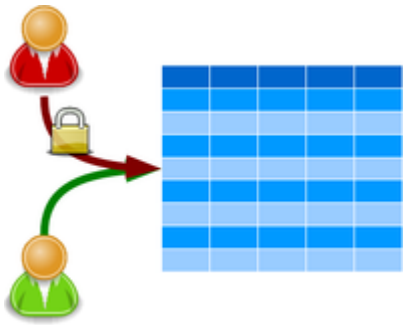
- Oracle proporciona concurrencia de datos, *consistencia e integridad* en las transacciones mediante sus mecanismos de *bloqueo*. Los bloqueos se realizan de forma automática y no requiere la actuación del usuario.

CONCURRENCIA A DATOS

POLITICAS DE BLOQUEO

Los bloqueos afectan a la interacción de lectores y escritores.

- **Lector** (*CONSULTA*) es una consulta sobre un recurso
- **Escritor** (*MODIFICACIÓN*) es una sentencia que realiza un modificación sobre un recurso.



Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13

CONCURRENCIA A DATOS

REGLAS SOBRE EL COMPORTAMIENTO DE LA CONCURRECIA

- Un registro es bloqueado sólo cuando es modificado por un escritor: *Cuando una sentencia actualiza un registro, la transacción obtiene un bloqueo sólo para ese registro.*

- Un escritor de un registro bloquea a otro escritor concurrente del mismo registro: *Si una transacción está modificando una fila, un bloqueo del registro impide que otra transacción modifique el mismo registro simultáneamente.*

- Un lector nunca bloquea a un escritor: *Puesto que un lector de un registro no lo bloquea, un escritor puede modificar dicho registro. La única excepción es la sentencia SELECT … FOR UPDATE, que es un tipo especial de sentencia SELECT que bloquea el registro que está siendo consultado*.

- Un escritor nunca bloquea a un lector: *Cuando un registro está siendo modificado, la base de datos proporciona al lector una vista del registro sin los cambios que se están realizando*.

CONCURRENCIA A DATOS

BLOQUEOS COMPARTIDOS Y EXCLUSIVOS

- **bloqueo exclusivo**: previene que sea compartido el recurso asociado. Una transacción obtiene un bloqueo exclusivo cuando modifica los datos. La 1ª transacción que bloquea un recurso exclusivamente, es la única que puede modificar el recurso hasta que el bloqueo es liberado.

- **bloqueo compartido**: permite que sea compartido el recurso asociado, dependiendo de la operación en la que se encuentra involucrado. Varios usuarios que estén leyendo datos pueden compartir los datos, realizando bloqueos compartidos para prevenir el acceso concurrente de un escritor que necesita un bloqueo exclusivo. Varias transacciones pueden obtener bloqueos compartidos del mismo recurso.

CONCURRENCIA A DATOS

BLOQUEOS COMPARTIDOS Y EXCLUSIVOS

Ej.: *SELECT… FOR UPDATE* para consultar un registro de una tabla.

La transacción obtiene:

- **un bloqueo exclusivo del registro** : mientras que el bloqueo de la tabla previene que otras sesiones modifiquen la estructura de la tabla
- **un bloqueo compartido de la tabla**: que permite a otras sesiones que modifiquen cualquier otro registro que no sea el registro bloqueado

Roberto Portabales Enríquez
IES JJ Calvo Miguel - Curso 12/13