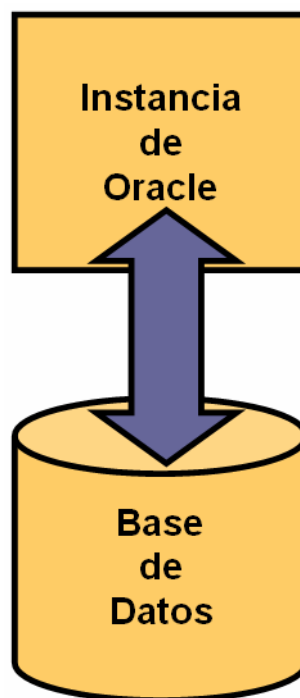


Oracle 9i

SQL & PL/SQL



Elaborado por: Gustavo Coronel

Febrero - 2005

ORACLE 9i – SQL & PL/SQL

Esta obra es de uso exclusivo del CEPS-UNI para el curso
ORACLE 9i – Nivel Inicial, esta prohibido su uso para otros fines.

Derechos Reservados © 2004 Eric Gustavo Coronel Castillo
Primera Edición

Presentación

Oracle es sin duda una de las mejores bases de datos que tenemos en el mercado, tiene muchas características que nos garantizan la seguridad e integridad de los datos; que las transacciones se efectuarán de manera correcta, sin causar inconsistencias; desarrollo en la capa de datos utilizando: procedimientos, funciones, desencadenantes, y paquetes; y el procesamiento de grandes volúmenes de información estará también asegurada.

Este manual esta compuesto por 12 lecciones, donde veremos de una manera práctica el lenguaje SQL y la programación con PL/SQL, no pretende ser un texto de consulta teórica, sino más bien, una guía de práctica de laboratorio.

Esta manual lo desarrolle para que sea usado exclusivamente en el curso **ORACLE 9i Nivel Inicial** que el CEPS-UNI ofrece en forma libre ó como parte de la carrera técnica **ORACLE DATABASE ADMINISTRATOR**.

Sería ingrato no mencionar los aportes de mis colegas Sergio Matsukawa, Ricardo Marcelo, Fortunato Veliz y Hugo Valencia, sin duda alguna que muchas de sus ideas y ejemplos están plasmados en este manual.

Como parte de mi esfuerzo por escribir mejores libros y manuales les agradecería me envíen sus comentarios a mi correo: gcoronel@viabcp.com, me sería de mucha utilidad conocer sus opiniones para poder mejorar mis futuras publicaciones.

Eric Gustavo Coronel Castillo

Contenido

Lección 01: Aspectos Generales de Oracle 9i

Introducción	2
Arquitectura de un servidor Oracle 9i	3
La instancia de Oracle	3
Conexión con una instancia de Oracle	6
Conceptos generales de almacenamiento.....	11

Lección 02: Esquemas Ejemplos de la Base de Datos

Esquema de Base de Datos	2
Esquema SCOTT	2
Esquema HR	3
Consultar la Estructura de una Tabla	5
Consultar el Contenido de una Tabla	5

Lección 03: Sentencias SQL Select Básicas

SQL Fundamentos.....	2
Escribiendo Consultas Simples	5
Otros Operadores	11
Ordenando Filas	13
Usando Expresiones.....	15

Lección 04: Funciones Simples de Fila

Funciones para Valores Nulos.....	2
Funciones para Caracteres	3
Funciones Numéricas	5
Funciones de Fecha	6
Funciones de Conversión	8
Otras Funciones	11

Lección 05: Totalizando Datos y Funciones de Grupo

Funciones de Grupo	2
GROUP BY.....	4
HAVING	5

Lección 06: Consultas Multitablas

¿Qué es un Join?	2
Consultas Simples	2
Consultas Complejas.....	3
Producto Cartesiano.....	7
Combinaciones Externas.....	8
Otras Consultas Multitablas.....	11
Operadores de Conjuntos.....	12

Lección 07: Subconsultas

Subconsultas de Solo una Fila	2
Subconsultas de Múltiples Filas	2
Subconsultas Correlacionadas	3
Subconsultas Escalares	3

Lección 08: Modificando Datos

Insertando Filas	2
Modificando Datos	5
Eliminando Filas	9
Transacciones	12

Lección 09: Creación de un Esquema de Base de Datos

Caso a Desarrollar	2
Creación del Usuario para el Esquema	3
Creación de Tablas	4
Restricción Primary Key (PK)	6
Restricción Foreign Key (FK)	8
Restricción Default (Valores por Defecto)	10
Restricción NOT NULL (Nulidad de una Columna)	11
Restricción Unique (Valores Únicos)	12
Restricción Check (Reglas de Validación)	13
Asignar Privilegios a Usuarios	14

Lección 10: PL/SQL - Fundamentos

Introducción a PL/SQL	2
Tipos de Datos y Variables	4
Estructuras de Control	6
Bucles	11
Otros Elementos de Programación	15

Lección 11: PL/SQL – Trabajando con Datos

Registros	2
SQL en PL/SQL	4
Cursores	12

Lección 12: PL/SQL – Tópicos Adicionales

Tratamiento de Errores	2
Procedimientos	8
Funciones	9
Parámetros	10
Paquetes	12
Desencadenantes	14

Oracle 9i Básico PL/SQL

Lección 01

Aspectos Generales de Oracle 9i

Contenido

Introducción	2
¿Qué es una base de datos?	2
¿Qué es un DBMS?	2
Tipos de bases de datos	2
Arquitectura de un servidor Oracle 9i	3
Esquema General	3
La instancia de Oracle	3
Procesos de fondo	4
Area Global del Sistema (SGA)	4
La base de datos	5
Estructuras Adicionales	5
Conexión con una instancia de Oracle	6
Verificación de los servicios	6
Esquema General	7
Conexión local utilizando SQL Plus	8
Vistas del Sistema	9
Comandos SQL/Plus	9
Conexión remota utilizando SQL Plus	10
Conceptos generales de almacenamiento	11
TableSpace	11
DataFile	12

Introducción

¿Qué es una base de datos?

Colección o depósito de datos integrados, almacenados en soporte secundario (no volátil) y con redundancia controlada.

La estructura de la base de datos debe responder a las necesidades del mundo real, en cuanto a sus interrelaciones y restricciones.

¿Qué es un DBMS?

Es el software que contiene una colección ordenada y sincronizada de programas, procedimientos y lenguajes, que permiten gestionar una base de datos.

Tipos de bases de datos

Desde el punto de vista de organización lógica:

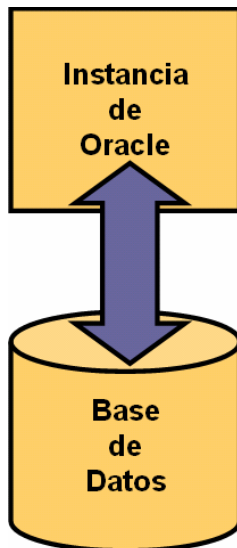
- a. Jerárquicas
- b. Relacionales (Oracle, SQL Server, DB2, Sybase, etc.)

Desde el punto de vista de números de usuarios:

- a. Mono usuarios
- b. Multiusuarios

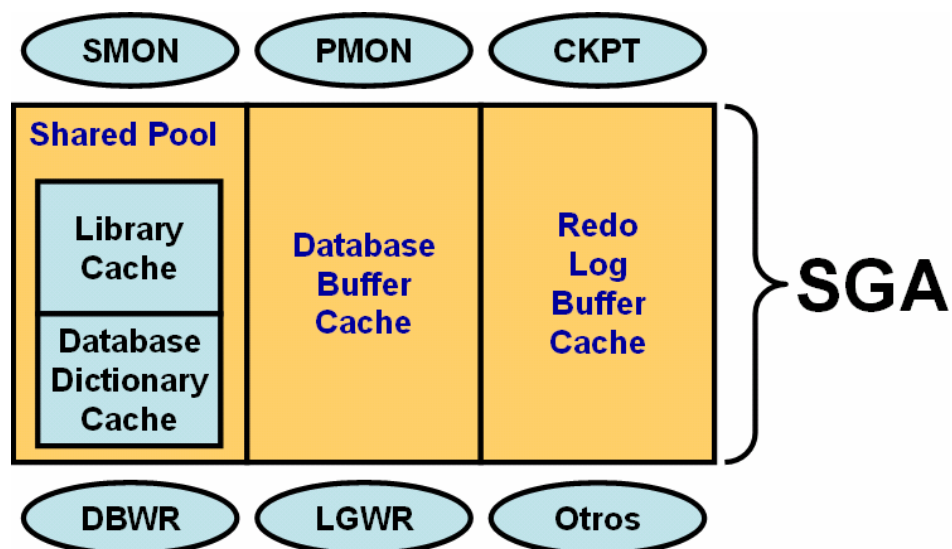
Arquitectura de un servidor Oracle 9i

Esquema General



- Por cada instancia de Oracle se tiene una sola base de datos
- En un servidor se pueden crear varias instancias, pero se recomienda solo una, por que cada instancia consume muchos recursos.

La instancia de Oracle

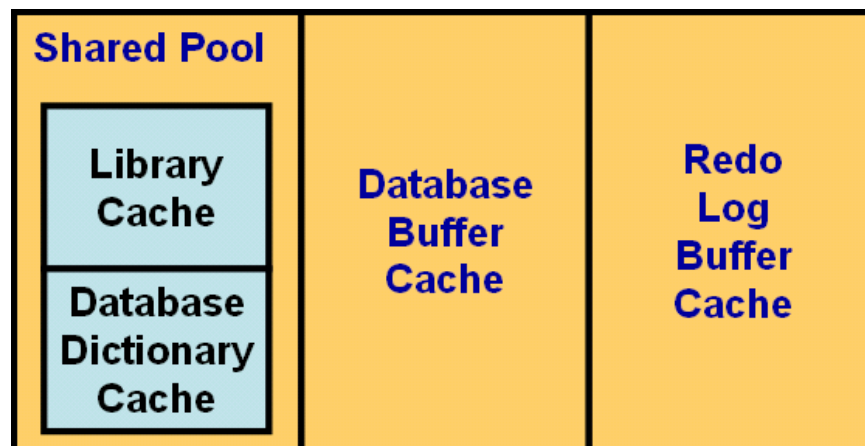


- Esta compuesta por procesos de fondo y un área de memoria compartida denominada SYSTEM GLOBAL AREA (SGA).
- El SGA es utilizado para el intercambio de datos entre el servidor y las aplicaciones cliente.
- Una instancia de Oracle solo puede abrir una sola base de datos a la vez.

Procesos de fondo

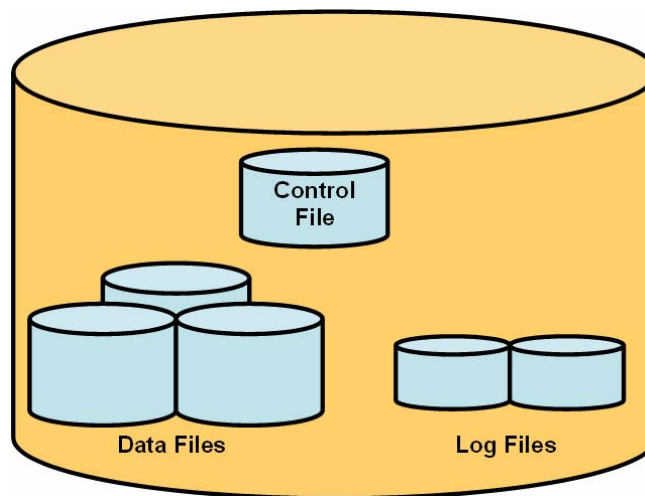
PMON	<i>Process Monitor.</i> Monitorea los procesos de los usuarios, en caso que la conexión falle.
SMON	<i>System Monitor.</i> Este proceso es el encargado de recuperar la instancia y abrir la base de datos, en caso que ocurra alguna falla.
CKPT	<i>CheckPoint Process.</i> Sintoniza las tareas de grabación en la base de datos.
DBWR	<i>Database Writer.</i> Escribe los bloques de datos de la memoria a la base de datos.
LGWR	<i>Log Writer.</i> Graba los bloques del Redo Log del buffer a los archivos Redo Log File.

Area Global del Sistema (SGA)



Library Cache	Almacena las sentencias SQL más recientes en memoria.
Database Dictionary Cache	Buffer para el diccionario de datos. Tablas, columnas, tipos, índices.
Database Buffer Cache	Buffer de la base de datos, contiene bloques de datos que han sido cargados desde los Data File.
Redo Log Buffer Cache	Bloques de datos que han sido actualizados.

La base de datos



Control File	Contiene información para mantener y controlar la integridad de la base de datos.
Data Files	Son los archivos donde se almacenan los datos de las aplicaciones.
Redo Log Files	Almacena los cambios hechos en la base de datos con propósito de recuperarlos en caso de falla.

Estructuras Adicionales

Archivo de Parámetros	Contiene parámetros y valores que definen las características de la instancia y de la base de datos, por ejemplo contiene parámetros que dimensionan el SGA.
Archivo de Password	Se utiliza para validar al usuario que puede bajar y subir la instancia de Oracle.
Archivos Archived Log Files	Los Archived Log Files son copias fuera de línea de los archivos Redo Log Files que son necesarios para el proceso de Recovery en caso de falla del medio de almacenamiento.

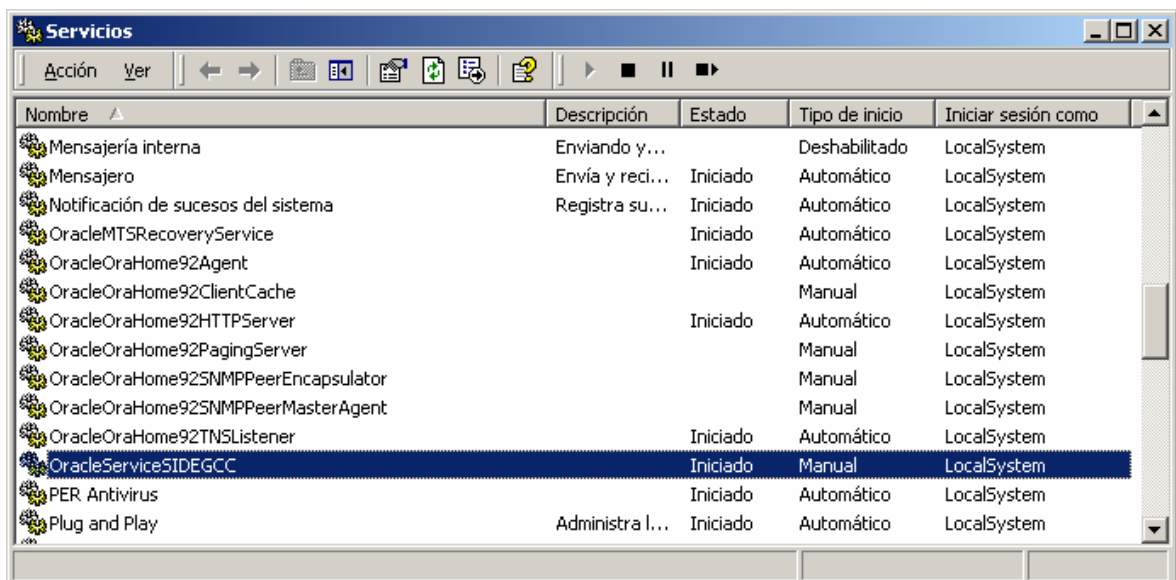
Conexión con una instancia de Oracle

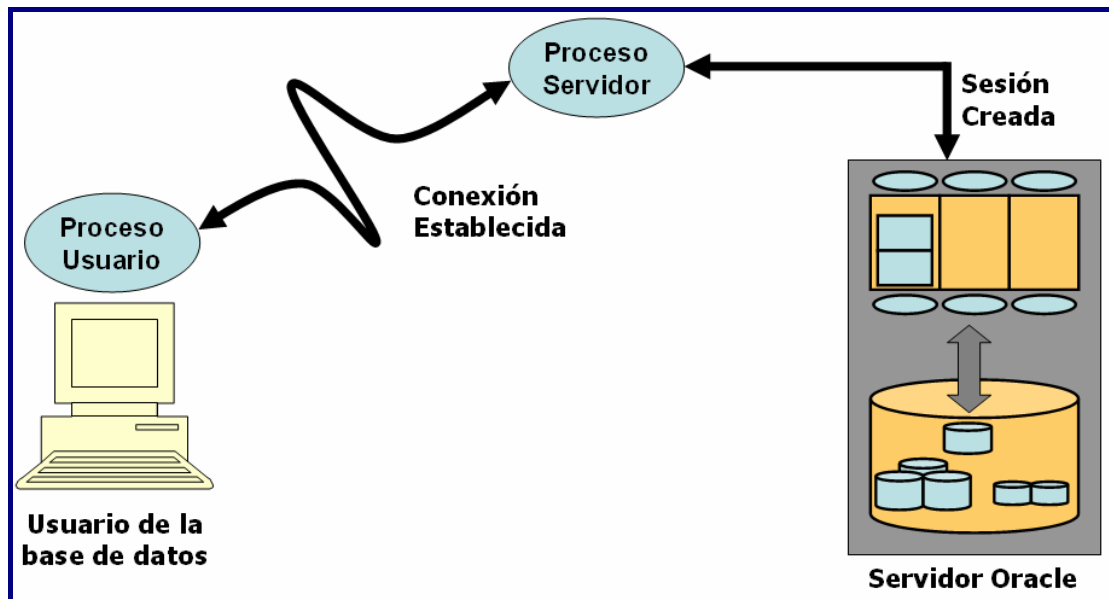
Verificación de los servicios

De la relación de servicios creados durante la instalación de Oracle, por ahora nos interesa básicamente dos:

- El servicio relacionado con la instancia y la base de datos, cuyo nombre tiene la siguiente estructura: OracleServiceXXX, donde XXX representa el nombre de la instancia. Por ejemplo, si la instancia tiene por nombre SIDEGCC, el servicio sería OracleServiceSIDEGCC.
- El servicio relacionado con la disponibilidad del servidor para el acceso remoto, el nombre de este servicio es: OracleOraHome92TNSListener.

Estos dos servicios deben estar ejecutándose, y su verificación se puede realizar en la venta de servicios, a la que accedemos desde el Panel de control / Herramientas administrativas.



Esquema General

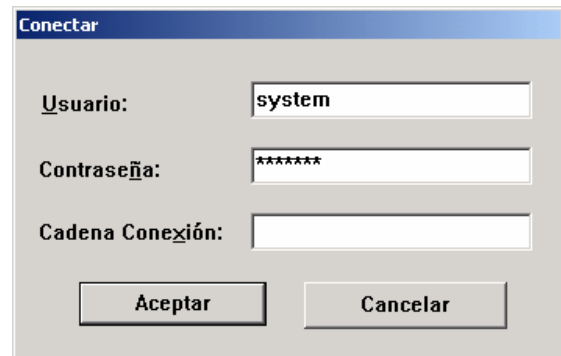
Proceso Usuario	Programa, aplicación ó herramienta que usa el usuario para iniciar un proceso de usuario y establecer una conexión.
Proceso Servidor	<p>Una ves que el proceso de usuario establece la conexión, un proceso servidor es iniciado, el cual manejará las peticiones del proceso usuario.</p> <p>Un proceso servidor puede ser dedicado, es decir solo atiende las peticiones de un solo proceso usuario, ó puede se compartido, con lo cual puede atender múltiples procesos usuarios.</p>
Sesión	<p>Una sesión es una conexión específica de un usuario a un servidor Oracle.</p> <ul style="list-style-type: none">• Se inicia cuando el usuario es validado por el servidor Oracle.• Finaliza cuando el usuario termina la sesión en forma normal (logout) ó aborta la sesión.

Conexión local utilizando SQL Plus

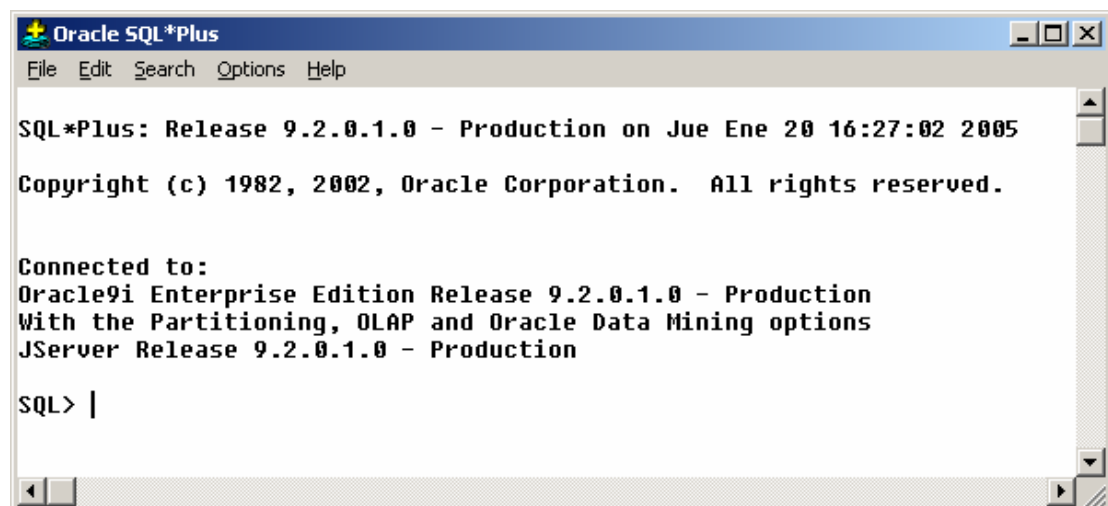
SQL Plus es una herramienta que permite al usuario comunicarse con el servidor, para procesar comandos SQL ó PL/SQL, tiene la flexibilidad de poder realizar inicio y parada (shutdown) de la base de datos.

En la ventana inicial de conexión debemos ingresar el usuario y su contraseña, por ejemplo podemos usar:

Usuario	Contraseña
system	manager
scott	tiger



La pantalla de bienvenida de SQL Plus mostrará los siguientes mensajes:



En estos momentos estamos listos para trabajar, por ejemplo si queremos conectarnos como scout, el comando es el siguiente:

```
SQL> connect scott/tiger
Conectado.

SQL>
```

Vistas del Sistema

Tenemos algunas vistas que podemos consultar para verificar nuestro servidor: v\$instance, v\$database y v\$sga.

Para realizar las consultas a las vistas, ejecutamos los siguientes comandos:

```
SQL> connect system/manager
Conectado.

SQL> select instance_name from v$instance;

INSTANCE_NAME
-----
sidegcc

SQL> select name from v$database;

NAME
-----
DBEGCC

SQL> select * from v$sga;

NAME                                VALUE
-----
Fixed Size                          282576
Variable Size                       83886080
Database Buffers                   33554432
Redo Buffers                       532480
```

Comandos SQL/Plus

También contamos con comandos SQL/Plus, algunos de ellos son:

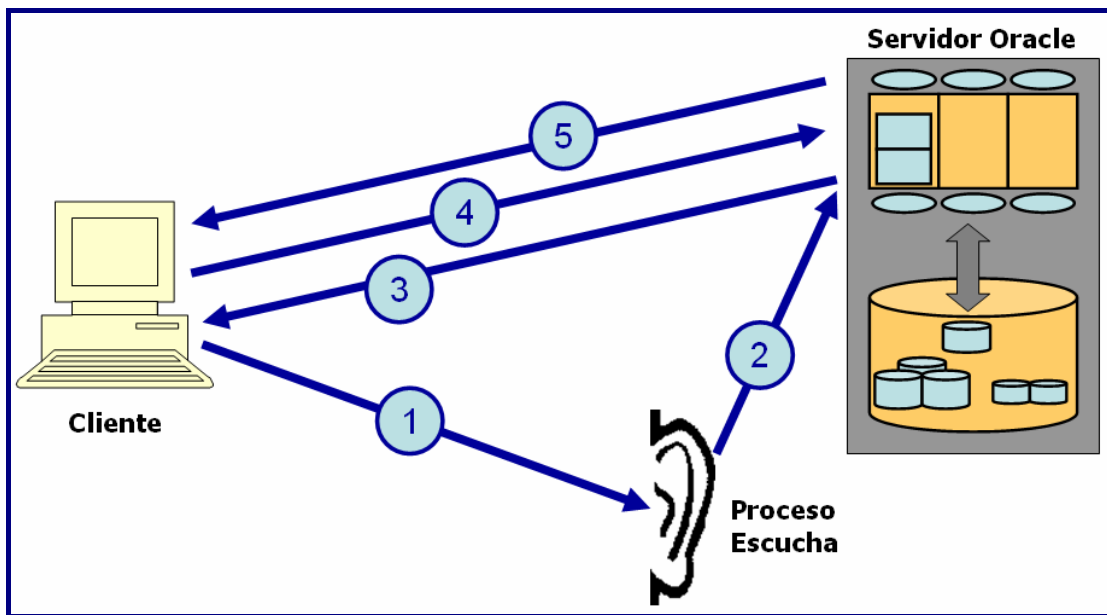
- RUN** Vuelve a ejecutar la última instrucción ejecutada.
- EDIT** Edita la última instrucción ejecutada.
- START** Ejecuta las instrucciones que se encuentran en un archivo.
- SPOOL** Envía la sesión de trabajo a un archivo.

Conexión remota utilizando SQL Plus

Oracle tiene su herramienta de red que permite a las aplicaciones en general conectarse a servidores Oracle. El nombre inicial de esta herramienta fue SQL*Net, luego fue renombrada con el nombre Net8, y hoy día se le conoce como Oracle Net.

Para que una aplicación pueda conectarse remotamente a un servidor Oracle, es necesario que el *Proceso Escucha* se encuentre ejecutándose en el servidor, específicamente el servicio OracleOraHome92TNSListener.

El esquema general de la conexión remota se puede apreciar en el siguiente gráfico.



El proceso se describe a continuación:

1. El cliente establece una conexión al Proceso Escucha usando el protocolo configurado y envía un paquete CONNECT.
2. El proceso escucha comprueba que el SID esté definido. Si es así, generará un nuevo proceso para ocuparse de la conexión. Una conexión se establece entre el proceso escucha y el nuevo proceso del servidor para pasarle la información del proceso de inicialización. Luego la conexión es cerrada.
3. El proceso del servidor envía un paquete al cliente.
4. Un nuevo paquete CONNECT es enviado al proceso servidor dedicado.
5. El proceso de servidor dedicado acepta la conexión entrante y remite un mensaje de ACEPTADO al nuevo al cliente.

Conceptos generales de almacenamiento

TableSpace

Unidad lógica en que se divide una base de datos. Es posible consultar los tablespace utilizando los siguientes comandos:

```
SQL> select * from v$tablespace
2  order by 1;

      TS# NAME                                INC
-----
0 SYSTEM                                YES
1 UNDOTBS1                             YES
2 TEMP                                 YES
3 CWM Lite                             YES
4 DRSYS                                YES
5 EXAMPLE                              YES
6 INDX                                 YES
7 ODM                                  YES
8 TOOLS                                YES
9 USERS                                YES
10 XDB                                 YES

11 rows selected.

SQL> select tablespace_name from dba_tablespaces
2  order by 1;

TABLESPACE_NAME
-----
CWM Lite
DRSYS
EXAMPLE
INDX
ODM
SYSTEM
TEMP
TOOLS
UNDOTBS1
USERS
XDB

11 rows selected.
```

DataFile

Es el archivo físico donde se almacenas los datos.

```
SQL> column file_name format a40

SQL> select tablespace_name, file_name from dba_data_files
       2 order by 1;
```

TABLESPACE_NAME	FILE_NAME
CWMLITE	E:\ORACLE\ORADATA\DBEGCC\CWMLITE01.DBF
DRSYS	E:\ORACLE\ORADATA\DBEGCC\DRSYS01.DBF
EXAMPLE	E:\ORACLE\ORADATA\DBEGCC\EXAMPLE01.DBF
INDX	E:\ORACLE\ORADATA\DBEGCC\INDX01.DBF
ODM	E:\ORACLE\ORADATA\DBEGCC\ODM01.DBF
SYSTEM	E:\ORACLE\ORADATA\DBEGCC\SYSTEM01.DBF
TOOLS	E:\ORACLE\ORADATA\DBEGCC\TOOLS01.DBF
UNDOTBS1	E:\ORACLE\ORADATA\DBEGCC\UNDOTBS01.DBF
USERS	E:\ORACLE\ORADATA\DBEGCC\USERS01.DBF
XDB	E:\ORACLE\ORADATA\DBEGCC\XDB01.DBF

10 rows selected.

Oracle 9i Básico PL/SQL

Lección 02 Esquemas Ejemplos de la Base de Datos

Contenido

Esquema de Base de Datos.....	2
Esquema SCOTT	2
Esquema HR	3
Consultar la Estructura de una Tabla	5
Consultar el Contenido de una Tabla.....	5

Esquema de Base de Datos

El conjunto de objetos que tiene una cuenta de usuario se denomina *esquema* del usuario, por lo tanto el nombre del esquema será también el nombre del usuario.

Cuando creamos la base de datos de Oracle, por defecto crea dos esquemas de ejemplo, para poder realizar nuestras pruebas.

Estos esquemas son los siguientes:

SCOTT Se trata de un esquema muy básico de recursos humanos, cuenta con tan solo 4 tablas.

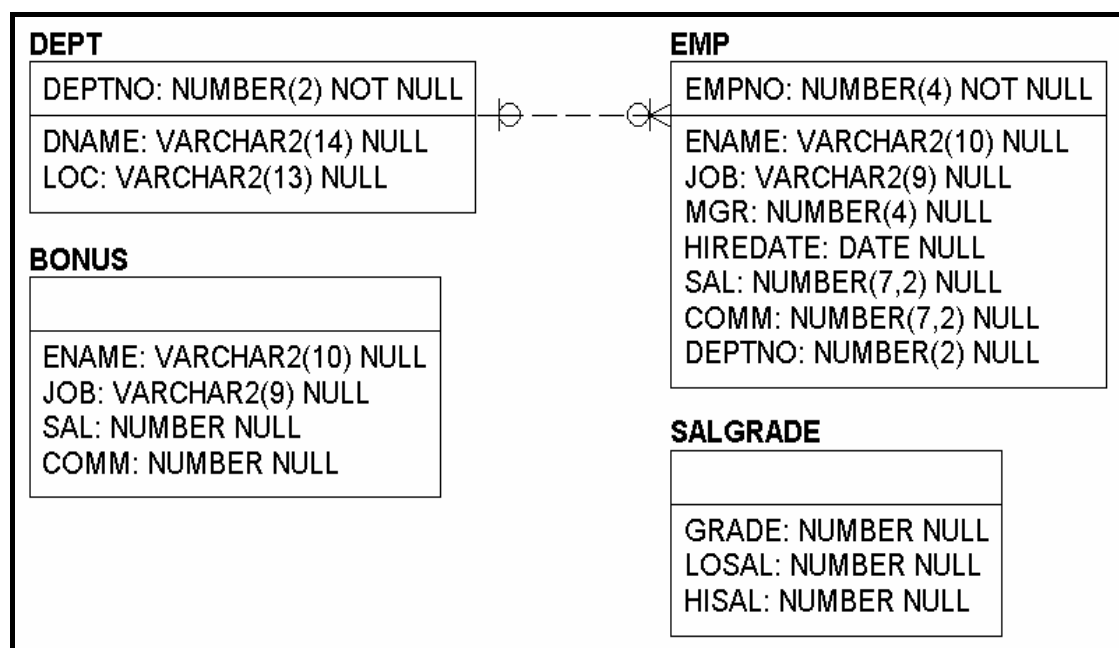
HR Se trata también de un esquema de recursos humanos, pero este esquema cuenta con 7 tablas.

Esquema SCOTT

Para poder iniciar una sesión en el esquema de scott debemos utilizar los siguientes datos:

Usuario	scott
Contraseña	tiger

Su esquema es el siguiente:



El siguiente script permite consultar el catalogo de scott:

Script 2.1

```
SQL> connect scott/tiger
Connected.
```

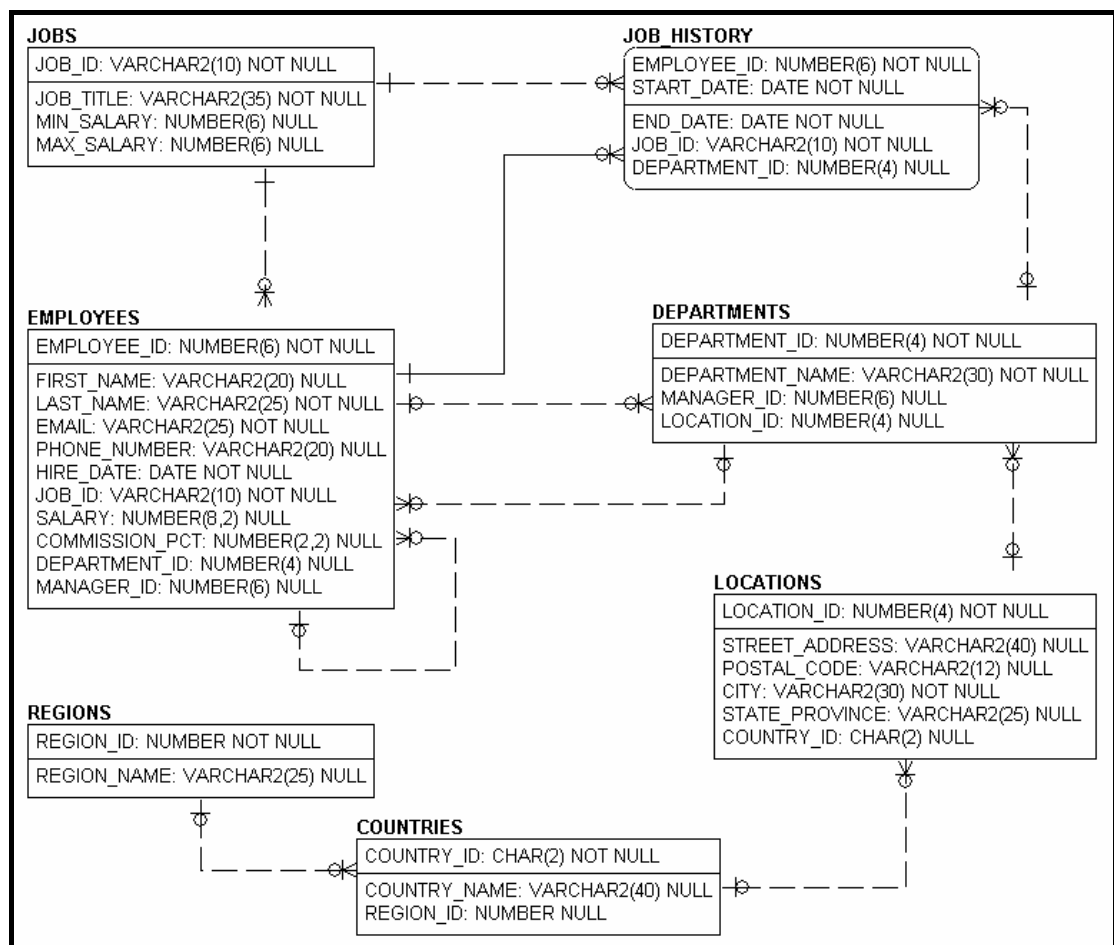
```
SQL> select * from cat;
```

TABLE_NAME	TABLE_TYPE
BONUS	TABLE
DEPT	TABLE
EMP	TABLE
SALGRADE	TABLE

4 rows selected.

Esquema HR

Su esquema es el siguiente:



La cuenta de usuario HR por defecto está bloqueada, así que lo primero que debemos hacer es desbloquearla, el script es el siguiente:

Script 2.2

```
SQL> connect system/manager
Connected.

SQL> alter user hr
  2  identified by hr
  3  account unlock;

User altered.
```

Ahora si podemos consultar el catalogo del esquema HR:

Script 2.3

```
SQL> connect hr/hr
Connected.

SQL> select * from cat;

TABLE_NAME                                TABLE_TYPE
-----
COUNTRIES                                TABLE
DEPARTMENTS                             TABLE
DEPARTMENTS_SEQ                          SEQUENCE
EMPLOYEES                                TABLE
EMPLOYEES_SEQ                            SEQUENCE
EMP_DETAILS_VIEW                         VIEW
JOBS                                     TABLE
JOB_HISTORY                              TABLE
LOCATIONS                                TABLE
LOCATIONS_SEQ                            SEQUENCE
REGIONS                                  TABLE

11 rows selected.
```

También podemos utilizar la siguiente consulta:

Script 2.4

```
SQL> select * from tab;

TNAME                                TABTYPE  CLUSTERID
-----
COUNTRIES                                TABLE
DEPARTMENTS                             TABLE
EMPLOYEES                                TABLE
EMP_DETAILS_VIEW                         VIEW
JOBS                                     TABLE
JOB_HISTORY                              TABLE
LOCATIONS                                TABLE
REGIONS                                  TABLE

8 rows selected.
```

Consultar la Estructura de una Tabla

Sintaxis

```
DESCRIBE Nombre_Tabla
```

Como ejemplo ilustrativo consultemos la estructura de la tabla EMP del esquema SCOTT:

Script 2.5

```
SQL> connect scott/tiger
Connected.

SQL> describe emp
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

Consultar el Contenido de una Tabla

Sintaxis

```
SELECT * FROM Nombre_Tabla
```

Como ejemplo ilustrativo consultemos el contenido de la tabla DEPT de SCOTT:

Script 2.6

```
SQL> select * from dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Página en Blanco

Oracle 9i Básico PL/SQL

Lección 03

Sentencias SQL SELECT Básicas

Contenido

SQL Fundamentos	2
Tipos de Datos de Oracle	3
Operadores y Literales	3
Operadores Aritméticos	3
Operador de Concatenación	3
Operadores de Conjuntos	4
Precedencia de Operadores	4
Literales	4
Escribiendo Consultas Simples	5
Usando la Sentencia SELECT	5
Consulta del contenido de una Tabla	5
Seleccionando Columnas	5
Alias para Nombres de Columnas	6
Asegurando Valores Únicos	6
La Tabla DUAL	7
Limitando las Filas	7
Operadores de Comparación	7
Operadores Lógicos	10
Otros Operadores	11
IN y NOT IN	11
BETWEEN	11
EXISTS	12
IS NULL y IS NOT NULL	12
LIKE	12
Ordenando Filas	13
Ordenando Nulos	14
Usando Expresiones	15
La Expresión CASE	15
Caso 1	15
Caso 2	15

SQL Fundamentos

Data Manipulation Language (DML)

Usado para acceder, crear, modificar, o eliminar data en una estructura de base de datos existente.

Data Definition Language (DDL)

Usado para crear, modificar, o eliminar objetos de base de datos y sus privilegios.

Transaction Control

Las instrucciones de control de transacciones garantizan la consistencia de los datos, organizando las instrucciones SQL en transacciones lógicas, que se completan o fallan como una sola unidad.

Session Control

Estas instrucciones permiten controlar las propiedades de sesión de un usuario. La sesión se inicia desde el momento en que el usuario se conecta a la base de datos hasta el momento en que se desconecta.

System Control

Usadas para manejar las propiedades de la base de datos.

Tipos de Datos de Oracle

Categoría	Tipos de Datos
Character	CHAR, NCHAR, VARCHAR2, NVARCHAR2
Number	NUMBER
Long and raw	LONG, LONG RAW, RAW
Date and time	DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIME STAMP WITH LOCAL TIME ZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
Large object	CLOB, NCLOB, BLOB, BFILE
Row ID	ROWID, UROWID

Operadores y Literales

Operadores Aritméticos

Operador	Propósito	Ejemplo
+ -	Operadores unarios: Usado para representar datos positivos y negativos. Para datos positivos, el + es opcional.	-234.56
+	Suma: Usado para sumar dos números o expresiones.	5 + 7
-	Resta: Usado para encontrar la diferencia entre dos números o expresiones.	56.8 - 18
*	Multiplicación: Usado para multiplicar dos números o expresiones.	7 * 15
/	División: Usado para dividir un número o expresión con otro.	8.67 / 3

Operador de Concatenación

Dos barras verticales (||) son usadas como operador de concatenación. La siguiente tabla muestra dos ejemplos.

Ejemplo	Resultado
'Alianza Lima' 'Campeón'	'Alianza LimaCampeón'
'Alianza Lima ' 'Campeón'	'Alianza Lima Campeón'

Operadores de Conjuntos

Estos Operadores son usados para combinar el resultado de dos consultas.

Operador	Propósito
UNION	Retorna todas las filas de cada consulta; no las filas duplicadas.
UNION ALL	Retorna todas las filas de cada consulta, incluyendo las filas duplicadas. no las filas duplicadas
INTERSECT	Retorna las filas distintas del resultado de cada consulta.
MINUS	Retorna las filas distintas que son retornadas por la primera consulta pero que no son retornadas por la segunda consulta.

Precedencia de Operadores

Precedencia	Operador	Propósito
1	- +	Operadores unarios, negación
2	* /	Multiplicación, división
3	+ -	Suma, resta, concatenación

Literales

Son valores que representan un valor fijo. Estos pueden ser de cuatro tipos diferentes:

Texto	<ul style="list-style-type: none">• 'CEPS-UNI'• 'Nos vemos en Peter's, la tienda del chino'• 'El curso es "Oracle", y lo dictan en CEPS-UNI'• '28-JUL-2006'
Entero	<ul style="list-style-type: none">• 45• -345
Número	<ul style="list-style-type: none">• 25• -456.78• 15E-15

Escribiendo Consultas Simples

Usando la Sentencia SELECT

Consulta del contenido de una Tabla

Script 3.1

```
SQL> conn hr/hr
Connected.

SQL> select * from jobs;

JOB_ID      JOB_TITLE                                MIN_SALARY  MAX_SALARY
-----
AD_PRES      President                                20000       40000
AD_VP        Administration Vice President           15000       30000
AD_ASST      Administration Assistant                3000        6000
FI_MGR       Finance Manager                        8200       16000
FI_ACCOUNT   Accountant                             4200        9000
... ..
IT_PROG      Programmer                             4000       10000
MK_MAN       Marketing Manager                       9000       15000
MK_REP       Marketing Representative                 4000        9000
HR_REP       Human Resources Representative           4000        9000
PR_REP       Public Relations Representative          4500       10500

19 rows selected.
```

Seleccionando Columnas

Script 3.2

```
SQL> select job_title, min_salary from jobs;

JOB_TITLE                                MIN_SALARY
-----
President                                20000
Administration Vice President           15000
Administration Assistant                3000
Finance Manager                        8200
Accountant                             4200
... ..
Programmer                             4000
Marketing Manager                       9000
Marketing Representative                 4000
Human Resources Representative           4000
Public Relations Representative          4500

19 rows selected.
```

Alias para Nombres de Columnas

Script 3.3

```
SQL> select job_title as Titulo,  
2 min_salary as "Salario Mínimo"  
3 from jobs;
```

TITULO	Salario Mínimo
President	20000
Administration Vice President	15000
Administration Assistant	3000
Finance Manager	8200
Accountant	4200
...	...
Programmer	4000
Marketing Manager	9000
Marketing Representative	4000
Human Resources Representative	4000
Public Relations Representative	4500

19 rows selected.

Asegurando Valores Únicos

Script 3.4

```
SQL> select distinct department_id from employees;
```

DEPARTMENT_ID
10
20
30
40
50
60
70
80
90
100
110

12 rows selected.

La Tabla DUAL

Script 3.5

```
SQL> select sysdate, user from dual;

SYSDATE  USER
-----
22/01/05 HR

SQL> select 'Yo soy ' || user || ' Hoy es ' || sysdate
2  from dual;

'YOSOY' || USER || 'HOYES' || SYSDATE
-----
Yo soy HR Hoy es 22/01/05
```

Limitando las Filas

Operadores de Comparación

Igualdad (=)

Script 3.6

```
SQL> select first_name || ' ' || last_name,
2  department_id
3  from employees
4  where department_id = 90;

FIRST_NAME || ' ' || LAST_NAME                                DEPARTMENT_ID
-----
Steven King                                                    90
Neena Kochhar                                                  90
Lex De Haan                                                    90
```

Diferente (!=, <>, ^=)

Script 3.7

```
SQL> select first_name || ' ' || last_name,
2  commission_pct
3  from employees
4  where commission_pct <> .35;

FIRST_NAME || ' ' || LAST_NAME                                COMMISSION_PCT
-----
John Russell                                                    ,4
Karen Partners                                                  ,3
Alberto Errazuriz                                              ,3
... ..
... ..
Jack Livingston                                                  ,2
Kimberely Grant                                                ,15
Charles Johnson                                                ,1

32 rows selected.
```

Menor Que (<)**Script 3.8**

```
SQL> select first_name || ' ' || last_name,  
2 commission_pct  
3 from employees  
4 where commission_pct < .15;
```

FIRST_NAME ' ' LAST_NAME	COMMISSION_PCT
Mattea Marvins	,1
David Lee	,1
Sundar Ande	,1
Amit Banda	,1
Sundita Kumar	,1
Charles Johnson	,1

6 rows selected.

Mayor Que (>)**Script 3.9**

```
SQL> select first_name || ' ' || last_name,  
2 commission_pct  
3 from employees  
4 where commission_pct > .35;
```

FIRST_NAME ' ' LAST_NAME	COMMISSION_PCT
John Russell	,4

Menor ó Igual Que (<=)**Script 3.10**

```
SQL> select first_name || ' ' || last_name,  
2 commission_pct  
3 from employees  
4 where commission_pct <= .15;
```

FIRST_NAME ' ' LAST_NAME	COMMISSION_PCT
Oliver Tuvault	,15
Danielle Greene	,15
Mattea Marvins	,1
David Lee	,1
Sundar Ande	,1
Amit Banda	,1
William Smith	,15
Elizabeth Bates	,15
Sundita Kumar	,1
Kimberely Grant	,15
Charles Johnson	,1

11 rows selected.

Mayor ó Igual Que (>=)**Script 3.11**

```
SQL> select first_name || ' ' || last_name,  
2 commission_pct  
3 from employees  
4 where commission_pct >= .35;
```

FIRST_NAME ' ' LAST_NAME	COMMISSION_PCT
John Russell	,4
Janette King	,35
Patrick Sully	,35
Allan McEwen	,35

ANY ó SOME**Script 3.12**

```
SQL> select first_name || ' ' || last_name,  
2 department_id  
3 from employees  
4 where department_id <= ANY (10,15,20,25);
```

FIRST_NAME ' ' LAST_NAME	DEPARTMENT_ID
Jennifer Whalen	10
Michael Hartstein	20
Pat Fay	20

ALL**Script 3.13**

```
SQL> select first_name || ' ' || last_name,  
2 department_id  
3 from employees  
4 where department_id >= ALL (80,90,100);
```

FIRST_NAME ' ' LAST_NAME	DEPARTMENT_ID
Nancy Greenberg	100
Daniel Faviat	100
John Chen	100
Ismael Sciarra	100
Jose Manuel Urman	100
Luis Popp	100
Shelley Higgins	110
William Gietz	110

8 rows selected.

Operadores Lógicos

NOT

Script 3.14

```
SQL> select first_name, department_id
  2   from employees
  3   where not (department_id >= 30);
```

FIRST_NAME	DEPARTMENT_ID
Jennifer	10
Michael	20
Pat	20

AND

Script 3.15

```
SQL> select first_name, salary
  2   from employees
  3   where last_name = 'Smith'
  4   and salary > 7500;
```

FIRST_NAME	SALARY
Lindsey	8000

OR

Script 3.16

```
SQL> select first_name, last_name
  2   from employees
  3   where first_name = 'Kelly'
  4   or last_name = 'Smith';
```

FIRST_NAME	LAST_NAME
Lindsey	Smith
William	Smith
Kelly	Chung

Otros Operadores

IN y NOT IN

Script 3.17

```
SQL> select first_name, last_name, department_id
  2   from employees
  3   where department_id in (10, 20, 90);
```

FIRST_NAME	LAST_NAME	DEPARTMENT_ID
Steven	King	90
Neena	Kochhar	90
Lex	De Haan	90
Jennifer	Whalen	10
Michael	Hartstein	20
Pat	Fay	20

6 rows selected.

```
SQL> select first_name, last_name, department_id
  2   from employees
  3   where department_id not in (10, 30, 40, 50, 60, 80, 90, 110, 100);
```

FIRST_NAME	LAST_NAME	DEPARTMENT_ID
Michael	Hartstein	20
Pat	Fay	20
Hermann	Baer	70

BETWEEN

Script 3.18

```
SQL> select first_name, last_name, salary
  2   from employees
  3   where salary between 5000 and 6000;
```

FIRST_NAME	LAST_NAME	SALARY
Bruce	Ernst	6000
Kevin	Mourgos	5800
Pat	Fay	6000

EXISTS

Script 3.19

```
SQL> select first_name, last_name, department_id
2   from employees e
3   where exists (select 1 from departments d
4                 where d.department_id = e.department_id
5                 and d.department_name = 'Administration');

FIRST_NAME      LAST_NAME      DEPARTMENT_ID
-----
Jennifer         Whalen         10
```

IS NULL y IS NOT NULL

Script 3.20

```
SQL> select last_name, department_id
2   from employees
3   where department_id is null;

LAST_NAME      DEPARTMENT_ID
-----
Grant
```

LIKE

Script 3.21

```
SQL> select first_name, last_name
2   from employees
3   where first_name like 'Su%'
4   and last_name not like 'S%';

FIRST_NAME      LAST_NAME
-----
Sundar          Ande
Sundita         Kumar
Susan           Mavris
```

Ordenando Filas

Script 3.22

```
SQL> select first_name, last_name
  2   from employees
  3   where department_id = 90
  4   order by first_name;

FIRST_NAME          LAST_NAME
-----
Lex                 De Haan
Neena               Kochhar
Steven              King

SQL> select first_name || ' ' || last_name "Employee Name"
  2   from employees
  3   where department_id = 90
  4   order by last_name;

Employee Name
-----
Lex De Haan
Steven King
Neena Kochhar

SQL> select first_name, hire_date, salary, manager_id mid
  2   from employees
  3   where department_id in (110,100)
  4   order by mid asc, salary desc, hire_date;

FIRST_NAME          HIRE_DAT      SALARY        MID
-----
Shelley             07/06/94      12000         101
Nancy               17/08/94      12000         101
Daniel             16/08/94      9000          108
John               28/09/97      8200          108
Jose Manuel        07/03/98      7800          108
Ismael             30/09/97      7700          108
Luis               07/12/99      6900          108
William            07/06/94      8300          205

8 rows selected.

SQL> select distinct 'Region ' || region_id
  2   from countries
  3   order by 'Region ' || region_id;

'REGION' || REGION_ID
-----
Region 1
Region 2
Region 3
Region 4
```

```
SQL> select first_name, hire_date, salary, manager_id mid
2   from employees
3   where department_id in (110,100)
4   order by 4, 2, 3;
```

FIRST_NAME	HIRE_DATE	SALARY	MID
Shelley	07/06/94	12000	101
Nancy	17/08/94	12000	101
Daniel	16/08/94	9000	108
John	28/09/97	8200	108
Ismael	30/09/97	7700	108
Jose Manuel	07/03/98	7800	108
Luis	07/12/99	6900	108
William	07/06/94	8300	205

8 rows selected.

Ordenando Nulos

Script 3.23

```
SQL> select last_name, commission_pct
2   from employees
3   where last_name like 'A%'
4   order by commission_pct asc;
```

LAST_NAME	COMMISSION_PCT
Ande	,1
Abel	,3
Austin	
Atkinson	

```
SQL> select last_name, commission_pct
2   from employees
3   where last_name like 'A%'
4   order by commission_pct asc nulls first;
```

LAST_NAME	COMMISSION_PCT
Austin	
Atkinson	
Ande	,1
Abel	,3

Usando Expresiones

La Expresión CASE

Caso 1

Formato

```
CASE <expresión>
  WHEN <Valor1> THEN <Valor de Retorno 1>
  WHEN <Valor2> THEN <Valor de Retorno 2>
  WHEN <Valor3> THEN <Valor de Retorno 3>
  . . .
  . . .
  [ELSE <Valor de Retorno>]
END
```

Script 3.24

```
SQL> select country_name, region_id,
2      case region_id
3          when 1 then 'Europa'
4          when 2 then 'America'
5          when 3 then 'Asia'
6          else 'Otro'
7      end as continente
8 from countries
9 where country_name like 'I%';
```

COUNTRY_NAME	REGION_ID	CONTINE
Israel	4	Otro
India	3	Asia
Italy	1	Europa

Caso 2

Formato

```
CASE
  WHEN <Condición1> THEN <Valor de Retorno 1>
  WHEN <Condición2> THEN <Valor de Retorno 2>
  WHEN <Condición3> THEN <Valor de Retorno 3>
  . . .
  . . .
  [ELSE <Valor de Retorno>]
END
```

Script 3.25

```
SQL> select first_name, department_id, salary,  
2      case  
3          when salary < 6000 then 'Bajo'  
4          when salary < 10000 then 'Regular'  
5          when salary >= 10000 then 'Alto'  
6      end as Categoria  
7 from employees  
8 where department_id <= 30  
9 order by first_name;
```

FIRST_NAME	DEPARTMENT_ID	SALARY	CATEGOR
Alexander	30	3100	Bajo
Den	30	11000	Alto
Guy	30	2600	Bajo
Jennifer	10	4400	Bajo
Karen	30	2500	Bajo
Michael	20	13000	Alto
Pat	20	6000	Regular
Shelli	30	2900	Bajo
Sigal	30	2800	Bajo

9 rows selected.

Oracle 9i Básico PL/SQL

Lección 04 Funciones Simples de Fila

Contenido

Funciones para Valores Nulos	2
Funciones NVL	2
Función NVL2	3
Funciones para Caracteres	3
Funciones Numéricas	5
Funciones de Fecha	6
Conversión de Formato de Fecha	6
Add_Months	6
Current_Date	6
Current_Timestamp	7
Extract	7
Last_Day	7
Month_Between	8
SysDate	8
Funciones de Conversión	8
Cast	8
To_Char	9
Conversión de Datos Tipo Fecha	9
Conversión de Datos Numéricos	9
To_Date	10
To_Number	10
Otras Funciones	11
NULLIF	11
Sys_Connect_By_Path	12
Sys_Context	12
UID	13
User	13

Funciones para Valores Nulos

Funciones NVL

Remplaza un valor nulo por otro valor.

Script 4.1

```
SQL> conn scott/tiger
Connected.

SQL> select ename, sal, comm, (sal + comm) as neto
2   from emp;
```

ENAME	SAL	COMM	NETO
SMITH	800		
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		
MARTIN	1250	1400	2650
BLAKE	2850		
CLARK	2450		
SCOTT	3000		
KING	5000		
TURNER	1500	0	1500
ADAMS	1100		
JAMES	950		
FORD	3000		
MILLER	1300		

```
14 rows selected.

SQL> select ename, sal, comm,
2   sal + nvl(comm,0) as neto
3   from emp;
```

ENAME	SAL	COMM	NETO
SMITH	800		800
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		2975
MARTIN	1250	1400	2650
BLAKE	2850		2850
CLARK	2450		2450
SCOTT	3000		3000
KING	5000		5000
TURNER	1500	0	1500
ADAMS	1100		1100
JAMES	950		950
FORD	3000		3000
MILLER	1300		1300

```
14 rows selected.
```

Función NVL2

Remplaza un valor nulo por otro valor, si no es nulo también lo remplaza por otro valor diferente.

Script 4.2

```
SQL> select ename, sal, comm,
2      nvl2(comm, sal + comm, sal) as neto
3      from emp;
```

ENAME	SAL	COMM	NETO
SMITH	800		800
ALLEN	1600	300	1900
WARD	1250	500	1750
JONES	2975		2975
MARTIN	1250	1400	2650
BLAKE	2850		2850
CLARK	2450		2450
SCOTT	3000		3000
KING	5000		5000
TURNER	1500	0	1500
ADAMS	1100		1100
JAMES	950		950
FORD	3000		3000
MILLER	1300		1300

14 rows selected.

Funciones para Caracteres

Función	Descripción	Ejemplo
ASCII	Retorna el valor ASCII equivalente de un carácter.	Ascii('A') = 65
CHR	Retorna el carácter determinado por el valor ASCII equivalente.	Chr(65) = A
CONCAT	Concatena dos cadena; equivalente al operador .	concat('Gustavo','Coronel') = GustavoCoronel
INITCAP	Retorna la cadena con la primera letra de cada palabra en mayúscula.	InitCaP('PACHERREZ') = Pacherrez
INSTR	Busca la posición de inicio de una cadena dentro de otra.	Instr('Mississippi','i') = 2 Instr('Mississippi','s',5) = 6 Instr('Mississippi','i',3,2) = 8
INSTRB	Similar a INSTR, pero cuenta bytes en lugar de caracteres.	InstrB('Mississippi','i') = 2 InstrB('Mississippi','s',5) = 6 InstrB('Mississippi','i',3,2) = 8
LENGTH	Retorna la longitud de una cadena en caracteres.	Length('Oracle is Powerful') = 18

Función	Descripción	Ejemplo
LENGTHB	Retorna la longitud de una cadena en bytes.	LengthB('Oracle is Powerful') = 18
LOWER	Convierte una cadena a minúsculas.	Lower('CHICLAYO') = chiclayo
LPAD	Ajustada a la derecha una cadena, rellenándola a la izquierda con otra cadena.	LPad('56.78',8,'#') = ###56.78
LTRIM	Elimina caracteres a la izquierda de una cadena, por defecto espacios en blanco.	LTrim(' Alianza') = Alianza LTrim('Mississippi','Mis') = ppi
RPAD	Ajustada a la izquierda una cadena, rellenándola a la derecha con otra cadena.	RPad('56.78',8,'#') = 56.78###
RTRIM	Elimina caracteres a la derecha de una cadena, por defecto espacios en blanco.	RTrim('Real ') 'Madrid' = RealMadrid RTrim('Mississippi','ip') = Mississ
REPLACE	Permite reemplaza parte de una cadena.	Replace('PagDown','Down','Up') = PagUp
SUBSTR	Permite extraer parte de una cadena.	SubStr('Trujillo',4,2) = ji
SUBSTRB	Similar a SUBSTR, pero la posición se indica en bytes.	SubStrB('Trujillo',4,2) = ji
SOUNDEX	Retorna la representación fonética de una cadena.	Soundex('HOLA') = H400
TRANSLATE	Reemplaza caracteres de una cadena por otros caracteres.	Translate('Lorena','orn','unr') = Lunera
TRIM	Elimina espacios en blanco a ambos lados de una cadena.	'Alianza' Trim(' ES ') 'Alianza' = AlianzaESAlianza
UPPER	Convierte a mayúsculas una cadena.	Upper('peru') = PERU

Script 4.3

```
SQL> select initcap( first_name || ' ' || last_name)
2   from employees
3   where department_id = 30;

INITCAP(FIRST_NAME||' '||LAST_NAME)
-----
Den Raphaely
Alexander Khoo
Shelli Baida
Sigal Tobias
Guy Himuro
Karen Colmenares

6 rows selected.
```

Funciones Numéricas

Función	Descripción	Ejemplo
ABS	Retorna el valor absoluto de un valor.	$\text{Abs}(-5) = 5$
ACOS	Retorna el arco coseno.	$\text{ACos}(-1) = 3.14159265$
ASIN	Retorna el arco seno.	$\text{ASin}(1) = 1.57079633$
ATAN	Retorna el arco tangente.	$\text{ATan}(0) = 0$
ATAN2	Retorna el arco tangente; tiene dos valores de entrada.	$\text{ATan2}(0, 3.1415) = 0$
BITAND	Retorna el resultado de una comparación a nivel de bits de números.	$\text{BitAnd}(3, 9) = 1$
CEIL	Retorna el siguiente entero más alto.	$\text{Ceil}(5.1) = 6$
COS	Retorna el coseno de un ángulo.	$\text{Cos}(0) = 1$
COSH	Retorna el coseno hiperbólico.	$\text{Cosh}(1.4) = 2.15089847$
EXP	Retorna la base del logaritmo natural elevado a una potencia.	$\text{Exp}(1) = 2.71828183$
FLOOR	Retorna el siguiente entero más pequeño.	$\text{Floor}(5.31) = 5$
LN	Retorna el logaritmo natural.	$\text{Ln}(2.7) = 0.99325177$
LOG	Retorna el logaritmo.	$\text{Log}(8, 64) = 2$
MOD	Retorna el residuo de una operación de división.	$\text{Mod}(13, 5) = 3$
POWER	Retorna un número elevado a una potencia.	$\text{Power}(2, 3) = 8$
ROUND	Redondea un número.	$\text{Round}(5467, -2) = 5500$ $\text{Round}(56.7834, 2) = 56.78$
SIGN	Retorna el indicador de signo de un número.	$\text{Sign}(-456) = -1$
SIN	Retorna el seno de un ángulo.	$\text{Sin}(0) = 0$
SQRT	Retorna el seno hiperbólico.	$\text{Sqrt}(16) = 4$
TAN	Retorna la tangente de un ángulo.	$\text{Tan}(0.785398165) = 1$
TANH	Retorna la tangente hiperbólica.	$\text{Tanh}(\text{Acos}(-1)) = 0.996272076$
TRUNC	Trunca un número.	$\text{Trunc}(456.678, 2) = 456.67$ $\text{Trunc}(456.678, -1) = 450$

Funciones de Fecha

Estableciendo el Formato de Fecha

Script 4.4

```
SQL> alter session set nls_date_format='DD-Mon-YYYY HH24:MI:SS';  
Session altered.
```

Add_Months

Adiciona un número de meses a una fecha.

Script 4.5

```
SQL> select  
2      sysdate as Hoy,  
3      add_months(sysdate,3) as TresMesesDespues  
4  from dual;  
  
HOY                                TRESMESESDESPUES  
-----  
26-Ene-2005 15:15:13 26-Abr-2005 15:15:13
```

Current_Date

Retorna la fecha actual.

Script 4.6

```
SQL> select current_date from dual;  
  
CURRENT_DATE  
-----  
26-Ene-2005 15:16:15
```

Current_Timestamp

Retorna la fecha y hora actual.

Script 4.7

```
SQL> select current_timestamp from dual;

CURRENT_TIMESTAMP
-----
26/01/05 03:17:41,394000 PM -05:00
```

Extract

Extrae y retorna un componente de una expresión Date/Time.

Script 4.8

```
SQL> select sysdate as Hoy, extract(year from sysdate) as Año
       2 from dual;

HOY                                AÑO
-----
26-Ene-2005 15:20:48              2005

SQL> select sysdate as Hoy, extract(month from sysdate) as Mes
       2 from dual;

HOY                                MES
-----
26-Ene-2005 15:21:38              1

SQL> select sysdate as Hoy, extract(day from sysdate) as Día
       2 from dual;

HOY                                DÍA
-----
26-Ene-2005 15:22:27              26
```

Last_Day

Retorna el último día del mes.

Script 4.9

```
SQL> select sysdate as Hoy,
       2         last_day(sysdate) as fin_del_mes,
       3         last_day(sysdate) + 1 as proximo_mes
       4 from dual;

HOY                                FIN_DEL_MES                                PROXINO_MES
-----
26-Ene-2005 15:33:48 31-Ene-2005 15:33:48 01-Feb-2005 15:33:48
```

Month_Between

Retorna el número de meses entre dos fechas.

Script 4.10

```
SQL> select months_between('19-Abr-2005','19-Dic-2004')
       2   from dual;

MONTHS_BETWEEN('19-ABR-2005','19-DIC-2004')
-----
4
```

SysDate

Retorna la fecha y hora actual.

Script 4.11

```
SQL> select sysdate from dual;

SYSDATE
-----
26-Ene-2005 15:57:42
```

Funciones de Conversión

Cast

Convierte una expresión a un tipo de dato específico.

Script 4.12

```
SQL> select cast(sysdate as varchar2(24)) from dual;

CAST(SYSDATEASVARCHAR2(2
-----
26-Ene-2005 17:46:02
```


To_Char

Convierte un dato tipo fecha ó número a una cadena con un formato específico.

Conversión de Datos Tipo Fecha

Script 4.13

```
SQL> select to_char(sysdate, 'Day, Month YYYY', 'NLS_DATE_LANGUAGE=English')
       2 from dual;

TO_CHAR(SYSDATE, 'DAY, MONT
-----
Wednesday, January    2005

SQL> select to_char(sysdate, 'Day, Month YYYY', 'NLS_DATE_LANGUAGE=Spanish')
       2 from dual;

TO_CHAR(SYSDATE, 'DAY, MONTH
-----
Miércoles, Enero      2005

SQL> Select to_char(sysdate, 'Day, DD "de" MONTH "de" YYYY')
       2 from dual;

TO_CHAR(SYSDATE, 'DAY, DD"DE"MONTH"DE
-----
Miércoles, 26 de ENERO      de 2005
```

Conversión de Datos Numéricos

Script 4.14

```
SQL> select to_char(15.6789, '99,999.00')
       2 from dual;

TO_CHAR(15
-----
      15.68

SQL> select to_char(45.78234, '00,000.00')
       2 from dual;

TO_CHAR(45
-----
    00,045.78

SQL> select to_char(346.4567, 'L99,999.00')
       2 from dual;

TO_CHAR(346.4567, 'L9
-----
      S/346.46
```

To_Date

Convierte una cadena con una fecha a un dato de tipo fecha.

Script 4.15

```
SQL> select to_date('15-01-2005','DD-MM-YYYY')
       2   from dual;

TO_DATE('15
-----
15-Ene-2005
```

To_Number

Convierte una cadena numérica a su respectivo valor numérico.

Script 4.16

```
SQL> select to_number('15.45','999.99') from dual;

TO_NUMBER('15.45','999.99')
-----
                        15,45
```

Otras Funciones

NULLIF

Compara dos expresiones expr1 y expr2, si ambas son iguales retorna NULL, de lo contrario retorna expr1. expr1 no puede ser el literal NULL.

Script 4.17

```
SQL> connect scott/tiger
Connected.
```

```
SQL> select ename, mgr, comm,
2      NULLIF(comm,0) test1,
3      NULLIF(0, comm) test2,
4      NULLIF(mgr,comm) test3
5 from emp
6 where empno in (7844,7839, 7654, 7369);
```

ENAME	MGR	COMM	TEST1	TEST2	TEST3
SMITH	7902			0	7902
MARTIN	7698	1400	1400	0	7698
KING				0	
TURNER	7698	0			7698

Sys_Connect_By_Path

SYS_CONNECT_BY_PATH es válido solamente en consultas jerárquicas. Devuelve la trayectoria de una columna desde el nodo raíz, con los valores de la columna separados por un carácter para cada fila devuelta según la condición especificada en CONNECT BY.

Script 4.18

```
SQL> connect hr/hr
Connected.

SQL> column path format a40

SQL> select last_name, sys_connect_by_path(last_name, '/') Path
   2   from employees
   3   start with last_name = 'Kochhar'
   4   connect by prior employee_id = manager_id;

LAST_NAME          PATH
-----
Kochhar            /Kochhar
Greenberg          /Kochhar/Greenberg
Faviet             /Kochhar/Greenberg/Faviet
Chen               /Kochhar/Greenberg/Chen
Sciarra            /Kochhar/Greenberg/Sciarra
Urman              /Kochhar/Greenberg/Urman
Popp               /Kochhar/Greenberg/Popp
Whalen             /Kochhar/Whalen
Mavris             /Kochhar/Mavris
Baer               /Kochhar/Baer
Higgins            /Kochhar/Higgins
Gietz              /Kochhar/Higgins/Gietz

12 rows selected.
```

Sys_Context

Retorna el parámetro asociado con un namespace.

Script 4.19

```
SQL> select sys_context('USERENV','HOST') from dual;

SYS_CONTEXT('USERENV','HOST')
-----
TECHSOFT\EGCC
```

UID

Devuelve un número entero que identifique únicamente a cada usuario.

Script 4.20

```
SQL> select uid from dual;

      UID
-----
      46
```

User

Retorna el nombre del usuario de la sesión actual

Script 4.21

```
SQL> select user from dual;

USER
-----
HR
```

Página en Blanco

Oracle 9i Básico PL/SQL

Lección 05 Totalizando Datos y Funciones de Grupo

Contenido

Funciones de Grupo	2
AVG	2
COUNT	2
MAX	2
MIN	3
SUM	3
GROUP BY	4
HAVING	5

Funciones de Grupo

AVG

Obtiene el promedio de una columna o expresión. Se puede aplicar la cláusula DISTINCT.

Script 5.1

```
SQL> connect hr/hr
Connected.

SQL> select avg(salary) from employees
       2  where department_id = 30;

AVG(SALARY)
-----
        4150
```

COUNT

Cuenta las filas de una consulta. Se puede aplicar DISTINCT.

Script 5.2

```
SQL> select count(*) from departments;

COUNT(*)
-----
        27

SQL> select count(distinct department_id) from employees;

COUNT(DISTINCTDEPARTMENT_ID)
-----
        11
```

MAX

Retorna el máximo valor de una columna ó expresión.

Script 5.3

```
SQL> select max(salary) from employees
       2  where department_id = 80;

MAX(SALARY)
-----
       14000
```


MIN

Retorna el mínimo valor de una columna ó expresión.

Script 5.4

```
SQL> select min(salary) from employees
      2  where department_id = 80;

MIN(SALARY)
-----
        6100
```

SUM

Retorna la suma de los valores de una columna. Se puede aplicar DISTINCT.

Script 5.5

```
SQL> select sum(salary) from employees
      2  where department_id = 80;

SUM(SALARY)
-----
       304500
```

GROUP BY

Se utiliza para agrupar data en base a una ó más columnas, para aplicar funciones de grupo.

Script 5.6

Cantidad de empleados por departamento.

```
SQL> select
  2     department_id as Departamento,
  3     count(*) as Empleados
  4   from employees
  5   group by department_id;
```

DEPARTAMENTO	EMPLEADOS
10	1
20	2
30	6
40	1
50	45
60	5
70	1
80	34
90	3
100	6
110	2
	1

12 rows selected.

Script 5.7

Cantidad de empleados por puesto de trabajo en los departamentos 50 y 80.

```
SQL> select
  2     department_id as Departamento,
  3     job_id as puesto,
  4     count(*) as Empleados
  5   from employees
  6   where department_id in (50,80)
  7   group by department_id, job_id;
```

DEPARTAMENTO	PUESTO	EMPLEADOS
50	ST_MAN	5
50	SH_CLERK	20
50	ST_CLERK	20
80	SA_MAN	5
80	SA_REP	29

Script 5.8

Cantidad de empleados que han ingresado por año.

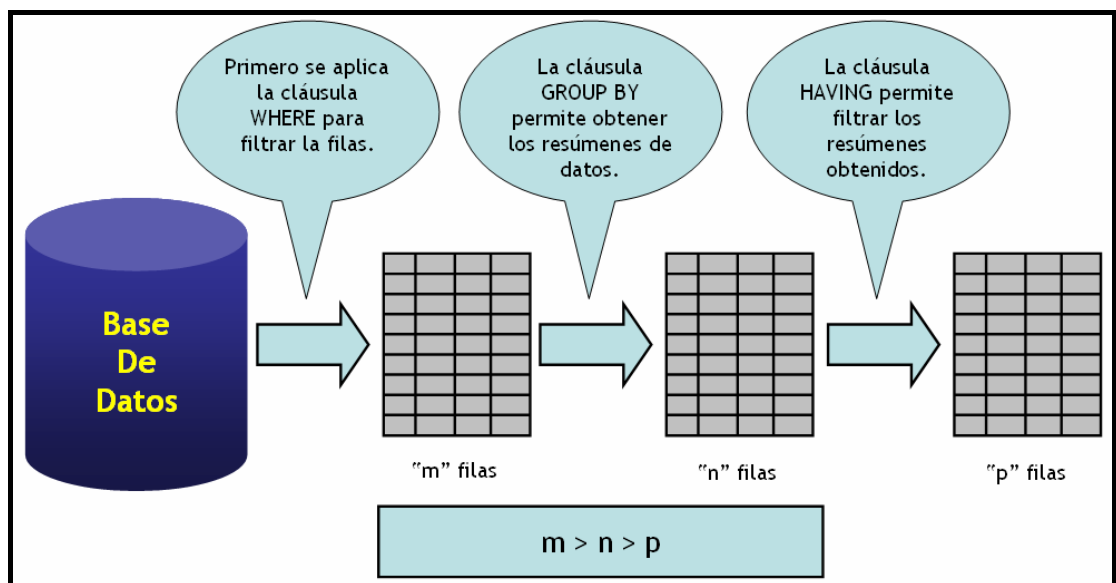
```
SQL> select
  2      extract(year from hire_date) as año,
  3      count(*) as empleados
  4  from employees
  5  group by extract(year from hire_date);
```

AÑO	EMPLEADOS
1987	2
1989	1
1990	1
1991	1
1993	1
1994	7
1995	4
1996	10
1997	28
1998	23
1999	18
2000	11

12 rows selected.

HAVING

Permite limitar mediante una condición de grupo el resultado obtenido después de aplicar GROUP BY, tal como se aprecia en el siguiente gráfico.



Script 5.9

Departamentos que tienen más de 10 empleados.

```
SQL> select department_id as Departamento,  
2 count(*) as Empleados  
3 from employees  
4 group by department_id  
5 having count(*) > 10;
```

DEPARTAMENTO	EMPLEADOS
50	45
80	34

Script 5.10

Los puestos de trabajo de los que solo hay un empleado en la empresa.

```
SQL> select job_id as Puesto,  
2 count(*) as Empleados  
3 from employees  
4 group by job_id  
5 having count(*) = 1;
```

PUESTO	EMPLEADOS
AC_ACCOUNT	1
AC_MGR	1
AD_ASST	1
AD_PRES	1
FI_MGR	1
HR_REP	1
MK_MAN	1
MK_REP	1
PR_REP	1
PU_MAN	1

10 rows selected.

Oracle 9i Básico PL/SQL

Lección 06 Consultas Multitablas

En esta lección veremos como escribir sentencias SELECT para acceder a los datos de dos o más tablas usando equality y non-equality joins (combinaciones por igualdad y por desigualdad). Visualizar datos que no se cumplirían normalmente con una condición de join usando outer joins (uniones externas). Combinar (Join) una tabla consigo misma.

Contenido

¿Qué es un Join?	2
Consultas Simples.....	2
Consultas Complejas	3
Uso de Alias	3
Usando Sintaxis ANSI	4
NATURAL JOIN	4
JOIN ... USING	5
JOIN ... ON	6
Producto Cartesiano.....	7
Combinaciones Externas	8
Usando Sintaxis ANSI	8
Left Outer Joins	8
Right Outer Join	9
Full Outer Join	10
Otras Consultas Multitablas	11
Autoreferenciadas (Self-joins)	11
Consultas Basadas en Desigualdades (Nonequality Joins)	12
Operadores de Conjuntos	12

¿Qué es un Join?

Un Join es usado para consultar datos desde más de una tabla. Las filas se combinan (joined) relacionando valores comunes, típicamente valores de primary key y foreign key.

Métodos de Join:

- Equijoin
- Non-equijoin
- Outer join
- Self join

Consultas Simples

Script 6.1

Consultar los países por región.

```
SQL> conn hr/hr
Connected.

SQL> select regions.region_id, region_name,
2  country_name
3  from regions, countries
4  where regions.region_id = countries.region_id;
```

REGION_ID	REGION_NAME	COUNTRY_NAME
1	Europe	United Kingdom
1	Europe	Netherlands
1	Europe	Italy
1	Europe	France
1	Europe	Denmark
1	Europe	Germany
1	Europe	Switzerland
1	Europe	Belgium
2	Americas	United States of America
2	Americas	Mexico
2	Americas	Canada
2	Americas	Brazil
2	Americas	Argentina
3	Asia	Singapore
3	Asia	Japan
3	Asia	India
3	Asia	HongKong
3	Asia	China
3	Asia	Australia
4	Middle East and Africa	Zimbabwe
4	Middle East and Africa	Zambia
4	Middle East and Africa	Nigeria
4	Middle East and Africa	Kuwait
4	Middle East and Africa	Israel
4	Middle East and Africa	Egypt

```
25 rows selected.
```

Consultas Complejas

Script 6.2

Consultar los departamentos que se encuentran fuera de EEUU, y su respectiva ciudad.

```
SQL> select locations.location_id, city, department_name
  2  from locations, departments
  3  where (locations.location_id = departments.location_id)
  4  and (country_id != 'US');
```

LOCATION_ID	CITY	DEPARTMENT_NAME
1800	Toronto	Marketing
2400	London	Human Resources
2700	Munich	Public Relations
2500	Oxford	Sales

Uso de Alias

Los alias simplifican la referencia a las columnas de las tablas que se utilizan en una consulta.

Script 6.3

Consultar los países de Asia.

```
SQL> select r.region_id, r.region_name, c.country_name
  2  from regions r, countries c
  3  where (r.region_id = c.region_id)
  4  and (r.region_name = 'Asia');
```

REGION_ID	REGION_NAME	COUNTRY_NAME
3	Asia	Australia
3	Asia	China
3	Asia	HongKong
3	Asia	India
3	Asia	Japan
3	Asia	Singapore

6 rows selected.

Usando Sintaxis ANSI

```
<table name> NATURAL [INNER] JOIN <table name>

<table name> [INNER] JOIN <table name> USING (<columns>)

<table name> [INNER] JOIN <table name> ON <condition>
```

NATURAL JOIN

Se combinan esta basada en todas las columnas con igual nombre entre ambas tablas.

Script 6.4

No es necesario utilizar alias.

```
SQL> select location_id, city, department_name
       2  from locations natural join departments;

SQL> select location_id, city, department_name
       2  from departments natural join locations;
```

El resultado en ambos casos es el mismo.

LOCATION_ID	CITY	DEPARTMENT_NAME
1700	Seattle	Administration
1800	Toronto	Marketing
1700	Seattle	Purchasing
2400	London	Human Resources
1500	South San Francisco	Shipping
1400	Southlake	IT
2700	Munich	Public Relations
2500	Oxford	Sales
1700	Seattle	Executive
1700	Seattle	Finance
1700	Seattle	Accounting
1700	Seattle	Treasury
1700	Seattle	Corporate Tax
1700	Seattle	Control And Credit
1700	Seattle	Shareholder Services
1700	Seattle	Benefits
1700	Seattle	Manufacturing
1700	Seattle	Construction
1700	Seattle	Contracting
1700	Seattle	Operations
1700	Seattle	IT Support
1700	Seattle	NOC
1700	Seattle	IT Helpdesk
1700	Seattle	Government Sales
1700	Seattle	Retail Sales
1700	Seattle	Recruiting
1700	Seattle	Payroll

27 rows selected.

Script 6.5

Las columnas comunes solo se muestran una vez en el conjunto de resultado.

```
SQL> select *
  2 from regions natural join countries
  3 where country_name like 'A%';
```

REGION_ID	REGION_NAME	CO	COUNTRY_NAME
2	Americas	AR	Argentina
3	Asia	AU	Australia

```
SQL> select region_name, country_name, city
  2 from regions
  3 natural join countries
  4 natural join locations;
```

REGION_NAME	COUNTRY_NAME	CITY
Europe	Netherlands	Utrecht
Europe	Switzerland	Bern
Europe	Switzerland	Geneva
Europe	Germany	Munich
Europe	United Kingdom	Stretford
Europe	United Kingdom	Oxford
Europe	United Kingdom	London
Europe	Italy	Venice
Europe	Italy	Roma
Americas	Mexico	Mexico City
Americas	Brazil	Sao Paulo
Americas	Canada	Whitehorse
Americas	Canada	Toronto
Americas	United States of America	Seattle
Americas	United States of America	South Brunswick
Americas	United States of America	South San Francisco
Americas	United States of America	Southlake
Asia	Singapore	Singapore
Asia	Australia	Sydney
Asia	India	Bombay
Asia	China	Beijing
Asia	Japan	Hiroshima
Asia	Japan	Tokyo

23 rows selected.

JOIN . . . USING

Permite indicar las columnas a combinar entre dos tablas.

Script 6.6

```
SQL> select location_id, city, department_name
  2 from locations join departments using (location_id)
  3 where city not like 'S%';
```

LOCATION_ID	CITY	DEPARTMENT_NAME
1800	Toronto	Marketing
2400	London	Human Resources
2700	Munich	Public Relations
2500	Oxford	Sales

```
SQL> select region_name, country_name, city
2   from regions
3   join countries using(region_id)
4   join locations using(country_id)
5   where country_id = 'US';
```

REGION_NAME	COUNTRY_NAME	CITY
Americas	United States of America	Southlake
Americas	United States of America	South San Francisco
Americas	United States of America	South Brunswick
Americas	United States of America	Seattle

JOIN ... ON

La condición que permite combinar ambas tablas se debe especificar en la cláusula ON.

Script 6.7

```
SQL> select region_name, country_name, city
2   from regions r
3   join countries c on (r.region_id=c.region_id)
4   join locations l on (c.country_id=l.country_id)
5   where country_id = 'US';
```

REGION_NAME	COUNTRY_NAME	CITY
Americas	United States of America	Southlake
Americas	United States of America	South San Francisco
Americas	United States of America	South Brunswick
Americas	United States of America	Seattle

Producto Cartesiano

Si dos tablas en una consulta no tienen ninguna condición de combinación, entonces Oracle vuelve su producto cartesiano. Oracle combina cada fila de una tabla con cada fila de la otra tabla. Un producto cartesiano genera muchas filas y es siempre raramente útil. Por ejemplo, el producto cartesiano de dos tablas, cada uno con 100 filas, tiene 10.000 filas.

Script 6.8

```
SQL> select region_name, country_name
       2 from regions, countries;
```

REGION_NAME	COUNTRY_NAME
Europe	Argentina
Europe	Australia
Europe	Belgium
Europe	Brazil
Europe	Canada
. . .	
. . .	
Middle East and Africa	Netherlands
Middle East and Africa	Singapore
Middle East and Africa	United Kingdom
Middle East and Africa	United States of America
Middle East and Africa	Zambia
Middle East and Africa	Zimbabwe

100 rows selected.

Script 6.9

En este script utilizaremos la sintaxis ANSI, el resultado es el mismo obtenido en el Script 6.8.

```
SQL> select region_name, country_name
       2 from regions cross join countries;
```

Combinaciones Externas

Una combinación externa amplía el resultado de una combinación simple. Una combinación externa devuelve todas las filas que satisfagan la condición de combinación y también vuelve todos o parte de las filas de una tabla para la cual ninguna filas de la otra satisfagan la condición de combinación.

Script 6.10

En este script se mostrar todos los países de la tabla countries.

```
SQL> select c.country_name, l.city
  2   from countries c, locations l
  3   where ( c.country_id = l.country_id (+) )
  4   and ( c.country_name like 'A%' );
```

COUNTRY_NAME	CITY
Argentina	
Australia	Sydney

Usando Sintaxis ANSI

Left Outer Joins

Script 6.11

Todos estos ejemplos producen el mismo resultado, y muy similar al del Script 6.10.

```
SQL> select c.country_name, l.city
  2   from countries c left outer join locations l
  3   on c.country_id = l.country_id;

SQL> select country_name, city
  2   from countries natural left join locations;

SQL> select country_name, city
  2   from countries left join locations
  3   using (country_id);

SQL> select c.country_name, l.city
  2   from countries c, locations l
  3   where l.country_id (+) = c.country_id;
```

Right Outer Join

Script 6.12

Todos estos ejemplos dan el mismo resultado, e igual al del Script 6.11.

```
SQL> select c.country_name, l.city
  2   from locations l right outer join countries c
  3   on l.country_id = c.country_id;

SQL> select country_name, city
  2   from locations natural right outer join countries;

SQL> select country_name, city
  2   from locations right outer join countries
  3   using ( country_id );

SQL> select c.country_name, l.city
  2   from locations l, countries c
  3   where c.country_id = l.country_id (+);
```

Full Outer Join

Script 6.13

```

SQL> select e.employee_id, e.last_name, d.department_id, d.department_name
  2   from employees e full outer join departments d
  3   on e.department_id = d.department_id;

SQL> select e.employee_id, e.last_name, d.department_id, d.department_name
  2   from employees e, departments d
  3   where e.department_id(+) = d.department_id
  4   union
  5   select e.employee_id, e.last_name, d.department_id, d.department_name
  6   from employees e, departments d
  7   where e.department_id = d.department_id(+);

```

El resultado de estas dos consultas es el mismo, y se muestra a continuación.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
200	Whalen	10	Administration
202	Fay	20	Marketing
201	Hartstein	20	Marketing
.	.		
.	.		
178	Grant		
		220	NOC
		170	Manufacturing
		240	Government Sales
		210	IT Support
		160	Benefits
		150	Shareholder Services
		250	Retail Sales
		140	Control And Credit
		260	Recruiting
		200	Operations
		120	Treasury
		270	Payroll
		130	Corporate Tax
		180	Construction
		190	Contracting
		230	IT Helpdesk

123 rows selected.

Otras Consultas Multitablas

Autoreferenciadas (Self-joins)

Script 6.14

```
SQL> select e.last_name Employee, m.last_name Manager
  2   from employees e, employees m
  3   where m.employee_id = e.manager_id;

SQL> select e.last_name Employee, m.last_name Manager
  2   from employees e inner join employees m
  3   on m.employee_id = e.manager_id;
```

Estas dos consultas muestran una lista de los empleados y sus respectivos jefes.

EMPLOYEE	MANAGER
Hartstein	King
Zlotkey	King
Cambrault	King
. . .	
. . .	
Abel	Zlotkey
Fay	Hartstein
Gietz	Higgins

106 rows selected.

Consultas Basadas en Desigualdades (Nonequality Joins)

Script 6.15

```

SQL> connect scott/tiger
Connected.

SQL> select ename, sal, grade
2   from emp, salgrade
3   where sal between losal and hisal;

```

ENAME	SAL	GRADE
SMITH	800	1
JAMES	950	1
ADAMS	1100	1
WARD	1250	2
MARTIN	1250	2
MILLER	1300	2
TURNER	1500	3
ALLEN	1600	3
CLARK	2450	4
BLAKE	2850	4
JONES	2975	4
SCOTT	3000	4
FORD	3000	4
KING	5000	5

```

14 rows selected.

```

Operadores de Conjuntos

La siguiente tabla describe los diferentes operadores de conjuntos.

Operador	Descripción
UNION	Retorna todas la filas únicas seleccionas por las consultas.
UNION ALL	Retorna todas las filas (incluidas las duplicadas) seleccionadas por las consultas.
INTERSECT	Retorna las filas seleccionadas por ambas consultas.
MINUS	Retorna las filas únicas seleccionadas por la primera consulta, pero que no son seleccionadas por la segunda consulta.

Script 6.16

Consideremos las siguientes consultas.

```
SQL> connect hr/hr
Connected.

SQL> alter session set nls_date_format='DD-Mon-YYYY';

Session altered.

SQL> select last_name, hire_date
  2   from employees
  3   where department_id = 90;

LAST_NAME                HIRE_DATE
-----
King                     17-Jun-1987
Kochhar                  21-Sep-1989
De Haan                  13-Ene-1993

SQL> select last_name, hire_date
  2   from employees
  3   where last_name like 'K%';

LAST_NAME                HIRE_DATE
-----
King                     17-Jun-1987
Kochhar                  21-Sep-1989
Khoo                     18-May-1995
Kaufling                 01-May-1995
King                     30-Ene-1996
Kumar                    21-Abr-2000

6 rows selected.
```

La operador UNION es usado para retornar las filas de ambas consultas pero sin considerar las duplicadas.

```
SQL> select last_name, hire_date
  2   from employees
  3   where department_id = 90
  4   UNION
  5   select last_name, hire_date
  6   from employees
  7   where last_name like 'K%';

LAST_NAME                HIRE_DATE
-----
De Haan                  13-Ene-1993
Kaufling                 01-May-1995
Khoo                     18-May-1995
King                     17-Jun-1987
King                     30-Ene-1996
Kochhar                  21-Sep-1989
Kumar                    21-Abr-2000

7 rows selected.
```

Página en Blanco

Oracle 9i Básico PL/SQL

Lección 07 Subconsultas

Contenido

Subconsultas de Solo una Fila	2
Subconsultas de Múltiples Filas	2
Subconsultas Correlacionadas	3
Subconsultas Escalares	3
Subconsulta Escalar en una Expresión CASE	3
Subconsulta Escalar en la Cláusula SELECT	4
Subconsultas Escalares en las Cláusulas SELECT y WHERE	4
Subconsultas Escalares en la Cláusula ORDER BY	5
Múltiples Columnas en una Subconsultas	6

Subconsultas de Solo una Fila

Script 7.1

```
SQL> connect hr/hr
Connected.

SQL> alter session set nls_date_format='DD-Mon-YYYY';

Session altered.

SQL> select last_name, first_name, salary
2   from employees
3  where salary = (select max(salary) from employees);
```

LAST_NAME	FIRST_NAME	SALARY
King	Steven	24000

Subconsultas de Múltiples Filas

Script 7.2

```
SQL> select last_name, first_name, department_id
2   from employees
3  where department_id in ( select department_id
4                           from employees
5                           where first_name = 'John' );
```

LAST_NAME	FIRST_NAME	DEPARTMENT_ID
Popp	Luis	100
Urman	Jose Manuel	100
Sciarra	Ismael	100
. . .		
. . .		
Errazuriz	Alberto	80
Partners	Karen	80
Russell	John	80

85 rows selected.

Subconsultas Correlacionadas

Script 7.3

```
SQL> select department_id, last_name, salary
  2   from employees e1
  3   where salary = ( select max(salary)
  4                     from employees e2
  5                     where e1.department_id = e2.department_id );
```

DEPARTMENT_ID	LAST_NAME	SALARY
10	Whalen	4400
20	Hartstein	13000
30	Raphaely	11000
40	Mavris	6500
50	Fripp	8200
60	Hunold	9000
70	Baer	10000
80	Russell	14000
90	King	24000
100	Greenberg	12000
110	Higgins	12000

11 rows selected.

Subconsultas Escalares

Retornan exactamente una columna y una sola fila.

Subconsulta Escalar en una Expresión CASE

Script 7.4

Esta consulta lista las ciudades, su código de país, y si es de la India ó no.

```
SQL> select city, country_id,
  2   ( case
  3   when country_id in ( select country_id
  4                         from countries
  5                         where country_name = 'India' ) then 'Indian'
  6   else 'Non-Indian'
  7   end) as "India?"
  8   from locations
  9   where city like 'B%';
```

CITY	CO India?
Beijing	CN Non-Indian
Bombay	IN Indian
Bern	CH Non-Indian

Subconsulta Escalar en la Cláusula SELECT

Script 7.5

```
SQL> select department_id, department_name,
2      ( select max(salary) from employees e
3        where e.department_id = d.department_id ) as "Salario Maximo"
4      from departments d;
```

DEPARTMENT_ID	DEPARTMENT_NAME	Salario Maximo
10	Administration	4400
20	Marketing	13000
30	Purchasing	11000
40	Human Resources	6500
50	Shipping	8200
60	IT	9000
70	Public Relations	10000
80	Sales	14000
90	Executive	24000
100	Finance	12000
110	Accounting	12000

Subconsultas Escalares en las Cláusulas SELECT y WHERE

Script 7.6

El propósito de la siguiente consulta es buscar los nombres de los departamentos y el nombre de sus jefes para todos los departamentos que están en Estados Unidos (United States of America) y Canadá (Canada).

```
SQL> select department_name, manager_id,
2      ( Select last_name from employees e
3        where e.employee_id = d.manager_id) as mgr_name
4      from departments d
5      where ( (select country_id from locations l
6              where d.location_id = l.location_id)
7              in (select country_id from countries c
8                  where c.country_name = 'United States of America'
9                    or c.country_name = 'Canada') )
10     and d.manager_id is not null;
```

DEPARTMENT_NAME	MANAGER_ID	MGR_NAME
Administration	200	Whalen
Marketing	201	Hartstein
Purchasing	114	Raphaely
Shipping	121	Frapp
IT	103	Hunold
Executive	100	King
Finance	108	Greenberg
Accounting	205	Higgins

8 rows selected.

Subconsultas Escalares en la Cláusula ORDER BY**Script 7.7**

La siguiente consulta ordena los nombres de las ciudades por sus respectivos nombres de país.

```
SQL> select country_id, city, state_province
  2   from locations l
  3   order by (select country_name
  4   from countries c
  5   where l.country_id = c.country_id);
```

CO	CITY	STATE_PROVINCE
AU	Sydney	New South Wales
BR	Sao Paulo	Sao Paulo
CA	Toronto	Ontario
CA	Whitehorse	Yukon
CN	Beijing	
DE	Munich	Bavaria
IN	Bombay	Maharashtra
IT	Roma	
IT	Venice	
JP	Tokyo	Tokyo Prefecture
JP	Hiroshima	
MX	Mexico City	Distrito Federal,
NL	Utrecht	Utrecht
SG	Singapore	
CH	Geneva	Geneve
CH	Bern	BE
UK	London	
UK	Stretford	Manchester
UK	Oxford	Oxford
US	Southlake	Texas
US	South San Francisco	California
US	South Brunswick	New Jersey
US	Seattle	Washington

23 rows selected.

Múltiples Columnas en una Subconsultas

Script 7.8

Consideremos las siguientes tablas.

State		
CNT_Code	ST_Code	ST_Name
1	TX	TEXAS
1	CA	CALIFORNIA
91	TN	TAMIL NADU
1	TN	TENNESSE
91	KL	KERALA

City			
CNT_Code	ST_Code	CTY_Code	CTY_Name
1	TX	1001	Dallas
91	TN	2243	Madras
1	CA	8099	Los Angeles

Se quiere listar todas las ciudades ubicadas en Texas.

```
SQL> select cty_name
  2   from city
  3   where (cnt_code, st_code) in
  4         ( select cnt_codr, st_code
  5           from state
  6           where st_name = 'TEXAS' );
```

```
CTY_NAME
-----
DALLAS
```


Oracle 9i Básico PL/SQL

Lección 08 Modificando Datos

Contenido

Insertando Filas	2
Inserciones una Sola Fila	2
Insertando Filas con Valores Nulos	2
Insertando Valores Especiales	3
Insertando Valores Específicos de Fecha	3
Usando & Sustitución para el Ingreso de Valores	3
Copiando Filas Desde Otra Tabla	4
Insertando en Múltiples Tablas.....	4
Modificando Datos.....	5
Actualizando una Columna de una Tabla.....	5
Seleccionando las Filas a Actualizar	6
Actualizando Columnas con Subconsultas.....	7
Actualizando Varias Columnas con una Subconsulta	7
Error de Integridad Referencial	8
Eliminando Filas	9
Eliminar Todas la Filas de una Tabla	9
Seleccionando las Filas a Eliminar	9
Creando una tabla de prueba	9
Eliminando una sola fila	10
Eliminando un grupo de filas.....	10
Uso de Subconsultas.....	10
Error de Integridad Referencial	11
Truncando una Tabla	11
Transacciones	12
Propiedades de una Transacción.....	12
Atomicidad	12
Coherencia	12
Aislamiento	12
Durabilidad.....	13
Operación de Transacciones.....	13
Inicio de una transacción	13
Confirmación de una transacción	14
Cancelar una transacción	14

Insertando Filas

Inserciones una Sola Fila

Script 8.1

```
SQL> connect hr/hr
Connected.

SQL> insert into
  2 departments(department_id, department_name, manager_id, location_id)
  3 values(300, 'Departamento 300', 100, 1800);

1 row created.

SQL> commit;
Commit complete.
```

Insertando Filas con Valores Nulos

Script 8.2

Método Implícito: Se omiten las columnas que aceptan valores nulos.

```
SQL> insert into
  2 departments(department_id, department_name)
  3 values(301, 'Departamento 301');

1 row created.

SQL> commit;
Commit complete.
```

Script 8.3

Método Explícito: Especificamos la palabra clave NULL en las columnas donde queremos insertar un valor nulo.

```
SQL> insert into departments
  2 values(302, 'Departamento 302', NULL, NULL);

1 row created.

SQL> commit;
Commit complete.
```

Insertando Valores Especiales

Script 8.4

```
SQL> insert into employees (employee_id,  
2   first_name, last_name,  
3   email, phone_number,  
4   hire_date, job_id, salary,  
5   commission_pct, manager_id,  
6   department_id)  
7   values (250,  
8   'Gustavo', 'Coronel',  
9   'gcoronel@miempresa.com', '511.481.1070',  
10  sysdate, 'FI_MGR', 14000,  
11  NULL, 102, 100);  
  
1 row created.  
  
SQL> commit;  
Commit complete.
```

Insertando Valores Específicos de Fecha

Script 8.5

```
SQL> insert into employees  
2   values (251, 'Ricardo', 'Marcelo',  
3   'rmarcelo@techsoft.com', '511.555.4567',  
4   to_date('FEB 4, 2005', 'MON DD, YYYY'),  
5   'AC_ACCOUNT', 11000, NULL, 100, 30);  
  
1 row created.  
  
SQL> commit;  
Commit complete.
```

Usando & Sustitución para el Ingreso de Valores

Script 8.6

```
SQL> insert into  
2   departments (department_id, department_name, location_id)  
3   values (&department_id, '&department_name', &location_id);  
Enter value for department_id: 3003  
Enter value for department_name: Departamento 303  
Enter value for location_id: 2800  
old 3: values (&department_id, '&department_name', &location_id)  
new 3: values (3003, 'Departamento 303', 2800)  
  
1 row created.  
  
SQL> commit;  
Commit complete.
```

Copiando Filas Desde Otra Tabla

Script 8.7

```
SQL> create table test
2  (
3    id number(6) primary key,
4    name varchar2(20),
5    salary number(8,2)
6  );

Table created.

SQL> insert into test (id, name, salary)
2  select employee_id, first_name, salary
3  from employees
4  where department_id = 30;

7 rows created.

SQL> commit;
Commit complete.
```

Insertando en Múltiples Tablas

Script 8.8

Primero creamos las siguientes tablas: test50 y test80.

```
SQL> create table test50
2  (
3    id number(6) primary key,
4    name varchar2(20),
5    salary number(8,2)
6  );

Table created.

SQL> create table test80
2  (
3    id number(6) primary key,
4    name varchar2(20),
5    salary number(8,2)
6  );

Table created.
```

Luego limpiamos la tabla **test**.

```
SQL> delete from test;
7 rows deleted.

SQL> commit;
Commit complete.
```

Ahora procedemos a insertar datos en las tres tablas a partir de la tabla **employees**.

```
SQL> insert all
  2  when department_id = 50 then
  3      into test50 (id, name, salary)
  4      values(employee_id, first_name, salary)
  5  when department_id = 80 then
  6      into test80 (id, name, salary)
  7      values (employee_id, first_name, salary)
  8  else
  9      into test(id, name, salary)
 10      values(employee_id, first_name, salary)
 11  select department_id, employee_id, first_name, salary
 12  from employees;

109 rows created.

SQL> commit;
Commit complete.
```

Modificando Datos

Actualizando una Columna de una Tabla

Script 8.9

Incrementar el salario de todos los empleados en 10%.

```
SQL> update employees
  2  set salary = salary * 1.10;

109 rows updated.

SQL> Commit;
Commit complete.
```

Seleccionando las Filas a Actualizar

Script 8.10

Ricardo Marcelo (Employee_id=251) ha sido trasladado de departamento de Compras (Department_id = 30) al departamento de Ventas (Department_id = 80).

```
SQL> select employee_id, first_name, department_id, salary
2   from employees
3   where employee_id = 251;
```

EMPLOYEE_ID	FIRST_NAME	DEPARTMENT_ID	SALARY
251	Ricardo	30	12100

```
SQL> update employees
2   set department_id = 80
3   where employee_id = 251;
```

1 row updated.

```
SQL> select employee_id, first_name, department_id, salary
2   from employees
3   where employee_id = 251;
```

EMPLOYEE_ID	FIRST_NAME	DEPARTMENT_ID	SALARY
251	Ricardo	80	12100

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

Actualizando Columnas con Subconsultas

Script 8.11

Gustavo Coronel (Employee_id = 250) ha sido trasladado al mismo departamento del empleado 203, y su salario tiene que ser el máximo permitido en su puesto de trabajo.

```
SQL> select employee_id, first_name, last_name, department_id, job_id, salary
  2  from employees
  3  where employee_id = 250;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	JOB_ID	SALARY
250	Gustavo	Coronel	100	FI_MGR	15400

```
SQL> update employees
  2  set department_id = (select department_id from employees
  3                      where employee_id = 203),
  4  salary = (select max_salary from jobs
  5             where jobs.job_id = employees.job_id)
  6  where employee_id = 250;

1 row updated.

SQL> commit;
Commit complete.

SQL> select employee_id, first_name, last_name, department_id, job_id, salary
  2  from employees
  3  where employee_id = 250;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	JOB_ID	SALARY
250	Gustavo	Coronel	40	FI_MGR	16000

Actualizando Varias Columnas con una Subconsulta

Asumiremos que tenemos la tabla **resumen_dept**, con la siguiente estructura:

Columna	Tipo de Dato	Nulos	Descripción
Department_id	Number(4)	No	Código de Departamento.
Emps	Number(4)	Si	Cantidad de Empleados en el departamento.
Planilla	Number(10,2)	Si	Emporte de la planilla en el departamento.

Esta tabla guarda la cantidad de empleados y el importe de la planilla por departamento.

Script 8.12

Este script crea la tabla resumen_dept e inserta los departamentos.

```
SQL> create table resumen_dept
2  (
3      department_id number(4) primary key,
4      emps number(4),
5      planilla number(10,2)
6  );
```

Table created.

```
SQL> insert into resumen_dept (department_id)
2  select department_id from departments;
```

31 rows created.

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

Script 8.13

Este script actualiza la tabla resumen_dept.

```
SQL> update resumen_dept
2  set (emps, planilla) = (select count(*), sum(salary)
3      from employees
4      where employees.department_id = resumen_dept.department_id);
```

31 rows updated.

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

Error de Integridad Referencial

Script 8.14

```
SQL> update employees
2  set department_id = 55
3  where department_id = 110;
update employees
*
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK) violated - parent key not found
```


Eliminando Filas

Eliminar Todas la Filas de una Tabla

Script 8.15

```
SQL> select count(*) from test;

COUNT(*)
-----
        30

SQL> delete from test;

30 rows deleted.

SQL> commit;

Commit complete.

SQL> select count(*) from test;

COUNT(*)
-----
         0
```

Seleccionando las Filas a Eliminar

Creando una tabla de prueba

Script 8.16

```
SQL> create table copia_emp
2 as select * from employees;

Table created.
```

Eliminando una sola fila

Script 8.17

```
SQL> delete from copia_emp
      2  where employee_id = 190;

1 row deleted.

SQL> commit;
Commit complete.
```

Eliminando un grupo de filas

Script 8.18

```
SQL> delete from copia_emp
      2  where department_id = 50;

44 rows deleted.

SQL> commit;
Commit complete.
```

Uso de Subconsultas

Script 8.19

Eliminar los empleados que tienen el salario máximo en cada puesto de trabajo.

```
SQL> delete from copia_emp
      2  where salary = (select max_salary from jobs
      3                    where jobs.job_id = copia_emp.job_id);

1 row deleted.

SQL> commit;

Commit complete.
```

Error de Integridad Referencial

Script 8.20

```
SQL> delete from departments
      2  where department_id = 50;
delete from departments
*
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK) violated - child record found
```

Truncando una Tabla

Script 8.21

```
SQL> select count(*) from copia_emp;

COUNT(*)
-----
        64

SQL> truncate table copia_emp;

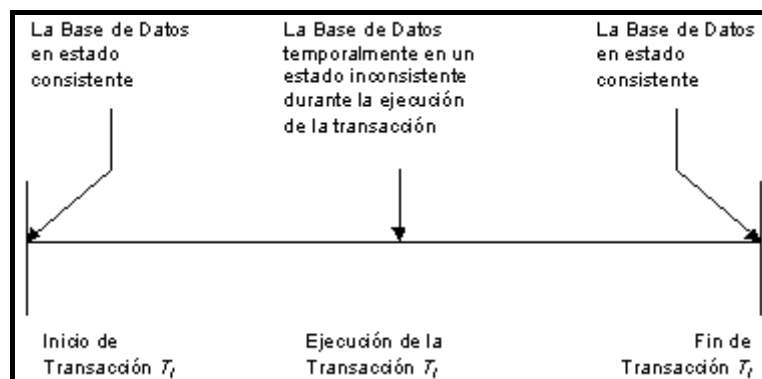
Table truncated.

SQL> select count(*) from copia_emp;

COUNT(*)
-----
         0
```

Transacciones

Una **transacción** es un grupo de acciones que hacen transformaciones consistentes en las tablas preservando la consistencia de la base de datos. Una base de datos está en un estado *consistente* si obedece todas las restricciones de integridad definidas sobre ella. Los cambios de estado ocurren debido a actualizaciones, inserciones, y eliminaciones de información. Por supuesto, se quiere asegurar que la base de datos nunca entre en un estado de inconsistencia. Sin embargo, durante la ejecución de una transacción, la base de datos puede estar temporalmente en un estado inconsistente. El punto importante aquí es asegurar que la base de datos regresa a un estado consistente al fin de la ejecución de una transacción.



Lo que se persigue con el manejo de transacciones es por un lado tener una transparencia adecuada de las acciones concurrentes a una base de datos y por otro lado tener una transparencia adecuada en el manejo de las fallas que se pueden presentar en una base de datos.

Propiedades de una Transacción

Una transacción debe tener las propiedades ACID, que son las iniciales en inglés de las siguientes características: Atomicity, Consistency, Isolation, Durability.

Atomicidad

Una transacción constituye una unidad atómica de ejecución y se ejecuta exactamente una vez; o se realiza todo el trabajo o nada de él en absoluto.

Coherencia

Una transacción mantiene la coherencia de los datos, transformando un estado coherente de datos en otro estado coherente de datos. Los datos enlazados por una transacción deben conservarse semánticamente.

Aislamiento

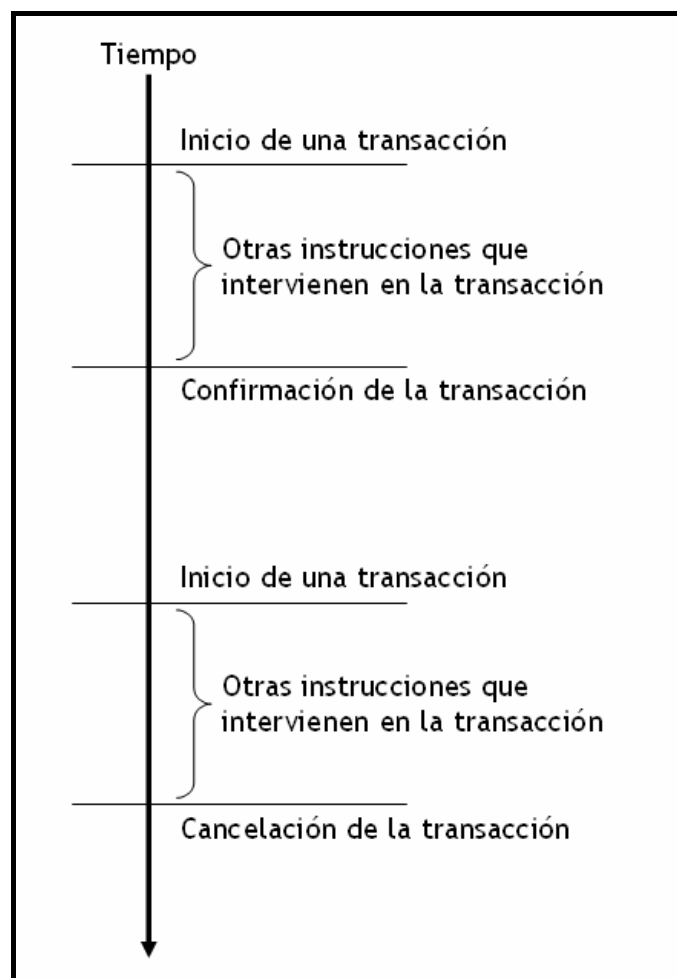
Una transacción es una unidad de aislamiento y cada una se produce aislada e independientemente de las transacciones concurrentes. Una transacción nunca debe ver las fases intermedias de otra transacción.

Durabilidad

Una transacción es una unidad de recuperación. Si una transacción tiene éxito, sus actualizaciones persisten, aun cuando falle el equipo o se apague. Si una transacción no tiene éxito, el sistema permanece en el estado anterior antes de la transacción.

Operación de Transacciones

El siguiente gráfico ilustra el funcionamiento de una transacción, cuando es confirmada y cuando es cancelada.



Inicio de una transacción

El inicio de una transacción es de manera automática cuando ejecutamos una sentencia insert, update, ó delete. La ejecución de cualquiera de estas sentencias da inicio a una transacción. Las instrucciones que se ejecuten a continuación formaran parte de la misma transacción.

Confirmación de una transacción

Para confirmar los cambios realizados durante una transacción utilizamos la sentencia **commit**.

Cancelar una transacción

Para cancelar los cambios realizados durante una transacción utilizamos la sentencia **rollback**.

Script 8. 22

Incrementar el salario al empleado Ricardo Marcelo (employee_id = 251) en 15%.

```
SQL> select employee_id, salary
       2  from employees
       3  where employee_id = 251;
```

EMPLOYEE_ID	SALARY
251	12100

```
SQL> update employees
       2  set salary = salary * 1.15
       3  where employee_id = 251;
```

1 row updated.

```
SQL> select employee_id, salary
       2  from employees
       3  where employee_id = 251;
```

EMPLOYEE_ID	SALARY
251	13915

```
SQL> commit;
```

Commit complete.

Oracle 9i Básico PL/SQL

Lección 09 Creación de un Esquema de Base de Datos

El objetivo de esta lección es la ejecución de sentencia SQL de tipo DDL.

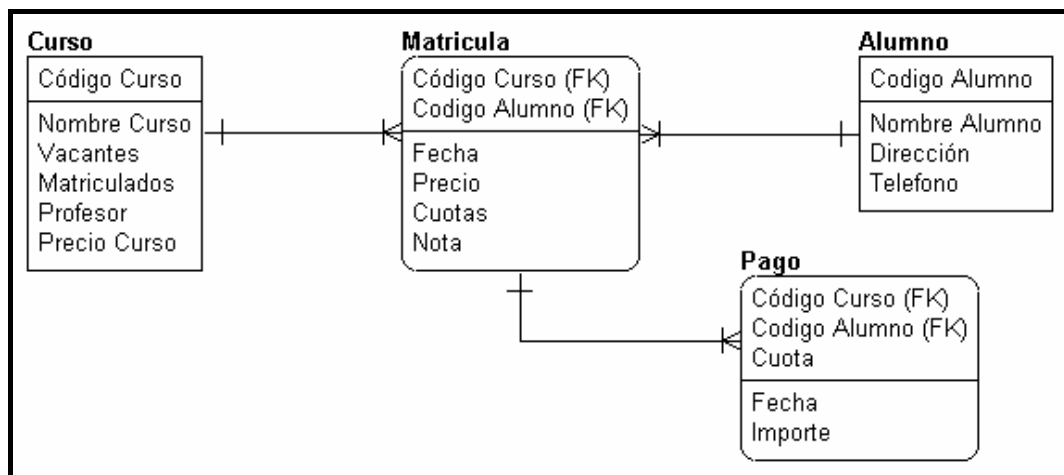
Contenido

Caso a Desarrollar	2
Modelo Lógico	2
Modelo Físico	2
Creación del Usuario para el Esquema	3
Creación del Usuario	3
Asignar Privilegios	3
Creación de Tablas	4
Tabla Curso	4
Tabla Alumno	4
Tabla Matricula	5
Tabla Pago	5
Restricción Primary Key (PK)	6
Tabla Curso	6
Tabla Alumno	6
Tabla Matricula	7
Tabla Pago	7
Restricción Foreign Key (FK)	8
Tabla Matricula	8
Tabla Pago	9
Restricción Default (Valores por Defecto)	10
Ejemplo	10
Restricción NOT NULL (Nulidad de una Columna)	11
Ejemplo	11
Restricción Unique (Valores Únicos)	12
Ejemplo	12
Restricción Check (Reglas de Validación)	13
Ejemplo	13
Asignar Privilegios a Usuarios	14
Ejemplo	14

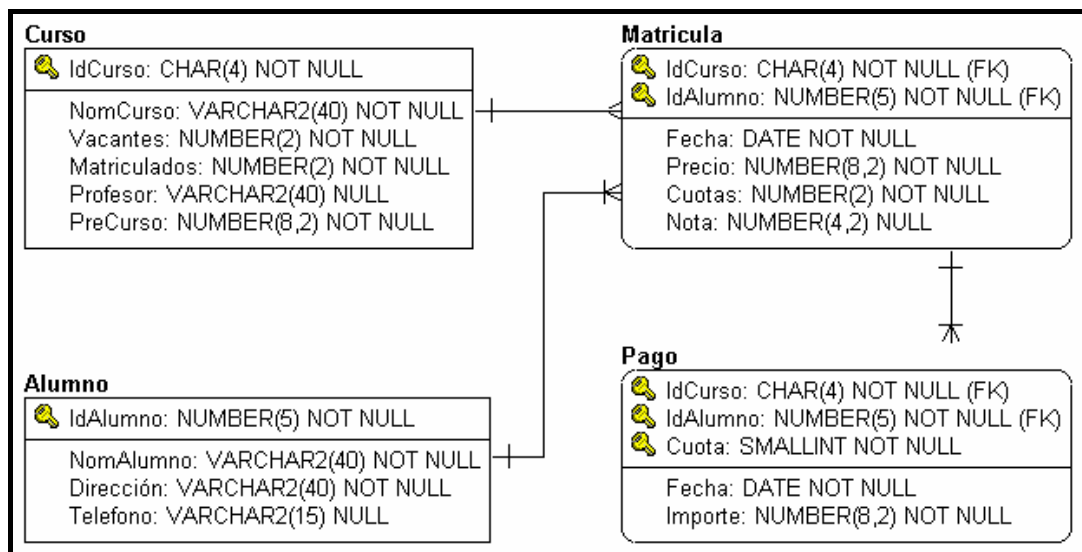
Caso a Desarrollar

El siguiente modelo trata de una empresa que ofrece cursos de extensión, los participantes tienen la libertad de matricularse sin ninguna restricción, y pueden tener facilidades de pago.

Modelo Lógico



Modelo Físico



Creación del Usuario para el Esquema

Creación del Usuario

Script 9.1

```
SQL> conn / as sysdba
Connected.

SQL> create user egcc
      2 identified by admin;

User created.
```

Asignar Privilegios

Asignaremos privilegios al usuario **egcc** a través de los roles **connect** y **resource**, los cuales le otorgan los privilegios necesarios para que pueda crear sus objetos.

Script 9.2

```
SQL> grant connect, resource to egcc;

Grant succeeded.
```

Ahora ya podemos ingresar como usuario **egcc** y crear los objetos que corresponden a su esquema.

Creación de Tablas

Sintaxis

```
Create Table NombreTabla(  
  Columna1 Tipo1 [ NULL | NOT NULL ],  
  Columna2 Tipo2 [ NULL | NOT NULL ],  
  Columna2 Tipo2 [ NULL | NOT NULL ],  
  . . .  
  . . .  
);
```

Tabla Curso

Script 9.3

```
SQL> connect egcc/admin  
Connected.  
  
SQL> CREATE TABLE Curso (  
2      IdCurso          CHAR(4) NOT NULL,  
3      NomCurso         VARCHAR2(40) NOT NULL,  
4      Vacantes         NUMBER(2) NOT NULL,  
5      Matriculados     NUMBER(2) NOT NULL,  
6      Profesor        VARCHAR2(40) NULL,  
7      PreCurso        NUMBER(8,2) NOT NULL  
8  );  
  
Table created.
```

Tabla Alumno

Escriba el script para crear la tabla Alumno.

Tabla Matricula

Escriba el script para crear la tabla Matricula.

Tabla Pago

Escriba el script para crear la tabla Pago.

Restricción Primary Key (PK)

La restricción Primary Key se utiliza para definir la clave primaria de una tabla, en el siguiente cuadro se especifica la(s) columna(s) que conforman la PK de cada tabla.

Tabla	Primary Key
Curso	Incurso
Alumno	IdAlumno
Matricula	IdCurso, IdAlumno
Pago	IdCurso, IdAlumno, Cuota

Sintaxis

```
Alter Table NombreTabla  
  Add Constraint PK_NombreTabla  
  Primary Key ( Columna1, Columna2, . . . );
```

Tabla Curso

Script 9.4

```
SQL> Alter Table Curso  
  2   Add Constraint PK_Curso  
  3   Primary Key ( IdCurso );  
  
Table altered.
```

Tabla Alumno

Escriba el script para crear la PK de la tabla Alumno.

Tabla Matricula

Escriba el script para crear la PK de la tabla Matricula.

Tabla Pago

Escriba el script para crear la PK de la tabla Pago.

Restricción Foreign Key (FK)

La restricción Foreign Key se utiliza para definir la relación entre dos tablas, en el siguiente cuadro se especifica la(s) columna(s) que conforman la FK de cada tabla.

Tabla	Foreign Key	Tabla Referenciada
Matricula	IdCurso	Curso
	IdAlumno	Alumno
Pago	IdCurso, IdAlumno	Matricula

Sintaxis

```
Alter Table NombreTabla
Add Constraint FK_NombreTabla_TablaReferenciada
Foreign Key ( Columna1, Columna2, . . . )
References TablaReferenciada;
```

Es necesario que en la tabla referenciada esté definida la PK, por que la relación se crea entre la PK de la tabla referenciada y las columnas que indicamos en la cláusula Foreign Key.

Tabla Matricula

1ra FK

La primera FK de esta tabla es IdCurso y la tabla referenciada es Curso, el script para crear esta FK es el siguiente:

Script 9.5

```
SQL> Alter table Matricula
2      Add Constraint FK_Matricula_Curso
3      Foreign Key ( IdCurso )
4      References Curso;

Table altered.
```

2da FK

La segunda FK de esta tabla es IdAlumno y la tabla referenciada es Alumno, escriba usted el script para crear ésta FK.

Tabla Pago

Esta tabla solo tiene una FK y esta compuesta por dos columnas: IdCurso e IdAlumno, y la tabla referenciada es Matricula, escriba usted el script para crear ésta FK.

Restricción Default (Valores por Defecto)

El *Valor por Defecto* es el que toma una columna cuando no especificamos su valor en una sentencia insert.

Sintaxis

```
Alter Table NombreTabla  
Modify ( NombreColumna Default Expresión );
```

Ejemplo

El número de vacantes por defecto para cualquier curso debe ser 20.

Script 9.6

```
SQL> Alter Table Curso  
2 Modify ( Vacantes default 20 );  
  
Table altered.
```

Para probar el default insertemos un registro en la tabla curso.

Script 9.7

IDCU	NOMCURSO	VACANTES	MATRICULADOS	PROFESOR	PRECURSO
C001	Oracle 9i - Nivel Inicial	20	10	Gustavo Coronel	350

Restricción NOT NULL (Nulidad de una Columna)

Es muy importante determinar la nulidad de una columna, y es muy importante para el desarrollador tener esta información a la mano cuando crea las aplicaciones.

Sintaxis

```
Alter Table NombreTabla  
  Modify ( NombreColumna [NOT] NULL );
```

Ejemplo

En la tabla alumno, la columna **Telefono** no debe aceptar valores nulos.

Script 9. 8

```
SQL> Alter Table Alumno  
  2  Modify ( Telefono NOT NULL );  
  
Table altered.  
  
SQL> describe alumno  
Name                                         Null?    Type  
-----  
IDALUMNO                                   NOT NULL NUMBER(5)  
NOMALUMNO                                  NOT NULL VARCHAR2(40)  
DIRECCIÓN                                  NOT NULL VARCHAR2(40)  
TELEFONO                                   NOT NULL VARCHAR2(15)
```

Si queremos insertar un alumno tendríamos que ingresar datos para todas las columnas.

Script 9. 9

```
SQL> insert into alumno  
  2  values(10001, 'Ricardo Marcelo', 'Ingeniería', NULL);  
insert into alumno  
*  
ERROR at line 1:  
ORA-01400: cannot insert NULL into ("EGCC"."ALUMNO"."TELEFONO")
```

El mensaje de error claramente nos indica que no se puede insertar valores nulos en la columna TELEFONO, de la tabla ALUMNO, que se encuentra en el esquema EGCC.

Restricción Unique (Valores Únicos)

En muchos casos debemos garantizar que los valores de una columna ó conjunto de columnas de una tabla acepten solo valores únicos.

Sintaxis

```
Alter Constraint NombreTabla  
  Add Constraint U_NombreTabla_NombreColumna  
  Unique ( Columna1, Columna2, . . . );
```

Ejemplo

No puede haber dos alumnos con nombres iguales.

Script 9.10

```
SQL> Alter Table alumno  
      2 Add Constraint U_Alumno_NomAlumno  
      3 Unique (NomAlumno);  
  
Table altered.
```

Para probar la restricción insertemos datos.

Script 9.11

```
SQL> Insert Into Alumno  
      2 Values( 10001, 'Sergio Matsukawa', 'San Miguel', '456-3456' );  
  
1 row created.  
  
SQL> Insert Into Alumno  
      2 Values( 10002, 'Sergio Matsukawa', 'Los Olivos', '521-3456' );  
Insert Into Alumno  
*  
ERROR at line 1:  
ORA-00001: unique constraint (EGCC.U_ALUMNO_NOMALUMNO) violated
```

El mensaje de error del segundo insert nos indica que esta violando el constraint de tipo unique de nombre U_ALUMNO_NOMALUMNO en el esquema EGCC.

Restricción Check (Reglas de Validación)

Las reglas de validación son muy importantes por que permiten establecer una condición a los valores que debe aceptar una columna.

Sintaxis

```
Alter Table NombreTabla  
  Add Constraint CK_NombreTabla_NombreColumna  
  Check ( Condición );
```

Ejemplo

El precio de un curso no puede ser cero, ni menor que cero.

Script 9.12

```
SQL> Alter Table Curso  
  2      Add Constraint CK_Curso_PreCurso  
  3      Check ( PreCurso > 0 );  
  
Table altered.
```

Probemos el constraint ingresando datos.

Script 9.13

```
SQL> Insert Into Curso  
  2  Values( 'C002', 'Asp.NET', 20, 7, 'Ricardo Marcelo', -400.00 );  
Insert Into Curso  
*  
ERROR at line 1:  
ORA-02290: check constraint (EGCC.CK_CURSO_PRECURSO) violated
```

Al intentar ingresar un curso con precio negativo, inmediatamente nos muestra el mensaje de error indicándonos que se está violando la regla de validación.

Asignar Privilegios a Usuarios

Si queremos que otros usuarios puedan operar los objetos de un esquema, debemos darle los privilegios adecuadamente.

Sintaxis

```
Grant Privilegio On Objeto To Usuario;
```

Ejemplo

Por ejemplo, el usuario scott necesita consultar la tabla curso.

Script 9.14

```
SQL> Grant Select On Curso To Scott;  
  
Grant succeeded.
```

Ahora hagamos la prueba respectiva.

Script 9.15

```
SQL> connect scott/tiger  
Connected.  
  
SQL> select * from egcc.curso;
```

IDCU	NOMCURSO	VACANTES	MATRICULADOS	PROFESOR	PRECURSO
C001	Oracle 9i - Nivel Inicial	20	10	Gustavo Coronel	350

Oracle 9i Básico PL/SQL

Lección 10 PL/SQL - Fundamentos

Contenido

Introducción a PL/SQL	2
Bloque	2
Bloque anónimo	2
Función.....	2
Procedimiento	3
Tipos de Datos y Variables	4
Tipos de Datos	4
Declaración de Variables.....	4
Asignar Valores a Variables	4
Caso 1: Sentencia de asignación	4
Caso 2: Utilizando la sentencia select	5
%Type	5
Estructuras de Control.....	6
Estructura IF.....	6
Caso 1:	6
Caso 2	7
Caso 3	8
Estructura Case.....	9
Caso 1	9
Caso 2	10
Bucles.....	11
Objetos previos	11
Secuencia sqtest	11
Tabla Temporal.....	11
Bucle Simple: LOOP	11
Bucle While	12
Bucle FOR.....	13
Otros Elementos de Programación	15
Etiquetas	15
Crear una etiqueta:	15
Saltar a una etiqueta:.....	15
Restricciones	16
Etiquetando los Bucles	16
Instrucción NULL	17

Introducción a PL/SQL

Bloque

Unidad básica de programación.

Bloque anónimo

Sintaxis

```
Declare
-----
-----
Begin
-----
-----
End;
```

Script 10.1

```
Declare
  sFecha Varchar2(40);
Begin
  select to_char(sysdate,'dd/mm/yyyy hh24:mm:ss')
  into sFecha from dual;
  dbms_output.put_line( 'Hoy es: ' || sFecha );
End;
```

Función

Sintaxis

```
Create Or Replace Function NombreFuncion( Parámetros ) Return TipoSalida
Is
-----
-----
Begin
-----
-----
End;
```

Script 10.2

```
create or replace function fnSuma( a number, b number ) return number
is
  c number;
begin
  c := a + b;
  return c;
end;
```

Ejecución

```
SQL> select fnSuma(12,25) from dual;

FNSUMA(12,25)
-----
          37
```

Procedimiento

Sintaxis

```
Create Or Replace Procedure NombreProcedimiento( Parámetros )
Is
-----
-----
Begin
-----
-----
End;
```

Script 10. 3

```
create or replace procedure prSuma( a number, b number )
is
  c number;
begin
  c := a + b;
  dbms_output.put_line( c );
end;
```

Ejecución

```
SQL> begin
  2   prSuma(15,15);
  3   end;
  4   /
30

PL/SQL procedure successfully completed.
```

Tipos de Datos y Variables

Tipos de Datos

Clase	Tipos de Datos
Escalares	Long, Number, Date, Char, Varchar2, Boolean
Compuestos	Record, Table
LOB	BFile, CLob, BLob
Referencia	Ref Cursor

Declaración de Variables

```
Nombre_Variable Tipo [CONSTANT] [NOT NULL] [:=valor];
```

Script 10. 4

```
Declare  
  v_Descripcion varchar2(50);  
  v_NroAsiento number := 35;  
  v_Contador   binary_integer := 0;
```

Asignar Valores a Variables

Caso 1: Sentencia de asignación

Script 10.5

```
v_NroAsiento := 78;  
v_Descuento := fnSuma(18,16);
```


Caso 2: Utilizando la sentencia select**Script 10. 6**

```
select count(*) into v_Emps from emp;
```

Script 10. 7

Procedimiento para consultar el nombre de un empleado.

```
create or replace procedure pr101(p_empno number)
is
    v_ename varchar2(10);
begin
    select ename into v_ename from emp where empno = p_empno;
    dbms_output.put_line(v_ename);
end;
```

Ejecución

```
SQL> execute pr101(7499);
ALLEN

PL/SQL procedure successfully completed.
```

%Type

Permite declarar variables del mismo tipo de una columna.

Script 10. 8

Función para consultar el nombre de un departamento.

```
create or replace function fn101(p_deptno dept.deptno%type)
return dept.dname%type
is
    v_dname dept.dname%type;
begin
    select dname into v_dname from dept where deptno = p_deptno;
    return(v_dname);
end;
```

Ejecución

```
SQL> select fn101(10) from dual;

FN101(10)
-----
ACCOUNTING
```

Estructuras de Control

Estructura IF

Caso 1:

```
if (condicion) then
    -----
    -----
    -----
end if
```

Script 10. 9

Función para encontrar el mayor de 3 números.

```
create or replace function fn102 (n1 number, n2 number, n3 number) return number
is
    mayor number := 0;
begin
    if (n1>mayor) then
        mayor := n1;
    end if;
    if (n2>mayor) then
        mayor := n2;
    end if;
    if (n3>mayor) then
        mayor := n3;
    end if;
    return mayor;
end;
```

Ejecución

```
SQL> select fn102(12,45,4) from dual;
```

```
FN102 (12,45,4)
-----
              45
```

Caso 2

```
if (condicion) then
    -----
    -----
else
    -----
    -----
end if
```

Script 10. 10

Función para encontrar el mayor de 3 números.

```
create or replace function fn103 (n1 number, n2 number, n3 number) return number
is
    mayor number;
begin
    if (n1>n2) then
        mayor := n1;
    else
        mayor := n2;
    end if;
    if (n3>mayor) then
        mayor := n3;
    end if;
    return mayor;
end;
```

Ejecución

```
SQL> select fn103(-4,-8,-23) from dual;

FN103 (-4,-8,-23)
-----
                -4
```

Caso 3

```
if (condicion1) then
    -----
    -----
elsif (condicion2) then
    -----
    -----
elsif (condicion3) then
    -----
    -----
...
[else
    -----
    -----]
end if;
```

Script 10. 11

Clasificar el salario de un empleado.

```
create or replace function fn104 (p_empno emp.empno%type) return varchar2
is
    v_sal emp.sal%type; v_msg varchar2(30);
begin
    select sal into v_sal from emp where empno = p_empno;
    if (v_sal<2500) then
        v_msg := 'Salario Bajo';
    elsif (v_sal<4000) then
        v_msg := 'Salario Regular';
    else
        v_msg := 'Salario Bueno';
    end if;
    v_msg := to_char(v_sal) || ' - ' || v_msg;
    return v_msg;
end;
```

Ejecución

```
SQL> select fn104(7698) from dual;
```

```
FN104 (7698)
-----
2850 - Salario Regular
```

Estructura Case

Caso 1

```
case selector
  when valor1 then
    -----
    -----
  when valor2 then
    -----
    -----
  ...
  [else
    -----
    -----]
end case;
```

Script 10. 12

Función para evaluar un número.

```
create or replace function fn105(n number) return varchar2
is
  rpta varchar2(30);
begin
  case n
    when 1 then
      rpta := 'Uno';
    when 2 then
      rpta := 'Dos';
    else
      rpta := 'None';
  end case;
  return rpta;
end;
```

Ejecución

```
SQL> select fn105(1) from dual;
```

```
FN105(1)
-----
Uno
```

Caso 2

```
case
  when condicion1 then
    -----
    -----
  when condicion2 then
    -----
    -----
  ...
  [else
    -----
    -----]
end case;
```

Script 10. 13

Realizar una función para evaluar el salario de un empleado.

```
create or replace function fn106(v_empno emp.empno%type) return varchar2 is
  v_msg varchar2(40);
  v_sal emp.sal%type;
begin
  select sal into v_sal from emp where empno = v_empno;
  case
    when (v_sal > 0 and v_sal <= 2500) then
      v_msg := 'Salario Bajo';
    when (v_sal > 2500 and v_sal <= 4000) then
      v_msg := 'Salario Regular';
    when (v_sal > 4000) then
      v_msg := 'Salario Bueno';
    else
      v_msg := 'Caso Desconocido';
  end case;
  v_msg := to_char(v_sal) || ' - ' || v_msg;
  return v_msg;
end;
```

Ejecución

```
SQL> select fn106(7782) from dual;
```

```
FN106(7782)
```

```
-----
2450 - Salario Bajo
```

Bucles

Objetos previos

Secuencia sqtest

Script 10.14

```
SQL> create sequence sqtest;  
  
Sequence created.
```

Tabla Temporal

Script 10.15

```
SQL> create global temporary table test (  
2   id number primary key,  
3   dato varchar2(30)  
4   ) on commit preserve rows;  
  
Table created.
```

Bucle Simple: LOOP

Sintaxis

```
loop  
-----  
-----  
if (Condición) then  
-----  
-----  
    exit;  
end if;  
-----  
-----  
exit when (condición);  
-----  
-----  
end loop;
```

Script 10. 16

Función para encontrar el factorial de un número.

```
create or replace function fn107 (n number) return number
is
  f number := 1;
  cont number := n;
begin
  loop
    f := f * cont;
    cont := cont - 1;
    exit when (cont=0);
  end loop;
  return f;
end;
```

Ejecución

```
SQL> select fn107(5) from dual;
```

```
FN107(5)
-----
      120
```

Bucle While

Sintaxis

```
while (condicion) loop
  -----
  -----
end loop;
```

Script 10. 17

Insertar datos en una tabla.

```
create or replace procedure pr102 (n number)
is
  k number := 0;
begin
  while (k<n) loop
    insert into test( id,dato )
      values( sqtest.nextval, 'Gustavo Coronel' );
    k := k + 1;
  end loop;
  commit;
  dbms_output.put_line('Proceso Ejecutado');
end;
```


Ejecución

```
SQL> execute pr102(15);  
Proceso Ejecutado  
  
PL/SQL procedure successfully completed.
```

Verificar la tabla

```
SQL> select * from test;  
  
      ID DATO  
-----  
      1 Gustavo Coronel  
      2 Gustavo Coronel  
      3 Gustavo Coronel  
      4 Gustavo Coronel  
      5 Gustavo Coronel  
      6 Gustavo Coronel  
      7 Gustavo Coronel  
      8 Gustavo Coronel  
      9 Gustavo Coronel  
     10 Gustavo Coronel  
     11 Gustavo Coronel  
     12 Gustavo Coronel  
     13 Gustavo Coronel  
     14 Gustavo Coronel  
     15 Gustavo Coronel  
  
15 rows selected.
```

Bucle FOR

Sintaxis

```
for contador in [reverse] limite_inferior .. limite_superior loop  
-----  
-----  
end loop;
```

Script 10.18

Calcular el factorial de un número.

```
create or replace function fn108 ( n number ) return number  
is  
  f number := 1;  
begin  
  for k in 1 .. n loop  
    f := f * k;  
  end loop;  
  return f;  
end;
```

Ejecución

```
SQL> select fn108(5) from dual;

  FN108(5)
-----
       120
```

Script 10. 19

Ilustrar que no es necesario declarar el contador del for.

```
create or replace procedure pr103 ( n number, msg varchar2 )
is
  k number := 1000;
begin
  for k in 1 .. n loop
    dbms_output.put_line( k || ' - ' || msg );
  end loop;
  dbms_output.put_line( 'k = ' || k );
end;
```

Ejecución

```
SQL> execute pr103(5,'ALIANZA CAMPEON');
1 - ALIANZA CAMPEON
2 - ALIANZA CAMPEON
3 - ALIANZA CAMPEON
4 - ALIANZA CAMPEON
5 - ALIANZA CAMPEON
k = 1000

PL/SQL procedure successfully completed.
```

Script 10. 20

Aplicación de reverse. Tabla de multiplicar.

```
create or replace procedure pr104 ( n number )
is
  cad varchar2(30);
begin
  for k in reverse 1 .. 12 loop
    cad := n || ' x ' || k || ' = ' || (n*k);
    dbms_output.put_line( cad );
  end loop;
end;
```

Ejecución

```
SQL> execute pr104(5);  
5 x 12 = 60  
5 x 11 = 55  
5 x 10 = 50  
5 x 9 = 45  
5 x 8 = 40  
5 x 7 = 35  
5 x 6 = 30  
5 x 5 = 25  
5 x 4 = 20  
5 x 3 = 15  
5 x 2 = 10  
5 x 1 = 5  
  
PL/SQL procedure successfully completed.
```

Otros Elementos de Programación

Etiquetas

Crear una etiqueta:

```
<<nombre_etiqueta>>
```

Saltar a una etiqueta:

```
goto nombre_etiqueta;
```

Script 10. 21

Elevar un número a una potencia.

```
create or replace function fn109( b number, p number ) return number
is
  r number := 1;
  k number := 0;
begin
  loop
    k := k + 1;
    r := r * b;
    if (k=p) then
      goto fin;
    end if;
  end loop;
  <<fin>>
  return r;
end;
```

Ejecución

```
SQL> select fn109(-5,3) from dual;

FN109 (-5,3)
-----
        -125
```

Restricciones

1. No se puede realizar un salto al interior de un if, bucle, ó bloque interno.
2. No se puede saltar de una cláusula if a otra, en la misma instrucción if.
3. No se puede saltar de un bloque de excepciones al bloque de instrucciones.

Etiquetando los Bucles**Formato**

```
<<abc>>
for ...
  -----
  -----
  <<xyz>>
  for ...
    -----
    -----
    if ...
      exit abc;
    end if;
  end loop xyz;
  -----
  -----
end loop abc;
```

Instrucción NULL

Formato

```
if (condición) then
    null;
else
    -----
    -----
end if;
```

Script 10.22

Desarrollar una función para determinar si número es impar.

```
create or replace function fn110( n number )
return varchar2
is
    rtn varchar2(30) := "";
begin
    if ( mod(n,2) = 0 ) then
        null;
    else
        rtn := n || ' es impar';
    end if;
    return rtn;
end;
```

Ejecución

```
SQL> select fn110(15) from dual;
```

```
FN110(15)
-----
15 es impar
```

Página en Blanco

Oracle 9i Básico PL/SQL

Lección 11

PL/SQL – Trabajando con Datos

Contenido

Registros	2
Definición.....	2
Declaración de variable	2
Acceso a los campos.....	2
%RowType	3
SQL en PL/SQL.....	4
Categorías.....	4
Uso de SQL en PL/SQL	4
SQL Dinámico	4
Sentencia Select	5
Sentencia Insert	6
Sentencia Update	6
Sentencia Delete	8
Nombres de Variables	9
Cláusula Returning	9
Referencias de Tablas.....	10
Enlaces de Base de Datos	10
Sinónimos.....	11
Cursores	12
Procesamiento de Cursores	12
Pasos a Seguir	12
Declarar un Cursor	12
Apertura de un Cursor	12
Extracción de datos desde un cursor.....	13
Cerrar un cursor.....	13
Atributos de los Cursores	14
Bucles de Extracción	14
Bucles Simples	14
Bucles While	15
Bucle For	17
Bucles For Implícitos	18
Cursores Select For Update	18
Cursores Implícitos.....	19

Registros

Definición

```
type tipo_registro is record (  
    campo1 tipo1 [not null] [:= valor1],  
    campo2 tipo2 [not null] [:= valor2],  
    -----  
    -----  
);
```

Declaración de variable

```
nombre_variable tipo_registro;
```

Acceso a los campos

```
nombre_variable.nombre_campo
```

Script 11.1

Consultar el nombre y salario de un empleado.

```
create or replace procedure pr105( cod emp.empno%type )  
is  
    type reg is record (  
        nombre emp.ename%type,  
        salario emp.sal%type  
    );  
    r reg;  
begin  
    select ename, sal into r  
    from emp where empno = cod;  
    dbms_output.put_line( 'Nombre: ' || r.nombre );  
    dbms_output.put_line( 'Salario: ' || r.salario );  
end;
```


Ejecución

```
SQL> execute pr105( 7698 );  
Nombre: BLAKE  
Salario: 2850  
  
PL/SQL procedure successfully completed.
```

%RowType

Se utiliza para declarar registros con la misma estructura de una tabla.

Formato:

```
NombreVariable NombreTable%RowType;
```

Script 11.2

Consultar los datos de un departamento.

```
create or replace procedure pr106( cod dept.deptno%type )  
is  
  r dept%rowtype;  
begin  
  select * into r  
    from dept where deptno = cod;  
  dbms_output.put_line('Codigo: ' || r.deptno);  
  dbms_output.put_line('Nombre: ' || r.dname);  
  dbms_output.put_line('Localización: ' || r.loc);  
end;
```

Ejecución

```
SQL> execute pr106(10);  
Codigo: 10  
Nombre: ACCOUNTING  
Localización: NEW YORK  
  
PL/SQL procedure successfully completed.
```

SQL en PL/SQL

Categorías

1. DML Lenguaje de Manipulación de Datos: Select, Insert, Update, Delete.
2. DDL Lenguaje de Definición de Datos: Create, Alter, Drop, Grant.
3. Control de Transacciones: Commit, Rollback.
4. Control de Sesiones: Alter Session.
5. Control del Sistema: Alter System.

Uso de SQL en PL/SQL

Las categorías permitidas de SQL en PL/SQL son solamente la 1 y 3: DML y Control de Transacciones.

SQL Dinámico

Permite ejecutar cualquier tipo de instrucción SQL desde PL/SQL.

Script 11.3

```
create or replace procedure pr107( cmd varchar2)
is
begin
    execute immediate cmd;
end;
```

Ejecución 1

```
SQL> exec pr107('create table t1 ( id number, dato varchar2(30) )');
PL/SQL procedure successfully completed.
```

Ejecución 2

```
SQL> exec pr107('insert into t1 values( 1, ''Oracle is Powerful'' )');
PL/SQL procedure successfully completed.
```

Verificando la tabla t1

```
SQL> select * from t1;

      ID DATO
-----
      1 Oracle is Powerful
```

Sentencia Select

Sintaxis

```
Select columnas into variables/registro
From NombreTabla
Where condición;
```

Script 11.4

Consultar la cantidad de empleados y el importe de la planilla de un departamento.

```
create or replace procedure pr108(cod dept.deptno%type)
is
  emps number;
  planilla number;
begin
  select count(*), sum(sal) into emps, planilla
    from emp
   where deptno = cod;
  dbms_output.put_line('Empleados: ' || emps);
  dbms_output.put_line('Planilla: ' || planilla);
end;
```

Ejecución

```
SQL> exec pr108( 20 );
Empleados: 5
Planilla: 10875

PL/SQL procedure successfully completed.
```

Sentencia Insert

Sintaxis 1

```
Insert into NombreTabla[(columnas)] values(datos);
```

Sintaxis 2

```
Insert into NombreTabla[(columns)] select ... ;
```

Script 11.5

Procedimiento para registrar un nuevo departamento.

```
create or replace procedure pr109( cod number, nom varchar2, loc varchar2)
is
begin
    insert into dept values(cod, nom, loc);
    commit;
    dbms_output.put_line('Proceso OK');
end;
```

Ejecución

```
SQL> exec pr109( 50, 'Deportes', 'Los Olivos' );
Proceso OK

PL/SQL procedure successfully completed.
```

Verificando tabla Dept

```
SQL> select * from dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	Deportes	Los Olivos

Sentencia Update

Sintaxis

```
Update Nombretabla
Set   columna1 = valor,
      (columna2, columna3, ...) = (sentencia_select),
      ...
where (condición);
```

Script 11.6

Para este ejemplo crearemos la tabla **resumen**, donde almacenaremos el número de empleados por departamento y el importe de su planilla.

Creación de la tabla resumen.

```
SQL> create table resumen(  
2     deptno number(2) primary key,  
3     emps number(2),  
4     planilla number(10,2)  
5 );  
  
Table created.
```

Insertando los códigos de los departamentos.

```
SQL> insert into resumen(deptno)  
2     select deptno from dept;  
  
5 rows created.
```

Procedimiento para actualizar las columnas emps y planilla.

```
create or replace procedure pr110  
is  
begin  
    update resumen  
        set (emps,planilla) = (select count(*), sum(sal) from emp  
                                where emp.deptno = resumen.deptno);  
    commit;  
    dbms_output.put_line('Proceso Ok');  
end;
```

Ejecución

```
SQL> exec pr110;  
Proceso Ok  
  
PL/SQL procedure successfully completed.
```

Verificar la ejecución.

```
SQL> select * from resumen;
```

DEPTNO	EMPS	PLANILLA
10	3	8750
20	5	10875
30	6	9400
40	0	
50	0	

Sentencia Delete

Sintaxis

```
Delete from NombreTabla  
Where condición;
```

Script 11.7

Desarrollar un procedimiento para eliminar un departamento, primero debe verificar que no tenga registros relacionados en la tabla **emp**.

```
create or replace procedure pr111(cod number)  
is  
    cont number;  
begin  
    select count(*) into cont from dept where deptno = cod;  
    if cont = 0 then  
        dbms_output.put_line('No existe');  
        return;  
    end if;  
    select count(*) into cont from emp where deptno = cod;  
    if cont > 0 then  
        dbms_output.put_line('No puede se eliminado');  
        return;  
    end if;  
    delete from dept where deptno = cod;  
    commit;  
    dbms_output.put_line('Proceso Ok');  
end;
```

Ejecución

```
SQL> exec pr111(10);  
No puede se eliminado  
  
PL/SQL procedure successfully completed.
```

Nombres de Variables

Se recomienda no utilizar nombres de variables y/o parámetros iguales a los nombres de las columnas, sobre todo si van a ser utilizadas en las instrucciones SQL.

Script 11.8

Desarrollar un procedimiento para eliminar un empleado. El siguiente procedimiento tiene un resultado inesperado.

```
create or replace procedure pr112(empno number)
is
begin
    delete from emp where empno = empno;
end;
```

Si intentamos eliminar un empleado, se eliminarán todos los registros de la tabla **emp**.

```
SQL> exec pr112(7654);

PL/SQL procedure successfully completed.
```

Verifiquemos el resultado.

```
SQL> select * from emp;

no rows selected
```

Esto sucede incluso si el código del empleado no existe. Ahora ejecutemos la sentencia **rollback** para recuperar los empleados.

```
SQL> rollback;

Rollback complete.
```

Si consultamos nuevamente la tabla **emp** tendremos los registros recuperados.

Cláusula Returning

Sirve para obtener información de la última fila modificada, puede ser utilizado con las sentencias insert, update, y delete.

Sintaxis

```
returning expresión, ... into variable, ... ;
```

Script 11.9

Ilustración de Returning. La tabla **test** y la secuencia **sqtest** se crearon en la lección 10.

```
create or replace procedure pr113(msg varchar2)
is
  v_rowid rowid;
  v_id number;
begin
  insert into test values(sqtest.nextval,msg)
  returning rowid, id into v_rowid, v_id;
  commit;
  dbms_output.put_line('RowId: ' || v_rowid);
  dbms_output.put_line('Id: ' || v_id);
end;
```

Ejecución

```
SQL> exec pr113('El deporte es salud.');
```

RowId:	AAQAEJAABAAAAEKAAA
Id:	21

Referencias de Tablas

Sintaxis

```
[esquema.]tabla[@enlace]
```

Enlaces de Base de Datos

Sintaxis

```
create [shared] [public] database link nombre_enlace
connect to nombre_usuario identified by contraseña
[using cadena_sqlnet];
```

Script 11.10

Crearemos un enlace remoto público para conectarnos como usuario hr, este enlace debe ser creado como usuario **sys**.

```
SQL> conn / as sysdba
Connected.

SQL> create public database link lnk_demo
2   connect to hr identified by hr
3   using 'dbegcc';

Database link created.
```


Ahora haremos una consulta al esquema **hr**.

```
SQL> conn scott/tiger
Connected.

SQL> select employee_id, first_name
  2     from employees@lnk_demo
  3     where department_id = 30;

EMPLOYEE_ID FIRST_NAME
-----
          114 Den
          115 Alexander
          116 Shelli
          117 Sigal
          118 Guy
          119 Karen

6 rows selected.
```

Sinónimos

Facilitan la referencia a tablas.

Sintaxis

```
Create [or replace] [public] synonym [esquema.] sinonimo
for [esquema.] objeto [@dblink]
```

Script 11.11

Crearemos un sinónimo público para acceder a la tabla **employees** del esquema **hr**. Debemos crearlo con **sys**.

```
SQL> conn / as sysdba
Connected.
SQL> create or replace public synonym hr_emp
  2     for employees@lnk_demo;

Synonym created.
```

Ahora como **scott** accederemos a la tabla mediante el sinónimo.

```
SQL> conn scott/tiger
Connected.

SQL> select employee_id, first_name
  2     from hr_emp
  3     where rownum = 1;

EMPLOYEE_ID FIRST_NAME
-----
          100 Steven
```

Cursores

Los cursores permiten realizar recorridos a través de los registros de una tabla.

Procesamiento de Cursores

Pasos a Seguir

1. Declarar el cursor
2. Apertura del cursor
3. Extracción de los resultados
4. Cerrar el cursor

Declarar un Cursor

Sintaxis

```
Cursor nombre_cursor[(Parámetros)]  
is sentencia_select;
```

Script 11.12

```
declare  
  cursor c_demo is select * from dept;
```

Apertura de un Cursor

Sintaxis

```
Open nombre_cursor [(argumentos)];
```

Script 11.13

```
open c_demo;
```

Extracción de datos desde un cursor

Sintaxis 1

```
Fetch nombre_cursor into lista_variables;
```

Sintaxis 2

```
Fetch nombre_cursor into registro;
```

Script 11.14

```
fetch c_demo into cod, nom, loc;
```

Cerrar un cursor

Sintaxis

```
Close nombre_cursor;
```

Script 11.15

```
create or replace procedure pr114
is
  cursor c_demo is select * from dept;
  r dept%rowtype;
begin
  open c_demo;
  fetch c_demo into r;
  close c_demo;
  dbms_output.put_line('deptno: ' || r.deptno);
  dbms_output.put_line('dname: ' || r.dname);
  dbms_output.put_line('loc: ' || r.loc);
end;
```

Ejecución

```
SQL> exec pr114;
deptno: 10
dname:  ACCOUNTING
loc:    NEW YORK

PL/SQL procedure successfully completed.
```

Atributos de los Cursores

%Found	Devuelve TRUE si la última sentencia fetch tuvo éxito.
%NotFound	Devuelve TRUE si la última sentencia fetch no tuvo éxito.
%IsOpen	Este atributo se utiliza para averiguar si un cursor esta abierto ó no.
%RowCount	Este atributo se utiliza para averiguar la cantidad de filas que se van extrayendo del cursor.

Bucles de Extracción

Bucles Simples

Formato

```
open cursor_name ;
loop
  fetch cursor_name into ... ;
  exit when cursor_name%notfound;
  -----
  -----
  -----
end loop;
close cursor_name;
```

Script 11.16

Procedimiento para listar los códigos y nombres de los empleados.

```
create or replace procedure pr115
is
  cursor c_emp is select * from emp;
  r emp%rowtype;
begin
  open c_emp;
  loop
    fetch c_emp into r;
    exit when c_emp%notfound;
    dbms_output.put_line(r.empno || ' - ' || r.ename);
  end loop;
  close c_emp;
end;
```

Ejecución

```
SQL> exec pr115;
7369 - SMITH
7499 - ALLEN
7521 - WARD
7566 - JONES
7654 - MARTIN
7698 - BLAKE
7782 - CLARK
7788 - SCOTT
7839 - KING
7844 - TURNER
7876 - ADAMS
7900 - JAMES
7902 - FORD
7934 - MILLER

PL/SQL procedure successfully completed.
```

Bucles While**Formato**

```
open cursor_name;
fetch cursor_name into ... ;
while cursor_name%found loop
    -----
    -----
    -----
    fetch cursor_name into ... ;
end loop;
close cursor_name;
```

Script 11.17

Tenemos una tabla de nombre **planillames** para guardar la planilla por mes por departamento, la estructura es la siguiente:

```
create table planillames(
    anio number(4),
    mes number(2),
    deptno number(2),
    emps number(2) not null,
    planilla number(10,2) not null,
    constraint pk_planillames primary key(anio,mes, deptno)
);
```

Ahora desarrollaremos un procedimiento para generar la planilla de un determinado mes. Este procedimiento debe verificar si la planilla ya fue generada.

```
create or replace procedure pr116(p_anio number, p_mes number)
is
  cursor c_dept is select deptno from dept;
  v_deptno dept.deptno%type;
  cont number;
  v_emps number;
  v_planilla number;
begin
  select count(*) into cont
    from planillames
   where anio = p_anio and mes = p_mes;
  if (cont > 0) then
    dbms_output.put_line('Ya esta procesado');
    return;
  end if;
  open c_dept;
  fetch c_dept into v_deptno;
  while c_dept%found loop
    select count(*), sum(sal) into v_emps, v_planilla
      from emp
     where deptno = v_deptno;
    insert into planillames
      values(p_anio, p_mes, v_deptno, v_emps, nvl(v_planilla,0));
    fetch c_dept into v_deptno;
  end loop;
  close c_dept;
  commit;
  dbms_output.put_line('Proceso ok.');
```

Ejecución

```
SQL> exec pr116(2005,2);
Proceso ok.

PL/SQL procedure successfully completed.
```

Consultemos el resultado

```
SQL> select * from planillames;
```

ANIO	MES	DEPTNO	EMPS	PLANILLA
2005	2	10	3	8750
2005	2	20	5	10875
2005	2	30	6	9400
2005	2	40	0	0
2005	2	50	0	0

Bucle For

Formato

```
for variable in cursor_name loop
    .....
end loop
```

Script 11.18

Procedimiento para determinar el número de empleados y el importe de la planilla, por departamento.

```
create or replace procedure pr117
is
    cursor c_dept is select * from dept;
    emps number;
    planilla number;
    cad varchar2(100);
begin
    for r in c_dept loop
        select count(*), sum(nvl(sal,0)) into emps, planilla
        from emp where deptno = r.deptno;
        cad := r.deptno || ' - ' || emps || ' - ' || nvl(planilla,0);
        dbms_output.put_line(cad);
    end loop;
end;
```

Ejecución

```
SQL> exec pr117;
10 - 3 - 8750
20 - 5 - 10875
30 - 6 - 9400
40 - 0 - 0
50 - 0 - 0

PL/SQL procedure successfully completed.
```

Ejercicio 1

Desarrollar un procedimiento que determine el empleado con mayor sueldo por departamento.

Bucles For Implícitos

Formato

```
for variable in (sentencia_select) loop
    -----
    -----
end loop
```

Script 11. 19

Determinar el sueldo promedio por departamento.

```
create or replace procedure pr118
is
    prom number;
begin
    for r in (select deptno from dept) loop
        select avg(nvl(sal,0)) into prom
        from emp where deptno = r.deptno;
        dbms_output.put_line(r.deptno || '-' || to_char(nvl(prom,0),'999,990.00'));
    end loop;
end;
```

Ejecución

```
SQL> exec pr118;
10- 2,916.67
20- 2,175.00
30- 1,566.67
40- 0.00
50- 0.00

PL/SQL procedure successfully completed.
```

Cursores Select For Update

Sintaxis

```
for update [of lista_columnas] [nowait | wait n]
```


Script 11.20

Listado de empleados.

```
create or replace procedure pr119
is
  cursor c_demo is select * from emp for update wait 2;
begin
  for r in c_demo loop
    dbms_output.put_line(r.empno || ' ' || r.ename);
  end loop;
end;
```

Antes de ejecutar el procedimiento, en otra ventana inicie una transacción sobre la tabla emp.

```
SQL> exec pr119;
BEGIN pr119; END;

*
ERROR at line 1:
ORA-30006: resource busy; acquire with WAIT timeout expired
ORA-06512: at "SCOTT.PR119", line 3
ORA-06512: at "SCOTT.PR119", line 5
ORA-06512: at line 1;
```

Cursores Implícitos

Se puede utilizar SQL%Atributo para verificar la ejecución de una sentencia SQL.

Script 11.21

Actualizar el salario de un empleado.

```
create or replace procedure pr120(cod number, delta number)
is
begin
  update emp
    set sal = sal + delta
    where empno = cod;
  if sql%notfound then
    dbms_output.put_line('no existe');
  else
    commit;
    dbms_output.put_line('proceso ok');
  end if;
end;
```

Ejecución

```
SQL> exec pr120(7369,200);
proceso ok

PL/SQL procedure successfully completed.
```

Página en Blanco

Oracle 9i Básico PL/SQL

Lección 12 PL/SQL – Tópicos Adicionales

Contenido

Tratamiento de Errores	2
Tipos de Errores	2
Excepciones	2
Esquema General	2
Excepciones Predefinidas	3
Instrucción RAISE	4
Excepciones de Usuario	5
Generación de Mensajes de Error	6
Procedimientos	8
Funciones	9
Parámetros	10
Uso de NOCOPY	11
Paquetes	12
Especificación del Paquete	12
Cuerpo del Paquete	13
Desencadenantes	14
Desencadenante a Nivel de Tabla	14
Desencadenante a Nivel de Esquema	15

Tratamiento de Errores

Tipos de Errores

Tipo de Error	Quién Informa	Cómo es tratado
De Compilación	Compilador PL/SQL	Interactivamente: el compilador informa de los errores y el programador debe corregirlos.
De Ejecución	Motor de Ejecución PL/SQL	Programáticamente: las excepciones son generadas e interceptadas por las rutinas de tratamiento de excepciones.

Excepciones

Es el mecanismo de tratamiento de errores en tiempo de ejecución. Tenemos dos tipos de excepciones: definidas por el usuario y predefinidas.

Esquema General

```
EXCEPTION
  WHEN nombre_excepción THEN
    secuencia_de_instrucciones;
  WHEN nombre_excepción THEN
    secuencia_de_instrucciones;
[ WHEN OTHERS THEN
  secuencia_de_instrucciones; ]
END;
```

Excepciones Predefinidas

Oracle ha definido diversas excepciones que corresponden con los errores Oracle más comunes.

INVALID_CURSOR	Ocorre cuando se hace referencia a un cursor que esta cerrado.
CURSOR_ALREADY_OPEN	Ocorre cuando se trata de abrir un cursor que ya esta abierto.
NO_DATA_FOUND	Ocorre cuando una sentencia SELECT no retorna ninguna fila.
TOO_MANY_ROWS	Ocorre cuando una sentencia SELECT retorna mas de una fila.
VALUE_ERROR	Ocorre cuando hay conflicto de tipos de datos.

Script 12. 1

Desarrollar un procedimiento para consultar el salario de un empleado.

```
create or replace procedure FindEmp( Cod Emp.EmpNo%Type )
is
    Salario Emp.Sal%Type;
Begin
    Select Sal Into Salario
    From Emp
    Where EmpNo = Cod;
    DBMS_Output.Put_Line( 'Salario: ' || Salario );
Exception
    When No_Data_Found Then
        DBMS_Output.Put_Line( 'Código no existe.' );
End;
```

Ejecución

```
SQL> exec FindEmp( 9999 );
Código no existe.

PL/SQL procedure successfully completed.
```

Instrucción RAISE

Permite generar una excepción.

Script 12.2

```
create or replace procedure UpdateSalEmp
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise No_Data_Found;
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
Exception
    When No_Data_Found Then
        DBMS_Output.Put_Line( 'Código no existe.' );
End;
```

Ejecución

```
SQL> exec UpdateSalEmp( 9999, 5000 );
Código no existe.

PL/SQL procedure successfully completed.
```

Excepciones de Usuario

Script 12.3

Desarrollar una segunda versión del procedimiento UpdateSalEmp, pero con una excepción de usuario.

```
create or replace procedure UpdateSalEmp2
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
    Excep1 Exception;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise Excep1;
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
Exception
    When Excep1 Then
        DBMS_Output.Put_Line( 'Código no existe.' );
End;
```

Ejecución

```
SQL> exec UpdateSalEmp2( 9999, 5000 );
Código no existe.

PL/SQL procedure successfully completed.
```

Generación de Mensajes de Error

Script 12.4

Desarrollar una tercera versión del procedimiento UpdateSalEmp, pero esta vez genere un mensaje de error.

```
create or replace procedure UpdateSalEmp3
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise_Application_Error( -20000, 'No existe empleado.' );
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
End;
```

Ejecución

```
SQL> exec UpdateSalEmp3( 9999, 5000 );
BEGIN UpdateSalEmp3( 9999, 5000 ); END;

*
ERROR at line 1:
ORA-20000: No existe empleado.
ORA-06512: at "SCOTT.UPDATESALEMP3", line 10
ORA-06512: at line 1
```


Script 12.5

Otra versión del mismo procedimiento.

```
create or replace procedure UpdateSalEmp4
( Codigo Emp.EmpNo%Type, Salario Emp.Sal%Type )
is
    Cont Number;
Begin
    Select Count(*) Into Cont
    From Emp
    Where EmpNo = Codigo;
    If (Cont=0) Then
        Raise_Application_Error( -20000, 'No existe empleado.' );
    End If;
    Update Emp
    Set Sal = Salario
    Where EmpNo = Codigo;
    Commit;
    DBMS_Output.Put_Line( 'Proceso OK' );
Exception
    When Others Then
        dbms_output.put_line ( 'Error Nro. ORA' || to_char(sqlcode) );
        dbms_output.put_line ( sqlerrm );
End;
```

Ejecución

```
SQL> exec UpdateSalEmp4( 9999, 5000 );
Error Nro. ORA-20000
ORA-20000: No existe empleado.

PL/SQL procedure successfully completed.
```

Procedimientos

Sintaxis

```
CREATE OR REPLACE PROCEDURE nombre_procedimiento (  
    Arg1 [ IN | OUT | IN OUT ] tipo,  
    Arg2 [ IN | OUT | IN OUT ] tipo,  
    ... )  
AS  
  
    Declaraciones_Locales  
  
BEGIN  
  
    Cuerpo_Procedimiento  
  
EXCEPTION  
  
    Tratamiento_Excepciones  
  
END nombre_procedimiento;
```

Script 12.6

```
create or replace procedure adddept(  
    p_deptno dept.deptno%type,  
    p_dname dept.dname%type,  
    p_loc dept.loc%type )  
as  
begin  
    insert into dept(deptno, dname, loc) values(p_deptno, p_dname, p_loc);  
end adddept;
```

Ejecución

```
SQL> exec adddept( 15, 'demo', 'lima');  
  
PL/SQL procedure successfully completed.
```

Funciones

Sintaxis

```
CREATE OR REPLACE FUNCTION nombre_función (  
    Arg1 [ IN | OUT | IN OUT ] tipo,  
    Arg2 [ IN | OUT | IN OUT ] tipo,  
    ... ) RETURN tipo  
AS  
  
    Declaraciones_Locales  
  
BEGIN  
  
    Cuerpo_Función  
  
EXCEPTION  
  
    Tratamiento_Excepciones  
  
END nombre_procedimiento;
```

Script 12.7

```
create or replace function cantemp( p_deptno dept.deptno%type) return number  
as  
    v_cont number;  
begin  
    select count(*) into v_cont from emp where deptno = p_deptno;  
    return v_cont;  
end cantemp;
```

Ejecución 1

```
SQL> declare  
2     v_cont number;  
3     begin  
4         v_cont := cantemp( 10 );  
5         dbms_output.put_line( v_cont || ' empleados' );  
6     end;  
7     /  
3 empleados  
  
PL/SQL procedure successfully completed.
```

Ejecución 2

```
declare
  cursor c_dept is select deptno from dept;
  v_cant number;
begin
  for v_dept in c_dept loop
    v_cant := cantemp(v_dept.deptno);
    dbms_output.put_line( 'dept=' || v_dept.deptno || ' empleados: ' || v_cant );
  end loop;
end;
```

Parámetros

Modo	Descripción
IN	El valor del parámetro real se pasa al procedimiento cuando se produce la llamada al mismo. Dentro del procedimiento, el parámetro formal se comporta como una constante PL/SQL, se considera de solo lectura y no puede ser modificado. Cuando el procedimiento finaliza y devuelve el control al entorno desde donde se produjo la llamada, el parámetro real no se modifica.
OUT	Se ignora cual valor que el parámetro real pueda tener cuando se produce la llamada al procedimiento. Dentro del procedimiento, el parámetro formal se comporta como una variable sin inicializar, por lo que su valor inicial es NULL. Puede leerse y escribir en dicha variable. Cuando el procedimiento finaliza y devuelve el control al entorno desde donde se produjo la llamada, se asigna el valor del parámetro formal al parámetro real.
IN OUT	Este modo es una combinación de los modos IN y OUT.

Script 12.8

```
Create or Replace Procedure prTestOUT1
( p_Raise IN Boolean, p_Dato Out Varchar2 )
Is
  Excep1 Exception;
Begin
  p_Dato := 'Alianza Campeon';
  If p_Raise Then
    Raise Excep1;
  Else
    Return;
  End If;
End;
```

Ejecución 1

```
SQL> Declare
  2   Rpta Varchar2(20) := 'Shakira';
  3   Begin
  4   DBMS_Output.Put_Line( 'Valor Inicial: ' || Rpta );
  5   prTestOUT1( False, Rpta );
  6   DBMS_Output.Put_Line( 'Valor Final: ' || Rpta );
  7   End;
  8   /
Valor Inicial: Shakira
Valor Final: Alianza Campeon

PL/SQL procedure successfully completed.
```

Ejecución 2

```
SQL> Declare
  2   Rpta Varchar2(20) := 'Shakira';
  3   Begin
  4   DBMS_Output.Put_Line( 'Valor Inicial: ' || Rpta );
  5   prTestOUT1( True, Rpta );
  6   DBMS_Output.Put_Line( 'Valor Final: ' || Rpta );
  7   Exception
  8   When Others Then
  9   DBMS_Output.Put_Line( 'Valor Después del error: ' || Rpta );
 10   End;
 11   /
Valor Inicial: Shakira
Valor Después del error: Shakira

PL/SQL procedure successfully completed.
```

Uso de NOCOPY

Script 12.9

```
Create or Replace Procedure prTestOUT2
( p_Raise IN Boolean, p_Dato Out NOCOPY Varchar2 )
Is
  Excep1 Exception;
Begin
  p_Dato := 'Alianza Campeon';
  If p_Raise Then
    Raise Excep1;
  Else
    Return;
  End If;
End;
```

Ejecución 1

```
SQL> Declare
  2   Rpta Varchar2(20) := 'Shakira';
  3   Begin
  4   DBMS_Output.Put_Line( 'Valor Inicial: ' || Rpta );
  5   prTestOUT2( False, Rpta );
  6   DBMS_Output.Put_Line( 'Valor Final: ' || Rpta );
  7   End;
  8   /
Valor Inicial: Shakira
Valor Final: Alianza Campeon

PL/SQL procedure successfully completed.
```

Ejecución 2

```
SQL> Declare
  2   Rpta Varchar2(20) := 'Shakira';
  3   Begin
  4   DBMS_Output.Put_Line( 'Valor Inicial: ' || Rpta );
  5   prTestOUT2( True, Rpta );
  6   DBMS_Output.Put_Line( 'Valor Final: ' || Rpta );
  7   Exception
  8   When Others Then
  9   DBMS_Output.Put_Line( 'Valor Después del error: ' || Rpta );
 10   End;
 11   /
Valor Inicial: Shakira
Valor Después del error: Alianza Campeon

PL/SQL procedure successfully completed.
```

Paquetes

Es una estructura PL/SQL que permite almacenar definiciones, funciones y procedimientos relacionados como una sola unidad.

Especificación del Paquete

Sintaxis

```
CREATE OR REPLACE PACKAGE nombre_paquete AS

  Definiciones

END nombre_paquete;
```

Script 12.10

```
CREATE OR REPLACE PACKAGE testpackage as
    function suma( n1 in number, n2 in number ) return number;
END testpackage;
```

Cuerpo del Paquete

Sintaxis

```
CREATE OR REPLACE PACKAGE BODY nombre_paquete AS

    Implementación

END nombre_paquete;
```

Script 12.11

```
CREATE OR REPLACE PACKAGE BODY testpackage as

    function suma( n1 in number, n2 in number ) return number
    as
        rtn number;
    begin
        rtn := n1 + n2;
        return rtn;
    end;

END testpackage;
```

Ejecución 1

```
SQL> select testpackage.suma( 12,13) from dual;

TESTPACKAGE.SUMA(12,13)
-----
                25
```

Ejecución 2

```
SQL> declare
2     v_suma number;
3     begin
4         v_suma := testpackage.suma( 12,13);
5         dbms_output.put_line ( 'Suma: ' || v_suma );
6     end;
7     /
Suma: 25

PL/SQL procedure successfully completed.
```

Desencadenantes

Desencadenante a Nivel de Tabla

Script 12.12

```
CREATE OR REPLACE TRIGGER tr_test_emp
AFTER INSERT OR DELETE OR UPDATE ON emp
BEGIN

    if inserting then
        dbms_output.put_line( 'nuevo empleado se ha insertado' );
    end if;

    if updating then
        dbms_output.put_line( 'un empleado se ha modificado' );
    end if;

    if deleting then
        dbms_output.put_line( 'un empleado se ha eliminado' );
    end if;

END tr_test_emp;
```

Ejecución 1

```
SQL> insert into emp( empno, ename ) values ( 1234, 'Sergio' );
nuevo empleado se ha insertado

1 row created.
```

Ejecución 2

```
SQL> update emp
2     set ename = 'Hugo'
3     where empno = 1234;
un empleado se ha modificado

1 row updated.
```

Ejecución 3

```
SQL> delete from emp
2     where empno = 1234;
un empleado se ha eliminado

1 row deleted.
```


Desencadenante a Nivel de Esquema

Script 12.13

```
Create or Replace Trigger tr_drop_object
Before drop on Scott.Schema
Begin
    Raise_Application_Error( -20000, 'No se puede eliminar el objeto !!!' );
End;
```

Ejecución

```
SQL> drop table emp;
drop table emp
*
ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-20000: No se puede eliminar el objeto !!!
ORA-06512: at line 2
```

Página en Blanco