

# 8

## Manipulación de Datos

ORACLE®

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

# Objetivos

**Al finalizar esta lección, debería estar capacitado para:**

- **Describir cada sentencia DML**
- **Insertar filas en una tabla**
- **Actualizar las filas de una tabla**
- **Suprimir filas de una tabla**
- **Fusionar las filas de una tabla**
- **Controlar transacciones**

ORACLE

8-2

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Objetivo de la Lección

En esta lección, aprenderá a insertar filas en una tabla y a actualizar y suprimir las filas existentes de una tabla. También aprenderá a controlar transacciones con las sentencias COMMIT, SAVEPOINT y ROLLBACK.

# Lenguaje de Manipulación de Datos

- Se ejecuta una sentencia DML cuando:
  - Agrega filas nuevas a una tabla.
  - Modifica las filas existentes de una tabla.
  - Elimina filas existentes de una tabla.
- Una *transacción* consta de una recopilación de sentencias DML que forman una unidad de trabajo lógica.

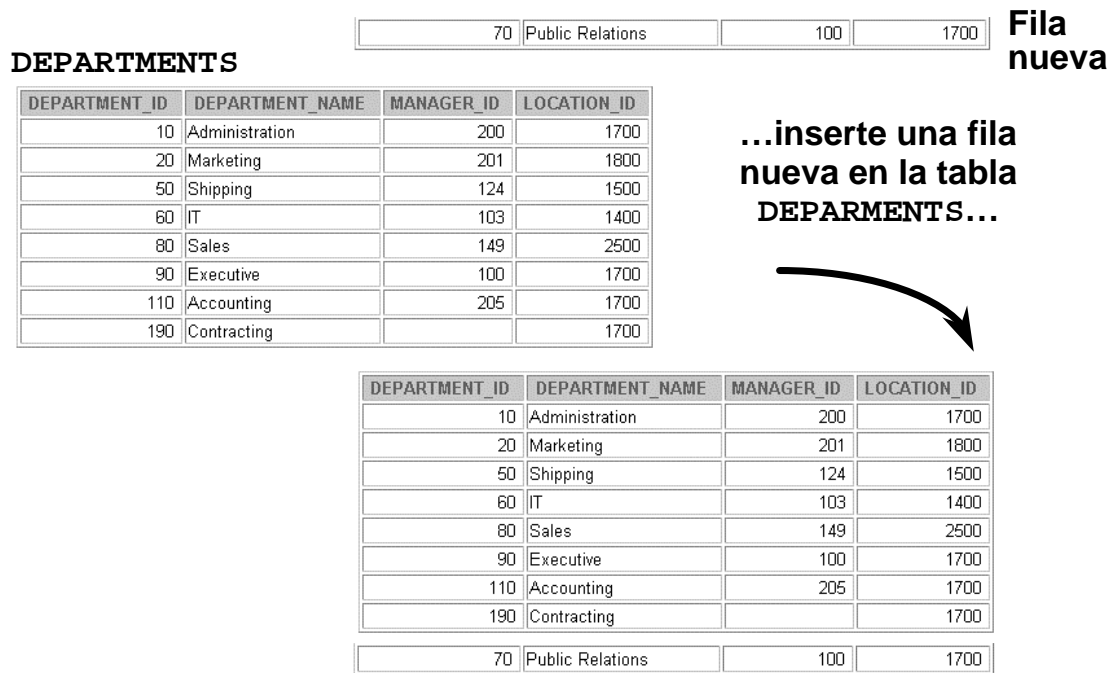
ORACLE

## Lenguaje de Manipulación de Datos

El lenguaje de manipulación de datos (DML) es una pieza central de SQL. Cuando desee agregar, actualizar o suprimir datos de la base de datos, ejecute una sentencia DML. Una recopilación de sentencias DML que forma una unidad de trabajo lógica se llama transacción.

Considere la base de datos de un banco. Cuando un cliente del banco transfiere dinero de una cuenta de ahorros a una de cheques, la transacción puede constar de tres operaciones distintas: disminución de la cuenta de ahorros, aumento de la cuenta de cheques y registro de la transacción en el diario de transacciones. Oracle Server debe garantizar que las tres sentencias SQL se realizan para mantener las cuentas en el equilibrio correcto. Si algo evita la ejecución de una de las sentencias de la transacción, las otras sentencias se deben deshacer.

## Adición de una Fila Nueva a una Tabla



ORACLE

### Adición de una Fila Nueva a una Tabla

El gráfico de la transparencia ilustra la adición de un departamento nuevo a la tabla DEPARTMENTS.

## Sintaxis de la Sentencia INSERT

- Agregue nuevas filas a una tabla mediante la sentencia INSERT.

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Con esta sintaxis sólo se inserta una fila cada vez.

ORACLE

8-5

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Adición de una Fila Nueva a una Tabla (continuación)

Puede agregar filas nuevas a una tabla emitiendo la sentencia INSERT.

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>column</i>	es el nombre de la columna de la tabla que se ha de rellenar.
<i>value</i>	es el valor correspondiente para la columna.

**Nota:** Esta sentencia con la cláusula VALUES sólo agrega una fila cada vez a una tabla.

## Inserción de Filas Nuevas

- Inserte una fila nueva que contenga valores para cada columna.
- Enumere los valores en el orden por defecto de las columnas de la tabla.
- Opcionalmente, liste las columnas en la cláusula INSERT.

```
INSERT INTO departments(department_id, department_name,  
                        manager_id, location_id)  
VALUES      (70, 'Public Relations', 100, 1700);  
1 row created.
```

- Escriba los valores de caracteres y de fecha entre comillas simples.

ORACLE

8-6

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Adición de una Fila Nueva a una Tabla (continuación)

Como puede insertar una fila nueva que contenga valores para cada columna, no se requiere la lista de columnas en la cláusula INSERT. Sin embargo, si no utiliza la lista de columnas, los valores se deben listar de acuerdo con el orden por defecto de las columnas de la tabla y se debe proporcionar un valor para cada columna.

```
DESCRIBE departments
```

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Para lograr una mayor claridad, utilice la lista de columnas en la cláusula INSERT.

Escriba los valores de caracteres y de fecha entre comillas simples; no se recomienda escribir los valores numéricos entre comillas simples.

Los valores numéricos no se deben escribir entre comillas simples, pues, si se hace, puede tener lugar una conversión implícita para los valores numéricos asignados a columnas del tipo de dato NUMBER.

## Inserción de Filas con Valores Nulos

- **Método implícito:** Omita la columna de la lista de columnas.

```
INSERT INTO departments (department_id,  
                           department_name )  
VALUES (30, 'Purchasing');  
1 row created.
```

- **Método explícito:** Especifique la palabra clave NULL en la cláusula VALUES.

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);  
1 row created.
```

ORACLE

### Métodos para Insertar Valores Nulos

Método	Description
Implícito	Omita la columna de la lista de columnas.
Explícito	Especifique la palabra clave NULL en la lista VALUES, especifique la cadena vacía ( ' ' ) en la lista VALUES para cadenas de caracteres y fechas.

Asegúrese de que puede utilizar valores nulos en la columna de destino verificando el estado de Null? con el comando DESCRIBE de iSQL\*Plus.

Oracle Server fuerza automáticamente todos los tipos de dato, rangos de datos y restricciones de integridad de datos. Toda columna no listada explícitamente obtiene un valor nulo en la fila nueva.

Errores habituales que se pueden producir durante la entrada del usuario:

- Falta un valor obligatorio para una columna NOT NULL.
- Un valor duplicado viola la restricción de unicidad.
- Se viola una restricción de clave ajena.
- Se viola una restricción CHECK.
- El tipo de dato no coincide.
- El valor es demasiado ancho para la columna.

## Inserción de Valores Especiales

La función **SYSDATE** registra la fecha y la hora actuales.

```
INSERT INTO employees (employee_id,
                        first_name, last_name,
                        email, phone_number,
                        hire_date, job_id, salary,
                        commission_pct, manager_id,
                        department_id)
VALUES                (113,
                        'Louis', 'Popp',
                        'LPOPP', '515.124.4567',
                        SYSDATE, 'AC_ACCOUNT', 6900,
                        NULL, 205, 100);

1 row created.
```

ORACLE

8-8

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Inserción de Valores Especiales Utilizando Funciones SQL

Puede utilizar funciones para introducir valores especiales en la tabla.

En el ejemplo de la transparencia se registra información del empleado Popp en la tabla **EMPLOYEES**, se proporciona la fecha y la hora actuales en la columna **HIRE\_DATE** y se utiliza la función **SYSDATE** para la fecha y la hora actuales.

También puede utilizar la función **USER** al insertar filas en una tabla. Esta función registra el nombre de usuario actual.

#### Confirmación de Adiciones a la Tabla

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
113	Popp	AC_ACCOUNT	27-SEP-01	



## Inserción de Valores de Fecha Específicos

- Agregue un empleado nuevo.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);
1 row created.
```

- Verifique la adición.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

ORACLE

### Inserción de Valores de Fecha y Hora Específicos

El formato DD-MON-YY se utiliza normalmente para insertar un valor de fecha. Con este formato, recuerde que el siglo por defecto es el actual. Como la fecha también contiene información de hora, la hora por defecto es la medianoche (00:00:00).

Si se debe introducir una fecha en un formato distinto del formato por defecto, por ejemplo, con otro siglo, o una hora específica, debe utilizar la función TO\_DATE.

En el ejemplo de la transparencia se registra información para el empleado Raphealy en la tabla EMPLOYEES y se define la columna HIRE\_DATE en 3 de febrero de 1999. Si utiliza la siguiente sentencia en lugar de la mostrada en la transparencia, el año de hire\_date se interpreta como 2099.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             '03-FEB-99',
             'AC_ACCOUNT', 11000, NULL, 100, 30);
```

Si se utiliza el formato RR, el sistema proporciona el siglo correcto de forma automática, aunque no sea el actual.

## Creación de un Archivo de Comandos

- Utilice la sustitución & en una sentencia SQL para solicitar valores.
- & es una variable pendiente de asignación para el valor de la variable.

```
INSERT INTO departments
      (department_id, department_name, location_id)
VALUES (&department_id, '&department_name', &location);
```

Define Substitution Variables

"department\_id"

"department\_name"

"location"

Submit for Execution

Cancel

1 row created.

ORACLE

8-10

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Creación de un Archivo de Comandos para Manipular Datos

Puede guardar comandos con variables de sustitución en un archivo y ejecutar los comandos del archivo. En el ejemplo anterior se registra información para un departamento en la tabla DEPARTMENTS.

Ejecute el archivo de comandos y se le pedirá que introduzca las variables de sustitución &. Los valores que introduzca se sustituirán más tarde en la sentencia. Esto le permite ejecutar el mismo archivo de comandos una y otra vez, suministrando un juego de valores distinto cada vez que lo ejecute.

## Copia de Filas desde otra Tabla

- **Escriba la sentencia INSERT con una subconsulta.**

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM employees
WHERE job_id LIKE '%REP%';
```

4 rows created.

- **No utilice la cláusula VALUES.**
- **Haga coincidir el número de columnas de la cláusula INSERT con el de la subconsulta.**

ORACLE

8-11

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Copia de Filas desde otra Tabla

Puede utilizar la sentencia INSERT para agregar filas a una tabla en la que los valores se derivan desde tablas existentes. En lugar de la cláusula VALUES, utilice una subconsulta.

#### Sintaxis

```
INSERT INTO table [ column (, column) ] subquery;
```

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>column</i>	es el nombre de la columna de la tabla que se ha de rellenar.
<i>subquery</i>	es la subconsulta que devuelve filas a la tabla.

El número de columnas y sus tipos de dato en la lista de columnas de la cláusula INSERT debe coincidir con el número de valores y sus tipos de dato en la subconsulta. Para crear una copia de las filas de una tabla, utilice SELECT \* en la subconsulta.

```
INSERT INTO copy_emp
SELECT *
FROM employees;
```

Para obtener más información, consulte la sección de subconsultas de *Oracle9i SQL Reference*, “SELECT”.

## Cambio de los Datos de una Tabla

### EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Actualice las filas de la tabla **EMPLOYEES**.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

ORACLE

### Cambio de los Datos de una Tabla

El gráfico de la transparencia ilustra el cambio del número de departamento para los empleados del departamento 60 al departamento 30.

## La Sintaxis de la Sentencia UPDATE

- Modifique las filas existentes con la sentencia UPDATE.

```
UPDATE      table
SET          column = value [, column = value, ...]
[WHERE      condition];
```

- Actualice más de una fila cada vez si es necesario.

ORACLE

8-13

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Actualización de Filas

Puede modificar las filas existentes utilizando la sentencia UPDATE.

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>column</i>	es el nombre de la columna de la tabla que se ha de rellenar.
<i>value</i>	es la subconsulta o el valor correspondiente para la columna.
<i>condition</i>	identifica las filas que se han de actualizar y está compuesta de expresiones de nombres de columna, constantes, subconsultas y operadores de comparación.

Confirme la operación de actualización mediante la consulta a la tabla para mostrar las filas actualizadas.

Para obtener más información, consulte *Oracle9i SQL Reference*, “UPDATE”.

**Nota:** En general, utilice la clave primaria para identificar una sola fila. Si usa otras columnas se pueden actualizar varias filas de forma inesperada. Por ejemplo, identificar una sola fila de la tabla EMPLOYEES por el nombre es peligroso, ya que es posible que varios empleados tengan el mismo nombre.

## Actualización de las Filas de una Tabla

- Si incluye la cláusula **WHERE**, las filas específicas se modifican.

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- Se modifican todas las filas de la tabla si omite la cláusula **WHERE**.

```
UPDATE    copy_emp
SET       department_id = 110;
22 rows updated.
```

ORACLE

8-14

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Actualización de Filas (continuación)

La sentencia **UPDATE** modifica filas específicas si se incluye la cláusula **WHERE**. En el ejemplo de la transparencia se transfiere al empleado 113 (Popp) al departamento 70.

Si omite la cláusula **WHERE**, se modifican todas las filas de la tabla.

```
SELECT last_name, department_id
FROM    copy_emp;
```

LAST_NAME	DEPARTMENT_ID
King	110
Kochhar	110
De Haan	110
Hunold	110
Ernst	110
Lorentz	110

■ ■ ■

22 rows selected.

**Nota:** La tabla **COPY\_EMP** tiene los mismos datos que la tabla **EMPLOYEES**.

## Actualización de Dos Columnas con una Subconsulta

Actualice el cargo y el salario del empleado 114 para que coincida con el del empleado 205 .

```
UPDATE employees
SET   job_id = (SELECT job_id
                FROM   employees
                WHERE  employee_id = 205),
      salary = (SELECT salary
                FROM   employees
                WHERE  employee_id = 205)
WHERE employee_id = 114;
1 row updated.
```

ORACLE

8-15

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Actualización de Dos Columnas con una Subconsulta

Puede actualizar varias columnas en la cláusula SET de una sentencia UPDATE escribiendo varias subconsultas.

#### Sintaxis

```
UPDATE table
SET   column =
      (SELECT column
       FROM table
       WHERE condition)
[ ,
  column =
      (SELECT column
       FROM table
       WHERE condition) ]
[WHERE condition] ;
```

**Nota:** Si no se actualiza ninguna fila, se devuelve el mensaje “0 rows updated”.

## Actualización de Filas Basándose en otra Tabla

Utilice subconsultas en sentencias UPDATE para actualizar las filas de una tabla basándose en valores de otra tabla.

```
UPDATE copy_emp
SET    department_id = (SELECT department_id
                        FROM employees
                        WHERE employee_id = 100)
WHERE  job_id        = (SELECT job_id
                        FROM employees
                        WHERE employee_id = 200);

1 row updated.
```

ORACLE

### Actualización de Filas Basándose en otra Tabla

Puede utilizar subconsultas en sentencias UPDATE para actualizar las filas de una tabla. En el ejemplo de la transparencia se actualiza la tabla COPY\_EMP basándose en los valores de la tabla EMPLOYEES, se cambia el número de departamento de todos los empleados con el identificador de cargo del empleado 200 por el número de departamento actual del empleado 100.



## Actualización de Filas: Error de Restricción de Integridad

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
UPDATE employees
*
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

**El número de departamento 55 no existe**

ORACLE

8-17

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Error de Restricción de Integridad

Si intenta actualizar un registro con un valor que esté unido a una restricción de integridad, se devuelve un error.

En el ejemplo de la transparencia, el número de departamento 55 no existe en la tabla principal, DEPARTMENTS, por lo que se recibe la violación de *clave principal* ORA-02291.

**Nota:** Las restricciones de integridad aseguran que los datos se adhieren a un juego de reglas predefinido. Una lección posterior cubre las restricciones de integridad en mayor profundidad.

# Eliminación de una Fila de una Tabla

## DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

## Suprima una fila de la tabla DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

ORACLE

## Eliminación de una Fila de una Tabla

En el gráfico de la transparencia se elimina el departamento Finance de la tabla DEPARTMENTS (asumiendo que no se han definido restricciones en la tabla DEPARTMENTS).

# La Sentencia DELETE

Puede eliminar las filas existentes de una tabla utilizando la sentencia DELETE.

```
DELETE [FROM] table
[WHERE condition];
```

ORACLE

8-19

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Supresión de Filas

Puede eliminar las filas existentes utilizando la sentencia DELETE.

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>condition</i>	identifica las filas que se van a suprimir y está compuesta de nombres de columna, expresiones, constantes, subconsultas y operadores de comparación.

**Nota:** Si no se suprime ninguna fila, se devuelve el mensaje “0 rows deleted”.

Para obtener más información, consulte *Oracle9i SQL Reference*, “DELETE”.

## Supresión de Filas de una Tabla

- Se suprimen filas específicas si incluye la cláusula **WHERE**.

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 row deleted.
```

- Se suprimen todas las filas de la tabla si omite la cláusula **WHERE**.

```
DELETE FROM copy_emp;
22 rows deleted.
```

ORACLE

8-20

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Supresión de Filas (continuación)

Puede suprimir filas específicas incluyendo la cláusula **WHERE** en la sentencia **DELETE**. En el ejemplo de la transparencia se suprime el departamento Finance de la tabla **DEPARTMENTS**. Puede confirmar la operación de supresión si visualiza las filas suprimidas utilizando la sentencia **SELECT**.

```
SELECT *
FROM departments
WHERE department_name = 'Finance';

no rows selected.
```

Si omite la cláusula **WHERE**, todas las filas de la tabla se suprimen. En el segundo ejemplo de la transparencia se suprimen todas las filas de la tabla **COPY\_EMP**, porque no se ha especificado ninguna cláusula **WHERE**.

#### Ejemplo

Elimine las filas identificadas en la cláusula **WHERE**.

```
DELETE FROM employees
WHERE employee_id = 114;

1 row deleted.

DELETE FROM departments
WHERE department_id IN (30, 40);

2 rows deleted.
```

## Supresión de Filas Basándose en otra Tabla

Utilice subconsultas en sentencias **DELETE** para eliminar las filas de una tabla basándose en los valores de otra tabla.

```
DELETE FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name LIKE '%Public%');

1 row deleted.
```

ORACLE

8-21

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Supresión de Filas Basándose en otra Tabla

Puede utilizar subconsultas para suprimir filas de una tabla basándose en valores de otra tabla. En el ejemplo de la transparencia se suprimen todos los empleados que están en un departamento cuyo nombre contiene la cadena “Public”. La subconsulta busca en la tabla **DEPARTMENTS** el número de departamento basándose en el nombre de departamento que contiene la cadena “Public”. A continuación, proporciona el número de departamento a la consulta principal, la cual suprime filas de datos de la tabla **EMPLOYEES** basándose en este número de departamento.

## Supresión de Filas: Error de Restricción de Integridad

```
DELETE FROM departments
WHERE      department_id = 60;
```

```
DELETE FROM departments
          *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

**No puede suprimir una fila que contenga una clave primaria utilizada como clave ajena en otra tabla.**

ORACLE

8-22

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Error de Restricción de Integridad

Si intenta suprimir un registro con un valor que esté unido a una restricción de integridad, se devuelve un error.

En el ejemplo de la transparencia se intenta suprimir el número de departamento 60 de la tabla DEPARTMENTS, pero el resultado es un error, porque el número de departamento se utiliza como clave ajena en la tabla EMPLOYEES. Si el registro principal que intenta suprimir tiene registros secundarios, recibirá la violación *child record found* ORA-02292.

La siguiente sentencia funciona porque no hay ningún empleado en el departamento 70:

```
DELETE FROM departments
WHERE      department_id = 70;
```

```
1 row deleted.
```

## Uso de una Subconsulta en una Sentencia INSERT

```
INSERT INTO
    (SELECT employee_id, last_name,
           email, hire_date, job_id, salary,
           department_id
     FROM employees
    WHERE department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR',
       TO_DATE('07-JUN-99', 'DD-MON-RR'),
       'ST_CLERK', 5000, 50);

1 row created.
```

ORACLE

### Uso de una Subconsulta en una Sentencia INSERT

Puede utilizar una subconsulta en lugar del nombre de tabla en la cláusula INTO de la sentencia INSERT.

La lista de selección de esta subconsulta debe tener el mismo número de columnas que la lista de columnas de la cláusula VALUES. Se deben seguir todas las reglas de las columnas de la tabla base para que la sentencia INSERT funcione correctamente. Por ejemplo, no puede poner un identificador de empleado duplicado, ni dejar fuera un valor para una columna obligatoria no nula.

## Uso de una Subconsulta en una Sentencia INSERT

```
SELECT employee_id, last_name, email, hire_date,  
       job_id, salary, department_id  
FROM   employees  
WHERE  department_id = 50;
```

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
124	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50
141	Rajs	TRAJS	17-OCT-95	ST_CLERK	3500	50
142	Davies	CDAVIES	29-JAN-97	ST_CLERK	3100	50
143	Matos	RMATOS	15-MAR-98	ST_CLERK	2600	50
144	Vargas	PVARGAS	09-JUL-98	ST_CLERK	2500	50
99999	Taylor	DTAYLOR	07-JUN-99	ST_CLERK	5000	50

6 rows selected.

ORACLE

### Uso de una Subconsulta en una Sentencia INSERT

En el ejemplo se muestran los resultados de la subconsulta utilizada para identificar la tabla para la sentencia INSERT.



## Uso de la Palabra Clave WITH CHECK OPTION en Sentencias DML

- Se utiliza una subconsulta para identificar la tabla y las columnas de la sentencia DML.
- La palabra clave WITH CHECK OPTION le prohíbe que cambie filas que no están en la subconsulta.

```
INSERT INTO (SELECT employee_id, last_name, email,
                hire_date, job_id, salary
            FROM employees
            WHERE department_id = 50 WITH CHECK OPTION)
VALUES (99998, 'Smith', 'JSMITH',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000);
INSERT INTO
*
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

ORACLE

### La Palabra Clave WITH CHECK OPTION

Especifique WITH CHECK OPTION para indicar que, si se utiliza la subconsulta en lugar de una tabla en una sentencia INSERT, UPDATE o DELETE, en la tabla no se permitirán cambios que produzcan filas que no estén incluidas en la subconsulta.

En el ejemplo mostrado, se utiliza la palabra clave WITH CHECK OPTION. La subconsulta identifica las filas que están en el departamento 50, pero el identificador de departamento no se encuentra en la lista SELECT y no se le proporciona un valor en la lista VALUES. La inserción de esta fila daría como resultado un identificador de departamento de valor nulo, que no se encuentra en la subconsulta.

## **Visión General de la Función Valor por Defecto Explícito**

- Con la función valor por defecto explícito, puede utilizar la palabra clave **DEFAULT** como valor de columna donde desee el valor por defecto de columna.
- Esta función se agrega para asegurar el cumplimiento con el estándar **SQL: 1999**.
- Esto permite al usuario controlar dónde y cuándo se debe aplicar el valor por defecto a los datos.
- Se pueden utilizar valores por defecto explícitos en sentencias **INSERT** y **UPDATE**.

ORACLE

8-26

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### **Valores por Defecto Explícitos**

La palabra clave **DEFAULT** se puede utilizar en sentencias **INSERT** y **UPDATE** para identificar un valor de columna por defecto. Si no existe ningún valor por defecto, se utiliza un valor nulo.

## Uso de Valores por Defecto Explícitos

- **DEFAULT con INSERT:**

```
INSERT INTO departments  
  (department_id, department_name, manager_id)  
VALUES (300, 'Engineering', DEFAULT);
```

- **DEFAULT con UPDATE:**

```
UPDATE departments  
SET manager_id = DEFAULT WHERE department_id = 10;
```

ORACLE

### Uso de Valores por Defecto Explícitos

Especifique **DEFAULT** para definir la columna en el valor especificado previamente como valor por defecto para la columna. Si no se ha especificado ningún valor por defecto para la columna correspondiente, Oracle define la columna en un valor nulo.

En el primer ejemplo mostrado, la sentencia **INSERT** utiliza un valor por defecto para la columna **MANAGER\_ID**. Si no hay ningún valor por defecto definido para la columna, se inserta en su lugar un valor nulo.

En el segundo ejemplo se utiliza la sentencia **UPDATE** para definir la columna **MANAGER\_ID** en un valor por defecto para el departamento 10. Si no hay ningún valor por defecto definido para la columna, cambia el valor a nulo.

**Nota:** Al crear una tabla, puede especificar un valor por defecto para una columna. Esto se describe en la lección “Creating and Managing Tables”.

## La Sentencia MERGE

- **Proporciona la capacidad de actualizar o insertar datos condicionalmente en una tabla de base de datos**
- **Realiza una actualización (UPDATE) si existe la fila y una inserción (INSERT) si es una fila nueva:**
  - **Evita actualizaciones separadas.**
  - **Aumenta el rendimiento y la facilidad de uso.**
  - **Es útil en aplicaciones de almacenes de datos.**

ORACLE

8-28

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Sentencias MERGE

SQL se ha extendido para incluir la sentencia MERGE. Con esta sentencia, puede actualizar o insertar una fila condicionalmente en una tabla, evitando así tener múltiples sentencias UPDATE. La decisión sobre si actualizar o insertar en la tabla de destino se basa en una condición en la cláusula ON.

Como el comando MERGE combina los comandos INSERT y UPDATE, necesita los dos privilegios INSERT y UPDATE en la tabla de destino y el privilegio SELECT en la tabla de origen.

La sentencia MERGE es determinista. No puede actualizar la misma fila de la tabla de destino varias veces en la misma sentencia MERGE.

Un enfoque alternativo es utilizar bucles PL/SQL y varias sentencias DML. Sin embargo, la sentencia MERGE es fácil de utilizar y se expresa de forma más simple como una sola sentencia SQL.

La sentencia MERGE es útil en diversas aplicaciones de almacenes de datos. Por ejemplo, en una aplicación de almacenes de datos, tal vez sea necesario trabajar con datos procedentes de varios orígenes, algunos de los cuales pueden estar duplicados. Con la sentencia MERGE, puede agregar o modificar filas condicionalmente.

# Sintaxis de la Sentencia MERGE

Puede insertar o actualizar filas condicionalmente en una tabla utilizando la sentencia MERGE.

```
MERGE INTO table_name table_alias
  USING (table/view/sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
      col1 = col_val1,
      col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```

ORACLE

8-29

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Fusión de Filas

Puede actualizar filas existentes e insertar filas nuevas condicionalmente utilizando la sentencia MERGE.

En la sintaxis:

Cláusula INTO	especifica la tabla de destino que está actualizando o en la que está insertando.
Cláusula USING	identifica el origen de los datos que se van a actualizar o insertar; puede ser una tabla, una vista o una subconsulta.
Cláusula ON	la condición sobre la cual la operación MERGE actualiza o inserta.
WHEN MATCHED	indica al servidor cómo responder a los resultados de la condición de unión.
WHEN NOT MATCHED	Para obtener más información, consulte <i>Oracle9i SQL Reference</i> , “MERGE”.

## Fusión de Filas

**Inserte o actualice filas en la tabla COPY\_EMP para que coincida con la tabla EMPLOYEES.**

```
MERGE INTO copy_emp c
  USING employees e
  ON (c.employee_id = e.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      ...
      c.department_id  = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
                  e.email, e.phone_number, e.hire_date, e.job_id,
                  e.salary, e.commission_pct, e.manager_id,
                  e.department_id);
```

ORACLE

8-30

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Ejemplo de Fusión de Filas

```
MERGE INTO copy_emp c
  USING employees e
  ON (c.employee_id = e.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      c.email           = e.email,
      c.phone_number    = e.phone_number,
      c.hire_date       = e.hire_date,
      c.job_id          = e.job_id,
      c.salary          = e.salary,
      c.commission_pct  = e.commission_pct,
      c.manager_id      = e.manager_id,
      c.department_id  = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
                  e.email, e.phone_number, e.hire_date, e.job_id,
                  e.salary, e.commission_pct, e.manager_id,
                  e.department_id);
```

En el ejemplo mostrado se compara EMPLOYEE\_ID de la tabla COPY\_EMP con EMPLOYEE\_ID de la tabla EMPLOYEES. Si se encuentra una coincidencia, la fila de la tabla COPY\_EMP se actualiza para que coincida con la fila de la tabla EMPLOYEES. Si no se encuentra la fila, se inserta en la tabla COPY\_EMP.

## Fusión de Filas

```
SELECT *  
FROM COPY_EMP;
```

```
no rows selected
```

```
MERGE INTO copy_emp c  
  USING employees e  
  ON (c.employee_id = e.employee_id)  
 WHEN MATCHED THEN  
   UPDATE SET  
   ...  
 WHEN NOT MATCHED THEN  
   INSERT VALUES...;
```

```
SELECT *  
FROM COPY_EMP;
```

```
20 rows selected.
```

ORACLE

### Ejemplo de Fusión de Filas

Se evalúa la condición `c.employee_id = e.employee_id`. Como la tabla `COPY_EMP` está vacía, la condición devuelve falso: no hay coincidencias. La lógica cae en la cláusula `WHEN NOT MATCHED` y el comando `MERGE` inserta las filas de la tabla `EMPLOYEES` en la tabla `COPY_EMP`.

Si existieran filas en la tabla `COPY_EMP` y coincidieran los identificadores de empleado de ambas tablas (`COPY_EMP` y `EMPLOYEES`), las filas existentes de la tabla `COPY_EMP` se actualizarían para coincidir con la tabla `EMPLOYEES`.

# Transacciones de Base de Datos

**Una transacción de base de datos consta de uno de los siguientes elementos:**

- **Sentencias DML, que constituyen un cambio consistente en los datos**
- **Una sentencia DDL**
- **Una sentencia DCL**

ORACLE

8-32

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Transacciones de Base de Datos

Oracle Server asegura la consistencia de datos basándose en transacciones. Las transacciones le ofrecen más flexibilidad y control al cambiar datos y aseguran una consistencia de datos en caso de fallo del proceso de usuario o fallo del sistema.

Las transacciones constan de sentencias DML que constituyen un cambio consistente en los datos. Por ejemplo, una transferencia de fondos entre dos cuentas debe incluir el débito a una cuenta y el cargo a otra por el mismo importe. Las dos acciones se deben realizar correcta o incorrectamente al mismo tiempo, de forma que el crédito no se debe validar sin el débito.

## Tipos de Transacción

Tipo	Descripción
Lenguaje de manipulación de datos (DML)	Consta de un número variable de sentencias DML que Oracle Server trata como entidad única o como unidad de trabajo lógica.
Lenguaje de definición de datos (DDL)	Consta de una sola sentencia DDL.
Lenguaje de control de datos (DCL)	Consta de una sola sentencia DCL.



# Transacciones de Base de Datos

- **Comienzan cuando se ejecuta la primera sentencia SQL DML.**
- **Terminan con uno de los siguientes eventos:**
  - **Se emite una sentencia COMMIT o ROLLBACK.**
  - **Se ejecuta una sentencia DDL o DCL (validación automática).**
  - **El usuario sale de iSQL\*Plus.**
  - **El sistema falla.**

ORACLE

8-33

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## ¿Cuándo Empieza y Termina una Transacción?

Una transacción comienza cuando se encuentra la primera sentencia DML y termina cuando se produce una de las siguientes opciones:

- Se emite una sentencia COMMIT o ROLLBACK .
- Se emite una sentencia DDL, como por ejemplo, CREATE.
- Se emite una sentencia DCL.
- El usuario sale de iSQL\*Plus.
- Se produce un fallo de una máquina o en el sistema.

Cuando termina una transacción, la siguiente sentencia SQL ejecutable inicia automáticamente la siguiente transacción.

Se valida automáticamente una sentencia DDL o una sentencia DCL y, por lo tanto, finaliza implícitamente una transacción.

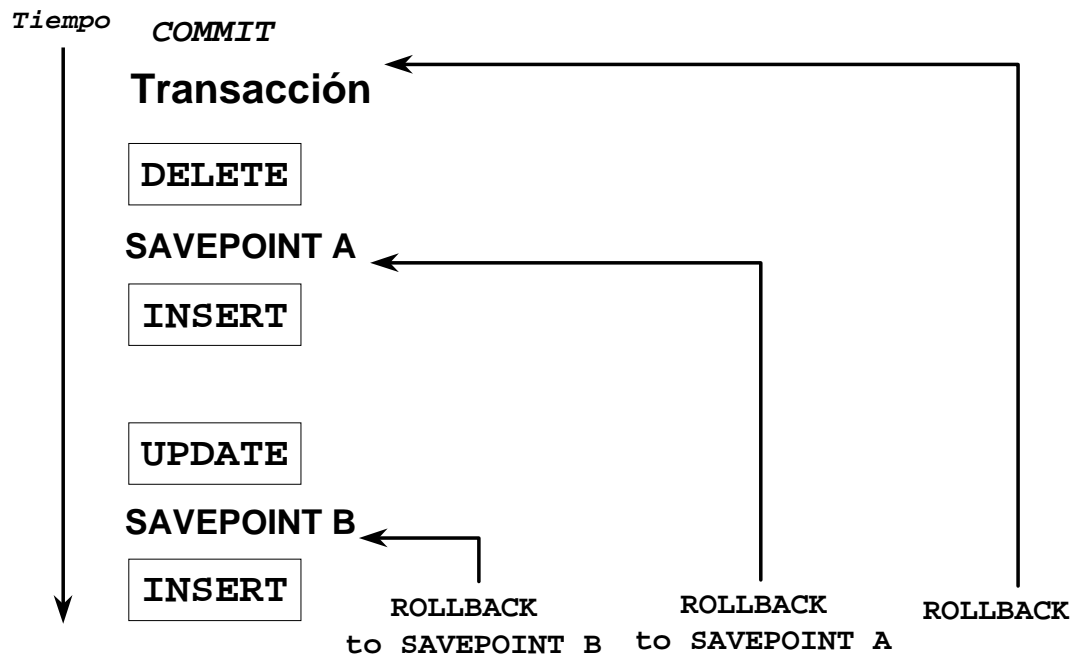
## **Ventajas de las Sentencias COMMIT y ROLLBACK**

**Con sentencias COMMIT y ROLLBACK, puede:**

- **Asegurar la consistencia de datos**
- **Realizar una presentación preliminar de los cambios de datos antes de hacer que estos sean permanentes**
- **Agrupar las operaciones relacionadas lógicamente**

**ORACLE**

## Control de Transacciones



ORACLE

8-35

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Sentencias Explícitas de Control de Transacciones

Puede controlar la lógica de las transacciones utilizando las sentencias COMMIT, SAVEPOINT y ROLLBACK.

Sentencia	Descripción
COMMIT	Termina la transacción actual haciendo permanentes todos los cambios de datos pendientes.
SAVEPOINT <i>name</i>	Marca un punto de grabación dentro de la transacción actual.
ROLLBACK	ROLLBACK termina la transacción actual desechando todos los cambios de datos pendientes.
ROLLBACK TO <i>SAVEPOINT name</i>	ROLLBACK TO SAVEPOINT realiza rollback de la transacción actual al punto de grabación especificado, desechando así los cambios o los puntos de grabación creados después del punto de grabación al que está realizando rollback. Si omite la cláusula TO SAVEPOINT, la sentencia ROLLBACK realiza rollback de toda la transacción. Como los puntos de grabación son lógicos, no hay ninguna forma de enumerar los puntos de grabación que ha creado.

**Nota:** SAVEPOINT no es SQL del estándar ANSI.

## Realización de Rollback de los Cambios a un Marcador

- Cree un marcador en una transacción actual utilizando la sentencia **SAVEPOINT**.
- Realice rollback a dicho marcador utilizando la sentencia **ROLLBACK TO SAVEPOINT**.

```
UPDATE...  
SAVEPOINT update_done;  
Savepoint created.  
INSERT...  
ROLLBACK TO update_done;  
Rollback complete.
```

ORACLE

### Realización de Rollback de los Cambios a un Punto de Grabación

Puede crear un marcador en la transacción actual utilizando la sentencia **SAVEPOINT**, que divide la transacción en secciones más pequeñas. A continuación, puede desechar los cambios pendientes hasta dicho marcador utilizando la sentencia **ROLLBACK TO SAVEPOINT**.

Si crea un segundo punto de grabación con el mismo nombre que uno anterior, el primero se suprime.

### Instructor Note

Savepoints are especially useful in PL/SQL and 3GL programs in which recent changes can be undone conditionally based on run-time conditions.

## Procesamiento Implícito de Transacciones

- Se produce una validación automática en las siguientes circunstancias:
  - Se emite una sentencia DDL.
  - Se emite una sentencia DCL.
  - Se sale normalmente de *iSQL\*Plus*, sin emitir explícitamente sentencias COMMIT o ROLLBACK.
- Se realiza un rollback automático tras una terminación anormal de *iSQL\*Plus* o un fallo del sistema.

ORACLE

8-37

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Procesamiento Implícito de Transacciones

Estado	Circunstancias
Validación automática	Se emite una sentencia DDL o DCL. Salida normal de <i>iSQL*Plus</i> , sin emitir explícitamente comandos COMMIT o ROLLBACK.
Rollback automático	Terminación anormal de <i>iSQL*Plus</i> o fallo del sistema.

**Nota:** En *iSQL\*Plus* está disponible un tercer comando. El comando AUTOCOMMIT se puede activar o desactivar. Si está *activado*, cada sentencia DML individual se valida en cuanto se ejecuta. No puede realizar rollback de los cambios. Si está *desactivado*, la sentencia COMMIT todavía se puede emitir explícitamente. Además, la sentencia COMMIT se emite cuando se emite una sentencia DDL o al salir de *iSQL\*Plus*.

### Fallos del Sistema

Cuando se interrumpe una transacción por un fallo del sistema, se realiza rollback automáticamente a toda la transacción. Esto evita que el error produzca cambios no deseados en los datos y devuelve las tablas al estado que tenían en el momento de la última validación. De esta forma, Oracle Server protege la integridad de las tablas.

Desde *iSQL\*Plus*, se consigue salir normalmente de la sesión haciendo clic en el botón Exit. Con *SQL\*Plus*, se consigue salir normalmente escribiendo el comando EXIT en el prompt. Cerrar la ventana se interpreta como una salida anormal.

## Estado de los Datos Antes de COMMIT o ROLLBACK

- Se puede recuperar el estado anterior de los datos.
- El usuario actual puede revisar los resultados de las operaciones DML utilizando la sentencia `SELECT`.
- Otros usuarios *no pueden* ver los resultados de las sentencias DML del usuario actual.
- Las filas afectadas se *bloquean*; otros usuarios no pueden cambiar los datos de las filas afectadas.

ORACLE

8-38

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Validación de Cambios

Todo cambio de datos realizado durante la transacción es temporal hasta que se valida la transacción.

Estado de los datos antes de que se emitan sentencias COMMIT o ROLLBACK:

- Las operaciones de manipulación de datos afectan principalmente al buffer de la base de datos; por lo tanto, el estado anterior de los datos se puede recuperar.
- El usuario actual puede revisar los resultados de las operaciones de manipulación de datos consultando las tablas.
- Otros usuarios no pueden ver los resultados de las operaciones de manipulación de datos realizados por el usuario actual. Oracle Server establece la consistencia de lectura para asegurar que cada usuario ve los datos como eran en la última validación.
- Las filas afectadas se bloquean; otros usuarios no pueden cambiar los datos de las filas afectadas.

## Estado de los Datos Después de COMMIT

- Los cambios de datos se hacen permanentes en la base de datos.
- El estado anterior de los datos se pierde permanentemente.
- Todos los usuarios pueden ver los resultados.
- Los bloqueos de las filas afectadas se liberan; estas filas están disponibles para que otros usuarios las manipulen.
- Todos los puntos de grabación se borran.

ORACLE

8-39

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Validación de Cambios (continuación)

Haga que todos los cambios pendientes sean permanentes utilizando la sentencia COMMIT. Tras una sentencia COMMIT:

- Los cambios de datos se escriben en la base de datos.
- El estado anterior de los datos se pierde permanentemente.
- Todos los usuarios pueden ver los resultados de la transacción.
- Los bloqueos de las filas afectadas se liberan; las filas están ahora disponibles para que otros usuarios realicen nuevos cambios de datos.
- Todos los puntos de grabación se borran.

## Validación de Datos

- **Realice los cambios.**

```
DELETE FROM employees
WHERE employee_id = 99999;
1 row deleted.

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 row inserted.
```

- **Valide los cambios.**

```
COMMIT;
Commit complete.
```

ORACLE

### Validación de Cambios (continuación)

En el ejemplo de la transparencia se suprime una fila de la tabla EMPLOYEES y se inserta una fila nueva en la tabla DEPARTMENTS. A continuación, se hace que el cambio sea permanente emitiendo la sentencia COMMIT.

#### Ejemplo

Elimine los departamentos 290 y 300 de la tabla DEPARTMENTS y actualice una fila en la tabla COPY\_EMP. Haga que el cambio de datos sea permanente.

```
DELETE FROM departments
WHERE department_id IN (290, 300);

2 rows deleted.
```

```
UPDATE copy_emp
SET department_id = 80
WHERE employee_id = 206;

1 row updated.
```

```
COMMIT;
```

```
Commit Complete.
```



## Estado de los Datos Después de ROLLBACK

Deseche todos los cambios pendientes utilizando la sentencia ROLLBACK:

- Los cambios de datos se deshacen.
- Se restaura el estado anterior de los datos.
- Se liberan los bloqueos de las filas afectadas.

```
DELETE FROM copy_emp;  
22 rows deleted.  
ROLLBACK;  
Rollback complete.
```

ORACLE

8-41

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Realización de Rollback de los Cambios

Deseche todos los cambios pendientes utilizando la sentencia ROLLBACK. Tras una sentencia ROLLBACK:

- Los cambios de datos se deshacen.
- El estado anterior de los datos se restaura.
- Los bloqueos de las filas afectadas se liberan.

#### Ejemplo

Al intentar eliminar un registro de la tabla TEST, puede vaciar accidentalmente la tabla. Puede corregir el error, volver a emitir la sentencia correcta y hacer que el cambio de datos sea permanente.

```
DELETE FROM test;  
25,000 rows deleted.
```

```
ROLLBACK;  
Rollback complete.
```

```
DELETE FROM test  
WHERE id = 100;  
1 row deleted.
```

```
SELECT *  
FROM test  
WHERE id = 100;  
No rows selected.
```

```
COMMIT;  
Commit complete.
```

## Rollback a Nivel de Sentencia

- Si una sola sentencia DML falla durante la ejecución, solamente se realiza rollback de dicha sentencia.
- Oracle Server implementa un punto de grabación implícito.
- Todos los demás cambios se retienen.
- El usuario debe terminar las transacciones explícitamente ejecutando una sentencia COMMIT o ROLLBACK.

ORACLE

8-42

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Rollbacks a Nivel de Sentencia

Un rollback implícito puede desechar parte de una transacción si se detecta un error de ejecución de sentencia. Si falla una sola sentencia DML durante la ejecución de una transacción, su efecto lo deshace un rollback a nivel de sentencia, pero los cambios realizados por las sentencias DML anteriores en la transacción no se desechan. El usuario puede validarlas o realizar rollback de ellas explícitamente.

Oracle emite una validación implícita antes y después de cualquier sentencia del lenguaje de definición de datos (DDL). Por ello, aunque la sentencia DDL no se ejecute correctamente, no puede realizar rollback de la sentencia anterior porque el servidor emitió una validación.

Termine las transacciones explícitamente ejecutando una sentencia COMMIT o ROLLBACK.

# Consistencia de Lectura

- **La consistencia de lectura garantiza una visualización consistente de los datos en todo momento.**
- **Los cambios realizados por un usuario no entran en conflicto con los cambios realizados por otro usuario.**
- **La consistencia de lectura asegura que en los mismos datos:**
  - **Los lectores no esperan a los escritores.**
  - **Los escritores no esperan a los lectores.**

ORACLE

8-43

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Consistencia de Lectura

Los usuarios de la base de datos acceden a ésta de dos formas:

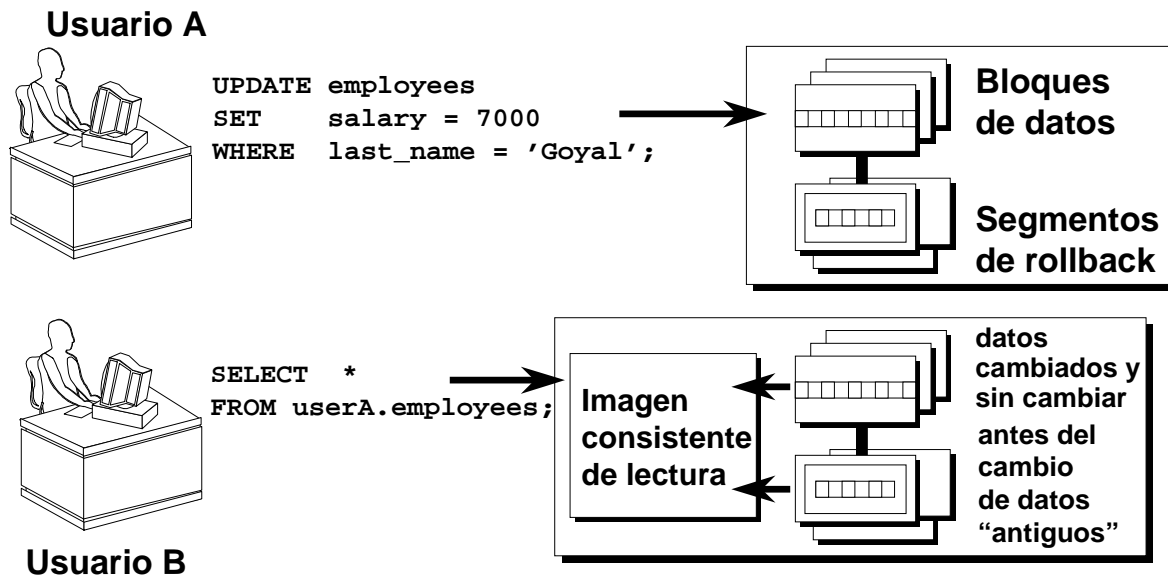
- Operaciones de lectura (sentencia SELECT)
- Operaciones de escritura (sentencias INSERT, UPDATE, DELETE)

Necesita consistencia de lectura para que:

- El escritor y el lector de la base de datos se aseguren una visualización consistente de los datos.
- Los lectores no visualicen los datos que están en proceso de cambio.
- Los escritores se aseguren de que los cambios en la base de datos se realizan de forma consistente.
- Los cambios realizados por un escritor no interrumpen a los que está haciendo otro escritor ni entren en conflicto con ellos.

El objetivo de la consistencia de lectura es asegurar que cada usuario ve los datos tal como eran en la última validación, antes de que comenzara una operación DML.

## Implementación de Consistencia de Lectura



ORACLE

8-44

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Implementación de Consistencia de Lectura

La consistencia de lectura es una implementación automática. Mantiene una copia parcial de la base de datos en segmentos deshacer.

Cuando se realiza una operación de inserción, actualización o supresión en la base de datos, Oracle Server realiza una copia de los datos antes de que se cambien y los escribe en un *segmento deshacer*.

Todos los lectores, excepto el que emitió el cambio, seguirán viendo la base de datos como existía antes de que comenzaran los cambios; ven la “instantánea” del segmento de rollback de los datos.

Antes de que se validen los cambios en la base de datos, sólo el usuario que está modificando los datos ve la base de datos con las modificaciones; los demás ven la instantánea en el segmento deshacer. Esto garantiza que los lectores de los datos lean datos consistentes que no están sufriendo ningún cambio actualmente.

Cuando se valida una sentencia DML, el cambio realizado en la base de datos se hace visible para cualquier persona que ejecute una sentencia SELECT. El espacio ocupado por los datos *antiguos* en el archivo del segmento deshacer se libera para su nuevo uso.

Si se realiza rollback de la transacción, los cambios se deshacen:

- La versión original (más antigua) de los datos del segmento deshacer se vuelve a escribir en la tabla.
- Todos los usuarios ven la base de datos como existía antes de que comenzara la transacción.

# Bloqueo

**En una base de datos Oracle, los bloqueos:**

- **Evitan la interacción destructiva entre transacciones simultáneas.**
- **No requieren la acción del usuario.**
- **Utilizan automáticamente el nivel más bajo de restricción.**
- **Se mantienen durante la transacción.**
- **Son de dos tipos: bloqueo explícito y bloqueo implícito.**

ORACLE

8-45

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## ¿Qué Son los Bloqueos?

Los bloqueos son mecanismos que evitan una interacción destructiva entre las transacciones que acceden al mismo recurso, ya sea un objeto de usuario (como tablas o filas) o un objeto de sistema no visible a los usuarios (como estructuras de datos compartidos y filas del diccionario de datos).

### **Cómo Bloquea Datos la Base de Datos Oracle**

El bloqueo de Oracle se realiza automáticamente y no requiere acción del usuario. El bloqueo implícito se produce para las sentencias SQL según sea necesario, dependiendo de la acción solicitada. Este bloqueo se produce para todas las sentencias SQL excepto `SELECT`.

Los usuarios también pueden bloquear los datos manualmente, lo que se denomina bloqueo explícito.

# Bloqueo Implícito

- **Dos modos de bloqueo:**
  - **Exclusivo:** Bloquea a otros usuarios.
  - **Compartido:** Permite que accedan otros usuarios.
- **Alto nivel de simultaneidad de datos:**
  - **DML:** Compartición de tabla, exclusivo de fila.
  - **Consultas:** No se requieren bloqueos.
  - **DDL:** Protege definiciones de objetos.
- **Bloqueos mantenidos hasta validación o rollback**

ORACLE

8-46

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Bloqueo DML

Al realizar operaciones del lenguaje de manipulación de datos (DML), Oracle Server proporciona simultaneidad de datos a través del bloqueo DML, que se produce a dos niveles:

- Un bloqueo compartido se obtiene automáticamente a nivel de tabla durante operaciones DML: Con el modo de bloqueo compartido, varias transacciones pueden adquirir bloqueos compartidos sobre el mismo recurso.
- Un bloqueo exclusivo se adquiere automáticamente para cada fila modificada por una sentencia DML. Los bloqueos exclusivos evitan que otras transacciones cambien la fila hasta que la transacción se valida o se realiza rollback. Este bloqueo asegura que ningún otro usuario puede modificar la misma fila al mismo tiempo ni sobrescribir los cambios que aún no han sido validados por otro usuario.
- Los bloqueos DDL se producen al modificar un objeto de base de datos como una tabla.

## Resumen

**En esta lección, debería haber aprendido a utilizar sentencias DML y a controlar transacciones.**

Sentencia	Descripción
INSERT	Agrega una fila nueva a la tabla.
UPDATE	Modifica las filas existentes de la tabla.
DELETE	Elimina las filas existentes de la tabla.
MERGE	Actualiza o inserta condicionalmente los datos de una tabla.
COMMIT	Hace que todos los cambios pendientes sean permanentes.
SAVEPOINT	Se utiliza para realizar rollback en el marcador del punto de grabación.
ROLLBACK	Desecha todos los cambios de datos pendientes.

ORACLE

8-47

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Resumen

En esta lección, debería haber aprendido a manipular datos de la base de datos Oracle utilizando las sentencias INSERT, UPDATE y DELETE. Controle los cambios de datos utilizando las sentencias COMMIT, SAVEPOINT y ROLLBACK.

Oracle Server garantiza una visualización consistente de los datos en todo momento.

El bloqueo puede ser implícito o explícito.

## Visión General de la Práctica 8

Esta práctica cubre los siguientes temas:

- Inserción de filas en las tablas
- Actualización y supresión de filas de la tabla
- Control de transacciones

ORACLE

8-48

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Visión General de la Práctica 8

En esta práctica, agregará filas a la tabla MY\_EMPLOYEE, actualizará y suprimirá datos de la tabla y controlará las transacciones.



## Práctica 8

Inserte datos en la tabla MY\_EMPLOYEE.

1. Ejecute la sentencia en el archivo de comandos lab8\_1.sql para crear la tabla MY\_EMPLOYEE que se utilizará para la práctica.
2. Describa la estructura de la tabla MY\_EMPLOYEE para identificar los nombres de columna.

Name	Null?	Type
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)

3. Agregue la primera fila de datos a la tabla MY\_EMPLOYEE desde los siguientes datos de ejemplo. No enumere las columnas en la cláusula INSERT.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

4. Rellene la tabla MY\_EMPLOYEE con la segunda fila de los datos de ejemplo de la lista anterior. Esta vez, enumere las columnas explícitamente en la cláusula INSERT.
5. Confirme la adición a la tabla.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

### Práctica 8 (continuación)

6. Escriba una sentencia INSERT en un archivo de texto llamado `loademp.sql` para cargar filas en la tabla `MY_EMPLOYEE`. Concatene la primera letra del nombre con los primeros siete caracteres del apellido para crear el identificador de usuario.
7. Rellene la tabla con las dos filas siguientes de los datos de ejemplo ejecutando la sentencia INSERT en el archivo de comandos que ha creado.
8. Confirme las adiciones a la tabla.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

9. Haga que las adiciones de datos sean permanentes.

Actualice y suprima datos en la tabla `MY_EMPLOYEE`.

10. Cambie el apellido del empleado 3 por Drexler.
11. Cambie el salario a 1000 para todos los empleados con un salario inferior a 900.
12. Verifique los cambios realizados a la tabla.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Dancs	Betty	bdancs	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

13. Suprima a Betty Dancs de la tabla `MY_EMPLOYEE`.
14. Confirme los cambios realizados a la tabla.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

## Práctica 8 (continuación)

15. Valide todos los cambios pendientes.

Controle la transacción de datos a la tabla MY\_EMPLOYEE.

16. Rellene la tabla con la última fila de los datos de ejemplo modificando las sentencias del archivo de comandos que creó en el paso 6. Ejecute las sentencias en el archivo de comandos.
17. Confirme la adición a la tabla.

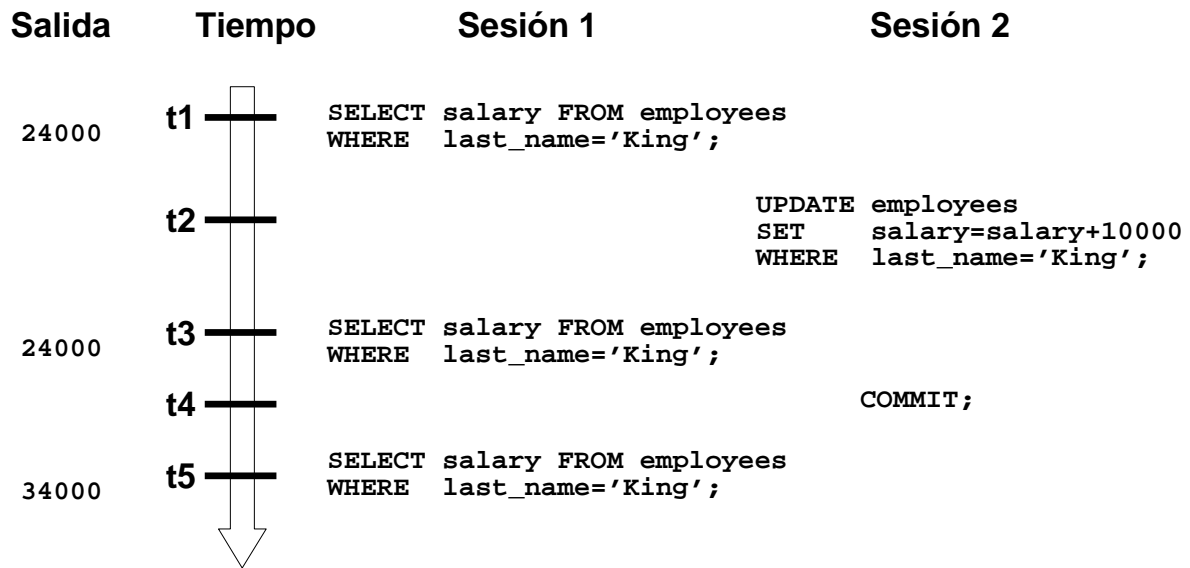
ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000
5	Ropeburn	Audrey	aropebur	1550

18. Marque un punto intermedio en el procesamiento de la transacción.
19. Vacíe toda la tabla.
20. Confirme que la tabla está vacía.
21. Deseche la operación DELETE más reciente sin desechar la operación INSERT anterior.
22. Confirme que la nueva fila sigue intacta.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000
5	Ropeburn	Audrey	aropebur	1550

23. Haga que la adición de datos sea permanente.

## Ejemplo de Consistencia de Lectura



ORACLE

# 9

## Creación y Gestión de Tablas

ORACLE

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

# Objetivos

**Al finalizar esta lección, debería estar capacitado para:**

- **Describir los principales objetos de base de datos**
- **Crear tablas**
- **Describir los tipos de dato que se pueden utilizar al especificar la definición de columna**
- **Modificar definiciones de tabla**
- **Borrar, cambiar el nombre y truncar tablas**

ORACLE

## Objetivo de la Lección

En esta lección, obtendrá información sobre las tablas, los principales objetos de base de datos y sus relaciones entre sí. También aprenderá a crear, modificar y borrar tablas.

# Objetos de Base de Datos

Objeto	Descripción
Tabla	Unidad básica de almacenamiento; está formada por filas y columnas.
Vista	Representa lógicamente subconjuntos de datos de una o varias tablas.
Secuencia	Generador de valor numérico.
Índice	Mejora el rendimiento de algunas consultas.
Sinónimo	Proporciona nombres alternativos a objetos.

ORACLE

9-3

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Objetos de Base de Datos

Una base de datos Oracle puede contener varias estructuras de datos, cada una de las cuales se debe describir en el diseño de base de datos de forma que se pueda crear durante la etapa de creación del desarrollo de la base de datos.

- Tabla: Almacena datos
- Vista: Subconjunto de datos de una o varias tablas
- Secuencia: Generador de valor numérico
- Índice: Mejora el rendimiento de algunas consultas
- Sinónimo: Proporciona nombres alternativos a objetos

## Estructuras de Tabla de Oracle9i

- Las tablas se pueden crear en cualquier momento, incluso mientras los usuarios utilizan la base de datos.
- No es necesario especificar el tamaño de las tablas, ya que se define en última instancia por la cantidad de espacio asignado a la base de datos entera. Sin embargo, es importante estimar cuánto espacio va a utilizar una tabla con el tiempo.
- La estructura de tabla se puede modificar en línea.

**Nota:** Hay más objetos de base de datos disponibles, pero no se cubren en este curso.

# Reglas de Nomenclatura

## Los nombres de tablas y columnas:

- Deben empezar por una letra.
- Deben tener entre 1 y 30 caracteres.
- Sólo deben contener A–Z, a–z, 0–9, \_, \$ y #.
- No deben duplicar el nombre de otro objeto propiedad del mismo usuario.
- No deben ser palabras reservadas de Oracle Server.

ORACLE

9-4

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Reglas de Nomenclatura

Asigne un nombre a las columnas y tablas de base de datos siguiendo las reglas estándar para la nomenclatura de cualquier objeto de base de datos Oracle:

- Los nombres de tablas y columnas deben empezar por una letra y deben tener entre 1 y 30 caracteres.
- Los nombres solamente deben contener los caracteres A–Z, a–z, 0–9, \_ (guión bajo), \$ y # (caracteres válidos, aunque su uso no se recomienda).
- Los nombres no deben duplicar el nombre de otro objeto propiedad del mismo usuario de Oracle Server.
- Los nombres no deben ser palabras reservadas de Oracle Server.

### Instrucciones de Nomenclatura

Utilice nombres descriptivos para tablas y otros objetos de base de datos.

**Nota:** Los nombres no son sensibles a mayúsculas/minúsculas. Por ejemplo, EMPLOYEES se trata como el mismo nombre que eMpLoYeeS o eMpLoYEEs.

Para obtener más información, consulte *Oracle9i SQL Reference*, “Object Names and Qualifiers”.



## La Sentencia CREATE TABLE

- **Debe tener:**
  - El privilegio **CREATE TABLE**
  - Un área de almacenamiento

```
CREATE TABLE [schema.]table  
              (column datatype [DEFAULT expr][, ...]);
```

- **Especifique:**
  - Nombre de tabla
  - Nombre de columna, tipo de dato de columna y tamaño de columna

ORACLE

9-5

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### La Sentencia CREATE TABLE

Cree tablas para almacenar datos ejecutando la sentencia SQL **CREATE TABLE**, que es una de las sentencias del lenguaje de definición de datos (DDL), que se cubren en lecciones posteriores. Las sentencias DDL son un subconjunto de sentencias SQL utilizadas para crear, modificar o eliminar estructuras de base de datos Oracle9i. Estas sentencias tienen un efecto inmediato sobre la base de datos y también registran información en el diccionario de datos.

Para crear una tabla, el usuario debe tener el privilegio **CREATE TABLE** y un área de almacenamiento en la que crear objetos. El administrador de la base de datos utiliza secuencias del lenguaje de control de datos (DCL), que se cubren en una lección posterior, para otorgar privilegios a los usuarios.

En la sintaxis:

<i>schema</i>	es lo mismo que el nombre del propietario.
<i>table</i>	es el nombre de la tabla.
DEFAULT <i>expr</i>	especifica un valor por defecto si se omite un valor en la sentencia INSERT.
<i>column</i>	es el nombre de la columna.
<i>datatype</i>	es la longitud y el tipo de dato de la columna.

## Referencia a Tablas de otro Usuario

- Las tablas pertenecientes a otros usuarios no están en el esquema del usuario.
- Debe utilizar el nombre de propietario como prefijo para dichas tablas.

ORACLE

### Referencia a Tablas de otro Usuario

Un *esquema* es una recopilación de objetos. Los objetos de esquema son las estructuras lógicas que hacen referencia directamente a los datos de una base de datos. Los objetos de esquema son tablas, vistas, sinónimos, secuencias, procedimientos almacenados, índices, agrupamientos y enlaces de base de datos.

Si una tabla no pertenece al usuario, el nombre del propietario debe ir como prefijo en la tabla. Por ejemplo, si hay un esquema llamado USER\_B y USER\_B tiene la tabla EMPLOYEES, debe especificar lo siguiente para recuperar datos de dicha tabla:

```
SELECT *  
FROM   user_b.employees;
```

## La Opción DEFAULT

- Especifique un valor por defecto para una columna durante una inserción.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Los valores literales, las expresiones o las funciones SQL son valores válidos.
- El nombre de otra columna o una pseudocolumna son valores no válidos.
- El tipo de dato por defecto debe coincidir con el tipo de dato de columna.

ORACLE

### La Opción DEFAULT

A una columna se le puede asignar un valor por defecto utilizando la opción DEFAULT. Esta opción evita que se introduzcan valores nulos en las columnas si se inserta una fila sin un valor para la columna. El valor por defecto puede ser un literal, una expresión o una función SQL, como SYSDATE y USER, pero el valor no puede ser el nombre de otra columna o una pseudocolumna, como NEXTVAL o CURRVAL. La expresión por defecto debe coincidir con el tipo de dato de la columna.

**Nota:** CURRVAL y NEXTVAL se explican más adelante.

# Creación de Tablas

- Cree la tabla.

```
CREATE TABLE dept
      (deptno NUMBER(2),
       dtype VARCHAR2(14),
       loc VARCHAR2(13));
Table created.
```

- Confirme la creación de la tabla.

```
DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

ORACLE

## Creación de Tablas

En el ejemplo de la transparencia se crea la tabla DEPT, con tres columnas: DEPTNO, DNAME y LOC. Además, se confirma la creación de la tabla emitiendo el comando DESCRIBE.

Como la creación de una tabla es una sentencia DDL, tiene lugar una validación automática cuando se ejecuta esta sentencia.

## Tablas de la Base de Datos Oracle

- **Tablas de Usuario:**
  - Son una recopilación de tablas creadas y mantenidas por el usuario
  - Contienen información de usuario
- **Diccionario de Datos:**
  - Es una recopilación de tablas creadas y mantenidas por Oracle Server
  - Contienen información de la base de datos

ORACLE

9-9

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tablas de la Base de Datos Oracle

Las tablas de usuario son tablas creadas por el usuario como, por ejemplo, EMPLOYEES. Hay otra recopilación de tablas y vistas en la base de datos Oracle conocida como el *diccionario de datos*. Esta recopilación la crea y la mantiene Oracle Server y contiene información sobre la base de datos.

Todas las tablas del diccionario de datos son propiedad del usuario SYS. El usuario no accede casi nunca a las tablas de base, ya que la información que contienen no es fácil de entender. Por lo tanto, los usuarios normalmente acceden a vistas del diccionario de datos, donde la información se presenta en un formato más fácil de entender. La información almacenada en el diccionario de datos incluye nombres de los usuarios de Oracle Server, privilegios otorgados a los usuarios, nombres de objetos de la base de datos, restricciones de tablas e información de auditoría.

Hay cuatro categorías de vistas del diccionario de datos; cada una tiene un prefijo distinto que refleja el uso que se pretende.

Prefijo	Descripción
USER_	Estas vistas contienen información sobre los objetos propiedad del usuario.
ALL_	Estas vistas contienen información sobre todas las tablas (tablas de objetos y tablas relacionales) accesibles para el usuario.
DBA_	Estas vistas están restringidas y sólo pueden acceder a ellas personas que tienen asignado el rol DBA.
V\$	Estas vistas son vistas de rendimiento dinámicas, rendimiento de servidor de base de datos, memoria y bloqueo.

## Consulta del Diccionario de Datos

- Vea tipos de objeto distintos propiedad del usuario.

```
SELECT table_name  
FROM   user_tables ;
```

- Vea tablas, vistas, sinónimos y secuencias propiedad del usuario.

```
SELECT DISTINCT object_type  
FROM   user_objects ;
```

- Vea los nombres de tablas propiedad del usuario.

```
SELECT *  
FROM   user_catalog ;
```

ORACLE

### Consulta del Diccionario de Datos

Puede consultar las tablas del diccionario de datos para visualizar los diversos objetos de base de datos que tenga en propiedad. Las tablas del diccionario de datos que se utilizan normalmente son:

- USER\_TABLES
- USER\_OBJECTS
- USER\_CATALOG

**Nota:** USER\_CATALOG tiene un sinónimo llamado CAT. Puede utilizar este sinónimo en lugar de USER\_CATALOG en sentencias SQL.

```
SELECT *  
FROM   CAT ;
```

## Tipos de Dato

Tipo de Dato	Descripción
<code>VARCHAR2(size)</code>	Dato de caracteres de longitud variable
<code>CHAR(size)</code>	Dato de caracteres de longitud fija
<code>NUMBER(p,s)</code>	Dato numérico de longitud variable
<code>DATE</code>	Valores de fecha y hora
<code>LONG</code>	Dato de caracteres de longitud variable de hasta 2 gigabytes
<code>CLOB</code>	Dato de caracteres de hasta 4 gigabytes
<code>RAW y LONG RAW</code>	Dato raw binario
<code>BLOB</code>	Dato binario de hasta 4 gigabytes
<code>BFILE</code>	Dato binario almacenado en un archivo externo; hasta 4 gigabytes
<code>ROWID</code>	Sistema numérico de base 64 que representa la dirección única de una fila en su tabla.

ORACLE

9-11

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Tipos de Dato

Tipo de dato	Descripción
<code>VARCHAR2(size)</code>	Dato de caracteres de longitud variable (se debe especificar un <i>tamaño</i> máximo: El <i>tamaño</i> mínimo es 1 y el máximo, 4000)
<code>CHAR [(size)]</code>	Datos de caracteres de longitud fija de bytes de <i>tamaño</i> de largo (el <i>tamaño</i> por defecto y mínimo es 1; el <i>tamaño</i> máximo es 2000)
<code>NUMBER [(p,s)]</code>	Número de precisión <i>p</i> y escala <i>s</i> (la precisión es el número total de dígitos decimales y la escala es el número de dígitos a la derecha de la coma decimal; el rango de valores de la precisión es de 1 a 38 y el de la escala, -84 a 127)
<code>DATE</code>	Valores de fecha y hora hasta el segundo más próximo entre el 1 de enero de 4712 a. C y el 31 de diciembre de 9999 d. C.
<code>LONG</code>	Dato de caracteres de longitud variable de hasta 2 GB
<code>CLOB</code>	Dato de caracteres de hasta 4 GB

## Tipos de Dato (continuación)

Tipo de dato	Descripción
RAW( <i>size</i> )	Dato raw binario de <i>tamaño</i> de longitud (se debe especificar un <i>tamaño</i> máximo. El <i>tamaño</i> máximo es 2000)
LONG RAW	Dato raw binario de longitud variable de hasta 2 GB
BLOB	Dato binario de hasta 4 GB
BFILE	Dato binario almacenado en un archivo externo; hasta 4 GB
ROWID	Sistema numérico de base 64 que representa la dirección única de una fila en su tabla

- No se copia una columna LONG cuando se crea una tabla utilizando una subconsulta.
- No se puede incluir una columna LONG en una cláusula GROUP BY u ORDER BY.
- Sólo se puede utilizar una columna LONG por tabla.
- No se pueden definir restricciones en una columna LONG.
- Puede utilizar una columna CLOB en lugar de una columna LONG.



## Tipos de Dato DateTime

### Mejoras de datetime con Oracle9i:

- Se han introducido nuevos tipos de dato datetime.
- Está disponible un nuevo almacenamiento de tipo de dato.
- Se han realizado mejoras en zonas horarias y en la zona horaria local.

Tipo de Dato	Descripción
<b>TIMESTAMP</b>	<b>Fecha con segundos fraccionarios</b>
<b>INTERVAL YEAR TO MONTH</b>	<b>Almacenado como intervalo de años y meses</b>
<b>INTERVAL DAY TO SECOND</b>	<b>Almacenado como intervalo de días a horas, minutos y segundos</b>

ORACLE

### Otros Tipos de Dato DateTime

Tipo de Dato	Descripción
<b>TIMESTAMP</b>	Permite que se almacene la hora como fecha con segundos fraccionarios. Hay diversas variaciones del tipo de dato.
<b>INTERVAL YEAR TO MONTH</b>	Permite que se almacene la hora como intervalo de años y meses. Se utiliza para representar la diferencia entre dos valores datetime, donde las únicas partes significativas son el año y el mes.
<b>INTERVAL DAY TO SECOND</b>	Permite que se almacene el tiempo como intervalo de días a horas, minutos y segundos. Resulta útil para representar la diferencia exacta entre dos valores datetime.

## Tipos de Dato DateTime

- El tipo de dato **TIMESTAMP** es una extensión del tipo de dato **DATE**.
- Almacena el año, el mes y el día del tipo de dato **DATE**, más valores de hora, minuto y segundo, así como el valor de segundo fraccionario.
- El tipo de dato **TIMESTAMP** se especifica como se indica a continuación:

```
TIMESTAMP[(fractional_seconds_precision)]
```

ORACLE

9-14

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tipos de Dato DateTime

`fractional_seconds_precision` especifica opcionalmente el número de dígitos de la parte fraccionaria del campo datetime **SECOND** y puede ser un número del rango 0 a 9. El valor por defecto es 6.

#### Ejemplo

```
CREATE TABLE new_employees
(employee_id NUMBER,
first_name VARCHAR2(15),
last_name VARCHAR2(15),
...
start_date TIMESTAMP(7),
...);
```

En el ejemplo anterior, estamos creando una tabla **NEW\_EMPLOYEES** con una columna **start\_date** con el tipo de dato **TIMESTAMP**. La precisión '7' indica la precisión de segundos fraccionarios que, si no se especifica, toma por defecto el valor '6'.

Suponga que se han insertado dos filas en la tabla **NEW\_EMPLOYEES**. La salida muestra las diferencias en la visualización. (El tipo de dato **DATE** muestra por defecto el formato **DD-MON-RR**):

```
SELECT start_date
FROM new_employees;
```

```
17-JUN-87 12.00.00.000000 AM
21-SEP-89 12.00.00.000000 AM
```

## Tipo de Dato **TIMESTAMP WITH TIME ZONE**

- **TIMESTAMP WITH TIME ZONE** es una variante de **TIMESTAMP** que incluye un cambio de zona horaria en su valor.
- El cambio de zona horaria es la diferencia, en horas y minutos, entre la hora local y UTC.

```
TIMESTAMP[(fractional_seconds_precision)]  
WITH TIME ZONE
```

ORACLE

9-15

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tipos de Dato Datetime

UTC son las siglas de Coordinated Universal Time (Hora Universal Coordinada), la antigua Hora Media de Greenwich. Dos valores **TIMESTAMP WITH TIME ZONE** se consideran idénticos si representan el mismo instante en UTC, independientemente de los offsets de **TIME ZONE** almacenados en los datos.

Como **TIMESTAMP WITH TIME ZONE** también puede almacenar información de zona horaria, está especialmente indicado para registrar información de fecha que se debe recopilar o coordinar entre distintas regiones geográficas.

Por ejemplo,

```
TIMESTAMP '1999-04-15 8:00:00 -8:00'
```

es lo mismo que

```
TIMESTAMP '1999-04-15 11:00:00 -5:00'
```

Es decir, 8:00 a.m. Hora Estándar del Pacífico es lo mismo que 11:00 a.m. Hora Estándar Oriental.

Esto también se puede especificar como

```
TIMESTAMP '1999-04-15 8:00:00 US/Pacific'
```

**Nota:** `fractional_seconds_precision` especifica opcionalmente el número de dígitos de la parte fraccionaria del campo datetime `SECOND` y puede ser un número del rango 0 a 9. El valor por defecto es 6.

## Tipo de Dato TIMESTAMP WITH LOCAL TIME

- **TIMESTAMP WITH LOCAL TIME ZONE es otra variante de TIMESTAMP que incluye un cambio de zona horaria en su valor.**
- **Los datos almacenados en la base de datos se normalizan en la zona horaria de la base de datos.**
- **El cambio de zona horaria no se almacena como parte del dato de columna; Oracle devuelve el dato en la zona horaria de la sesión local del usuario.**
- **El tipo de dato TIMESTAMP WITH LOCAL TIME ZONE se especifica como se indica a continuación:**

```
TIMESTAMP[(fractional_seconds_precision)]  
WITH LOCAL TIME ZONE
```

ORACLE

9-16

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tipos de Dato DateTime

A diferencia de **TIMESTAMP WITH TIME ZONE**, puede especificar columnas de tipo **TIMESTAMP WITH LOCAL TIME ZONE** como parte de una clave primaria o única. El cambio de zona horaria es la diferencia (en horas y minutos) entre la hora local y UTC. No hay ningún literal para **TIMESTAMP WITH LOCAL TIME ZONE**.

**Nota:** `fractional_seconds_precision` especifica opcionalmente el número de dígitos de la parte fraccionaria del campo datetime **SECOND** y puede ser un número del rango 0 a 9. El valor por defecto es 6.

#### Ejemplo

```
CREATE TABLE time_example  
  (order_date TIMESTAMP WITH LOCAL TIME ZONE);  
  
INSERT INTO time_example VALUES('15-NOV-00 09:34:28 AM');  
  
SELECT *  
FROM   time_example;  
  
order_date  
-----  
15-NOV-00 09.34.28.000000 AM
```

El tipo **TIMESTAMP WITH LOCAL TIME ZONE** es adecuado para aplicaciones de dos capas en las que desee visualizar fechas y horas utilizando la zona horaria del sistema cliente.

## Tipo de Dato INTERVAL YEAR TO MONTH

- **INTERVAL YEAR TO MONTH almacena un período de tiempo que utiliza los campos datetime YEAR y MONTH.**

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

```
INTERVAL '123-2' YEAR(3) TO MONTH
Indicates an interval of 123 years, 2 months.
```

```
INTERVAL '123' YEAR(3)
Indicates an interval of 123 years 0 months.
```

```
INTERVAL '300' MONTH(3)
Indicates an interval of 300 months.
```

```
INTERVAL '123' YEAR
Returns an error, because the default precision is 2,
and '123' has 3 digits.
```

ORACLE

9-17

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tipo de Dato INTERVAL YEAR TO MONTH

INTERVAL YEAR TO MONTH almacena un período de tiempo que utiliza los campos datetime YEAR y MONTH. Utilice INTERVAL YEAR TO MONTH para representar la diferencia entre dos valores datetime, donde las únicas partes significativas son el año y el mes. Por ejemplo, puede utilizar este valor para definir un recordatorio para una fecha dentro de 120 meses, o para comprobar si han transcurrido 6 meses desde una fecha determinada.

Especifique INTERVAL YEAR TO MONTH como se indica a continuación:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

En la sintaxis:

`year_precision` es el número de dígitos en el campo datetime YEAR. El valor por defecto de `year_precision` es 2.

#### Ejemplo

```
CREATE TABLE time_example2
(loan_duration INTERVAL YEAR (3) TO MONTH);

INSERT INTO time_example2 (loan_duration)
VALUES (INTERVAL '120' MONTH(3));

SELECT TO_CHAR( sysdate+loan_duration, 'dd-mon-yyyy')
FROM   time_example2;           --today's date is 26-Sep-2001
```

TO\_CHAR(SYS

26-sep-2011

## Tipo de Dato INTERVAL DAY TO SECOND

- **INTERVAL DAY TO SECOND almacena un período de tiempo en términos de días, horas, minutos y segundos.**

```
INTERVAL DAY [(day_precision)]  
TO SECOND [(fractional_seconds_precision)]
```

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)  
Indicates 4 days, 5 hours, 12 minutes, 10 seconds,  
and 222 thousandths of a second. INTERVAL '123' YEAR(3).
```

```
INTERVAL '7' DAY  
Indicates 7 days.
```

```
INTERVAL '180' DAY(3)  
Indicates 180 days.
```

ORACLE

9-18

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tipo de Dato INTERVAL DAY TO SECOND

INTERVAL DAY TO SECOND almacena un período de tiempo en términos de días, horas, minutos y segundos.

Utilice INTERVAL DAY TO SECOND para representar la diferencia exacta entre dos valores datetime. Por ejemplo, puede utilizar este valor para definir un recordatorio para una hora dentro de 36 horas o para registrar el tiempo entre el comienzo y el final de una carrera. Para representar espacios de tiempo grandes, incluyendo varios años, con gran precisión, puede utilizar un valor mayor para la porción de días.

Especifique INTERVAL DAY TO SECOND como se indica a continuación:

```
INTERVAL DAY [(day_precision)]  
TO SECOND [(fractional_seconds_precision)]
```

En la sintaxis:

`day_precision`

es el número de dígitos en el campo datetime DAY. Los valores válidos son 0 a 9. El valor por defecto es 2.

`fractional_seconds_precision` es el número de dígitos en la parte fraccionaria del campo datetime SECOND. Los valores válidos son 0 a 9. El valor por defecto es 6.

## Tipo de Dato INTERVAL DAY TO SECOND

- **INTERVAL DAY TO SECOND almacena un período de tiempo en términos de días, horas, minutos y segundos.**

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)
Indicates 4 days, 5 hours, 12 minutes, 10 seconds,
and 222 thousandths of a second.

INTERVAL '4 5:12' DAY TO MINUTE
Indicates 4 days, 5 hours and 12 minutes.

INTERVAL '400 5' DAY(3) TO HOUR
Indicates 400 days 5 hours.

INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)
indicates 11 hours, 12 minutes, and 10.2222222 seconds.
```

ORACLE

9-19

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Tipo de Dato INTERVAL DAY TO SECOND

#### Ejemplo

```
CREATE TABLE time_example3
(day_duration INTERVAL DAY (3) TO SECOND);

INSERT INTO time_example3 (day_duration)
VALUES (INTERVAL '180' DAY(3));

SELECT sysdate + day_duration "Half Year"
FROM   time_example3;           --today's date is 26-Sep-2001
```

Half Year
25-MAR-02

## Creación de una Tabla Utilizando una Sintaxis de Subconsulta

- Cree una tabla e inserte filas combinando la sentencia **CREATE TABLE** y la opción **AS subquery**.

```
CREATE TABLE table
      [(column, column...)]
AS subquery;
```

- Haga coincidir el número de columnas especificadas con el número de columnas de subconsulta.
- Defina columnas con nombres de columna y valores por defecto.

ORACLE

9-20

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Creación de una Tabla a partir de Filas de otra Tabla

Un segundo método para crear una tabla es aplicar la cláusula **AS subquery**, que crea la tabla e inserta filas devueltas desde la subconsulta.

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>column</i>	es el nombre de la columna, el valor por defecto y la restricción de integridad.
<i>subquery</i>	es la sentencia <b>SELECT</b> que define el juego de filas que se van a insertar en la nueva tabla.

### Instrucciones

- La tabla se crea con los nombres de columna especificados y las filas recuperadas por la sentencia **SELECT** se insertan en la tabla.
- La definición de columna sólo puede contener el nombre de columna y el valor por defecto.
- Si se proporcionan especificaciones de columna, el número de columnas debe ser igual al número de columnas de la lista **SELECT** de la subconsulta.
- Si no se proporcionan especificaciones de columna, los nombres de columna de la tabla son los mismos que los nombres de columna de la subconsulta.
- Las reglas de integridad no se transmiten a la nueva tabla, sólo las definiciones de tipo de dato de columna.



## Creación de una Tabla Utilizando una Subconsulta

```
CREATE TABLE dept80
```

```
AS
```

```
SELECT employee_id, last_name,  
       salary*12 ANNSAL,  
       hire_date  
FROM   employees  
WHERE  department_id = 80;
```

```
Table created.
```

```
DESCRIBE dept80
```

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ORACLE

9-21

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Creación de una Tabla a partir de Filas de otra Tabla (continuación)

En el ejemplo de la transparencia se crea una tabla con el nombre DEPT80, que contiene detalles de todos los empleados que trabajan en el departamento 80. Observe que los datos para la tabla DEPT80 proceden de la tabla EMPLOYEES.

Puede verificar la existencia de una tabla de base de datos y comprobar las definiciones de columna utilizando el comando DESCRIBE de iSQL\*Plus..

Be Asegúrese de proporcionar un alias de columna al seleccionar una expresión. A la expresión SALARY\*12 se le ha asignado el alias ANNSAL. Sin el alias, se genera este error:

```
ERROR at line 3:
```

```
ORA-00998: must name this expression with a column alias
```

## La Sentencia ALTER TABLE

Utilice la sentencia ALTER TABLE para:

- Agregar una columna nueva
- Modificar una columna existente
- Definir un valor por defecto para la nueva columna
- Borrar una columna

ORACLE

9-22

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### La Sentencia ALTER TABLE

Después de crear una tabla, es posible que tenga que cambiar la estructura de la misma por haber omitido una columna, porque sea necesario cambiar la definición de columna o porque tenga que eliminar columnas. Para ello, utilice la sentencia ALTER TABLE.

# La Sentencia ALTER TABLE

Utilice la sentencia ALTER TABLE para agregar, modificar o borrar columnas.

```
ALTER TABLE table
ADD          (column datatype [DEFAULT expr]
             [, column datatype]...);
```

```
ALTER TABLE table
MODIFY       (column datatype [DEFAULT expr]
             [, column datatype]...);
```

```
ALTER TABLE table
DROP         (column);
```

ORACLE

9-23

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Sentencia ALTER TABLE (continuación)

Puede agregar, modificar y borrar columnas de una tabla utilizando la sentencia ALTER TABLE.

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
ADD   MODIFY   DROP	es el tipo de modificación.
<i>column</i>	es el nombre de la nueva columna.
<i>datatype</i>	es el tipo de dato y la longitud de la nueva columna.
ULT <i>expr</i>	especifica el valor por defecto para una columna nueva.

**Nota:** En la transparencia se ofrece una sintaxis abreviada de ALTER TABLE. En una lección posterior se cubren más aspectos de ALTER TABLE.

## Adición de una Columna

### Columna nueva

#### DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

JOB_ID

“Agregue una columna nueva a la tabla DEPT80”.

#### DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

ORACLE

## Adición de una Columna

En el gráfico se agrega la columna JOB\_ID a la tabla DEPT80. Observe que la nueva columna pasa a ser la última de la tabla.

## Adición de una Columna

- Puede utilizar la cláusula **ADD** para agregar columnas.

```
ALTER TABLE dept80
ADD      (job_id VARCHAR2(9));
Table altered.
```

- La nueva columna pasa a ser la última.

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

ORACLE

### Instrucciones para la Adición de una Columna

- Puede agregar o modificar columnas.
- No puede especificar dónde debe aparecer la columna. La nueva columna para a ser la última.

En el ejemplo de la transparencia se agrega una columna con el nombre **JOB\_ID** a la tabla **DEPT80**. La columna **JOB\_ID** pasa a ser la última de la tabla.

**Nota:** Si una tabla ya contiene filas cuando se agrega una columna, la nueva columna es inicialmente nula para todas las filas.

## Modificación de una Columna

- Puede cambiar el tipo de dato, el tamaño y el valor por defecto de una columna.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30));
Table altered.
```

- Un cambio en el valor por defecto sólo afecta a las inserciones posteriores en la tabla.

ORACLE

9-26

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Modificación de una Columna

Puede modificar una definición de columna utilizando la sentencia `ALTER TABLE` con la cláusula `MODIFY`. La modificación de columna puede incluir cambios en el tipo de dato, el tamaño y el valor por defecto de la columna.

#### Instrucciones

- Puede aumentar el ancho o la precisión de una columna numérica.
- Puede aumentar el ancho de columnas numéricas o de caracteres.
- Puede disminuir el ancho de una columna sólo si ésta contiene sólo valores nulos o si la tabla no tiene filas.
- Puede cambiar el tipo de dato sólo si la columna contiene valores nulos.
- Puede convertir una columna `CHAR` en el tipo de dato `VARCHAR2` o convertir una columna `VARCHAR2` en el tipo de dato `CHAR` sólo si la columna contiene valores nulos o si no cambia el tamaño.
- Un cambio en el valor por defecto de una columna sólo afecta a las inserciones posteriores en la tabla.

## Eliminación de una Columna

Utilice la cláusula **DROP COLUMN** para borrar las columnas que ya no necesite de la tabla.

```
ALTER TABLE dept80  
DROP COLUMN job_id;  
Table altered.
```

ORACLE

9-27

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Eliminación de una Columna

Puede borrar una columna de una tabla utilizando la sentencia **ALTER TABLE** con la cláusula **DROP COLUMN**. Esta función está disponible a partir de Oracle8i.

#### Instrucciones

- La columna puede o no puede contener datos.
- Con la sentencia **ALTER TABLE**, sólo se puede borrar una columna cada vez.
- La tabla debe tener al menos una columna después de la modificación.
- Una vez borrada una columna, no se puede recuperar.

## La Opción SET UNUSED

- Utilice la opción **SET UNUSED** para marcar una o varias columnas como no utilizadas.
- Utilice la opción **DROP UNUSED COLUMNS** para eliminar las columnas marcadas como no utilizadas.

```
ALTER TABLE table
SET UNUSED (column);
OR
ALTER TABLE table
SET UNUSED COLUMN column;
```

```
ALTER TABLE table
DROP UNUSED COLUMNS;
```

ORACLE

9-28

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### La Opción SET UNUSED

La opción **SET UNUSED** marca una o varias columnas como no utilizadas para que se puedan borrar cuando la demanda de los recursos del sistema sea menor. Esta función está disponible a partir de Oracle8i. Al especificar esta cláusula no se eliminan realmente las columnas de destino de cada fila de la tabla (es decir, no se restaura el espacio en disco utilizado por estas columnas). Por lo tanto, el tiempo de respuesta es menor que si ejecutara la cláusula **DROP**. Las columnas no utilizadas se consideran borradas, aunque sus datos permanezcan en las filas de la tabla. Una vez marcada una columna como no utilizada, no tendrá acceso a ella. Una consulta **SELECT \*** no recuperará datos de columnas no utilizadas. Además, los nombres y los tipos de las columnas marcadas como no utilizadas no se mostrarán durante un comando **DESCRIBE** y puede agregar una nueva columna a la tabla con el mismo nombre que la no utilizada. La información de **SET UNUSED** se almacena en la vista de diccionario **USER\_UNUSED\_COL\_TABS**.

### La Opción DROP UNUSED COLUMNS

**DROP UNUSED COLUMNS** elimina de la tabla todas las columnas marcadas actualmente como no utilizadas. Puede usar esta sentencia para reclamar el espacio en disco adicional de las columnas no utilizadas de la tabla. Si la tabla no contiene columnas no utilizadas, la sentencia no devuelve ningún error.

```
ALTER TABLE dept80
SET UNUSED (last_name);
Table altered.
```

```
ALTER TABLE dept80
DROP UNUSED COLUMNS;
Table altered.
```



## Eliminación de una Tabla

- Se suprimen todos los datos y la estructura de la tabla.
- Se valida cualquier transacción pendiente.
- Se borran todos los índices.
- *No puede* realizar rollback de la sentencia DROP TABLE.

```
DROP TABLE dept80;  
Table dropped.
```

ORACLE

9-29

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Eliminación de una Tabla

La sentencia DROP TABLE elimina la definición de una tabla Oracle. Al borrar una tabla, la base de datos pierde todos los datos de la tabla y todos los índices asociados a ella.

#### Sintaxis

```
DROP TABLE table
```

En la sintaxis:

*table* es el nombre de la tabla.

#### Instrucciones

- Todos los datos se suprimen de la tabla.
- Las vistas y los sinónimos permanecen, pero no son válidos.
- Se valida cualquier transacción pendiente.
- Sólo pueden eliminar una tabla el creador de la misma o un usuario con el privilegio DROP ANY TABLE.

**Nota:** La sentencia DROP TABLE, una vez ejecutada, es irreversible. Oracle Server no cuestiona la acción cuando emite la sentencia DROP TABLE. Si es propietario de dicha tabla o tiene un privilegio de alto nivel, la tabla se elimina inmediatamente. Al igual que con todas las sentencias DDL, DROP TABLE se valida automáticamente.

## Cambio del Nombre de un Objeto

- Para cambiar el nombre de una tabla, una vista, una secuencia o un sinónimo, ejecute la sentencia **RENAME**.

```
RENAME dept TO detail_dept;  
Table renamed.
```

- Debe ser el propietario del objeto.

ORACLE

9-30

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Cambio de Nombre de una Tabla

Las sentencias DDL adicionales incluyen la sentencia **RENAME**, que se utiliza para cambiar el nombre a una tabla, una vista, una secuencia o un sinónimo.

#### Sintaxis

```
RENAME    old_name    TO    new_name ;
```

En la sintaxis :

*old\_name* es el nombre antiguo de la tabla, la vista, la secuencia o el sinónimo.

*new\_name* es el nombre nuevo de la tabla, la vista, la secuencia o el sinónimo.

Debe ser el propietario del objeto al que cambia el nombre.

## Truncamiento de una Tabla

- **La sentencia TRUNCATE TABLE:**
  - Elimina todas las filas de una tabla.
  - Libera el espacio de almacenamiento utilizado por dicha tabla.

```
TRUNCATE TABLE detail_dept;  
Table truncated.
```

- **No puede realizar rollback de la eliminación de filas si utiliza TRUNCATE.**
- **También puede eliminar filas utilizando la sentencia DELETE.**

ORACLE

9-31

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Truncamiento de una Tabla

La sentencia TRUNCATE TABLE es otra sentencia DDL que se utiliza para eliminar todas las filas de una tabla y para liberar el espacio de almacenamiento utilizado por dicha tabla. Al utilizar la sentencia TRUNCATE TABLE, no puede realizar rollback de la eliminación de filas.

#### Sintaxis

```
TRUNCATE TABLE table;
```

En la sintaxis:

*table* es el nombre de la tabla

Debe ser el propietario de la tabla o tener privilegios de sistema DELETE TABLE para truncar una tabla.

La sentencia DELETE también puede eliminar todas las filas de una tabla, pero no libera espacio de almacenamiento. El comando TRUNCATE es más rápido. Eliminar filas con la sentencia TRUNCATE es más rápido que con la sentencia DELETE por los siguientes motivos:

- La sentencia TRUNCATE es una sentencia del lenguaje de definición de datos (DDL) y no genera información de rollback.
- Truncar una tabla no arranca los disparadores de supresión de la tabla.
- Si la tabla es la principal de una restricción de integridad referencial, no puede truncarla. Desactive la restricción antes de emitir la sentencia TRUNCATE.

## Adición de Comentarios a una Tabla

- Puede agregar comentarios a una tabla o a una columna utilizando la sentencia **COMMENT**.

```
COMMENT ON TABLE employees  
IS 'Employee Information';  
Comment created.
```

- Los comentarios se pueden visualizar a través de las vistas del diccionario de datos:
  - ALL\_COL\_COMMENTS
  - USER\_COL\_COMMENTS
  - ALL\_TAB\_COMMENTS
  - USER\_TAB\_COMMENTS

ORACLE

9-32

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Adición de un Comentario a una Tabla

Puede agregar un comentario de hasta 2.000 bytes acerca de una columna, una tabla, una vista o una instantánea utilizando la sentencia **COMMENT**. El comentario se almacena en el diccionario de datos y se puede visualizar en una de las siguientes vistas del diccionario de datos en la columna **COMMENTS**:

- ALL\_COL\_COMMENTS
- USER\_COL\_COMMENTS
- ALL\_TAB\_COMMENTS
- USER\_TAB\_COMMENTS

#### Sintaxis

```
COMMENT ON TABLE table | COLUMN table.column  
IS 'text';
```

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>column</i>	es el nombre de la columna de la tabla.
<i>text</i>	es el texto del comentario.

Puede borrar un comentario de la base de datos definiéndolo en una cadena vacía (' '):

```
COMMENT ON TABLE employees IS ' ';
```

# Resumen

En esta lección, debería haber aprendido a utilizar sentencias DDL para crear, modificar, borrar y cambiar el nombre a tablas.

Sentencia	Descripción
CREATE TABLE	Crea una tabla.
ALTER TABLE	Modifica estructuras de tabla.
DROP TABLE	Elimina las filas y la estructura de la tabla.
RENAME	Cambia el nombre de una tabla, vista, secuencia o sinónimo.
TRUNCATE	Elimina todas las filas de una tabla y libera el espacio de almacenamiento.
COMMENT	Agrega comentarios a una tabla o una vista.

ORACLE

9-33

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Resumen

En esta lección, debería haber aprendido a utilizar comandos DDL para crear, modificar, borrar y cambiar el nombre a tablas. También debería haber aprendido a truncar una tabla y a agregar comentarios a una tabla.

### CREATE TABLE

- Crea una tabla.
- Crea una tabla basada en otra tabla utilizando una subconsulta.

### ALTER TABLE

- Modifica estructuras de tabla.
- Cambia anchos y tipos de dato de columna y agrega columnas.

### DROP TABLE

- Elimina filas y la estructura de una tabla.
- Una vez ejecutada, no se puede realizar rollback de esta sentencia.

### RENAME

- Cambia el nombre de una tabla, una vista, una secuencia o un sinónimo.

### TRUNCATE

- Elimina todas las filas de una tabla y libera el espacio de almacenamiento utilizado por la tabla.
- La sentencia DELETE sólo elimina filas.

### COMMENT

- Agrega un comentario a una tabla o a una columna.
- Consulte el diccionario de datos para visualizar el comentario.

## Visión General de la Práctica 9

Esta práctica cubre los siguientes temas:

- Creación de tablas nuevas
- Creación de una tabla nueva utilizando la sintaxis `CREATE TABLE AS`
- Modificación de definiciones de columna
- Verificación de que existen las tablas
- Adición de comentarios a tablas
- Eliminación de tablas
- Modificación de tablas

ORACLE

9-34

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Visión General de la Práctica 9

Cree tablas nuevas utilizando la sentencia `CREATE TABLE`. Confirme que la nueva tabla se ha agregado a la base de datos. Cree la sintaxis en el archivo de comandos y ejecute éste para crear la tabla.

## Práctica 9

1. Cree una tabla DEPT basándose en el siguiente gráfico de instancias de tabla. Coloque la sintaxis en un archivo de comandos llamado lab9\_1.sql y, a continuación, ejecute la sentencia en el archivo de comandos para crear la tabla. Confirme que se ha creado la tabla.

<b>Nombre de columna</b>	ID	NAME
<b>Tipo de clave</b>		
<b>Nulos/Únicos</b>		
<b>Tabla de claves ajenas</b>		
<b>Columna de claves ajenas</b>		
<b>Tipo de dato</b>	NUMBER	VARCHAR2
<b>Longitud</b>	7	25

Name	Null?	Type
ID		NUMBER(7)
NAME		VARCHAR2(25)

2. Rellene la tabla DEPT con datos de la tabla DEPARTMENTS. Incluya sólo columnas que necesite.
3. Cree una tabla EMP basándose en el siguiente gráfico de instancias de tabla. Coloque la sintaxis en un archivo de comandos llamado lab9\_3.sql y, a continuación, ejecute la sentencia en el archivo de comandos para crear la tabla. Confirme que se ha creado la tabla.

<b>Nombre de columna</b>	ID	LAST_NAME	FIRST_NAME	DEPT_ID
<b>Tipo de clave</b>				
<b>Nulos/Únicos</b>				
<b>Tabla de claves ajenas</b>				
<b>Columna de claves ajenas</b>				
<b>Tipo de dato</b>	NUMBER	VARCHAR2	VARCHAR2	NUMBER
<b>Longitud</b>	7	25	25	7

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

### Práctica 9 (continuación)

4. Modifique la tabla EMP para permitir un apellido de empleado más largo. Confirme la modificación.

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

5. Confirme que las tablas DEPT y EMP se almacenan en el diccionario de datos. (Indicación: USER\_TABLES)

TABLE_NAME
DEPT
EMP

6. Cree la tabla EMPLOYEES2 basándose en la estructura de la tabla EMPLOYEES. Incluya sólo las columnas EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY y DEPARTMENT\_ID. Llame a las columnas de la nueva tabla ID, FIRST\_NAME, LAST\_NAME, SALARY y DEPT\_ID, respectivamente.
7. Borre la tabla EMP.
8. Cambie el nombre de la tabla EMPLOYEES2 a EMP.
9. Agregue un comentario a las definiciones de las tablas DEPT y EMP que describa las tablas. Confirme las adiciones en el diccionario de datos.
10. Borre la columna FIRST\_NAME de la tabla EMP. Confirme la modificación comprobando la descripción de la tabla.
11. En la tabla EMP, marque la columna DEPT\_ID de la tabla EMP como UNUSED. Confirme la modificación comprobando la descripción de la tabla.
12. Borre todas las columnas UNUSED de la tabla EMP. Confirme la modificación comprobando la descripción de la tabla.



# 10

## Inclusión de Restricciones

ORACLE®

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

# Objetivos

**Al finalizar esta lección, debería estar capacitado para:**

- **Describir restricciones**
- **Crear y mantener restricciones**

ORACLE

10-2

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Objetivo de la Lección

En esta lección, aprenderá a implementar reglas de negocio incluyendo restricciones de integridad.

## ¿Qué Son las Restricciones?

- Las restricciones fuerzan las reglas a nivel de tabla.
- Las restricciones evitan la supresión de una tabla si hay dependencias.
- Son válidos los siguientes tipos de restricción:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

ORACLE

10-3

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Restricciones

Oracle Server utiliza *restricciones* para evitar la introducción de datos no válidos en las tablas. Se puede utilizar restricciones para:

- Forzar reglas sobre los datos de una tabla cuando se inserta, actualiza o suprime una fila de dicha tabla. La restricción se debe satisfacer para que se realice correctamente la operación.
- Evitar la supresión de una tabla si hay dependencias de otras tablas.
- Proporcionar reglas para herramientas Oracle, como Oracle Developer.

### Restricciones de Integridad de Datos

Restricción	Descripción
NOT NULL	Especifica que la columna no puede contener un valor nulo.
UNIQUE	Especifica una columna o una combinación de columnas cuyos valores deben ser únicos para todas las filas de la tabla.
PRIMARY KEY	Identifica de forma única cada fila de la tabla.
FOREIGN KEY	Establece y fuerza una relación de clave ajena entre la columna y una columna de la tabla a la que se hace referencia.
CHECK	Especifica una condición que debe ser verdadera.

For more information, see *Oracle9i SQL Reference*, “CONSTRAINT.”

## Instrucciones sobre Restricciones

- **Asigne un nombre a una restricción; si no lo hace, Oracle Server genera un nombre con el formato `SYS_Cn`.**
- **Cree una restricción:**
  - **Al mismo tiempo que se crea la tabla, o bien**
  - **Una vez creada la tabla.**
- **Defina una restricción a nivel de columna o de tabla.**
- **Visualice una restricción en el diccionario de datos.**

ORACLE

10-4

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Instrucciones sobre Restricciones

Todas las restricciones se almacenan en el diccionario de datos y es sencillo hacer referencia a ellas si se les asigna un nombre significativo. Los nombres de restricciones deben seguir las reglas estándar de nomenclatura de objetos. Si no asigna nombre a la restricción, Oracle Server genera un nombre con el formato `SYS_Cn`, donde *n* es un entero de forma que el nombre de la restricción sea único.

Las restricciones se pueden definir en el momento de la creación de la tabla o una vez creada ésta.

Puede visualizar las restricciones definidas para una tabla específica en la tabla `USER_CONSTRAINTS` del diccionario de datos.

## Definición de Restricciones

```
CREATE TABLE [schema.]table
    (column datatype [DEFAULT expr]
    [column_constraint],
    ...
    [table_constraint][,...]);
```

```
CREATE TABLE employees(
    employee_id  NUMBER(6),
    first_name   VARCHAR2(20),
    ...
    job_id       VARCHAR2(10) NOT NULL,
    CONSTRAINT emp_emp_id_pk
               PRIMARY KEY (EMPLOYEE_ID));
```

ORACLE

10-5

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Definición de Restricciones

La transparencia proporciona la sintaxis para definir restricciones mientras se crea una tabla.

En la sintaxis:

<i>schema</i>	es lo mismo que el nombre del propietario.
<i>table</i>	es el nombre de la tabla.
LT <i>expr</i>	especifica un valor por defecto que se va a utilizar si se omite un valor en la sentencia INSERT.
<i>column</i>	es el nombre de la columna.
<i>datatype</i>	es el tipo de dato y la longitud de la columna.
<i>column_constraint</i>	es una restricción de integridad como parte de la definición de columna.
<i>table_constraint</i>	es una restricción de integridad como parte de la definición de tabla.

Para obtener más información, consulte *Oracle9i SQL Reference*, "CREATE TABLE".

## Definición de Restricciones

- Nivel de restricción de columna

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Nivel de restricción de tabla

```
column,...  
[CONSTRAINT constraint_name] constraint_type  
(column, ...),
```

ORACLE

10-6

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Definición de Restricciones (continuación)

Las restricciones se crean normalmente al mismo tiempo que la tabla, aunque se pueden agregar a una tabla después de su creación y también se pueden desactivar temporalmente.

Las restricciones se pueden definir a dos niveles.

Nivel de Restricción	Descripción
Columna	Hace referencia a una sola columna y se define dentro de una especificación para la columna propietaria; puede definir cualquier tipo de restricción de integridad.
Tabla	Hace referencia a una o varias columnas y se define por separado desde las definiciones de las columnas de la tabla; puede definir cualquier restricción excepto NOT NULL.

En la sintaxis:

*constraint\_name*            es el nombre de la restricción.  
*constraint\_type*           es el tipo de restricción.

## La Restricción NOT NULL

**Asegura que no se permiten valores nulos para la columna:**

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

...  
20 rows selected.

↑  
**Restricción NOT NULL**  
(Ninguna fila puede  
contener  
un valor nulo para  
esta columna)

↑  
**Restricción  
NOT NULL**

↑  
**Ausencia de restricción  
NOT NULL**  
(Cualquier fila puede  
contener un valor nulo  
para esta columna)

ORACLE

10-7

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### La Restricción NOT NULL

La restricción NOT NULL asegura que la columna no contiene valores nulos. Las columnas sin la restricción NOT NULL pueden contener valores nulos por defecto.

# La Restricción NOT NULL

Se define a nivel de columna:

```
CREATE TABLE employees(  
  employee_id    NUMBER(6),  
  last_name      VARCHAR2(25) NOT NULL,  
  salary         NUMBER(8,2),  
  commission_pct NUMBER(2,2),  
  hire_date      DATE  
                CONSTRAINT emp_hire_date_nn  
                NOT NULL,  
  ...  
);
```

Nombrada por el sistema

Nombrada por el usuario

ORACLE

10-8

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción NOT NULL (continuación)

La restricción NOT NULL sólo se puede especificar a nivel de columna, no a nivel de tabla.

En el ejemplo de la transparencia se aplica la restricción NOT NULL a las columnas LAST\_NAME y HIRE\_DATE de la tabla EMPLOYEES. Como estas restricciones no tienen nombre, Oracle Server crea nombres para ellas.

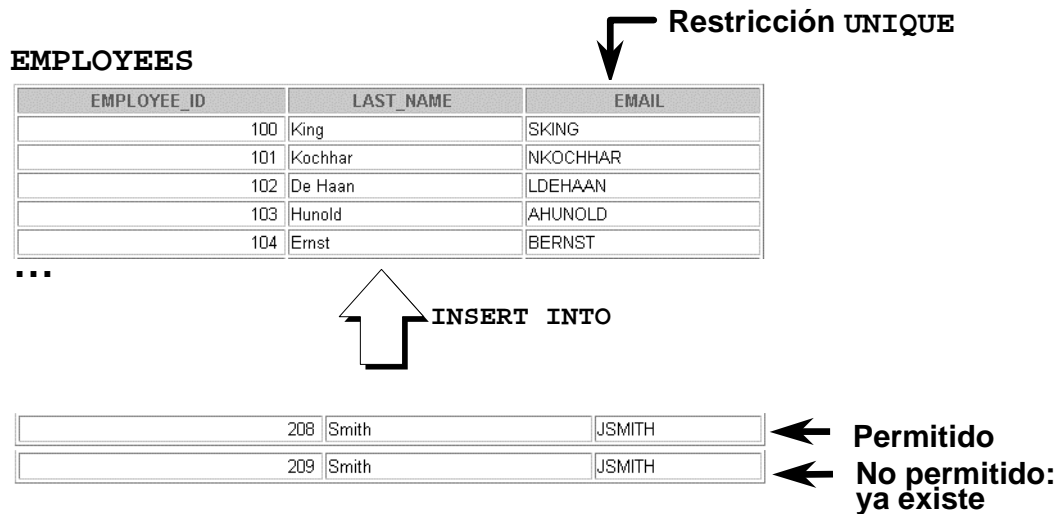
Puede especificar el nombre de la restricción al especificarla:

```
... last_name VARCHAR2(25)  
    CONSTRAINT emp_last_name_nn NOT NULL...
```

**Nota:** Los ejemplos de restricciones descritos en esta lección pueden no estar presentes en las tablas de ejemplo que se proporcionan con el curso. Si se desea, se pueden agregar estas restricciones a las tablas.



# La Restricción UNIQUE



ORACLE

10-9

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción UNIQUE

Una restricción de integridad de clave UNIQUE requiere que todos los valores de una columna o de un juego de columnas (clave) sean únicos, es decir, que no puede haber dos filas de una tabla que tengan valores duplicados en una columna o juego de columnas especificado. La columna (o juego de columnas) incluida en la definición de la restricción de clave UNIQUE se llama *clave única*. Si la restricción UNIQUE está formada por más de una columna, el grupo de columnas se llama *clave única compuesta*.

Las restricciones UNIQUE permiten la entrada de valores nulos a menos que también defina restricciones NOT NULL para las mismas columnas. En realidad, cualquier número de filas puede incluir valores nulos para columnas sin restricciones NOT NULL, pues los valores nulos no se consideran igual a cualquier cosa. Un valor nulo en una columna (o en todas las columnas de una clave UNIQUE compuesta) siempre satisface una restricción UNIQUE.

**Nota:** Debido al mecanismo de búsqueda para restricciones UNIQUE sobre más de una columna, no puede tener valores idénticos en las columnas no nulas de una restricción de clave UNIQUE compuesta parcialmente nula.

# La Restricción UNIQUE

Definida a nivel de tabla o de columna:

```
CREATE TABLE employees(  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

10-10

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción UNIQUE (continuación)


Las restricciones UNIQUE se pueden definir a nivel de columna o de tabla. Se crea una clave única compuesta utilizando la definición a nivel de tabla.

En el ejemplo de la transparencia se aplica la restricción UNIQUE a la columna EMAIL de la tabla EMPLOYEES. El nombre de la restricción es EMP\_EMAIL\_UK.

**Nota:** Oracle Server fuerza la restricción UNIQUE creando implícitamente un índice único en la columna o las columnas de clave única.

# La Restricción PRIMARY KEY

## DEPARTMENTS

 PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...

No permitido  
(valor nulo)

 INSERT INTO

	Public Accounting		1400
50	Finance	124	1500

No permitido  
(50 ya existe)

ORACLE

10-11

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción PRIMARY KEY

Una restricción PRIMARY KEY crea una clave primaria para la tabla. Sólo se puede crear una clave primaria para cada tabla. La restricción PRIMARY KEY es una columna o un juego de columnas que identifica de forma única a cada fila de una tabla. Esta restricción fuerza la unicidad de la columna o de la combinación de columnas y asegura que ninguna columna que sea parte de la clave primaria puede contener un valor nulo.

# La Restricción PRIMARY KEY

Definida a nivel de tabla o de columna:

```
CREATE TABLE departments(  
    department_id      NUMBER(4),  
    department_name     VARCHAR2(30)  
        CONSTRAINT dept_name_nn NOT NULL,  
    manager_id         NUMBER(6),  
    location_id         NUMBER(4),  
    CONSTRAINT dept_id_pk PRIMARY KEY(department_id));
```

ORACLE

10-12

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción PRIMARY KEY (continuación)

Las restricciones PRIMARY KEY se pueden definir a nivel de columna o de tabla. Se crea una clave primaria compuesta utilizando la definición a nivel de tabla.

Una tabla puede tener una sola restricción PRIMARY KEY pero varias restricciones UNIQUE.

En el ejemplo de la transparencia se define una restricción PRIMARY KEY en la columna DEPARTMENT\_ID de la tabla DEPARTMENTS. El nombre de la restricción es DEPT\_ID\_PK.

**Nota:** Se crea un índice UNIQUE automáticamente para una columna PRIMARY KEY.

# La Restricción FOREIGN KEY

## DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

**PRIMARY  
KEY** →

...

## EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

...

← **FOREIGN  
KEY**



**INSERT INTO**

200	Ford	9
201	Ford	60

← **No permitido  
(9 no existe)**

← **Permitido**

ORACLE

10-13

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción FOREIGN KEY

La restricción FOREIGN KEY, o restricción de integridad referencial, designa una columna o una combinación de columnas como clave ajena y establece una relación entre una clave primaria o una clave única en la misma tabla o en una diferente. En el ejemplo de la transparencia, DEPARTMENT\_ID se ha definido como la clave ajena en la tabla EMPLOYEES (tabla dependiente o secundaria); hace referencia a la columna DEPARTMENT\_ID de la tabla DEPARTMENTS (la tabla a la que se hace referencia o principal).

Un valor de clave ajena debe coincidir con un valor existente en la tabla principal o ser NULL.

Las claves ajenas se basan en valores de datos y son punteros meramente lógicos, no físicos.

# La Restricción FOREIGN KEY

Definida a nivel de tabla o de columna:

```
CREATE TABLE employees(  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary            NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    department_id    NUMBER(4),  
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)  
        REFERENCES departments(department_id),  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

10-14

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## La Restricción FOREIGN KEY (continuación)

Las restricciones FOREIGN KEY se pueden definir a nivel de restricción de tabla o de columna. Se debe crear una clave ajena compuesta utilizando la definición a nivel de tabla.

En el ejemplo de la transparencia se define una restricción FOREIGN KEY en la columna DEPARTMENT\_ID de la tabla EMPLOYEES, utilizando sintaxis a nivel de tabla. El nombre de la restricción es EMP\_DEPTID\_FK.

La clave ajena también se puede definir a nivel de columna, siempre que la restricción se base en una sola columna. La sintaxis se diferencia en que no aparecen las palabras clave FOREIGN KEY. Por ejemplo:

```
CREATE TABLE employees  
(  
    ...  
    department_id NUMBER(4) CONSTRAINT emp_deptid_fk  
        REFERENCES departments(department_id),  
    ...  
)
```

## Palabras Clave de la Restricción FOREIGN KEY

- **FOREIGN KEY:** Define la columna de la tabla secundaria a nivel de restricción de tabla.
- **REFERENCES:** Identifica la tabla y la columna en la tabla principal.
- **ON DELETE CASCADE:** Suprime las filas dependientes de la tabla secundaria cuando se suprime una fila en la tabla principal.
- **ON DELETE SET NULL:** Convierte los valores de clave ajena dependientes en valores nulos.

ORACLE

10-15

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### La Restricción FOREIGN KEY (continuación)

La clave ajena se define en la tabla secundaria y la tabla que contiene la columna a la que se hace referencia es la tabla principal. La clave ajena se define utilizando una combinación de las siguientes palabras clave:

- **FOREIGN KEY** se utiliza para definir la columna de la tabla secundaria a nivel de restricción de tabla.
- **REFERENCES** identifica la tabla y la columna en la tabla principal.
- **ON DELETE CASCADE** indica que, cuando se suprime la fila de la tabla principal, también se suprimirán las filas dependientes de la tabla secundaria.
- **ON DELETE SET NULL** convierte valores de clave ajena en nulos cuando se elimina el valor principal.

El comportamiento por defecto se llama regla de restricción, que no permite la actualización o la supresión de los datos a los que se hace referencia.

Sin las opciones **ON DELETE CASCADE** u **ON DELETE SET NULL**, la fila de la tabla principal no se puede suprimir si se hace referencia a ella en la tabla secundaria.

## La Restricción CHECK

- Define una condición que debe satisfacer cada fila.
- No se permiten las siguientes expresiones:
  - Referencias a las pseudocolumnas CURRVAL, NEXTVAL, LEVEL y ROWNUM
  - Llamadas a las funciones SYSDATE, UID, USER y USERENV
  - Consultas que hagan referencia a otros valores de otras filas

```
..., salary NUMBER(2)
      CONSTRAINT emp_salary_min
      CHECK (salary > 0),...
```

ORACLE

10-16

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### La Restricción CHECK

La restricción CHECK define una condición que debe satisfacer cada fila. La condición puede utilizar las mismas construcciones como condiciones de consulta, con las siguientes excepciones:

- Referencias a las pseudocolumnas CURRVAL, NEXTVAL, LEVEL y ROWNUM
- Llamadas a las funciones SYSDATE, UID, USER y USERENV
- Consultas que hagan referencia a otros valores de otras filas

Una sola columna puede tener varias restricciones CHECK que hacen referencia a la columna en su definición. No hay límite al número de restricciones CHECK que puede definir en una columna.

Se pueden definir restricciones CHECK a nivel de columna o de tabla.

```
CREATE TABLE employees
(
  ...
  salary NUMBER(8,2) CONSTRAINT emp_salary_min
  CHECK (salary > 0),
  ...
)
```



# Adición de una Sintaxis de Restricción

Utilice la sentencia `ALTER TABLE` para:

- Agregar o borrar una restricción, sin modificar su estructura
- Activar o desactivar restricciones
- Agregar una restricción `NOT NULL` utilizando la cláusula `MODIFY`

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

ORACLE

10-17

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Adición de una Restricción

Puede agregar una restricción para tablas existentes utilizando la sentencia `ALTER TABLE` con la cláusula `ADD`.

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>constraint</i>	es el nombre de la restricción.
<i>type</i>	es el tipo de restricción.
<i>column</i>	es el nombre de la columna afectada por la restricción.

La sintaxis de nombre de restricción es opcional, aunque se recomienda. Si no asigna un nombre a las restricciones, lo hará el sistema.

### Instrucciones

- Puede agregar, borrar, activar o desactivar una restricción, pero no modificar su estructura.
- Puede agregar una restricción `NOT NULL` a una columna existente utilizando la cláusula `MODIFY` de la sentencia `ALTER TABLE`.

**Nota:** Puede definir una columna `NOT NULL` sólo si la tabla está vacía o si la columna tiene un valor para cada fila.

## Adición de una Restricción

**Agregue una restricción FOREIGN KEY a la tabla EMPLOYEES que indique que ya debe existir un director como empleado válido en la tabla EMPLOYEES.**

```
ALTER TABLE      employees
ADD CONSTRAINT    emp_manager_fk
    FOREIGN KEY(manager_id)
    REFERENCES employees(employee_id);
Table altered.
```

ORACLE

### Adición de una Restricción (continuación)

En el ejemplo de la transparencia se crea una restricción FOREIGN KEY en la tabla EMPLOYEES. La restricción asegura que existe un director como empleado válido en la tabla EMPLOYEES.

## Eliminación de una Restricción

- Elimine la restricción de director de la tabla **EMPLOYEES**.

```
ALTER TABLE      employees
DROP CONSTRAINT    emp_manager_fk;
Table altered.
```

- Elimine la restricción **PRIMARY KEY** de la tabla **DEPARTMENTS** y borre la restricción **FOREIGN KEY** asociada en la columna **EMPLOYEES.DEPARTMENT\_ID**.

```
ALTER TABLE      departments
DROP PRIMARY KEY CASCADE;
Table altered.
```

ORACLE

10-19

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Eliminación de una Restricción

Para borrar una restricción, puede identificar el nombre de la misma desde las vistas del diccionario de datos **USER\_CONSTRAINTS** y **USER\_CONS\_COLUMNS**. A continuación, utilice la sentencia **ALTER TABLE** con la cláusula **DROP**. La opción **CASCADE** de la cláusula **DROP** hace que todas las restricciones dependientes también se borren.

#### Sintaxis

```
ALTER TABLE table
DROP PRIMARY KEY | UNIQUE (column) |
CONSTRAINT constraint [CASCADE];
```

En la sintaxis:

<i>table</i>	es el nombre de la tabla.
<i>column</i>	es el nombre de la columna afectada por la restricción.
<i>constraint</i>	es el nombre de la restricción.

Al borrar una restricción de integridad, ésta ya no estará forzada por Oracle Server ni estará disponible en el diccionario de datos.

# Desactivación de Restricciones

- Ejecute la cláusula **DISABLE** de la sentencia **ALTER TABLE** para desactivar una restricción de integridad.
- Aplique la opción **CASCADE** para desactivar restricciones de integridad dependientes.

```
ALTER TABLE      employees
DISABLE CONSTRAINT emp_emp_id_pk CASCADE;
Table altered.
```

ORACLE

10-20

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Desactivación de una Restricción

Puede desactivar una restricción sin borrarla ni volver a crearla utilizando la sentencia **ALTER TABLE** con la cláusula **DISABLE**.

### Sintaxis

```
ALTER TABLE table
DISABLE CONSTRAINT constraint [CASCADE];
```

En la sintaxis:

*table* es el nombre de la tabla.

*constraint* es el nombre de la restricción.

### Instrucciones

- Puede utilizar la cláusula **DISABLE** tanto en la sentencia **CREATE TABLE** como en la sentencia **ALTER TABLE**.
- La cláusula **CASCADE** desactiva restricciones de integridad dependientes.
- La desactivación de una restricción de clave primaria o única elimina el índice único.

# Activación de Restricciones

- **Active una restricción de integridad desactivada actualmente en la definición de tabla utilizando la cláusula `ENABLE`.**

```
ALTER TABLE      employees
ENABLE CONSTRAINT  emp_emp_id_pk;
Table altered.
```

- **Se crea un índice `UNIQUE` o `PRIMARY KEY` automáticamente si activa una restricción de clave `UNIQUE` o `PRIMARY KEY`.**

ORACLE

10-21

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Activación de una Restricción

Puede activar una restricción sin borrarla ni volver a crearla utilizando la sentencia `ALTER TABLE` con la cláusula `ENABLE`.

### Sintaxis

```
ALTER   TABLE      table
ENABLE  CONSTRAINT  constraint;
```

En la sintaxis:

*table*                      es el nombre de la tabla.  
*constraint*                es el nombre de la restricción.

### Instrucciones

- Si activa una restricción, se aplica a todos los datos de la tabla. Todos los datos de la tabla deben ajustarse a la restricción.
- Si activa una restricción de clave `UNIQUE` o `PRIMARY KEY`, se crea automáticamente un índice `UNIQUE` o `PRIMARY KEY`.
- Puede utilizar la cláusula `ENABLE` tanto en la sentencia `CREATE TABLE` como en la sentencia `ALTER TABLE`.
- La activación de una restricción de clave primaria desactivada con la opción `CASCADE` no activa ninguna clave ajena dependiente de la primaria.

## Restricciones en Cascada

- La cláusula **CASCADE CONSTRAINTS** se utiliza junto con la cláusula **DROP COLUMN**.
- La cláusula **CASCADE CONSTRAINTS** borra todas las restricciones de integridad referenciales que hacen referencia a las claves primaria y única definidas en las columnas borradas.
- La cláusula **CASCADE CONSTRAINTS** también borra todas las restricciones de varias columnas definidas en las columnas borradas.

ORACLE

10-22

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Restricciones en Cascada

Esta sentencia ilustra el uso de la cláusula **CASCADE CONSTRAINTS**. Suponga que la tabla **TEST1** se crea como sigue:

```
CREATE TABLE test1 (  
    pk NUMBER PRIMARY KEY,  
    fk NUMBER,  
    col1 NUMBER,  
    col2 NUMBER,  
    CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES test1,  
    CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),  
    CONSTRAINT ck2 CHECK (col2 > 0));
```

Se devuelve un error para las siguientes sentencias:

```
ALTER TABLE test1 DROP (pk);    -- pk es una clave principal.  
ALTER TABLE test1 DROP (col1);  -- se hace referencia a col1 por la restricción ck1  
                                  de varias columnas.
```

# Restricciones en Cascada

## Ejemplo:

```
ALTER TABLE test1  
DROP (pk) CASCADE CONSTRAINTS;  
Table altered.
```

```
ALTER TABLE test1  
DROP (pk, fk, coll) CASCADE CONSTRAINTS;  
Table altered.
```

ORACLE

10-23

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Restricciones en Cascada (continuación)

Al ejecutar la siguiente sentencia se borra la columna PK, la restricción de clave primaria, la restricción de clave ajena `fk_constraint` y la restricción de control, CK1:

```
ALTER TABLE test1 DROP (pk) CASCADE CONSTRAINTS;
```

Si todas las columnas a las que se hace referencia por las restricciones definidas en las columnas borradas también se borran, no se requiere `CASCADE CONSTRAINTS`. Por ejemplo, suponiendo que ninguna otra restricción referencial procedente de otras tablas hace referencia a la columna PK, es válido ejecutar la siguiente sentencia sin la cláusula `CASCADE CONSTRAINTS`:

```
ALTER TABLE test1 DROP (pk, fk, coll);
```

## Visualización de Restricciones

Consulte la tabla `USER_CONSTRAINTS` para visualizar todas las definiciones y los nombres de restricciones.

```
SELECT  constraint_name, constraint_type,
        search_condition
FROM    user_constraints
WHERE   table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	C	SEARCH_CONDITION
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL
EMP_SALARY_MIN	C	salary > 0
EMP_EMAIL_UK	U	

...

ORACLE

10-24

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Visualización de Restricciones

Después de crear una tabla, puede confirmar su existencia emitiendo un comando `DESCRIBE`. La única restricción que puede verificar es la restricción `NOT NULL`. Para visualizar todas las restricciones de la tabla, consulte la tabla `USER_CONSTRAINTS`.

En el ejemplo de la transparencia se muestran las restricciones de la tabla `EMPLOYEES`.

**Nota:** Las restricciones a las que el propietario de la tabla no asigna ningún nombre, se lo asigna el sistema. En tipo de restricción, C significa `CHECK`; P, `PRIMARY KEY`; R, integridad referencial; y U, clave `UNIQUE`. Observe que la restricción `NOT NULL` es realmente una restricción `CHECK`.



## Visualización de las Columnas Asociadas a Restricciones

Visualice las columnas asociadas a los nombres de restricción en la vista `USER_CONS_COLUMNS`.

```
SELECT  constraint_name, column_name
FROM    user_cons_columns
WHERE   table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPT_FK	DEPARTMENT_ID
EMP_EMAIL_NN	EMAIL
EMP_EMAIL_UK	EMAIL
EMP_EMP_ID_PK	EMPLOYEE_ID
EMP_HIRE_DATE_NN	HIRE_DATE
EMP_JOB_FK	JOB_ID
EMP_JOB_NN	JOB_ID

...

ORACLE

### Visualización de Restricciones (continuación)

Puede visualizar los nombres de las columnas implicadas en restricciones consultando la vista `USER_CONS_COLUMNS` del diccionario de datos. Esta vista resulta especialmente útil para restricciones que utilizan nombres asignados por el sistema.

# Resumen

**En esta lección, debería haber aprendido a crear restricciones.**

- **Tipos de restricción:**
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
- **Puede consultar la tabla `USER_CONSTRAINTS` para visualizar todas las definiciones y los nombres de restricciones.**

ORACLE

10-26

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

## Resumen

En esta lección, debería haber aprendido cómo utiliza Oracle Server restricciones para evitar la entrada de datos no válidos en las tablas. También debería haber aprendido a implementar las restricciones en sentencias DDL.

Son válidos los siguientes tipos de restricción:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Puede consultar la tabla `USER_CONSTRAINTS` para visualizar todas las definiciones y los nombres de restricciones.

## Visión General de la Práctica 10

Esta práctica cubre los siguientes temas:

- Adición de restricciones a tablas existentes
- Adición de más columnas a una tabla
- Visualización de información en vistas del diccionario de datos

ORACLE

10-27

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

### Visión General de la Práctica 10

En esta práctica, agregará restricciones y más columnas a una tabla utilizando las sentencias que se cubren en esta lección.

**Nota:** Se recomienda que asigne nombre a las restricciones que defina durante las prácticas.

## Práctica 10

1. Agregue una restricción `PRIMARY KEY` de nivel de tabla a la tabla `EMP` en la columna `ID`. Debe asignar el nombre `my_emp_id_pk` a la restricción en el momento de su creación.

**Indicación:** La restricción se activa tan pronto como se ejecuta el comando `ALTER TABLE` correctamente.

2. Cree una restricción `PRIMARY KEY` en la tabla `DEPT` utilizando la columna `ID`. Debe asignar el nombre `my_dept_id_pk` a la restricción en el momento de su creación.

**Indicación:** La restricción se activa tan pronto como se ejecuta el comando `ALTER TABLE` correctamente.

3. Agregue una columna `DEPT_ID` a la tabla `EMP`. Agregue una referencia de clave ajena en la tabla `EMP` que asegure que no se asigna el empleado a un departamento no existente. Llame a la restricción `my_emp_dept_id_fk`.

4. Confirme que se agregaron las restricciones consultando la vista `USER_CONSTRAINTS`. Anote los tipos y los nombres de las restricciones. Guarde el texto de sentencias en un archivo llamado `lab10_4.sql`.

CONSTRAINT_NAME	C
MY_DEPT_ID_PK	P
SYS_C002541	C
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

5. Visualice los nombres y los tipos de objeto de la vista `USER_OBJECTS` del diccionario de datos para las tablas `EMP` y `DEPT`. Observe que se crearon las nuevas tablas y un nuevo índice.

Si tiene tiempo, complete el siguiente ejercicio:

6. Modifique la tabla `EMP`. Agregue una columna `COMMISSION` del tipo de dato `NUMBER`, precisión 2, escala 2. Agregue una restricción a la columna `COMMISSION` que asegure que el valor de la comisión sea mayor que cero.