

# Oracle Database 10g

## S Q L

Primera Edición

Eric Gustavo Coronel Castillo

Febrero - 2009

## **Oracle Database 10g SQL**

Derechos Reservados © 2009 Eric Gustavo Coronel Castillo

**Primera Edición**

### **Contacto**

Email: [gcoronelc@gmail.com](mailto:gcoronelc@gmail.com)

Teléfono: (511) 996-664-457

Lima - Perú

## 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 9 lecciones, donde veremos de una manera práctica el lenguaje SQL y la creación de esquemas de base de datos, no pretende ser un texto de consulta teórica, sino más bien, una guía de práctica de laboratorio.

Sería ingrato no mencionar los aportes de mis amigos y colegas Sergio Matsukawa, Ricardo Marcelo, Fortunato Veliz, Julio Flores 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: gcoronelc@gmail.com, me sería de mucha utilidad conocer sus opiniones para poder mejorar mis futuras publicaciones.

Atentamente,

Eric Gustavo Coronel Castillo  
gcoronelc@gmail.com

# Resumen

- Capítulo 01** En esta lección se describe la estructura de la base de datos Oracle 10g: la instancia y la base de datos; así como una breve descripción de los procesos y componentes de la base de datos. También se describe aspectos sobre los servicios que debemos verificar y la conexión con la base de datos desde SQP\*Plus, y conceptos generales sobre el almacenamiento de datos.
- Capítulo 02** En esta lección se detalla los esquemas de ejemplo que tiene servidor Oracle para desarrollar nuestros ejercicios, cabe mencionar que muchos textos y ejemplos en diversos artículos están desarrollados con estos esquemas.
- Capítulo 03** En esta lección se desarrolla consultas básicas, específicamente consultas a una sola tabla y aplicando la cláusula where y diversos operadores para construir filtros.
- Capítulo 04** En esta lección se estudia la aplicaciones de funciones: funciones de cadenas, funciones de fechas, funciones de conversión, etc.
- Capítulo 05** Una de las tareas comunes que se desarrollan en los diversos tipos de aplicaciones es el proceso de los datos y obtener diversos tipos resúmenes. En esta lección se verá como aplicar las cláusulas group by y having para obtener resúmenes de datos.
- Capítulo 06** Definitivamente la mayoría de consultas se realizan a diversas tablas de manera simultáneamente, en esta lección veremos como desarrollar este tipo de consultas.
- Capítulo 07** Existen consultas que parecen imposibles, pero la aplicación de subconsultas nos ayuda a resolver muchas de este tipo de consultas. Esta lección esta dedicada al tema de subconsultas.
- Capítulo 08** La tarea común en los sistemas de información es la manipulación de datos, y esto se realiza a través de transacciones. Esta lección se centra en el manejo de transacciones y como desarrollar operaciones de manipulación de datos que involucren a una ó más tablas.
- Capítulo 09** La creación de esquemas de base de datos es también una operación muy importante en las bases de datos Oracle. En esta lección veremos como crear un esquema, sus objetos y asignación de permisos a otros usuarios para que puedan manipular los objetos del esquema.

# Contenido

## Lección 01: Aspectos Generales de Oracle 10g

Arquitectura de un servidor Oracle .....	2
La instancia de Oracle .....	2
Conexión con una instancia de Oracle .....	5
Conceptos generales de almacenamiento .....	12

## Lección 02: Esquemas Ejemplos de la Base de Datos

Esquema de Base de Datos .....	16
Esquema SCOTT .....	16
Esquema HR .....	18
Consultar la Estructura de una Tabla .....	20
Consultar el Contenido de una Tabla .....	20

## Lección 03: Sentencias SQL SELECT Básicas

SQL Fundamentos .....	22
Escribiendo Consultas Simples .....	25
Otros Operadores .....	31
Ordenando Filas .....	33
Usando Expresiones .....	35

## Lección 04: Funciones Simples de Fila

Funciones para Valores Nulos .....	38
Funciones para Caracteres .....	39
Funciones Numéricas .....	41
Funciones de Fecha .....	42
Funciones de Conversión .....	42
Otras Funciones .....	47

## Lección 05: Totalizando Datos y Funciones de Grupo

Funciones de Grupo .....	52
GROUP BY .....	54
HAVING .....	55

## Lección 06: Consultas Multitablas

¿Qué es un Join? .....	57
Consultas Simples .....	58
Consultas Complejas .....	59
Producto Cartesiano .....	63
Combinaciones Externas .....	65
Otras Consultas Multitablas .....	67
Operadores de Conjuntos .....	68

## Lección 07: Subconsultas

Subconsultas de Solo una Fila .....	72
Subconsultas de Múltiples Filas .....	72
Subconsultas Correlacionadas .....	73
Subconsultas Escalares .....	73

## Lección 08: Modificando Datos

Insertando Filas .....	78
Modificando Datos .....	81
Eliminando Filas .....	85
Transacciones .....	88

## Lección 09: Creación de un Esquema de Base de Datos

Caso a Desarrollar .....	92
Creación del Usuario para el Esquema .....	93
Creación de Tablas .....	94
Restricción Primary Key (PK) .....	96
Restricción Foreign Key (FK) .....	98
Restricción Default (Valores por Defecto) .....	100
Restricción NOT NULL (Nulidad de una Columna) .....	101
Restricción Unique (Valores Únicos) .....	102
Restricción Check (Reglas de Validación) .....	103
Asignar Privilegios a Usuarios .....	104



# Oracle Database 10g SQL

---

## Capítulo 01 Aspectos Generales

### Contenido

Arquitectura de un servidor Oracle

- Esquema General

La instancia de Oracle

- Procesos de fondo

- Area Global del Sistema (SGA)

- La base de datos

- Estructuras Adicionales

Conexión con una instancia de Oracle

- Verificación de los servicios

- Esquema General

- Conexión local utilizando SQL Plus

  - Vistas del Sistema

  - Comandos SQL/Plus

- Conexión remota utilizando SQL Plus

- Conexión Utilizando iSQL\*Plus

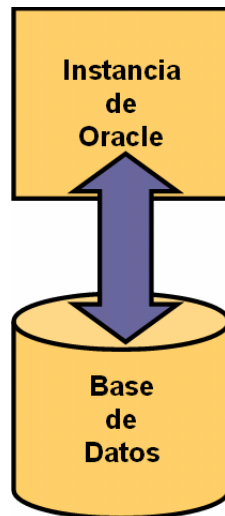
Conceptos generales de almacenamiento

- TableSpace

- DataFile

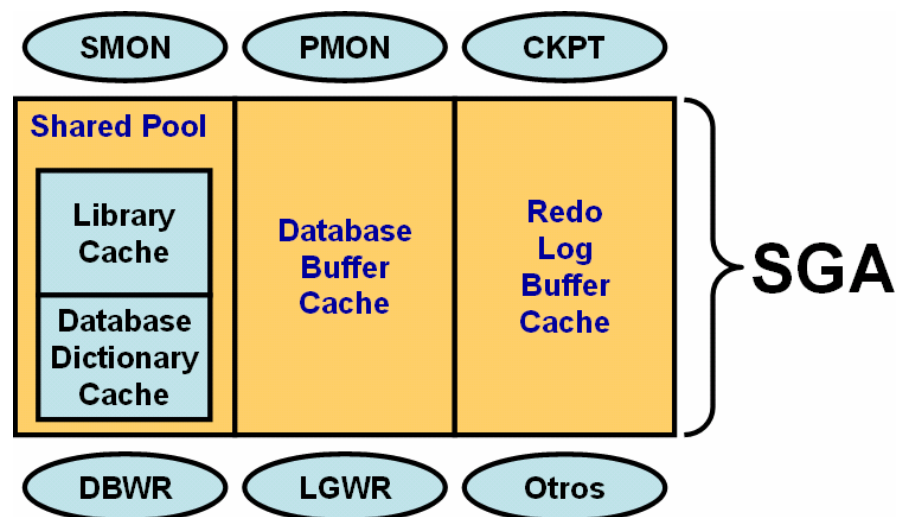
## Arquitectura de un servidor Oracle

### 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



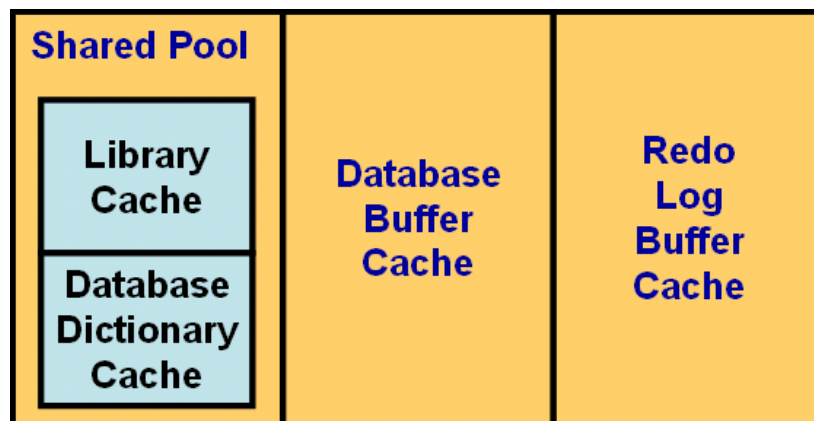
- Esta compuesta por procesos de fondo y un área de memoria compartida denominada SYSTEM GLOBAL AREA (SGA).
- El SGA es utilizado para el intercambio de datos entre el servidor y las aplicaciones cliente.
- Una instancia de Oracle solo puede abrir una sola base de datos a la vez.



## Procesos de fondo

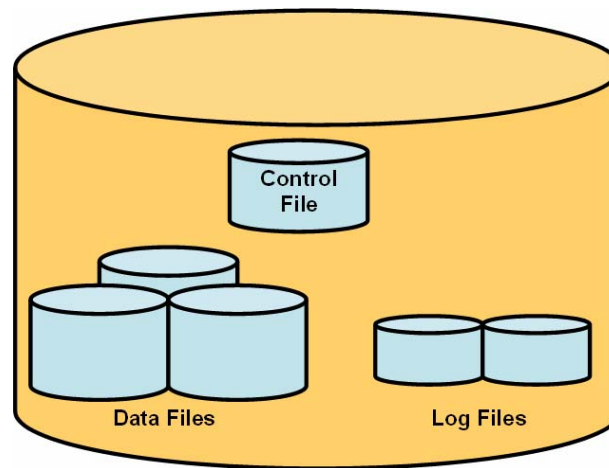
<b>PMON</b>	<i>Process Monitor.</i> Monitorea los procesos de los usuarios, en caso que la conexión falle.
<b>SMON</b>	<i>System Monitor.</i> Este proceso es el encargado de recuperar la instancia y abrir la base de datos, en caso que ocurra alguna falla.
<b>CKPT</b>	<i>CheckPoint Process.</i> Sintoniza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

## Área Global del Sistema (SGA)



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

## La base de datos



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

## Estructuras Adicionales

<b>Archivo de Parámetros</b>	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.
<b>Archivo de Password</b>	Se utiliza para validar al usuario que puede bajar y subir la instancia de Oracle.
<b>Archivos Archived Log Files</b>	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.

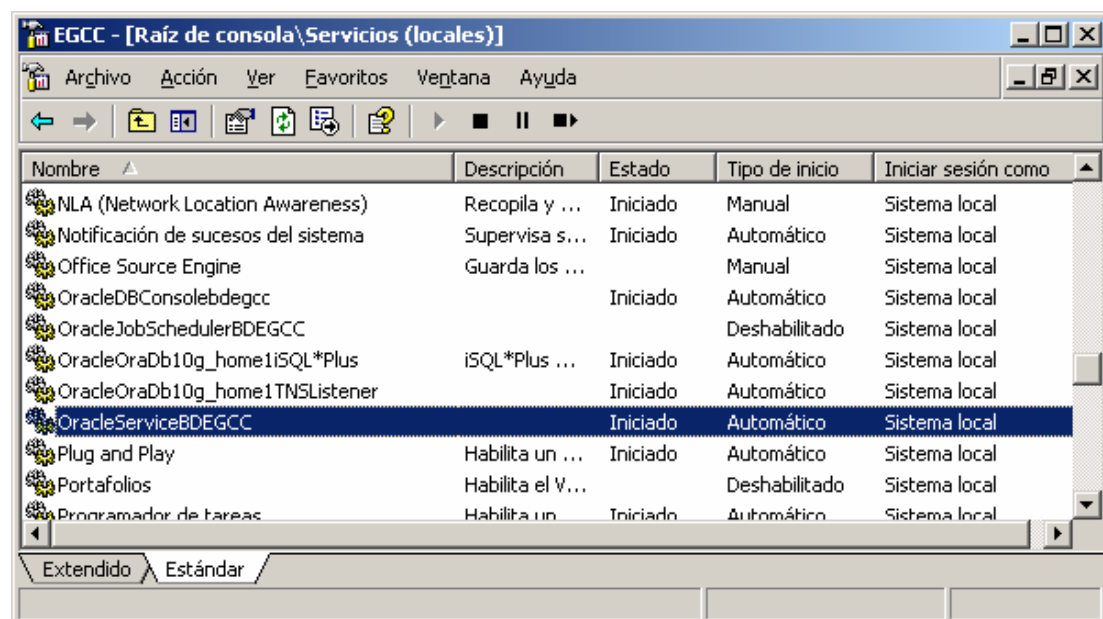
## Conexión con una instancia de Oracle

### Verificación de los servicios

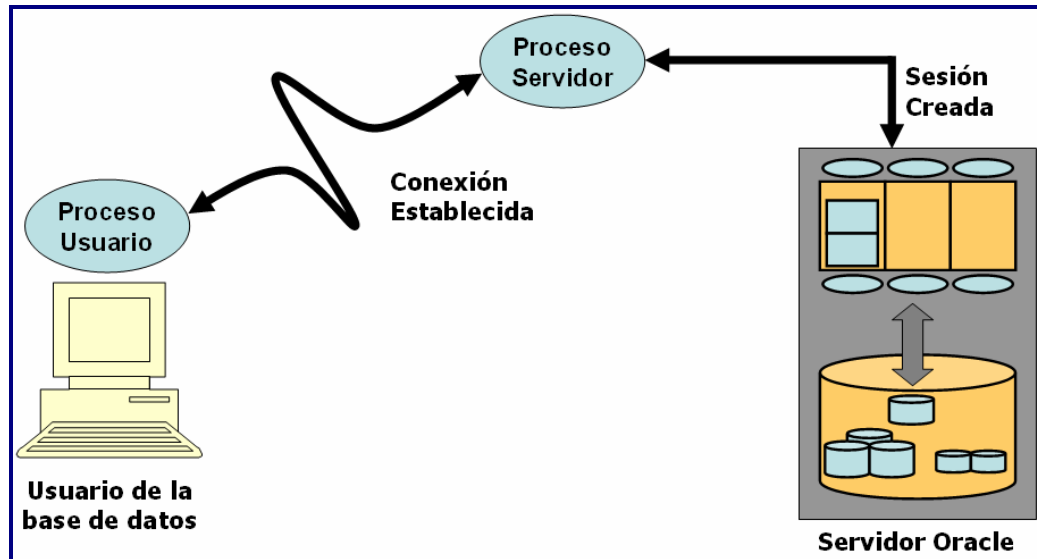
De la relación de servicios creados durante la instalación de Oracle, por ahora nos interesa básicamente tres:

- 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 **BDEGCC**, el servicio sería **OracleServiceBDEGCC**.
- El servicio relacionado con la disponibilidad del servidor para el acceso remoto, el nombre de este servicio es: **OracleOraDb10g\_home1TNSListener**.
- El servicio relacionado con la aplicación **iSQL\*Plus**, este servicio permite ejecutar esta aplicación desde cualquier equipo de la red vía el protocolo HTTP haciendo uso de un navegador Web, el nombre de este servicio es **OracleOraDb10g\_home1iSQL\*Plus**.

Estos tres 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



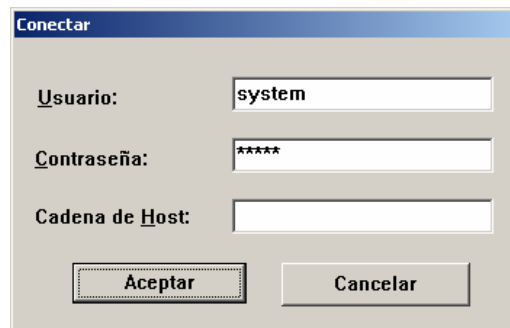
<b>Proceso Usuario</b>	Programa, aplicación ó herramienta que usa el usuario para iniciar un proceso de usuario y establecer una conexión.
<b>Proceso Servidor</b>	<p>Una vez que el proceso de usuario establece la conexión, un proceso servidor es iniciado, el cual manejará las peticiones del proceso usuario.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

## Conexión local utilizando SQL Plus

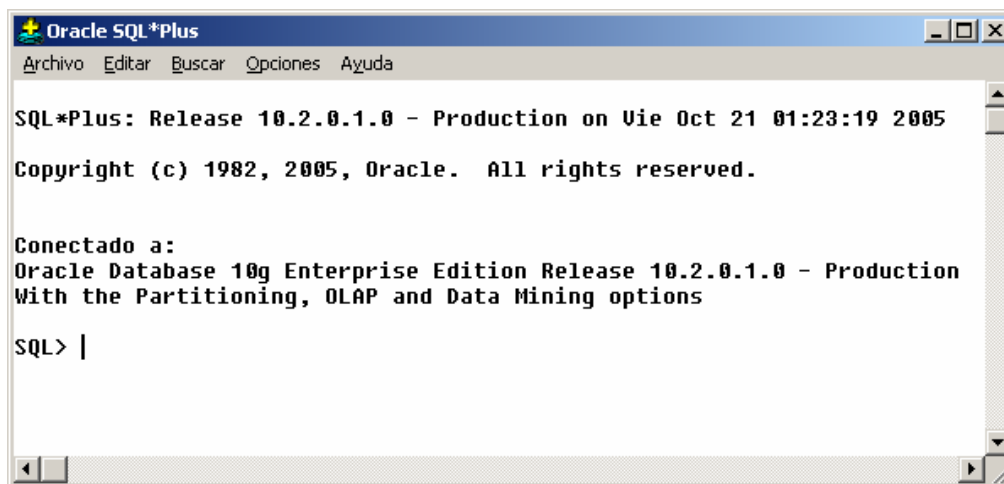
SQL Plus es una herramienta que permite al usuario comunicarse con el servidor, para procesar comandos SQL ó PL/SQL, tiene la flexibilidad de poder realizar inicio y parada (shutdown) de la base de datos.

En la ventana inicial de conexión debemos ingresar el usuario y su contraseña, debe recordar la contraseña que estableció para los usuarios **sys** y **system**.

Usuario	Contraseña
sys	admin.
system	admin



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> show user
USER es "SYSTEM"

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:

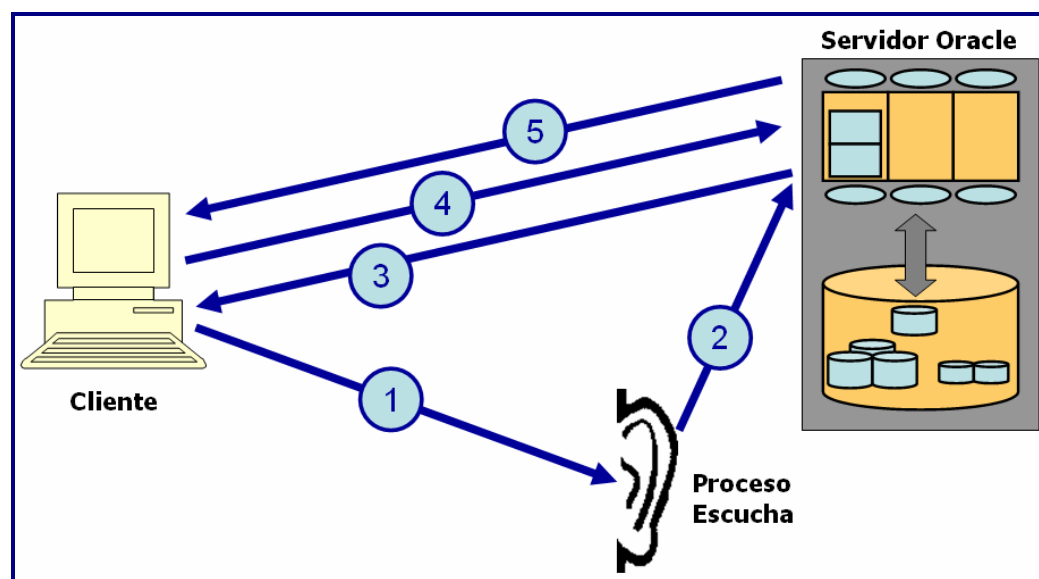
- RUN**        Vuelve a ejecutar la última instrucción ejecutada.
- EDIT**       Edita la última instrucción ejecutada.
- START**      Ejecuta las instrucciones que se encuentran en un archivo.
- SPOOL**      Envía la sesión de trabajo a un archivo.

## 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 **OracleOraDb10g\_home1TNSListener**.

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



El proceso se describe a continuación:

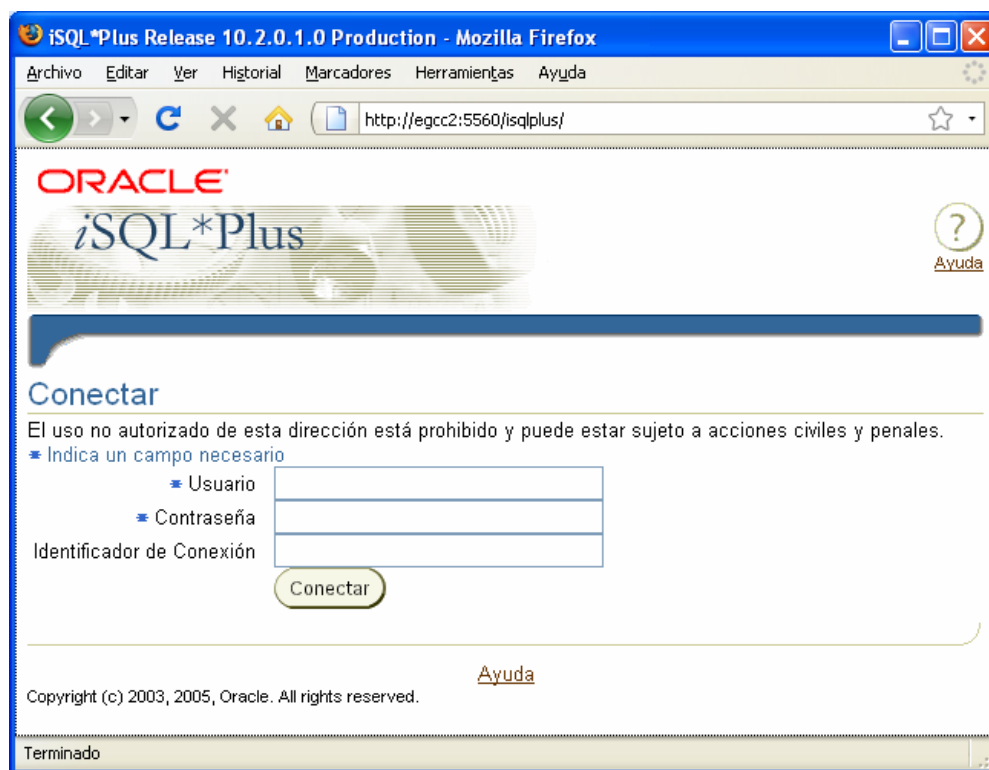
1. El cliente establece una conexión al Proceso Escucha usando el protocolo configurado y envía un paquete CONNECT.
2. El proceso escucha comprueba que el SID esté definido. Si es así, generará un nuevo proceso para ocuparse de la conexión. Una conexión se establece entre el proceso escucha y el nuevo 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.

## Conexión Utilizando iSQL\*Plus

Para utilizar **iSQL\*Plus** es necesario que el servicio **OracleOraDb10g\_home1iSQL\*Plus** se encuentre **Iniciado**, y la conexión se puede realizar desde cualquier equipo de la red utilizando un Navegador Web y escribiendo la siguiente **URL** en el campo **Dirección**:

**http://nombre\_servidor:5560/isqlplus**

La ventana inicial que se muestra en la siguiente figura, solicita el **Usuario** y **Contraseña** para poder iniciar sesión.





Después de iniciar sesión ingresa a la aplicación, tal como se ilustra en la siguiente figura:



Desde aquí usted podrá ejecutar sentencias SQL y bloques de programa PL/SQL.

## 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;

      TS# NAME                                INC BIG FLA ENC
-----
      0 SYSTEM                                YES NO  YES
      1 UNDOTBS1                              YES NO  YES
      2 SYSAUX                                YES NO  YES
      4 USERS                                 YES NO  YES
      3 TEMP                                  NO  NO  YES
      6 EXAMPLE                                YES NO  YES

6 filas seleccionadas.

SQL> select tablespace_name from dba_tablespaces
      2 order by 1;

TABLESPACE_NAME
-----
EXAMPLE
SYSAUX
SYSTEM
TEMP
UNDOTBS1
USERS

6 filas seleccionadas.
```

## DataFile

Es el archivo físico donde se almacenas los datos.

```
SQL> column tablespace_name format a20
SQL> column file_name format a55
SQL> select tablespace_name, file_name from dba_data_files
       2 order by 1;
```

TABLESPACE_NAME	FILE_NAME
EXAMPLE	H:\ORACLE\PRODUCT\10.2.0\ORADATA\BDEGCC\EXAMPLE01.DBF
SYS_AUX	H:\ORACLE\PRODUCT\10.2.0\ORADATA\BDEGCC\SYS_AUX01.DBF
SYSTEM	H:\ORACLE\PRODUCT\10.2.0\ORADATA\BDEGCC\SYSTEM01.DBF
UNDOTBS1	H:\ORACLE\PRODUCT\10.2.0\ORADATA\BDEGCC\UNDOTBS01.DBF
USERS	H:\ORACLE\PRODUCT\10.2.0\ORADATA\BDEGCC\USERS01.DBF

```
SQL>
```

*Página en Blanco*



## Oracle Database 10g SQL

---

### Capítulo 02 Esquemas Ejemplos de la Base de Datos

#### Contenido

Esquema de Base de Datos

Esquema SCOTT

Esquema HR

Consultar la Estructura de una Tabla

Consultar el Contenido de una Tabla

## 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.

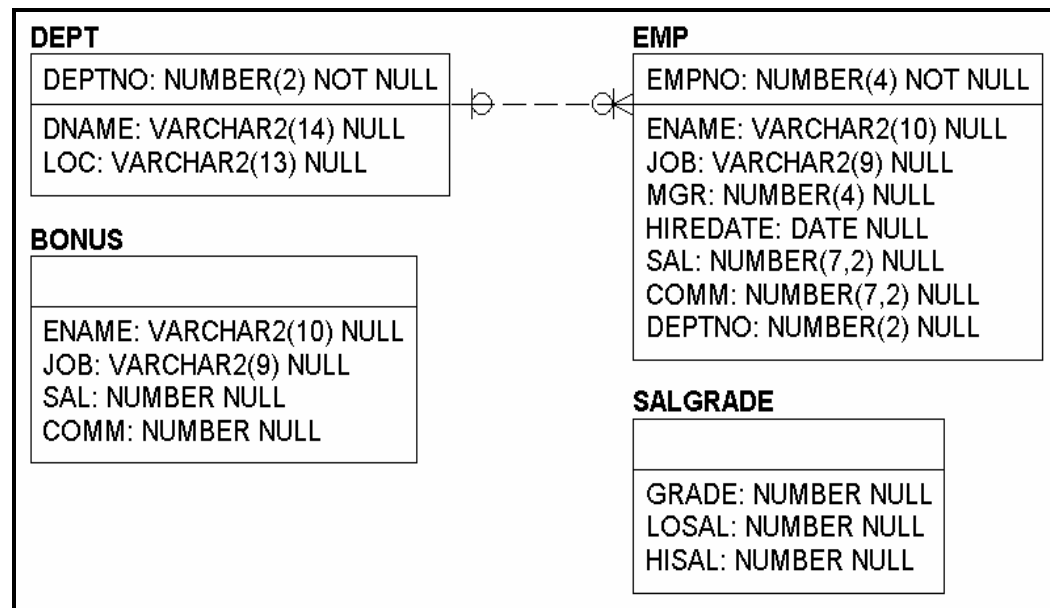
**HR** Se trata también de un esquema de recursos humanos, pero este esquema cuenta con 7 tablas.

## Esquema SCOTT

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

<b>Usuario</b>	scott
<b>Contraseña</b>	tiger

Su esquema es el siguiente:



El siguiente script permite consultar el catalogo de scott:

### Script 2.1

```
SQL> conn / as sysdba
Conectado.

SQL> alter user scott
2 account unlock;

Usuario modificado.

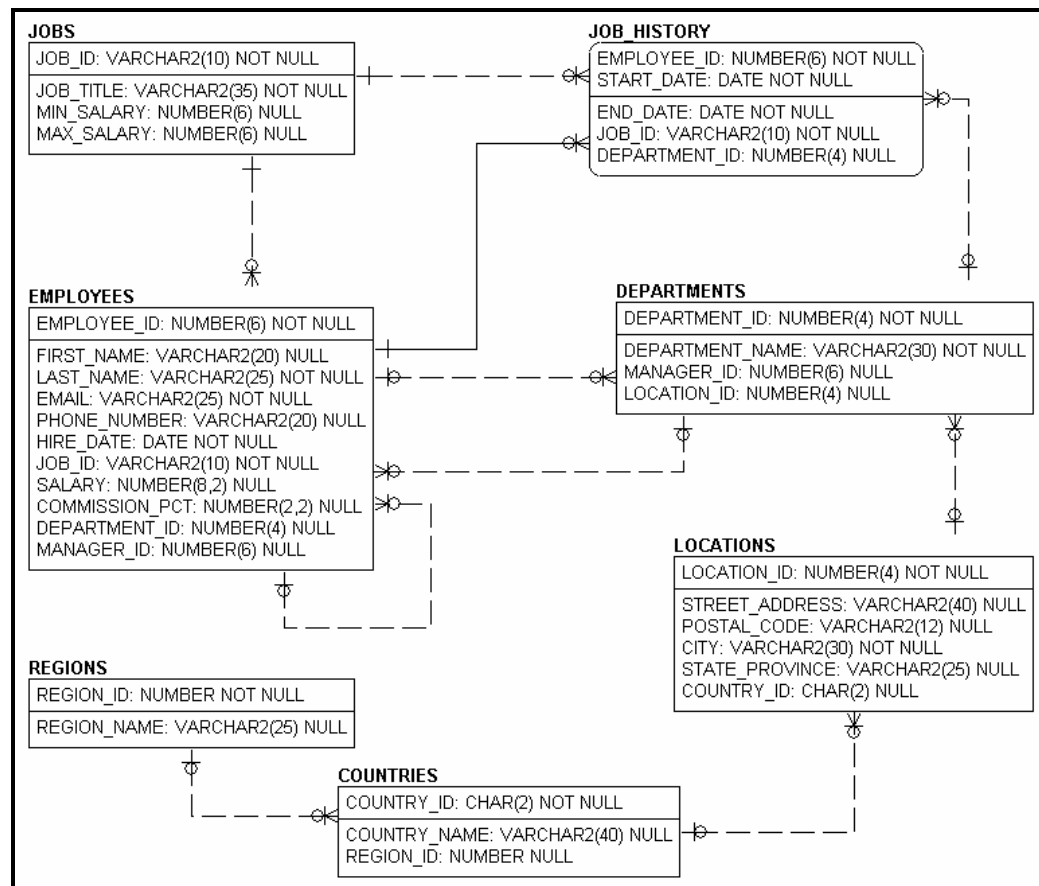
SQL> connect scott/tiger
Connected.

SQL> select * from cat;
```

TNAME	TABTYPE	CLUSTERID
DEPT	TABLE	
EMP	TABLE	
BONUS	TABLE	
SALGRADE	TABLE	

## 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 / as sysdba
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



# Oracle Database 10g SQL

---

## Capítulo 03 Sentencias SQL SELECT Básicas

### Contenido

#### SQL Fundamentos

- Tipos de Datos de Oracle

- Operadores y Literales

#### Escribiendo Consultas Simples

- Usando la Sentencia SELECT

- Limitando las Filas

#### Otros Operadores

- IN y NOT IN

- BETWEEN

- EXISTS

- IS NULL y IS NOT NULL

- LIKE

#### Ordenando Filas

- Ordenando Nulos

#### Usando Expresiones

- La Expresión CASE

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

## 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_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
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 Categoría  
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 Database 10g SQL

---

## Capítulo 04 Funciones Simples de Fila

### Contenido

#### Funciones para Valores Nulos

- Funciones NVL

- Función NVL2

#### Funciones para Caracteres

#### Funciones Numéricas

#### Funciones de Fecha

- Conversión de Formato de Fecha

- Add\_Months

- Current\_Date

- Current\_Timestamp

- Extract

- Last\_Day

- Month\_Between

- SysDate

#### Funciones de Conversión

- Cast

- To\_Char

- Conversión de Datos Tipo Fecha

- Conversión de Datos Numéricos

- To\_Date

- To\_Number

#### Otras Funciones

- NULLIF

- Sys\_Connect\_By\_Path

- Sys\_Context

- UID

- User

## Funciones para Valores Nulos

### Funciones NVL

Reemplaza 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

Reemplaza un valor nulo por otro valor, si no es nulo también lo reemplaza 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.	Abs(-5) = 5
ACOS	Retorna el arco coseno.	ACos(-1) = 3.14159265
ASIN	Retorna el arco seno.	ASin(1) = 1.57079633
ATAN	Retorna el arco tangente.	ATan(0) = 0
ATAN2	Retorna el arco tangente; tiene dos valores de entrada.	ATan2(0,3.1415) = 0
BITAND	Retorna el resultado de una comparación a nivel de bits de números.	BitAnd(3,9) = 1
CEIL	Retorna el siguiente entero más alto.	Ceil(5.1) = 6
COS	Retorna el coseno de un ángulo.	Cos(0) = 1
COSH	Retorna el coseno hiperbólico.	Cosh(1.4) = 2.15089847
EXP	Retorna la base del logaritmo natural elevado a una potencia.	Exp(1) = 2.71828183
FLOOR	Retorna el siguiente entero más pequeño.	Floor(5.31) = 5
LN	Retorna el logaritmo natural.	Ln(2.7) = 0.99325177
LOG	Retorna el logaritmo.	Log(8,64) = 2
MOD	Retorna el residuo de una operación de división.	Mod(13,5) = 3
POWER	Retorna un número elevado a una potencia.	Power(2,3) = 8
ROUND	Redondea un número.	Round(5467,-2) = 5500 Round(56.7834,2) = 56.78
SIGN	Retorna el indicador de signo de un número.	Sign(-456) = -1
SIN	Retorna el seno de un ángulo.	Sin(0) = 0
SQRT	Retorna el seno hiperbólico.	Sqrt(16) = 4
TAN	Retorna la tangente de un ángulo.	Tan(0.785398165) = 1
TANH	Retorna la tangente hiperbólica.	Tanh(Acos(-1)) = 0.996272076
TRUNC	Trunca un número.	Trunc(456.678,2) = 456.67 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                TRESMESESESPUES  
-----  
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 unívocamente 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 Database 10g SQL

---

# Capítulo 05 Totalizando Datos y Funciones de Grupo

### Contenido

Funciones de Grupo

AVG

COUNT

MAX

MIN

SUM

GROUP BY

HAVING

## 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.

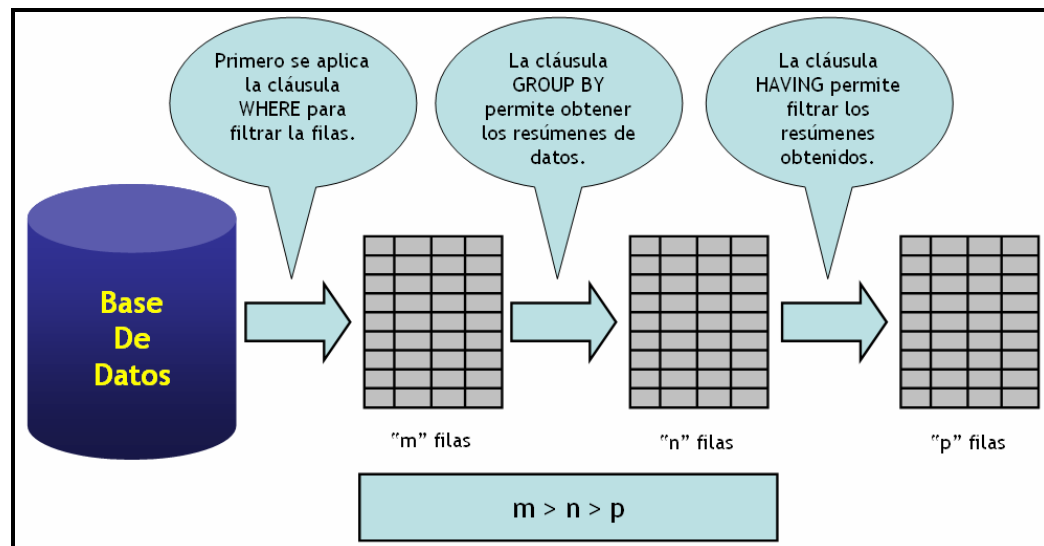
```
SQL> select
2      extract(year from hire_date) as año,
3      count(*) as empleados
4  from employees
5  group by extract(year from hire_date);
```

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 Database 10g SQL

---

## Capítulo 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?

- Consultas Simples

- Consultas Complejas

  - Uso de Alias

  - Usando Sintaxis ANSI

    - NATURAL JOIN

    - JOIN . . . USING

    - JOIN ... ON

- Producto Cartesiano

- Combinaciones Externas

  - Usando Sintaxis ANSI

    - Left Outer Joins

    - Right Outer Join

    - Full Outer Join

- Otras Consultas Multitablas

  - Autoreferenciadas (Self-joins)

  - Consultas Basadas en Desigualdades (Nonequality Joins)

- Operadores de Conjuntos

## ¿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 una 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 fila 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 Database 10g SQL

---

## Capítulo 07 Subconsultas

### Contenido

Subconsultas de Solo una Fila

Subconsultas de Múltiples Filas

Subconsultas Correlacionadas

Subconsultas Escalares

- Subconsulta Escalar en una Expresión CASE

- Subconsulta Escalar en la Cláusula SELECT

- Subconsultas Escalares en las Cláusulas SELECT y WHERE

- Subconsultas Escalares en la Cláusula ORDER BY

- Múltiples Columnas en una Subconsultas

## 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	Fripp
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 Database 10g SQL

---

## Capítulo 08 Modificando Datos

### Contenido

#### Insertando Filas

- Insertiones una Sola Fila
- Insertando Filas con Valores Nulos
- Insertando Valores Especiales
- Insertando Valores Específicos de Fecha
- Usando & Sustitución para el Ingreso de Valores
- Copiando Filas Desde Otra Tabla
- Insertando en Múltiples Tablas

#### Modificando Datos

- Actualizando una Columna de una Tabla
- Seleccionando las Filas a Actualizar
- Actualizando Columnas con Subconsultas
- Actualizando Varias Columnas con una Subconsulta
- Error de Integridad Referencial

#### Eliminando Filas

- Eliminar Todas la Filas de una Tabla
- Seleccionando las Filas a Eliminar
- Uso de Subconsultas
- Error de Integridad Referencial
- Truncando una Tabla

#### Transacciones

#### Propiedades de una Transacción

#### Operación de Transacciones

- Inicio de una transacción
- Confirmación de una transacción
- Cancelar una transacción

## 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;

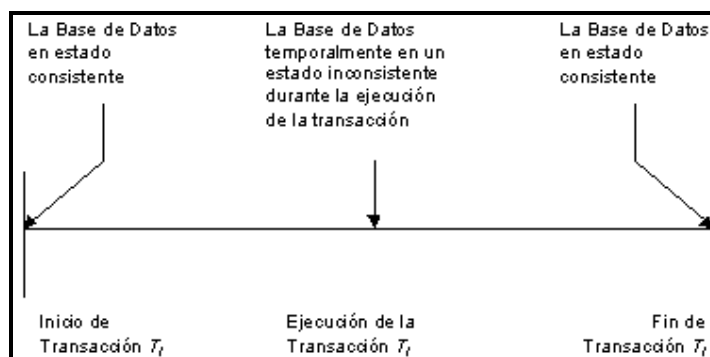
Table truncated.

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

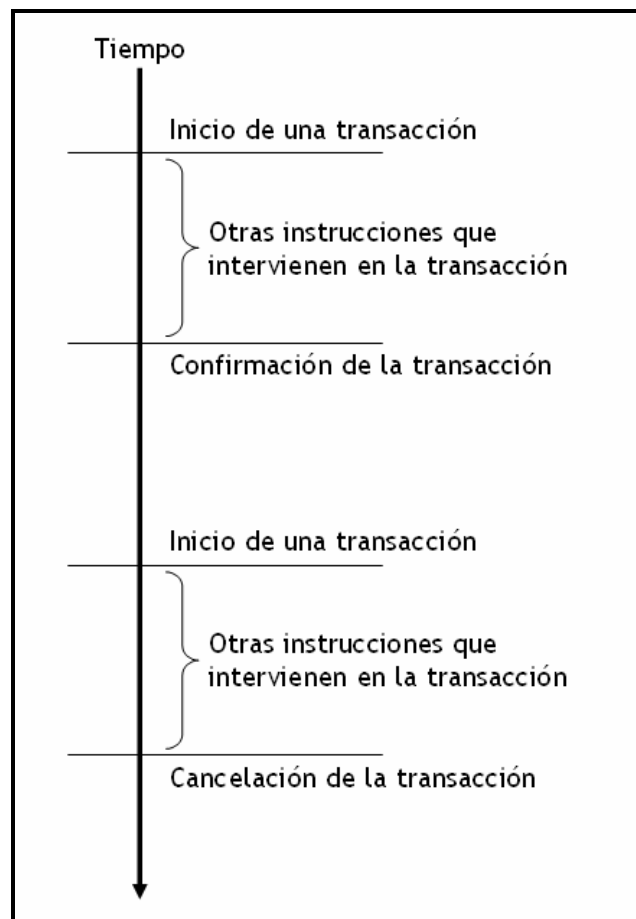
Una transacción es una unidad de aislamiento y cada una se produce aislada e independientemente de las transacciones concurrentes. Una transacción nunca debe ver las fases intermedias de otra transacción.

## 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 Database 10g SQL

---

## Capítulo 09 Creación de un Esquema de Base de Datos

### Contenido

#### Caso a Desarrollar

- Modelo Lógico

- Modelo Físico

#### Creación del Usuario para el Esquema

- Creación del Usuario

- Asignar Privilegios

#### Creación de Tablas

- Tabla Curso

- Tabla Alumno

- Tabla Matricula

- Tabla Pago

#### Restricción Primary Key (PK)

- Tabla Curso

- Tabla Alumno

- Tabla Matricula

- Tabla Pago

#### Restricción Foreign Key (FK)

- Tabla Matricula

- Tabla Pago

#### Restricción Default (Valores por Defecto)

- Ejemplo

#### Restricción NOT NULL (Nulidad de una Columna)

- Ejemplo

#### Restricción Unique (Valores Únicos)

- Ejemplo

#### Restricción Check (Reglas de Validación)

- Ejemplo

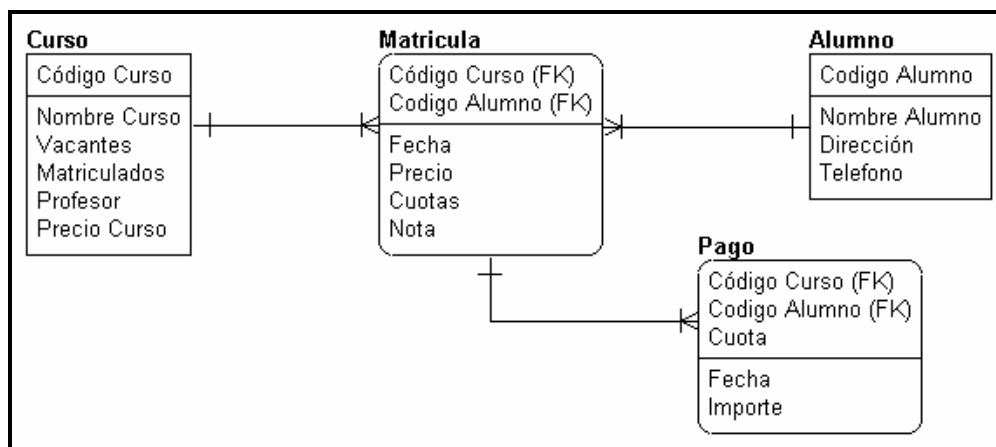
#### Asignar Privilegios a Usuarios

- Ejemplo

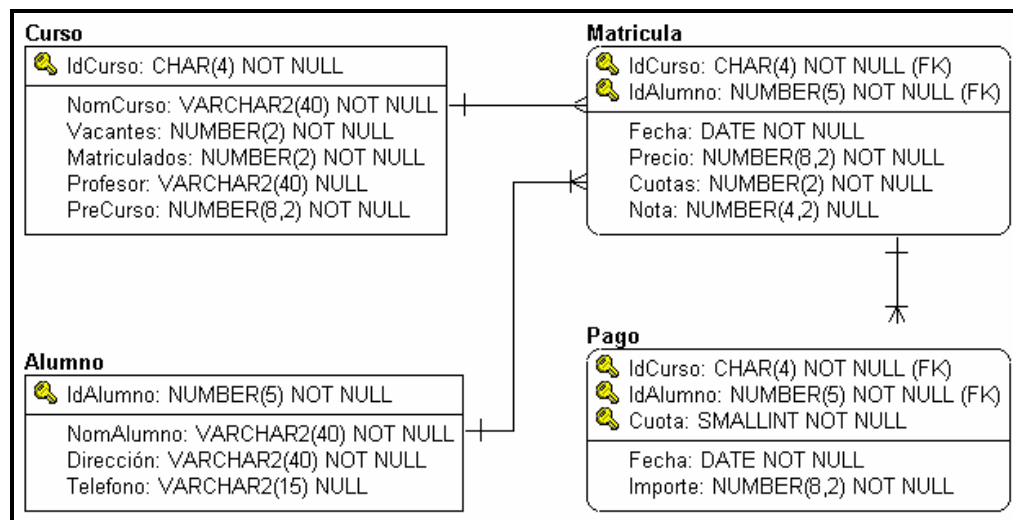
## Caso a Desarrollar

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 ( Column1, Column2, . . . );
```

## 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_NombreTable_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