"El producto más valioso que conozco es la información"

06

TALLER DE BASE DE DATOS - INF 272 SEMESTRE 2/2022

D.M.L.

DATA MANIPULATION LANGUAGE

INSERT – UPDATE - DELETE

INSERT

Permite insertar filas dentro de una tabla.

Formato:

INSERT INTO TABLA (C1,C2,C3,…) VALUES (V1,V2,V3,...);

```
insert  into departamentos (depnro,nom_dep,ciudad)
values (1,'Marketing','Santa Cruz de la Sierra');

insert  into departamentos values (1,'Marketing','Santa Cruz de la Sierra');

insert
into empleados  (empnro,nomemp,apeemp,cargo,fecha_ing,salario,deptnro)
values (1,'Jorge Choque','Vendedor',TO_DATE('15/01/2019','dd/mm/yyyy'),2000,1);
```

UPDATE

Para modificar uno o varios datos de uno o varios registros utilizamos "update" (actualizar).

Formato:

```
UPDATE TABLE SET
COLUMNA1 = VALOR1,COLUMNA2=VALOR2,…….
WHERE CONDICION;
```

Si no se pone condición actualizará el 100% de los registros

```
update empleados set email='correoprueba@prueba.com'
where email is null;

update empleados set salario=salario*1,1;

update empleados set salario=salario*1,1 where cargo='Vendedor';
```

DELETE

Para eliminar uno o varios registros (fila) de una tabla "delete" (eliminar). Y una condición, si no se especifica asumirá que borrara el 100% de los registros.

Formato:

```
DELETE FROM TABLA
WHERE CONDICION;
```

TRUNCATE

Para eliminar todos los registros de una tabla.

Formato:

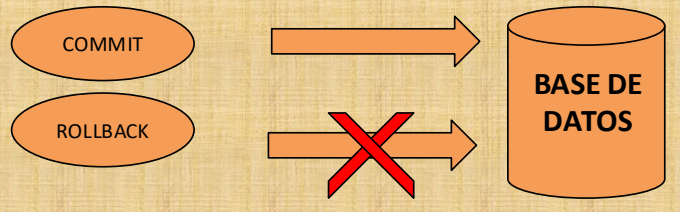
```
TRUNCATE TABLE nombre_tabla;
```

DIFERENCIAS ENTRE TRUNCATE/DELETE

- DELETE borra registros uno a uno y hace lento el proceso a diferencia de TRUNCATE.
- DELETE, al eliminar verifica sino viola ningún CONSTRAINT, al contrario TRUNCATE verifica si no hay ninguna tabla con la que esta vinculada (llave foránea), aunque estén vacías las tablas no deja ejecutar.
- Al usar TRUNCATE si existe un campo auto numerado este reinicia, mientras que DELETE continua con el ultimo valor asignado.

TRANSACCIONES

- Una transacción es un conjunto de operaciones que se lanzan contra la base de datos.
- Las transacciones deben ser aceptadas o rechazadas todas. No se pueden aceptar transacciones parciales.
- Es el usuario que acepta o rechaza las mismas.
- Para finalizar una transacción se emplea un COMMIT o un ROLLBACK



COMMIT / ROLLBACK

- Una transacción se inicia cuando se realiza cualquier tipo de operación, INSERT, UPDATE, DELETE
- Una transacción finaliza cuando se pone el comando COMMIT y se confirman los cambios, o el comando ROLLBACK si se rechazan los cambios.
- Cuando se lanza un comando DDL, automáticamente se hace COMMIT sin necesidad de escribir el comando. Por ejemplo con el comando CREATE TABLE, CREATE VIEW.
- Los comandos DCL como GRANT, REVOKE, generan un commit implícito.
- Los comandos DML necesitan COMMIT.
- Cuando hay un fallo de la maquina de la BD o de la conexión, ORACLE hace un ROLLBACK automático.

SAVEPOINT

- Permite hacer ROLLBACK parciales.
- Primero se debe crear un punto hasta donde se desea recuperar.

```
SAVEPOINT etiqueta;
```

- Para poder recuperar hasta el punto marcado se emplea:

```
ROLLBACK TO SAVEPOINT etiqueta;
```

BLOQUEOS

- Cuando un usuario esta haciendo alguna operación sobre una tabla o sobre un registro, este permanece bloqueado hasta que se confirme o rechace la misma.
- Esto impide que otro usuario acceda al registro o tabla.

D.D.L.

DATA DEFINITION LANGUAGE

CREATE –ALTER – DROP

TABLAS – INDICES – VISTAS - SINONIMOS

CREATE TABLE

- Permite crear tablas dentro de la BD.

```
CREATE TABLE nombretabla (
Nombrecolumna1 tipodedatos [DEFAULT valor]
[restricciones],
Nombrecolumna2 tipodedatos [DEFAULT valor]
[restricciones],
.....
);
```

```
CREATE TABLE PENSUM (
SIGLA          VARCHAR2(6),
ASIGNATURA     VARCHAR2(30),
SEMESTRE       NUMBER DEFAULT 0)
```

CONSTRAINTS (RESTRICCION)

- Se definen al crear una tabla para limitar o restringir algo.
- Hay diferentes tipos por ejemplo NOT NULL, que no permite que se ingresen valores nulos en esa columna.
- UNIQUE no permite ingresar valores duplicados.
- PRIMARY KEY para definir una clave primaria, que no se puede repetir y tampoco tener valores nulos.
- FOREIGN KEY para definir un columna mediante la cual se unen una tabla detalle con una tabla maestra.
- CHECK poner un tipo de condición a alguna columna.

```
CREATE TABLE PENSUM (
SIGLA          VARCHAR2(6) PRIMARY KEY,
ASIGNATURA     VARCHAR2(30) NOT NULL,
SEMESTRE       NUMBER DEFAULT 0)
```

Restricción UNIQUE

La restricción "unique" impide la duplicación de claves alternas (no primarias), es decir, especifica que dos registros no puedan tener el mismo valor en un campo.
Se permiten valores nulos.
Se pueden aplicar varias restricciones de este tipo a una misma tabla, y pueden aplicarse a uno o varios campos que no sean clave primaria.

```
create table DEPARTAMENTOS (
  depnro      number PRIMARY KEY,
  nom_dep     varchar2(50) UNIQUE,
  ciudad      varchar2(50) not null
);
```

```
create table DEPARTAMENTOS (
depnro      number,
nom_dep     varchar2(50),
ciudad      varchar2(50) not null,
constraint pk_departamentos primary key (depnro),
Constraint uq_departamentos unique(nom_dep)
);
```

FOREING KEY

Permite crear una relación entre una tabla detalle y una tabla maestra.

```
create table EMPLEADOS (
  empnro        number,
  nomemp        varchar2(50) not null,
  apeemp        varchar2(50) not null,
  cargo         varchar2(50),
  fechaing      date DEFAULT sysdate,
  salario       number(8,2) default 0,
  email         varchar2(50),
  depnro        number references DEPARTAMENTOS(depnro),
  constraint pk_empleados primary key (empnro)
);
```

FOREING KEY

Permite crear una relación entre una tabla detalle y una tabla maestra.

```
create table EMPLEADOS (
 empnro       number,
 nomemp       varchar2(50) not null,
 apeemp       varchar2(50) not null,
 cargo        varchar2(50),
 fechaing     date DEFAULT sysdate,
 salario      number(8,2) default 0,
 email        varchar2(50),
 depnro       number,
 constraint pk_empleados primary key (empnro),
 constraint fk_emp_dept foreign key (depnro)
 references DEPARTAMENTOS(depnro)
);
```

Restricción CHECK

La restricción "check" especifica los valores que acepta un campo, evitando que se ingresen valores inapropiados.

```
create table EMPLEADOS (
 empnro      number primary key,
 nomemp      varchar2(50) not null,
 apeemp      varchar2(50) not null,
 sexo        char,
 salario     number(8,2) CHECK (salario>1000),
 constraint CK_EMP_sexo check (sexo='F' or sexo ='M')
);
```

---->>>>> check (sexo in ('F','M'));

Crear tabla a partir de otra

Podemos crear una tabla e insertar datos en ella en una sola sentencia consultando otra tabla (o varias) con esta sintaxis:

```
create table NOMBRENUEVATABLA
 as SUBCONSULTA;
```

Es decir, se crea una nueva tabla y se inserta en ella el resultado de una consulta a otra tabla.

```
5  create table vendedores as (select * from empleados where cargo='Vendedor');
6
7
8  select * from vendedores;
```

EMPNRO	NOMBRE	CARGO	FECHA_ING	SALARIO	DEPTNRO
1	Luis Alberto Palle Rodriguez	Vendedor	15-JAN-19	2000	4
13	Noemi Guarachi Vasquez	Vendedor	15-JAN-19	1500	4
14	Maria Ramirez Chipana	Vendedor	15-JAN-19	2500	4
18	Brilda Condori Chura	Vendedor	15-JAN-19	2000	4
19	Bismarck Mayta Tintaya	Vendedor	25-MAR-20	2000	4
20	Mauricio Luna Romero	Vendedor	05-OCT-20	2000	4

Otras formas de insertar

Se pueden insertar datos en una tabla que pueden ser el resultado de una sub-consulta.

```
insert into empleados (SELECT * FROM EMPLOYEES WHERE JOB_ID = 'ST_CLERK');
```

Modificar Tablas
ADD - MODIFY

- Para poder modificar una tabla se emplea el comando ALTER TABLE.
- Podremos añadir una nueva columna.
- Podremos modificar las características de una columna.
- Podremos definir valores por defecto.
- Podremos borrar columnas con ciertas restricciones.
- Poner una tabla en modo read only.

AÑADIR UNA COLUMNA

```
ALTER TABLE EMPLEADOS1
ADD (EMAIL VARCHAR2(30));
```

No se puede poner la clausula NOT NULL

MODIFICAR UNA COLUMNA (TAMAÑO,TIPO,VALOR POR DEFECTO)

```
ALTER TABLE EMPLEADOS1
MODIFIY  (EMAIL VARCHAR2(50));
```

No se puede reducir tamaño si tiene datos

Modificar Tablas
DROP – READ ONLY

- Se puede eliminar una columna

```
ALTER TABLE EMPLEADOS1
DROP (EMAIL);
```

- Se puede poner una tabla en READ ONLY

```
ALTER TABLE EMPLEADOS1 READ ONLY;
```

```
ALTER TABLE EMPLEADOS1 READ WRITE;
```

Borrar Tablas

- Se puede eliminar una tabla
  DROP TABLE nombretabla;

  Se puede eliminar siempre y cuando no existan restricciones que lo impidan, o eliminar primero la restricción.

  DROP TABLE nombretabla CASCADE CONSTRAINTS;

Recuperar tablas borradas

Cuando se borra una tabla, la base de datos no libera inmediatamente el espacio asociado con la tabla. La base de datos renombra la tabla y la coloca junto sus objetos asociados en el recycle bin, donde en caso de que se haya borrado por error, puede ser recuperada posteriormente.
Esta opción se llama Flashback Drop y se utiliza la sentencia FLASBACK TABLE para restaurar la tabla.
El recycle bin es una tabla del diccionario de datos que contienen información sobre objetos borrados.

```
select object_name, original_name, type,
can_undrop, can_purge
from recyclebin;
flashback table DEPARTAMENTO to before drop;
```

Ver Estructura de Tabla

```
select table_name, tablespace_name, status
from user_tables
where tablespace_name = 'SYSAUX';
```

LIVESQL_USERS

```
select column_id, column_name , data_type
from user_tab_columns
where table_Name = 'EMPLEADOS'
order by column_id;
```

Consultar las Restricciones

El catálogo "user_constraints" muestra la información referente a todas las restricciones establecidas en las tablas del usuario actual, devuelve varias columnas:

```
select * from user_constraints
where table_name in ('EMPLOYEES','COUNTRIES');
```

El catálogo "user_cons_columns" muestra la información referente a todas las restricciones establecidas en las tablas del usuario actual:

```
select * from user_cons_columns;
```

Eliminar Restricciones

Para eliminar una restricción, la sintaxis básica es la siguiente:

```
alter table NOMBRETABLA
 drop constraint NOMBRERESTRICCION;
```

Para eliminar la restricción "UQ_DEPARTAMENTO_NOMDEP" de la tabla DEPARTAMENTO:

```
alter table DEPARTAMENTO
 drop constraint UQ_DEPARTAMENTO_NOMDEP;
```

Cuando eliminamos una tabla, todas las restricciones que fueron establecidas en ella, se eliminan también. Una restricción de control no puede modificarse, hay que eliminar la restricción y volver a crearla.

VISTAS

Una vista es un objeto. Una vista es una alternativa para mostrar datos de varias tablas; es como una tabla virtual que almacena una consulta. Los datos accesibles a través de la vista no están almacenados en la base de datos, en la base de datos se guarda la definición de la vista y no el resultado de ella.

Entonces, una vista almacena una consulta como un objeto para utilizarse posteriormente. Las tablas consultadas en una vista se llaman tablas base. En general, se puede dar un nombre a cualquier consulta y almacenarla como una vista. Una vista suele llamarse también tabla virtual porque los resultados que retorna y la manera de referenciarlas es la misma que para una tabla.

VISTAS

create view NOMBREVISTA as
SUBCONSULTA;

El contenido de una vista se muestra con un "select":

select * from NOMBREVISTA;

```
create view vista_empleados as
  select nombre,salario,nom_dep as Departamento
   from empleados join departamento
   on departamento.depnro=empleados.depnro;

   select * from vista_empleados;
   describe vista_empleados
   drop view nombrevista;
```

elimina la vista similar a una tabla

Indices

Objetivo es acelerar la recuperación de información y que es útil cuando la tabla contiene miles de registros, cuando se realizan operaciones de ordenamiento y agrupamiento, etc.

Los campos por los que sería útil crear un índice, son aquellos por los cuales se realizan búsquedas con frecuencia: claves primarias, claves foráneas o campos que combinan tablas. No recomendable crear índices sobre campos que no se usan con frecuencia en consultas o en tablas muy pequeñas.

```
create index NOMBREINDICE
on NOMBRETABLA(CAMPOS);
```

Indices

Si se intenta crear un índice único para un campo que tiene valores duplicados, Oracle no lo permite.
Los campos de tipo "long" y "long raw" no pueden indexarse.
Una tabla puede indexarse por un campo (o varios).
Cuando creamos una restricción "primary key" o "unique" sobre una tabla, Oracle automáticamente crea un índice sobre el campo (o los campos) de la restricción y le da el mismo nombre que la restricción. En caso que la tabla ya tenga un índice, Oracle lo usa, no crea otro.

all_indexes all_constraints VISTAS de CONSULTA

Eliminar Indices

Los índices se eliminan con "drop index"; la siguiente es la sintaxis básica:

drop index NOMBREINDICE;

Los índices usados por las restricciones "primary key" y "unique" no pueden eliminarse con "drop index", se eliminan automáticamente cuando quitamos la restricción.

Si eliminamos una tabla, todos los índices asociados a ella se eliminan.

Secuencias

Una secuencia (sequence) se emplea para generar valores enteros secuenciales únicos y asignárselos a campos numéricos; se utilizan generalmente para las claves primarias de las tablas garantizando que sus valores no se repitan.
Una secuencia es una tabla con un campo numérico en el cual se almacena un valor y cada vez que se consulta, se incrementa tal valor para la próxima consulta.
Sintaxis general:

```
create sequence NOMBRESECUENCIA
 start with VALORENTERO
 increment by VALORENTERO
 maxvalue VALORENTERO
 minvalue VALORENTERO
 cache NRO  cycle | nocycle;
```

Si no se especifica ninguna cláusula, excepto el nombre de la secuencia, por defecto, comenzará en 1, se incrementará en 1, el mínimo valor será 1, el máximo será 999999999999999999999999999 y "nocycle".

Secuencias

NEXTVAL (Siguiente valor de la secuencia)
CURRVAL (Valor actual de la secuencia)

```
create sequence s_coddep;

INSERT INTO EMPLEADOS
VALUES (s_coddep.nextval,'Depto PRUEBA','Ciudad Prueba');

select * from all_sequences;

alter sequence NOMBRE_SECUENCIA;

DROP SEQUENCE nombre_secuencia;
```