

**Antes de comenzar la clase una Reflexión: LA CULPA ES DE LA VACA.**

Se estaba promoviendo la exportación de artículos colombianos de cuero a los Estados Unidos, y un investigador de la firma Monitor decidió entrevistar a los representantes de dos mil almacenes en Colombia. La conclusión de la encuesta fue determinante: los precios de tales productos son altos , y calidad muy baja.

El investigador se dirigió entonces a los fabricantes para preguntarle sobre esta conclusión. Recibió esta respuesta: no es culpa nuestra; las curtiembres(**Curtidoras de Cuero**) tienen una tarifa arancelaria de protección de quince por ciento para impedir la entrada de cueros argentinos.

A continuación . le pregunto a los propietarios de las curtiembres, ellos contestaron: no es culpa nuestra; el problema radica en los mataderos, porque sacan cueros de mala calidad. Como la venta de carne les reporta mayores ganancias con menor esfuerzo, los cueros les importan muy poco.

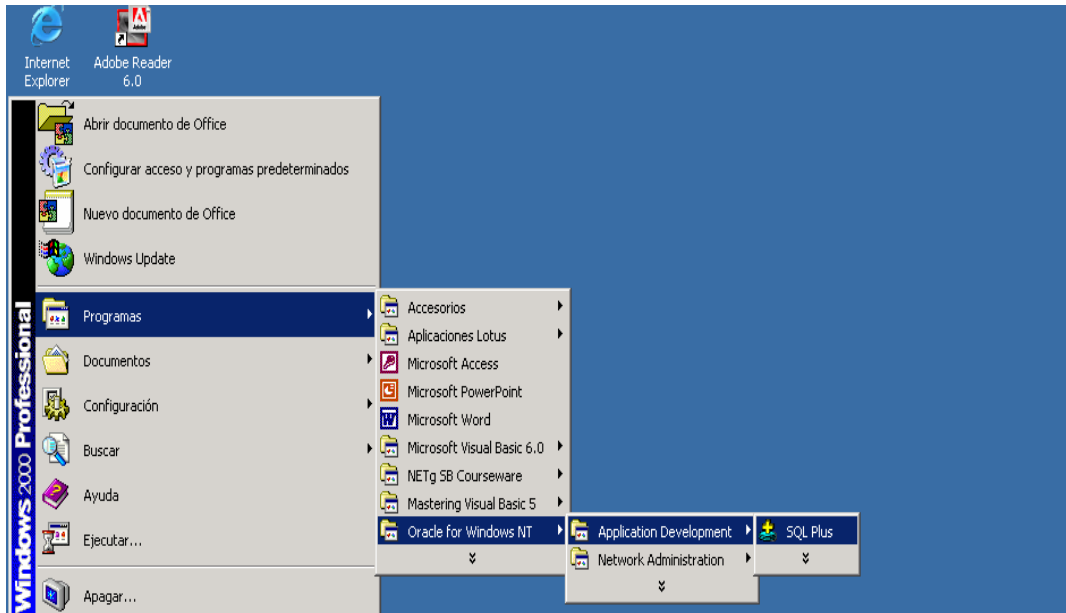
Entonces el investigador, armado de toda su paciencia. Se fue a un matadero. Allí le dijeron: no es culpa nuestra; el problema es que los ganaderos gastan muy poco en venenos contra las garrapatas y además marcan por todas partes a las reses para evitar que se las roben, practicas que destruyen los cueros.

Finalmente, el investigador decidió visitar a los ganaderos. Ellos también dijeron : no es culpa nuestra; esas estúpidas vacas se restriegan contra los alambres de púas para aliviarse de las picaduras.

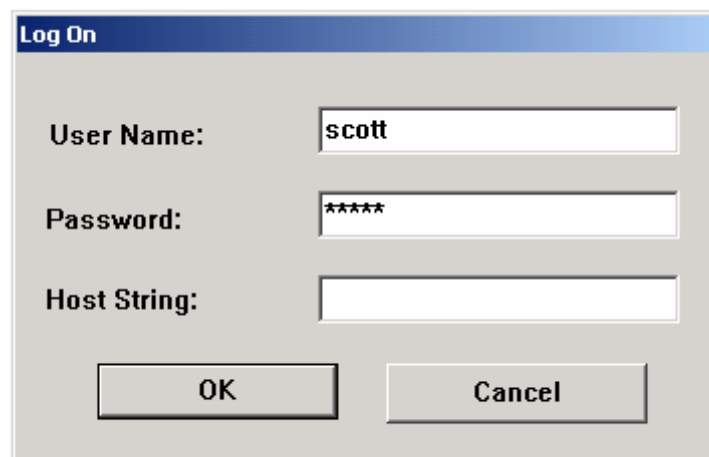
La conclusión del consultor extranjero fue muy simple: los productores colombianos de carteras de cuero no pueden competir en el mercado de Estados Unidos “¡ porque sus vacas son estúpidas!

## Sección No. 1: Ingreso al SQL\*Plus de ORACLE.

1. Entrar al SQL PLUS de ORACLE. Para ello ir a “Inicio”, “Programas”, “Oracle”, “Application Development”, “SQL Plus”.



2. Cuando solicite “User Name” responder “scott” y el “Password” es “tiger”. Luego seleccionar “OK”.



En estas Notas lo primero que el estudiante aprenderá, será a crear una Tabla en la Base de Datos, con el objetivo de que pueda crear sus propias tablas para luego realizar los Ejercicios.

En este orden de ideas, tenemos que el nombre de una tabla en ORACLE debe cumplir con las reglas y estándares para cualquier objeto in ORACLE.

## **REGLAS**

1. La longitud debe estar entre 1 – 30 caracteres, el primer carácter debe ser alfabético.
2. Debe contener sólo caracteres.
3. No debe ser una palabra reservada de ORACLE
4. No debe existir otro objeto con el mismo nombre y del mismo usuario.

## **NOTA**

Las letras mayúscula y minúsculas son tratadas de igual manera.

## **CREANDO UNA TABLA**

La sintaxis para crear una Tabla puede ser exigente debido al Grupo de Parámetros opcionales que puede llevar. Vamos a revisar la Sintaxis completa, pero se hará un uso sencillo en esta parte inicial.

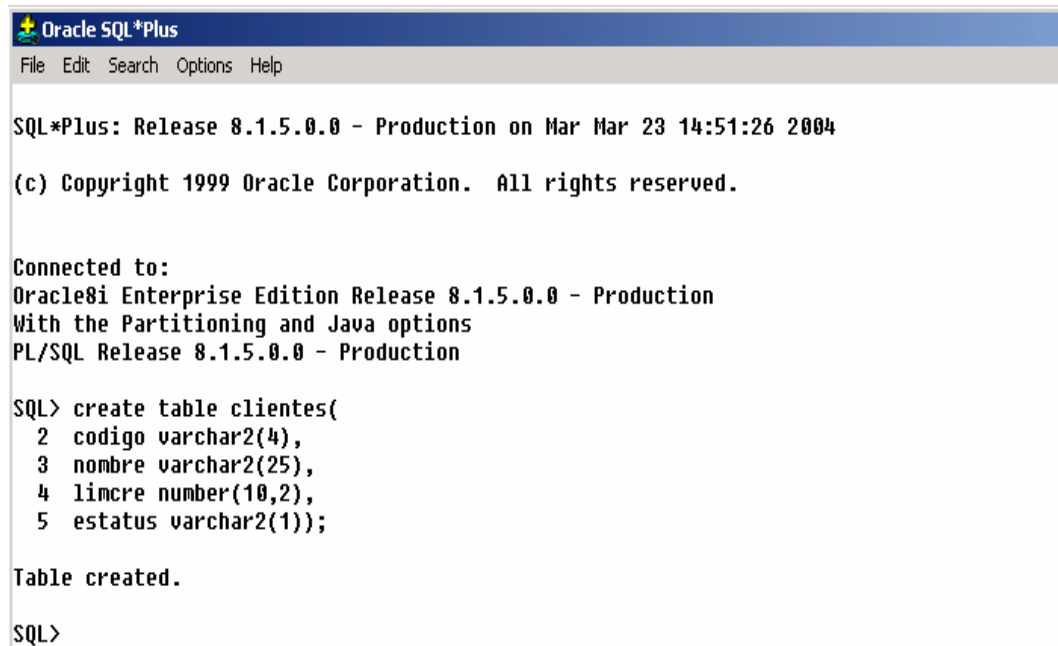
## Sintaxis

```
CREATE TABLE Nombre (  
    Columna1 TIPO NOT NULL,  
    Columna2 TIPO NOT NULL,  
    Columna3 TIPO NOT NULL,  
    Columna4 TIPO NOT NULL,  
    Columna5 TIPO NOT NULL,  
    Columna6 TIPO NOT NULL,  
    Columna7 TIPO          ,  
    Columna8 TIPO          ,  
    Columna9 TIPO          ,  
    PRIMARY KEY (columna1, columna2) CONSTRAINT ConPrimNombre,  
    FOREIGN KEY (columna3)  
        REFERENCES tabla (columna) CONSTRAINT ConForNombre,  
    UNIQUE (columna5, columna6) CONSTRAINT ConUniNombre,  
    CHECK (columna7 condición) );
```

Ahora bien, utilizando una sintaxis sencilla podemos en esta parte resumir la creación de una tabla como sigue:

```
CREATE TABLE Nombre (  
    Columna1 TIPO,  
    Columna2 TIPO,  
    Columna3 TIPO);
```

Por ello, para crear la Tabla de Clientes se haría lo siguiente:



```
Oracle SQL*Plus  
File Edit Search Options Help  
SQL*Plus: Release 8.1.5.0.0 - Production on Mar Mar 23 14:51:26 2004  
(c) Copyright 1999 Oracle Corporation. All rights reserved.  
  
Connected to:  
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production  
With the Partitioning and Java options  
PL/SQL Release 8.1.5.0.0 - Production  
  
SQL> create table clientes(  
  2  codigo varchar2(4),  
  3  nombre varchar2(25),  
  4  lincre number(10,2),  
  5  estatus varchar2(1));  
  
Table created.  
  
SQL>
```

Los otros parámetros de la Instrucción se describen a continuación:

CONSTRAINT	USO
PRIMARY KEY	Para identificar a una o varias columnas como clave primaria de la tabla.
CHECK	Para especificar que el contenido de una columna debe cumplir con una condición dada.
FOREIGN KEY	Para identificar a una o varias columnas como clave foránea de una tabla. La (s) columna (s) debe (n) estar definidas como clave primaria en la tabla referenciada.
NOT NULL	Especifica que la columna tiene que tener un valor no nulo.
UNIQUE	Para especificar a una o varias columnas como clave única o clave alterna.

## ESTRUCTURA DE UNA TABLA

Para revisar la estructura de una tabla, se utiliza la siguiente instrucción:

DESCRIBE Nombre de la Tabla

En este caso se puede resumir el comando como: DESC Nombre de la Tabla.  
ORACLE responderá con la estructura de la Tabla.

### Ejemplo

```
SQL> desc clientes
Name                               Null?    Type
-----
CODIGO                             VARCHA2(4)
NOMBRE                             VARCHA2(25)
LIMCRE                             NUMBER(10,2)
ESTATUS                             VARCHA2(1)
SQL>
```

## SALIR DEL SQL/PLUS DE ORACLE.

La Instrucción para salir es EXIT.

## INCLUYENDO INFORMACION EN UNA TABLA

La instrucción INSERT permite insertar filas en una tabla.

### Sintaxis

```
INSERT INTO nombre_tabla [(columna1, columna2 . . .)]  
VALUES (valor1, valor2 . . .)
```

### Incluyendo algunas columnas

Cuando la información que se va a incluir en una tabla no corresponde a todas las columnas de dicha tabla, se deben especificar todas las columnas de dicha tabla, es decir, se deben especificar todas las columnas que vamos a utilizar. Debe existir una correspondencia univoca entre las columnas que se van a incluir y los valores especificados en la cláusula VALUES.

### Sintaxis

```
SQL> INSERT INTO nombre_tabla (columna1, columna2, . . . , columnaN)  
2 VALUES (valor1, valor2, . . . , valorN);
```

### Ejemplo

Incluir un nuevo Cliente en la tabla Clientes, del cual solo tenemos su Código y su nombre y por supuesto su estatus será 'A'. En este caso no conocemos su Límite de crédito.

```
SQL> insert into clientes(codigo,nombre,estatus)  
2 values ('0001','MAKRO','A');  
  
1 row created.  
  
SQL>
```

### Incluyendo todas las columnas

Insertando valores en todas las columnas. Si no se especifican las columnas, se asume, que las columnas están en el mismo orden en que se suministran los valores.

### Sintaxis

```
SQL> INSERT INTO nombre_tabla  
      2  VALUES (valor1, valor2, . . . , valorN);
```

### Ejemplo

Incluir los Datos Completos de un Cliente.

```
SQL> insert into clientes  
      2  values ('0007','SIDETUR',7500,'A');  
  
1 row created.  
SQL> |
```

## CONSULTANDO LA INFORMACION EN UNA TABLA

Ahora bien, hemos ejecutado dos instrucciones de Inclusión, pero todavía no hemos constatado si se está incluyendo correctamente. Para ello, necesitamos revisar si las Filas o Registros creados, realmente se encuentran en la Tabla de Clientes. En consecuencia, vamos a revisar un Comando que en mi opinión es el más poderoso de SQL. Me refiero al comando **SELECT**, el cual permite seleccionar Filas de cualquier Tabla, aunado al hecho, de que este comando es capaz de ejecutar un grupo importante de funciones adicionales.

La instrucción **SELECT** nos permite obtener información de una o varias tablas. Cuando utilizamos varias tablas en la instrucción **SELECT** y necesitamos hacer referencia a una columna que se encuentra definida con el mismo nombre en dos o más tablas debemos cualificar las columnas, que no es más que preceder a dicha columna con el nombre de la tabla.

### Sintaxis

```
SELECT campo1, campo2. . . . , campoN  
FROM tabla
```

### Ejemplo

Mostrar el código y el nombre de todos los clientes que se encuentran en la tabla CLIENTES.

```
SQL> select codigo,nombre  
2 from clientes;  
  
CODI NOMBRE  
-----  
0001 MAKRO  
0007 SIDETUR
```

### La cláusula WHERE en el SELECT

Adicionando la Cláusula WHERE a una Instrucción SELECT, podemos restringir la consulta a valores específicos que deseamos consultar.

### Ejemplo

Mostrar el Código, Nombre y Límite de Crédito del Cliente cuyo Código es igual a 0007.

```
SQL> select codigo,nombre,limcre  
2 from clientes  
3 where codigo='0007';  
  
CODI NOMBRE LIMCRE  
-----  
0007 SIDETUR 7500
```

Mas adelante se verá en detalle, que la cláusula WHERE tiene un abanico importante de opciones para ser utilizada en conjunto con el SELECT.

Ya en este punto se ha aprendido a crear una Tabla y se le han agregado valores a esa Tabla. Adicionalmente, se sabe como consultar los Datos en la Tabla. Por ello, se puede aprender como Modificar y Agregar Columnas a una Tabla.



## MODIFICAR Y AGREGAR COLUMNAS A UNA TABLA

Una vez creada una Tabla, se puede Modificar su estructura mediante el uso del ALTER TABLE.

### Sintaxis

```
ALTER TABLE Tabla  
[ ADD Tipo de datos de la Columna]  
[Modify Columna (Tipo de datos)]
```

### Ejemplo

Ya se sabe la estructura de la Tabla de Clientes:

```
SQL> desc clientes  
Name                               Null?   Type  
-----  
CODIGO                             VARCHAR2(4)  
NOMBRE                             VARCHAR2(25)  
LIMCRE                             NUMBER(10,2)  
ESTATUS                             VARCHAR2(1)  
  
SQL>
```

Como ejemplo, se desea adicionar la columna Teléfono a la Tabla de Clientes:

```
SQL> alter table clientes  
2 add telefono varchar2(10);
```

Tabla modificada.

```
SQL> desc clientes  
Nombre                               ¿Nulo?   Tipo  
-----  
CODIGO                             VARCHAR2(4)  
NOMBRE                             VARCHAR2(25)  
LIMCRE                             NUMBER(10,2)  
ESTATUS                             VARCHAR2(1)  
TELEFONO                             VARCHAR2(10)
```

Seguidamente se desea que la columna Nombre sea de 30 caracteres:

```
SQL> alter table clientes
2 modify nombre varchar2(30);
```

Tabla modificada.

```
SQL> desc clientes
```

Nombre	¿Nulo?	Tipo
CODIGO		VARCHAR2(4)
NOMBRE		VARCHAR2(30)
LIMCRE		NUMBER(10,2)
ESTATUS		VARCHAR2(1)
TELEFONO		VARCHAR2(10)

```
SQL> select * from clientes;
```

CODI	NOMBRE	LIMCRE	E	TELEFONO
0001	MAKRO		A	
0007	SIDETUR	7500	A	

Surge una incógnita. ¿Qué pasa si se desea dejar la Tabla de Clientes como estaba?. Es decir sin la Columna Teléfono y el Nombre de tipo VARCHAR2 de 25. El lector debe investigar como lograrlo.

Alter table clientes  
Drop column telefono;

## Volviendo a la Inclusión de Datos.

En este momento, ya el lector ha colocado la Tabla de Clientes en su forma original:

```
SQL> desc clientes
```

Name	Null?	Type
CODIGO		VARCHAR2(4)
NOMBRE		VARCHAR2(25)
LIMCRE		NUMBER(10,2)
ESTATUS		VARCHAR2(1)

```
SQL>
```

En consecuencia, se seguirá trabajando con esta estructura.

## Incluyendo usando parámetro

Cuando se desea incluir información de manera iterativa se emplea la macrosustitución.

### Sintaxis

```
SQL> insert into clientes
  2 values ('&codigo','&nombre',&limcre,'A');
Enter value for codigo: 0002
Enter value for nombre: BECO
Enter value for limcre: 2500
old  2: values ('&codigo','&nombre',&limcre,'A')
new  2: values ('0002','BECO',2500,'A')

1 row created.
```

&codigo me permite leer el código, va entre comillas porque es Alfanumérico.  
&limcre me permite leer el Límite de Crédito no lleva comillas. El estatus en una Inclusión siempre será de valor A para que quede activo.

Para revisar todos los Campos de una tabla utilizo el comodín \* en la Instrucción SELECT como sigue:

```
SQL> select * from clientes;

CODI NOMBRE                                LIMCRE E
-----
0001 MAKRO                                A
0007 SIDETUR                               7500 A
0002 BECO                                  2500 A
```

### Incluyendo Múltiples Filas usando Query.

Cuando se desea incluir información proveniente de otra tabla, se procede de la siguiente manera:

### Sintaxis

```
SQL> INSERT INTO nombre_tabla_destino (C1, C2, C3)
```

- 2 SELECT c1, c2, c3
- 2 FROM nombre\_tabla\_fuente
- 3 WHERE condicion;

### Ejemplo

Imaginemos que nos piden crear una tabla de Proveedores con una estructura idéntica a la de Clientes. Luego nos piden que pasemos a todos nuestros clientes a la Tabla de proveedores. Eso lo podemos realizar de manera automática de la siguiente forma:

1. Usted debe crear una Tabla idéntica a la de Clientes, pero ahora llamarla Proveedores.(Ya sabe como hacerlo).
2. Crear una Secuencia, que puede comenzar en 1 e incrementarse de 1 en uno. Los Proveedores no van a quedar con los mismos códigos que los clientes, pero se van a incluir de manera automática.

```
SQL> create sequence seqproveedor  
2 start with 1 increment by 1;
```

Sequence created.

3. Ejecutar el comando SELECT necesario para realizar la Inclusión automática.

```
SQL> insert into proveedores  
2 (codigo,nombre,limcre,estatus)  
3 select seqproveedor.nextval,nombre,limcre,estatus  
4 from clientes;
```

3 rows created.

4. Revisar si se hizo la inclusión de manera correcta.

```
SQL> select * from proveedores;
```

CODI	NOMBRE	LIMCRE	E
1	MAKRO		A
2	SIDETUR	7500	A
3	BECO	2500	A

BORRAR SECUENCIA con DROP SÉQUENSE.

En este momento ya se conoce una vía para respaldar una Tabla utilizando una secuencia. Ahora bien, existe otra vía para emitir un respaldo de una tabla sin utilizar una secuencia. En este caso, la intención sería respaldar una Tabla de manera idéntica a la existente. Para ello existe una instrucción que me permite realizar este proceso de una manera sencilla. La sintaxis es la siguiente:

```
CREATE TABLE NombreTablaRespaldo  
AS SELECT * FROM TablaExistente;
```

Se requiere respaldar la Tabla de Clientes mediante una instrucción que haga lo siguiente:

1. Cree una Tabla idéntica a la Tabla que se desea respaldar.(Clientes en este caso).
2. Pase las Filas de una manera idéntica desde la Tabla ya existente a una Tabla de respaldo.

Para lograr los puntos 1 y 2, se procede como sigue:

```
SQL> create table respaldo_clientes  
2 as select * from clientes;
```

**Tabla creada.**

Ya se ha creado un respaldo de la Tabla de Clientes denominado respaldo\_clientes. Ya en esta tabla han sido incluidos los clientes existentes. Para verificar si se hizo la inclusión, se escribe:

```
SQL> select * from respaldo_clientes;
```

CODI	NOMBRE	LIMCRE	E
0007	SIDETUR	7500	A
0001	MAKRO		A
0002	BECO	2500	A

## MODIFICANDO INFORMACIÓN DE UNA TABLA

La instrucción UPDATE permite modificar la información contenida en una tabla.

## Sintaxis

```
UPDATE nombre_tabla
SET campo1 = valor1,
    Campo2 = valor2,
    C3      = (select A
                From nombre_tabla
                Where condición),
    (c4, c5) = (Select A, B
                From nombre_tabla
                Where condicion)
[ WHERE condición ]
```

## Modificando una Fila(Un registro).

### Ejemplo

Se desea cambiarle el Límite de Crédito al Proveedor SIDETUR cuyo código es 2.

Para ello se procede como sigue:

```
SQL> update proveedores
2 set limcre = 5600
3 where codigo= 2;

1 row updated.
```

Para chequear la actualización:

```
SQL> select * from proveedores;
CODI NOMBRE                LIMCRE E
-----
1    MAKRO                  A
2    SIDETUR                5600 A
3    BECO                   2500 A
```

**NOTA:** el uso del WHERE en el UPDATE para un caso como este, es muy importante, de lo contrario habría cambiado todas las Filas (Registros) de la Tabla.

### Modificando varias Filas.

### Ejemplo

A los Proveedores que tengan un Límite de crédito Inferior a 3000, colocarlo en Cero y además cambiar su Estatus a la letra E.

```
SQL> update proveedores
2   set limcre = 0,
3     estatus = 'E'
4   where limcre<3000;

1 row updated.
```

Luego:

```
SQL> select * from proveedores;
CODI NOMBRE                LIMCRE E
-----
1    MAKRO                  A
2    SIDETUR                5600 A
3    BECO                   0 E
```

Ahora bien, se generó una diferencia en los Datos que tiene BECO como cliente y los que tiene como Proveedor. Como ejemplo se van a traer los datos que tiene como proveedor y se le colocarán al cliente, para así ejemplificar el uso del UPDATE utilizando datos de otra Tabla.

```
SQL> update clientes
  2  set (limcre,estatus)=(select limcre,estatus
  3                        from proveedores
  4                        where codigo=3)
  5  where codigo='0002';

1 fila actualizada.

SQL> select * from clientes;

CODI NOMBRE                                LIMCRE E
-----
0007 SIDETUR                                7500 A
0001 MAKRO                                    A
0002 BECO                                    0 E

SQL>
```

## ELIMINANDO INFORMACION DE UNA TABLA

La instrucción DELETE elimina las filas de una tabla.

### Sintaxis

```
DELETE FROM Nombre_Tabla
WHERE Condición;
```

### Ejemplo



Eliminar todos los Proveedores cuyo Estatus sea igual a E.

```
SQL> delete from proveedores
2  where estatus='E';

1 row deleted.
```

**NOTA:** el uso del WHERE en el DELETE para un caso como este, es muy importante, de lo contrario habría Eliminado todas las Filas (Registros) de la Tabla.

Luego:

```
SQL> select * from proveedores;

CODI NOMBRE                                LIMCRE E
-----
1     MAKRO                                A
2     SIDETUR                               5600 A
```

## USO DE TRANSACCIONES. (COMMIT y ROLLBACK).

Varios Manejadores de Bases de Datos, incluyendo ORACLE, poseen una herramienta basada en TRANSACCIONES.

En ORACLE, la herramienta en cuestión utiliza los comando COMMIT y ROLLBACK. Por ello, cuando se están haciendo actualizaciones sobre las Tablas de la Base de Datos, realmente existe un proceso que pudiera catalogarse de “intermedio”, en donde las tablas no se actualizan “Completamente” hasta que no se ejecuta un COMMIT (El cual, se puede traducir como: Fin exitoso de una

Transacción). Si por el contrario, no se desea que las operaciones se actualicen completamente sobre las tablas, se ejecuta un ROLLBACK. (Se puede traducir como: Retroceder todas las operaciones sobre las Base de Datos hasta el último COMMIT).

### **Ejemplo.**

Se desea insertar un nuevo proveedor, denominado BECO con código 3 y Límite de Crédito 3500.

Para poder entender esta parte, lo primero que el estudiante hará será ejecutar la instrucción COMMIT así todas las operaciones que ha realizado hasta ahora quedarán grabadas y no se podrán retroceder.

```
SQL> commit;  
Commit complete.
```

Ahora se va a realizar la inclusión del nuevo proveedor:

```
SQL> insert into proveedores  
2 values ('3','BECO',3500,'A');  
1 row created.
```

Para verificar si se realizó la Inclusión:

```
SQL> select * from proveedores;
```

CODI	NOMBRE	LIMCRE	E
1	MAKRO		A
2	SIDETUR	5600	A
3	BECO	3500	A

Ahora se desea colocarle 6000 como límite de crédito a MAKRO, para ello se realiza lo siguiente:

```
SQL> update proveedores
  2  set limcre=6000
  3  where codigo='1';
```

1 row updated.

```
SQL> select * from proveedores;
```

CODI	NOMBRE	LIMCRE	E
1	MAKRO	6000	A
2	SIDETUR	5600	A
3	BECO	3500	A

Sin embargo, se pueden retroceder estas operaciones mediante el uso de la instrucción ROLLBACK. Es decir, al aplicar esta instrucción el Proveedor MAKRO estará nuevamente sin Límite de Crédito y desaparecerá el Proveedor BECO.

```
SQL> rollback;

Rollback complete.

SQL> select * from proveedores;

CODI NOMBRE                                LIMCRE E
-----
1     MAKRO                                5600  A
2     SIDETUR                               5600  A
```

Ahora bien, ¿Por qué solo retrocedió las últimas dos operaciones y no retrocedió las anteriores?.

Porque el ROLLBACK retrocede las operaciones hasta el último COMMIT. Debe recordarse, que antes de incluir al Proveedor BECO se ejecutó un COMMIT, en consecuencia se le dijo al manejador que desde allí hacia atrás las operaciones no se pueden devolver.

Se van a realizar nuevamente las dos últimas operaciones.

```
SQL> insert into proveedores
2 values ('3','BECO',3500,'A');

1 row created.

SQL> select * from proveedores;

CODI NOMBRE                                LIMCRE E
-----
3     BECO                                3500  A
1     MAKRO                                5600  A
2     SIDETUR                               5600  A
```

Ahora se va a ejecutar un COMMIT.

```
SQL> commit;
```

Commit complete.

Ahora se le va a cambiar el Límite de Crédito a MAKRO:

```
SQL> update proveedores
  2  set limcre=6000
  3  where codigo='1';
```

1 row updated.

```
SQL> select * from proveedores;
```

CODI	NOMBRE	LIMCRE	E
1	MAKRO	6000	A
2	SIDETUR	5600	A
3	BECO	3500	A

Ahora nuevamente se ejecutará un ROLLBACK:

```
SQL> rollback;
```

Rollback complete.

Cuando se ejecute nuevamente el SELECT, se notará que solo devolvió la última operación, porque la Inclusión del proveedor BECO no se puede retroceder en vista de que ya se ejecutó un COMMIT.

```
SQL> select * from proveedores;
```

CODI	NOMBRE	LIMCRE	E
1	MAKRO		A
2	SIDETUR	5600	A
3	BECO	3500	A

Al salir de ORACLE se ejecuta un COMMIT automático.

## ELIMINACIÓN FÍSICA VS ELMINACIÓN LÓGICA.

Ya se ha visto la manera de Eliminar Filas(Registros) de una Tabla utilizando el Comando DELETE. Es necesario acotar, que este tipo de Eliminación no deja pistas de lo ocurrido, porque la Información se Elimina Físicamente, es decir, deja de existir en las Tablas de la Base de Datos. Este proceso desde el punto de vista de Integridad y Seguridad de los datos se considera inadecuado.

Todo profesional de las Ciencias de la Computación, debe aprender que las eliminaciones en las Tablas deben hacerse de manera lógica. Es decir, la Fila (o Registro) va a seguir físicamente en la Tabla, pero no se tomará en cuenta desde el punto de vista lógico. Para ello, el profesional en cuestión se puede valer de la Columna (Campo) Estatus. Por lo general, se entiende que cuando una Fila tiene la Columna estatus en A es porque se encuentra activa y cuando tiene una E, se encuentra eliminada.

En otras palabras, si se usa la Eliminación Lógica, para eliminar una Fila de una Tabla, no se utilizará el comando DELETE, sino, el Comando UPDATE para cambiar la Columna Estatus por una E. Cambiar este Paradigma, presenta las siguientes ventajas:

1. Se deja una pista de que el Registro existió y fue eliminado en un proceso ejecutado por un usuario del Sistema. Algunas veces ocurre, que los

usuarios culpan al Sistema por desapariciones de información, lo cual afecta también la imagen de sus Diseñadores. Cuando esto ocurre, se puede demostrar que el Registro no ha sido eliminado por error, porque todavía va a existir físicamente en la Tabla. Mas aún, en el futuro el Estudiante aprenderá que además del Estatus cada Tabla debe llevar una Columna(Campo) que identifique el código del usuario que le hizo la última actualización, otra columna para la hora en que se hizo la última actualización y, por supuesto, otra columna que identifique la fecha de la última actualización. Así se sabrá, quién lo borró a que hora y en que fecha.

2. Es una operación mas rápida. Cuando se elimina una Columna(Registro), el Manejador tiene que reordenar las otras columnas, tanto en el área de datos como en el (las) área(s) de Indices asociada(s). En cambio al realizar una Eliminación Lógica solo se cambia el valor de una Columna por ende la respuesta es inmediata. En el ambiente académico no suele notarse esa diferencia porque los estudiantes utilizan tablas, por lo general, con pocos datos. En el ambiente empresarial, cuando los volúmenes de Datos crecen es cuando puede notarse esta diferencia de manera significativa.
3. Se trabaja con estilo y dominio de la Arquitectura de la Base de Datos y sus Tablas. Con regularidad, le insisto a mis alumnos que son estos detalles los que diferencia a los Profesionales de los Empíricos, porque de esta manera se implementan soluciones tecnológicas mas eficientes.

En consecuencia, una operación de eliminación, en lo sucesivo se hará de la siguiente manera:

### Ejemplo

Se desea Eliminar el Proveedor cuyo código es 1.

```
SQL> update proveedores
  2  set estatus='E'
  3  where codigo=1;

1 row updated.
```

Luego para consultar, tendremos que agregar obligatoriamente la cláusula WHERE a la instrucción SELECT.

```
SQL> select *
  2  from proveedores
  3  where estatus='A';
```

CODI	NOMBRE	LIMCRE	E
2	SIDETUR	5600	A

De ahora en adelante, las Filas(Registros) que existen, son aquellas cuyo Estatus es igual a A, el resto existe físicamente, pero desde el punto de vista lógico no existen.

Los Registros Eliminados de manera Lógica, ¿se quedarán en sus Tablas para Siempre?.La respuesta es NO. Cada cierto tiempo, se ejecutará un proceso de depuración que pasará los Registros con estatus E a una tabla Histórica y luego los eliminará físicamente de su tabla original. Este proceso será



ejecutado, en la mayoría de los casos, por el Administrador de la Base de Datos en un horario no laborable para no molestar en lo posible a los usuarios del Sistema.

## PRACTICA I.

1. Crear una Tabla de Facturas con la Siguiete Estructura:

```
SQL> describe facturas;
Name                               Null?    Type
-----
NUMERO                              UARCHAR2(4)
CODCLI                              UARCHAR2(4)
FECHA                               DATE
DIASVIGENCIA                       NUMBER(3)
MONTOTAL                            NUMBER(12,2)
ESTATUS                             UARCHAR2(1)
```

2. Crear una tabla llamada DetFact, con la siguiente estructura:

```
SQL> desc detfacturas;
Name                               Null?    Type
-----
NUMFAC                              UARCHAR2(4)
CODART                              UARCHAR2(4)
CANTIDAD                            NUMBER(5,2)
PRECIO                              NUMBER(7,2)
```

3. Crear una Tabla llamada Artículos con la siguiente estructura:

```
SQL> desc articulos;
Name                               Null?    Type
-----
CODIGO                              UARCHAR2(4)
DESCRIPCION                         UARCHAR2(30)
EXISTENCIA                          NUMBER(10,2)
COSTO                               NUMBER(10,2)
ESTATUS                             CHAR(1)
```

4. Insertar Filas en la Tabla de Clientes para que quede como sigue:

```
SQL> SELECT * FROM CLIENTES;

CODI  NOMBRE                               LIMCRE  E
-----
0001  MAKRO                                1100    A
0004  UCLA                                7800    A
0005  EXITO                                7000    A
0006  LOCATEL                             2500    A
0002  BECO                                 9500    A
0008  RENE DESSES                          7500    A
0009  BANCOR                               9900    A
0010  UPEL                                 1100    A
0003  EPA                                  2800    A

10 rows selected.
```

5. Insertar Filas en la Tabla de Facturas para que quede como sigue:

```
SQL> select * from facturas;
```

NUME	CODC	FECHA	DIASUIGENCIA	MONTOTAL	E
1851	0004	05/01/04	30	4500	A
1852	0003	05/01/04	60	6000	A
1853	0007	06/01/04	45	7500	A
1854	0004	06/01/04	30	1000	A
1855	0002	07/01/04	15	4000	A
1856	0007	07/01/04	30	2000	A
1857	0006	08/01/04	15	4500	A
1858	0009	09/01/04	60	7500	A
1859	0008	09/01/04	15	10000	A
1860	0009	10/01/04	20	2500	A
1861	0008	11/01/04	15	5000	A
1862	0008	12/01/04	10	7500	A

12 rows selected.

6. Insertar Filas en la Tabla de Artículos para que quede como sigue:

```
SQL> select * from articulos;
```

CODI	DESCRIPCION	EXISTENCIA	COSTO	E
0001	Nevera	230	500	A
0002	Enfriador	110	1630	A
0003	Dvd	615	420	A
0004	Cava Cuarto Pequeña	4	1300	A
0005	Ventilador Mediano	700	53	A
0006	Ventilador Pequeño	518	29	A
0007	Licuadaora	200	459	A
0008	Quemador de Cds	133	391	A

8 rows selected.

7. Insertar Filas en la Tabla DetFacturas para que quede como sigue:

```
SQL> select * from detfacturas;
```

NUMF	CODA	CANTIDAD	PRECIO
1851	0002	1	2500
1851	0003	2	1000
1852	0001	4	1500
1853	0004	3	2500
1854	0001	1	1000
1855	0002	1	2500
1855	0001	1	1500
1856	0005	5	200
1856	0006	4	100
1856	0007	1	600
1857	0005	20	200
1857	0008	1	500
1858	0002	2	2500
1858	0004	1	2500
1859	0002	4	2500
1860	0006	5	100
1860	0003	1	1000
1860	0008	2	500
1861	0004	1	2500
1861	0002	1	2500
1862	0007	1	600

NUMF	CODA	CANTIDAD	PRECIO
1862	0006	4	100
1862	0008	2	500
1862	0001	2	1500
1862	0004	1	2500

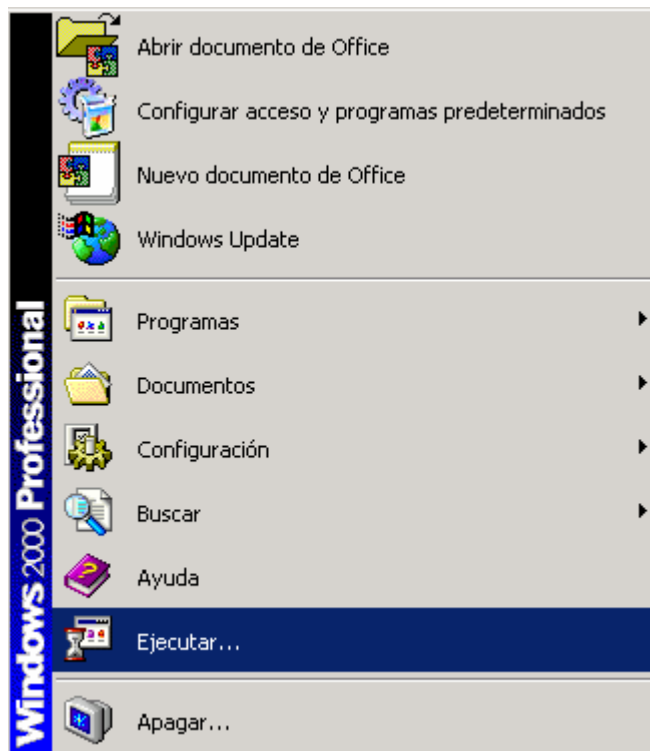
25 rows selected.

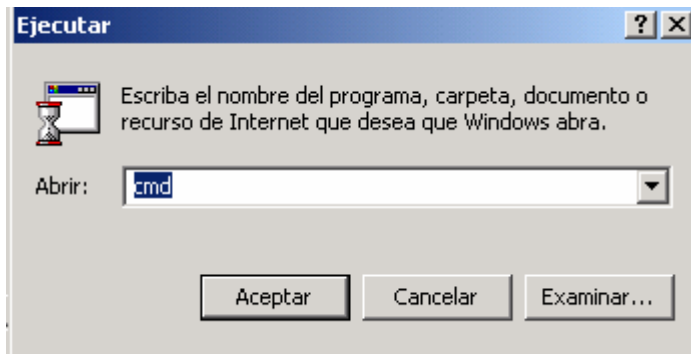
8. Colocar a todos los clientes como Proveedores y que queden con el mismo código en ambas tablas. Utilice un Query sin generar ninguna secuencia. Así también aprenderá a respaldar el contenido de una tabla en otra tabla.

9. Coloque el Límite de Crédito en 7500 al Proveedor SIDETUR.

Ahora bien, usted ha creado las tablas necesarias para trabajar en el resto del curso, sin embargo, al cambiarse de Computador tendría que crear nuevamente esas tablas, lo cual no es eficiente. Lo ideal es que usted se puede llevar las tablas creadas en un DISKETTE o en un PENDRIVE y colocarlas en cualquier Base de Datos que se encuentre en cualquier Computador. Para ello, debe proceder como sigue:

1. Ingresar a los Comandos MS-DOS. Para ello, ir por Inicio, Ejecutar, colocar CMD y Aceptar.





2. Al estar en la Pantalla de Comandos del MS-DOS, ya puede exportar las tablas que desee desde la Base de Datos a un Archivo intermedio. Para ello debe realizar lo siguiente:

2.1. Cambiarse a la Unidad A: para Grabar las Tablas en un DISKETTE.

2.2. Luego ejecutar el comando EXP que sirve para exportar las Tablas que se desean llevar a otra Base de Datos (U otra Computadora).

```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Versión 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Lponte.INF>A:
A:\>exp scott/tiger file=respaldo tables=facturas,detfacturas,clientes,articulos

Export: Release 8.1.5.0.0 - Production on Mar May 25 18:51:59 2004
(c) Copyright 1999 Oracle Corporation. All rights reserved.

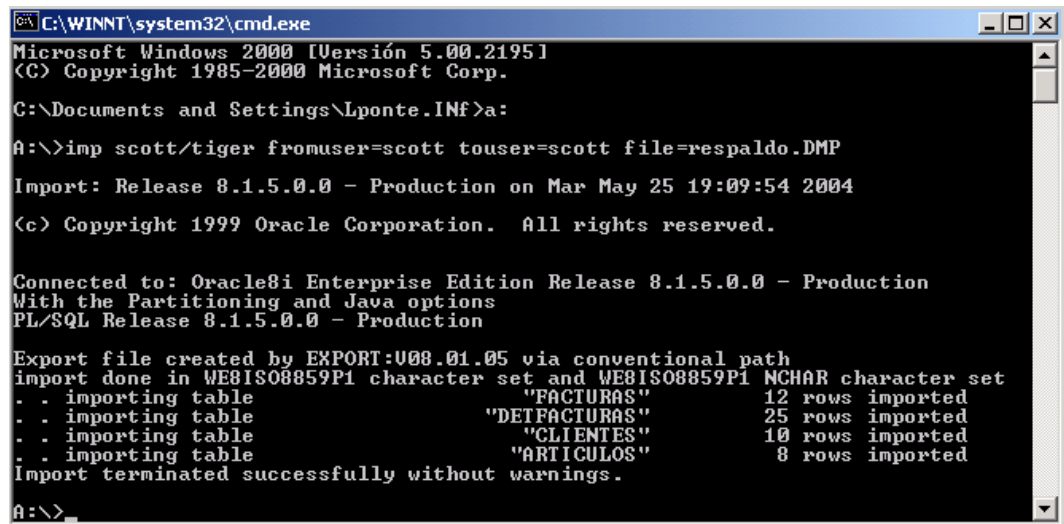
Connected to: Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
Export done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set

About to export specified tables via Conventional Path ...
. . exporting table          FACTURAS          12 rows exported
. . exporting table          DETFACTURAS       25 rows exported
. . exporting table          CLIENTES         10 rows exported
. . exporting table          ARTICULOS         8 rows exported
Export terminated successfully without warnings.
```

2.3 Para verificar que Archivo de respaldo existe en el Diskette, se utiliza el Comando DIR, y mostrará un Archivo llamado respaldo.DMP, el cual contiene las Tablas exportadas.

2.4 Luego para llevar las Tablas a otra Base de Datos (U otra Computadora). Debe ingresar el DISKETTE en la Unidad de la

Computadora en donde colocará los datos. De la misma forma anterior, irá al MS-DOS. Cuando esté allí, debe cambiarse a la Unidad A: y ejecutar el Comando IMP que permite Importar los Datos a la otra Base de Datos.



```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Versión 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\Documents and Settings\Lponte.INf>a:
A:\>imp scott/tiger fromuser=scott touser=scott file=respaldo.DMP
Import: Release 8.1.5.0.0 - Production on Mar May 25 19:09:54 2004
(c) Copyright 1999 Oracle Corporation. All rights reserved.

Connected to: Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production

Export file created by EXPORT:U08.01.05 via conventional path
import done in WE8ISO8859P1 character set and WE8ISO8859P1 NCHAR character set
. . importing table          "FACTURAS"          12 rows imported
. . importing table          "DETFACTURAS"       25 rows imported
. . importing table          "CLIENTES"         10 rows imported
. . importing table          "ARTICULOS"         8 rows imported
Import terminated successfully without warnings.
A:\>
```

**DESPUÉS DE ESTA CLASE TANNNNNNN!!!!!! LARGA: Algo de RELAX.**

**Tomado de la revista Selecciones (Reader's Digest):**

Cuenta una Viejita:

Después de un día pesado de trabajo en la casa, me dispuse a disfrutar de una ducha. Así que me desnudé, y me puse mi gran Gorro de Baño amarillo adornado con rosas rojas. En ese momento entró mi nieto de 4 años diciendo que quería orinar.

No me dio tiempo de cubrirme y me quedé allí avergonzada porque mi nieto estaba mirando mis partes íntimas. Sin embargo, me miró de arriba abajo y me dijo:

Yo ya sé para que sirve eso abuelita.

Ah sí?, le dije con voz débil y avergonzada.

Sí respondió el Niño. Sirve para que no te mojes el Cabello.

**(MAL PENSADOS!!!!!!!!!!!!!!!!!!!!!!).**