

# Oracle Lenguaje SQL y PLSQL (90 horas)



ORACLE®

## Objetivos del Curso

Este curso está orientado a que los alumnos:

1. Desarrollen aplicaciones en Lenguaje SQL y PLSQL que trabajen con la base de datos Oracle Database
2. Adquieran todo el conocimiento necesario de cara a la programación contra la base de datos Oracle Database.
3. Conozcan las técnicas para aumentar el rendimiento de códigos SQL y PLSQL.

## Oracle Lenguaje SQL y PLSQL

Módulos	Horas	Contenido
1	30	Oracle SQL Workshop I y II
2	30	Oracle PL/SQL Fundamentals
3	30	Developer PL/SQL Program Units

1 - 3

# 1

## Introducción



ORACLE®

## Objetivos de la Lección

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Definir los objetivos del curso
- Enumerar las funciones de base de datos Oracle 12c
- Describir las características más destacadas Oracle Cloud
- Discutir los aspectos teóricos y físicos de una base de datos relacional
- Identificar los entornos de desarrollo que se pueden utilizar para este curso
- Describir la base de datos y el esquema utilizado en este curso

1 - 5

## Agenda

- Objetivos del curso, orden del día, y apéndices utilizados en el curso
- Información general sobre 12c de base de datos Oracle y productos afines
- Descripción de los conceptos y la terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema Recursos Humanos (HR) y las tablas utilizadas en el curso
- La documentación de Oracle de base de datos SQL 12c y Recursos Adicionales

1 - 6

## Objetivos del Curso

Después de completar este curso, usted debe ser capaz de:

- Identificar los principales componentes de base de datos Oracle
- Recuperar filas y columnas de las tablas de datos con la instrucción `SELECT`
- Crear informes de datos ordenados y restringidos  
Emplean funciones SQL para generar y recuperar datos personalizados
- Ejecutar consultas complejas para recuperar datos de varias tablas
- Ejecutar lenguaje de manipulación de datos (DML) para actualizar datos en base de datos Oracle
- Ejecutar el lenguaje de definición de datos (DDL) para crear y administrar objetos de esquema

1 - 7

## Agenda

- Día 1:
  - Introducción
  - Recuperación de datos usando la sentencia `SELECT`
  - Filtrado y Ordenado de datos
  - Funciones de una sola fila
- Día 2:
  - Uso de las funciones de conversión y las expresiones condicionales
  - Informes agregados de datos utilizando las funciones de grupo
  - Viendo los datos de las tablas múltiples usando `JOINS`
  - El uso de subconsultas para resolver consultas

1 - 8

## Course Agenda

- Día 3:
  - Uso de operadores Set
  - Gestión de tablas Utilización de sentencias DML
  - Introducción a Data Definition Language

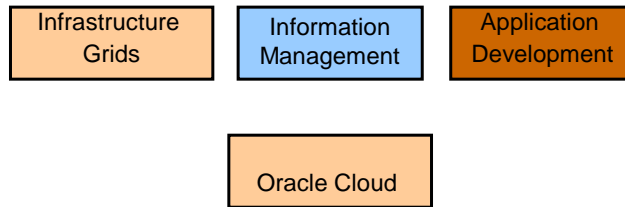
1 - 9

## Lesson Agenda

- Objetivos del curso, orden del día, y apéndices utilizados en el curso
- Información general sobre 12c de base de datos Oracle y productos afines
- Descripción de los conceptos y la terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema Recursos Humanos (HR) y las tablas utilizadas en el curso
- La documentación de Oracle de base de datos SQL 12c y Recursos Adicionales

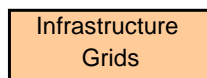
1 - 11

## Oracle Database 12c: Areas



1 - 12

## Oracle Database 12c: Areas



- Oracle GRID permite la **agrupación** de servidores de bajo costo y de almacenamiento para formar sistemas que proporcionan la más alta calidad de servicio en términos de manejabilidad, alta disponibilidad y rendimiento.
- Oracle Database 12c consolida y amplía los beneficios de la computación grid.
  - Permite una gestión mucho mas eficiente de Bases de Datos con características únicas

1 - 13

## Oracle Database 12c: Areas

### Information Management



- Oracle 12c base de datos amplía las capacidades de gestión de información existentes en versiones anteriores de Oracle:
  - Permite:
    - la centralización de información de productos de sistemas heterogéneos,
    - la creación de una única visión de la información del producto que se puede aprovechar en todos los departamentos funcionales.
- Oracle proporciona la gestión de contenidos de tipos avanzados de datos tales como lenguaje de marcado extensible (XML), el texto, espacial, multimedia, imágenes médicas y tecnologías semánticas.

1 - 14

## Oracle Database 12c: Areas

### Application Development



- Oracle 12c tiene capacidad para utilizar y gestionar todos los principales entornos de desarrollo de aplicaciones, tales como
  - PL / SQL
  - Java / JDBC
  - .NET y Windows,
  - PHP,
  - SQL Developer
  - Application Express (APEX)

1 - 15

## Oracle Database 12c: Areas

Oracle Cloud

**ORACLE**  
DATABASE **12<sup>c</sup>**

- La nube de Oracle es **una nube empresarial para los negocios**.
- Proporciona una colección integrada de servicios de aplicaciones y permite la creación de una plataforma basada en los mejores productos de su clase.
- Se basa en productos Open Source, Java y estándares SQL.

1 - 16

## Oracle Database 12c

**ORACLE**  
DATABASE

Manageability  
High Availability  
Performance  
Security  
Information Integration

Oracle Database 12c está diseñado con las siguientes características para ayudar a las organizaciones a gestionar sus datos:

- Manageability (Manejabilidad)
  - los DBA pueden aumentar su productividad, reducir costes, minimizar los errores y maximizar la calidad del servicio de forma muy fácil.
  - **Enterprise Manager Database Express 12c** es una herramienta basada en Web que facilita las gestiones habituales
- High Availability
  - Mediante la Alta disponibilidad, se puede reducir el riesgo de tiempo de inactividad y pérdida de datos.
  - Estas características mejoran las operaciones en línea y permiten actualizaciones de bases de datos más rápidas.

1 - 17



## Oracle Database 12c



Oracle Database 12c está diseñado con las siguientes características para ayudar a las organizaciones a gestionar sus datos:

Manageability  
High Availability  
Performance  
Security  
Information Integration

- Performance (Rendimiento)
  - Disponemos de funcionalidades como, SecureFiles, compression for online transaction processing (OLTP), Real Application Clusters (RAC) optimizations, Result Caches.
- Security (Seguridad)
  - Oracle Database 12c dispone de herramientas para proteger su información con configuraciones únicas seguras, encriptación de datos y el enmascaramiento, y auditoría
- Information Integration (integridad)
  - Mejoras en la integración de datos en toda la empresa y capacidades de gestión de ciclo de vida

1 - 18

## Oracle Fusion Middleware

**Suite de productos de software**, basado en estándares, probados por el cliente que se extiende por una amplia gama de herramientas y servicios de Java EE y herramientas para desarrolladores, a través de servicios de integración, inteligencia de negocio, colaboración y gestión de contenidos



1 - 19

## Oracle Enterprise Manager Cloud Control

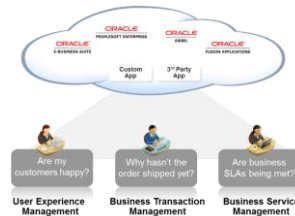
- Control Manager Enterprise Cloud es una herramienta de gestión que proporciona capacidades de monitorización y gestión para componentes Oracle y no Oracle.
- Permite gestionar todo el ciclo de vida de los servicios en la nube.
- Permite la integración de completa con servicios básicos (databases, Application Server, Listeners, etc)



Ciclo de Vida Completo



Stack Completo



Integración Completa

Self-Service IT | Simple and Automated | Business-Driven

1 - 20

## Oracle Cloud

La nube de Oracle es una nube empresarial para los negocios.

Se compone de muchos servicios diferentes que comparten algunas características comunes:

- Servicio bajo demanda (On-demand self-service)
- Pooling de Recursos
- Mediciones de Servicios
- Acceso a través de Red

[www.cloud.oracle.com](http://www.cloud.oracle.com)



### RESULTADO:

- las aplicaciones y bases de datos desplegados en la nube de Oracle son portátiles y se pueden mover fácilmente hacia o desde una nube privada o en las instalaciones de medio ambiente.

1 - 21

## Oracle Cloud Services

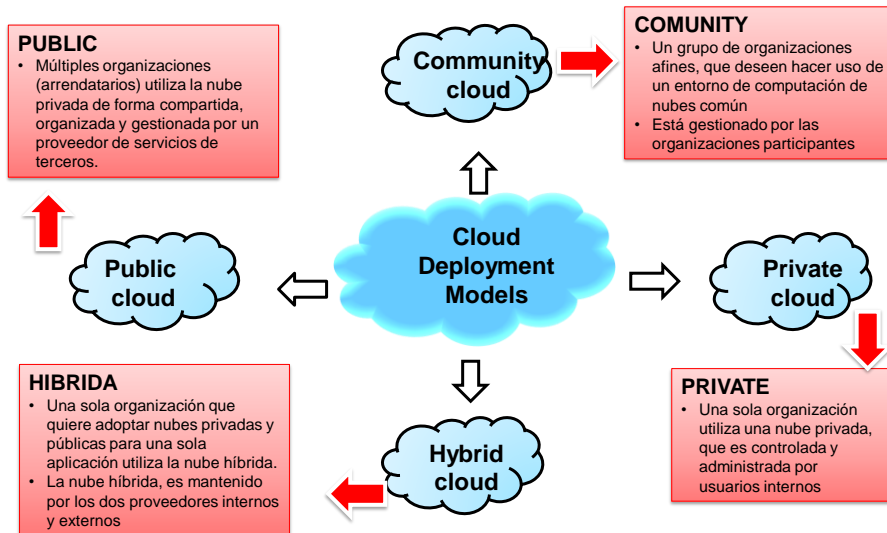
Oracle Cloud provee tres tipos de servicios:

- **Software as a Service (SaaS)**
  - Aplicaciones desarrolladas y terminadas que se entregan a los usuarios a través de Internet.
  - Oracle CRM On Demand es un ejemplo de una oferta SaaS.
- **Platform as a Service (PaaS)**
  - Plataforma de desarrollo y despliegue de aplicaciones entregado como un servicio a los desarrolladores, lo que les permite crear y desplegar una aplicación SaaS a los usuarios finales de forma rápida
- **Infrastructure as a Service (IaaS)**
  - Se refiere al hardware de computación (servidores, almacenamiento y red) se ofrece como servicio



1 - 22

## Cloud Deployment Models



1 - 23

## Agenda

- Objetivos del curso, orden del día, y apéndices utilizados en el curso
- Información general sobre 12c de base de datos Oracle y productos afines
- Descripción de los conceptos y la terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema Recursos Humanos (HR) y las tablas utilizadas en el curso
- La documentación de Oracle de base de datos SQL 12c y Recursos Adicionales

1 - 24

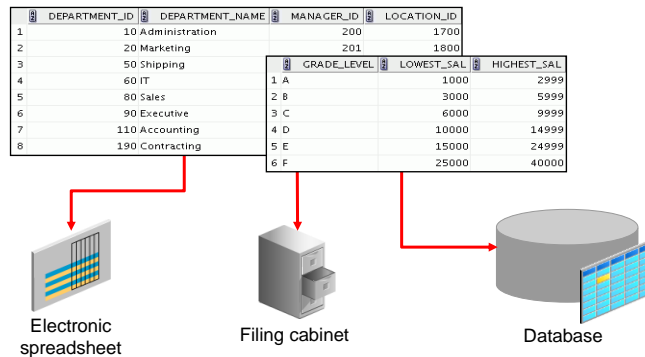
## Sistema de Gestión de Base de Datos Relacional

- El servidor de Oracle es compatible tanto con modelos de BBDD Relacionales como BBDD Relacionales de Objetos.
  - Oracle amplía las capacidades de modelado de datos para apoyar **un modelo de base de datos relacional objeto** que proporciona:
    - Programación orientada a objetos
    - Tipos de datos complejos
    - Objetos de negocio complejas, etc
- El modelo es compatible con Oracle cliente / servidor y aplicaciones basadas en web que son distribuidos y de varios niveles.



1 - 25

## Almacenamiento de datos en diferentes medios



- Las organizaciones pueden almacenar datos en diversos medios y en diferentes formatos, tales como:
  - Documentos, archivadores, Hojas Electrónicas, BBDD's
- Una BBDD es una colección organizada de información,
- Para gestionar las bases de datos, se necesita un sistema de gestión de base de datos (DBMS)

1 - 26

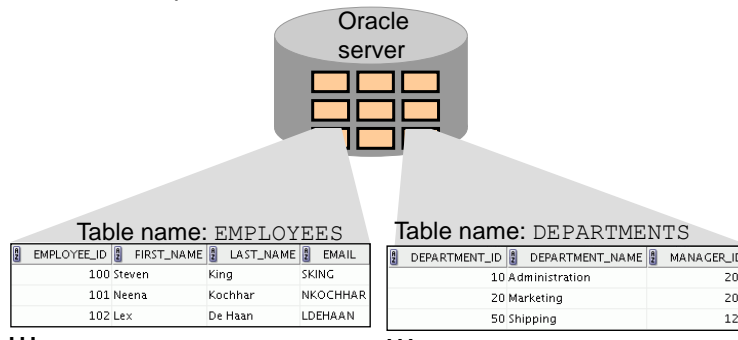
## Conceptos de BBDD Relacional

- El modelo relacional se describió por primera vez en un documento en 1970 por el Dr. Codd.
  - Los modelos comunes utilizados para BBDD hasta ese momento eran jerárquicos y de red
- Este modelo es la base para el sistema de gestión de base de datos relacional (RDBMS).
- El modelo relacional consiste en lo siguiente:
  - Colección de objetos o relaciones de datos almacenados
  - Conjunto de operadores para actuar sobre las relaciones
  - integridad de los datos para la exactitud y consistencia

1 - 27

## Definición de Base de Datos Relacional

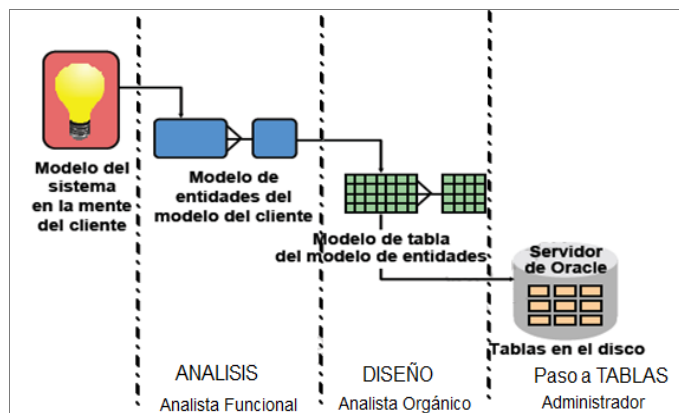
- Una base de datos relacional es un conjunto de relaciones o tablas de dos dimensiones controladas por el servidor Oracle.
- Una base de datos relacional utiliza relaciones o tablas de dos dimensiones para almacenar información.



1 - 28

## Tipos de Bases de Datos

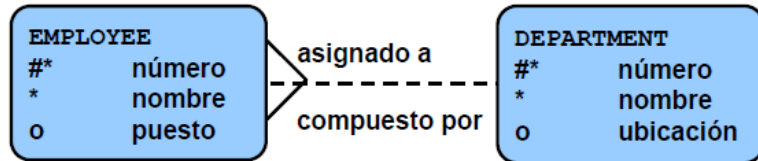
Para poder realizar de forma correcta una Aplicación, se deben de realizar una serie de pasos ( MODELIZACION )



1 - 29

## Modelo de Relación de Entidades

- Crear un diagrama de entidad/relación a partir de narrativas o especificaciones de negocio:



- Supuesto:
  - "... Asignar uno o más empleados a un departamento ..."
  - "... Algunos departamentos aún no tienen empleados asignados. ..."

1 - 30

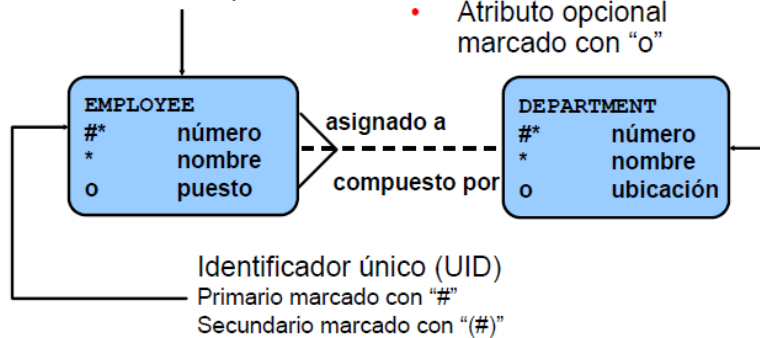
## Convenciones de Modelado de Relación de Entidades

### Entidad:

- Nombre único en singular
- Mayúsculas
- Recuadro editable
- Sinónimo entre paréntesis

### Atributo:

- Nombre en singular
- Minúsculas
- Atributo obligatorio marcado con **\***
- Atributo opcional marcado con **o**



1 - 31

## Relación de Varias Tablas

- Cada fila de datos de una tabla se identifica como única mediante una clave primaria.
- Puede relacionar de forma lógica desde varias tablas mediante claves ajenas.

Nombre de la tabla: **EMPLOYEES**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
103	Alexander	Hunold	60
104	Bruce	Ernst	60
107	Diana	Lorentz	60
124	Kevin	Mourgos	50
141	Trenna	Rajs	50
142	Curtis	Davies	50

Clave primaria

Clave ajena

Nombre de la tabla: **DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
120	Contracting	(null)	1700

Clave primaria

1 - 32

## Terminología de Bases de Datos Relacionales

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	90
101	Neena	Kochhar	17000	(null)	90
102	Lex	De Haan	17000	(null)	90
103	Alexander	Hunold	9000	(null)	60
104	Bruce	Ernst	6000	(null)	60
107	Diana	Lorentz	4200	(null)	60
124	Kevin	Mourgos	5800	(null)	50
141	Trenna	Rajs	3500	(null)	50
142	Curtis	Davies	3100	(null)	50
143	Randall	Mates	2600	(null)	50
144	Peter	Vargas	2500	(null)	50
149	Eleni	Zlotkey	10500	0.3	80
174	Ellen	Abel	11000	0.3	80
176	Jonathan	Taylor	8500	0.3	80
178	Timothy	Grant	7000	0.15	(null)
200	Jennifer	Whalen	4400	(null)	10
201	Michael	Hartstein	13000	(null)	20
202	Pat	Fay	6000	(null)	20
205	Shelley	Higgins	12000	(null)	110
206	William	Gietz	8300	(null)	110

1.- Fila      2.- Columna Clave      3.- Columna NO Clave      4.- Columna FK      5.- Campo      6.- Campo NULL

1 - 33



## Agenda

- Objetivos del curso, orden del día, y apéndices utilizados en el curso
- Información general sobre 12c de base de datos Oracle y productos afines
- Descripción de los conceptos y la terminología de gestión de bases de datos relacionales
- **Introducción a SQL y sus entornos de desarrollo**
- Esquema Recursos Humanos (HR) y las tablas utilizadas en el curso
- La documentación de Oracle de base de datos SQL 12c y Recursos Adicionales

1 - 34

## Uso de SQL para Consultar Base de Datos

- SQL es un conjunto de instrucciones que se utilizan para acceder a datos en la base de datos Oracle.
- Uso y aprendizaje sencillos y eficaces
- SQL proporciona comandos para una variedad de tareas, incluyendo:
  - Consulta de datos
  - Insertar, actualizar y eliminar filas en una tabla
  - Creación, sustitución, modificación y eliminación de objetos
  - Control del acceso a la base de datos ya sus objetos
  - Garantizar la coherencia e integridad de la base de datos



1 - 35

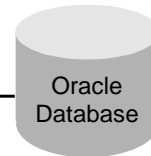
## Uso de SQL para Consultar Base de Datos

Se introduce una sentencia SQL.

```
SELECT department_name  
FROM departments;
```

DEPARTMENT_NAME
1 Administration
2 Marketing
3 Purchasing
4 Human Resources
5 Shipping
6 IT
7 Public Relations
8 Sales

La sentencia se envía al servidor de la base de datos Oracle.



Los datos se recuperan y devuelven al usuario.

1 - 36

## Sentencias SQL

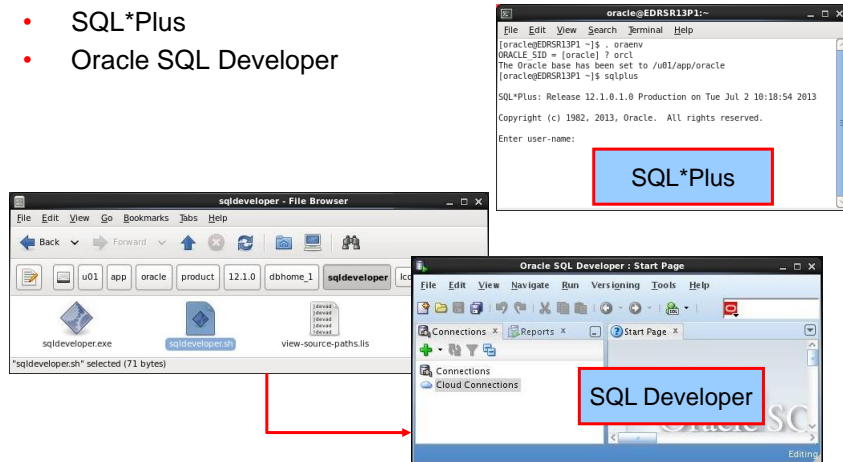
SELECT	Recuperación de datos
INSERT UPDATE DELETE MERGE	Data manipulation language (DML)
CREATE ALTER DROP RENAME TRUNCATE	Data definition language (DDL)
COMMIT ROLLBACK SAVEPOINT	Control de transacciones
GRANT REVOKE	Data control language (DCL)

1 - 37

## Entornos de Desarrollo para SQL

Existen dos entornos de desarrollo para este curso:

- SQL \*Plus
- Oracle SQL Developer



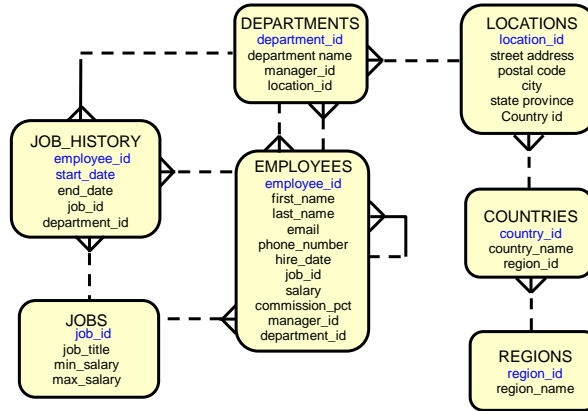
1 - 38

## Agenda

- Objetivos del curso, orden del día, y apéndices utilizados en el curso
- Información general sobre 12c de base de datos Oracle y productos afines
- Descripción de los conceptos y la terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- **Esquema Recursos Humanos (HR) y las tablas utilizadas en el curso**
- La documentación de Oracle de base de datos SQL 12c y Recursos Adicionales

1 - 39

## Human Resources (HR) Schema



1 - 40

## Tablas usadas en el Curso

EMPLOYEES

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	AC_MGR	12008
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000
6	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200
7	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-07	ST_MAN	5800
8	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-03	ST_CLERK	3500
9	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-05	ST_CLERK	3100
10	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-06	ST_CLERK	2600
11	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-06	ST_CLERK	2500
12	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-08	SA_MAN	10500
13	174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-04	SA_REP	11000
14	176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-06	SA_REP	8600
15	178	Klaberely	Grant	KGRANT	011.44.1644.429263	24-MAY-07	SA_REP	7000
16	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-03	AD_ASST	4400
17	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000
18	202	Pat	Fay	PFAY	603.123.6666	17-AUG-05	MK_REP	6000
19	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-02	AC_MGR	12008
20	206	William	Gietz	WGIEZT	515.123.8181	07-JUN-02	AC_ACCOUNT	8300

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

JOB\_GRADES

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

DEPARTMENTS

1 - 41

## Agenda

- Objetivos del curso, orden del día, y apéndices utilizados en el curso
- Información general sobre 12c de base de datos Oracle y productos afines
- Descripción de los conceptos y la terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema Recursos Humanos (HR) y las tablas utilizadas en el curso
- La documentación de Oracle de base de datos SQL 12c y Recursos Adicionales

1 - 42

## Oracle Database Documentation

- *Oracle Database New Features Guide*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Concepts*
- *Oracle Database SQL Developer User's Guide*

<http://st-doc.us.oracle.com/12/121/index.htm>

1 - 43

## Resumen

En esta lección, usted debe haber aprendido:

- Definir los objetivos del curso
- Enumerar las funciones de base de datos Oracle 12c
- Describir las características más destacadas Oracle Cloud
- Discutir los aspectos teóricos y físicos de una base de datos relacional
- Identificar los entornos de desarrollo que se pueden utilizar para este curso
- Describir la base de datos y el esquema utilizado en este curso

1 - 44

## Práctica 1:

Esta práctica se abordan los siguientes temas:

- Comenzando con Oracle SQL Developer
- Creando una nueva conexión a la BBDD
- Navegación por las tablas de HR

1 - 45

# 2

## Recuperación de datos usando la instrucción SQL SELECT



ORACLE®

### Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Enumerar las capacidades de las sentencias SQL SELECT
- Ejecutar una instrucción SELECT básica

## Agenda

- Capacidades de la sentencia SQL `SELECT`
- Expresiones aritméticas y valores `NULL` en la instrucción `SELECT`
- Alias de columna
- El uso del operador de concatenación, cadenas de caracteres literales, operador comilla alternativa, y la palabra clave `DISTINCT`
- Operador `DESCRIBE`

1 - 48

## Sentencia Básica `SELECT`

```
SELECT *|{[DISTINCT] column [alias],...}  
FROM table;
```

- `SELECT` identifica las columnas que se mostrarán.
- `FROM` identifica la tabla que contiene las columnas

```
SELECT  
*  
DISTINCT  
column|expression  
alias  
FROM table
```

Lista de **una o mas** columnas  
Selección **todas** las columnas  
Suprime duplicados  
Selecciona columna o expresión  
Da un alias a la columna indicada  
Especifica la tabla que contiene las columnas

1 - 49



## Seleccionando todas las columnas

- Se pueden visualizar todas las columnas de datos en una tabla añadiendo un Asterisco (\*) después de la palabra SELECT

```
SELECT *  
FROM departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

SQL Developer  
Pulsar →F9

- También podemos mostrar todas las columnas colocando el nombre de las mismas después de SELECT

1 - 50

## Seleccionando columnas específicas

- Puede utilizar la instrucción SELECT para mostrar columnas específicas de la tabla especificando los nombres de columna, separados por comas

```
SELECT department_id, location_id  
FROM departments;
```

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

- En la cláusula SELECT, especificar las columnas que desee en el orden en el que desea que aparezcan en la salida

1 - 51

## Escribiendo sentencias SQL

- Las sentencias SQL no diferencian entre mayúsculas y minúsculas.
- Las sentencias SQL se pueden introducir en una o más líneas.
- Las palabras clave no se pueden abreviar ni en diferentes líneas.
- Cláusulas se colocan generalmente en líneas separadas.
- El sangrado se utilizan para mejorar la legibilidad.
- En SQL \* Plus, se le requiere para terminar cada sentencia SQL con un punto y coma (;).

1 - 52

## Ejecutando sentencias SQL

- En `SQL Developer`, haga clic en el icono **Ejecutar script** o pulse **[F5]** para ejecutar el comando o comandos en la Hoja de trabajo de SQL.
- También puede hacer clic en el icono **Execute Statement** o pulse **[F9]**
- El resultado de la ejecución puede ser presentado en una página con pestañas **[F9]** o pantalla de emulación de SQL\*Plus **[F5]**
- En `SQL * Plus`, terminar la instrucción SQL con un punto y coma, y luego presione **[Enter]** para ejecutar el comando.

1 - 53

## Agenda

- Capacidades de la sentencia SQL SELECT
- Expresiones aritméticas y valores NULL en la instrucción SELECT
- Alias de columna
- El uso del operador de concatenación, cadenas de caracteres literales, operador comilla alternativa, y la palabra clave DISTINCT
- Operador DESCRIBE

1 - 55

## Expresiones Aritméticas

- Mediante el uso de expresiones aritméticas:
  - Modificar la forma en que se muestran los datos, o
  - Realizar cálculos de los datos.
- Una expresión aritmética puede contener:
  - Nombres de columna
  - Valores numéricos constantes
  - Operadores aritméticos.

**Nota:**

Con los tipos de datos de fecha y hora, puede utilizar sólo los operadores de suma y resta.

Operator	Description
+	Suma
-	Resta
*	Multipliación
/	División

1 - 56

## Usando Operadores Aritméticos

- En el ejemplo siguiente se realizar una suma de 300 dólares a la columna salario.
- La columna resultante no es una columna Física, sólo visual

```
SELECT last_name, salary, salary + 300
FROM employees;
```

	LAST_NAME	SALARY	SALARY+300
1	King	24000	24300
2	Kochhar	17000	17300
3	De Haan	17000	17300
4	Hunold	9000	9300
5	Ernst	6000	6300
6	Lorentz	4200	4500
7	Mourgos	5800	6100
8	Rajs	3500	3800
9	Davies	3100	3400
10	Matos	2600	2900

...

### Reglas de Precedencia

- Multiplicación y división se producen antes de la suma y resta.
- Los operadores de la misma prioridad se evalúan de izquierda a derecha.
- Los paréntesis se utilizan para anular la precedencia predeterminada

1 - 57

## Prioridad de Operadores. Ejemplos

```
SELECT last_name, salary, 12*salary+100
FROM employees;
```

1

	LAST_NAME	SALARY	12*SALARY+100
1	King	24000	288100
2	Kochhar	17000	204100
3	De Haan	17000	204100
4	Hunold	9000	108100

...

```
SELECT last_name, salary, 12*(salary+100)
FROM employees;
```

2

	LAST_NAME	SALARY	12*(SALARY+100)
1	King	24000	289200
2	Kochhar	17000	205200
3	De Haan	17000	205200
4	Hunold	9000	109200

...

1 - 58

## Definición de un valor nulo

- NULL es un valor que no está disponible, sin asignar, es desconocido o inaplicable.
- NULL no es lo mismo que cero o un espacio en blanco.

```
SELECT last_name, job_id, salary, commission_pct  
FROM employees;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	King	AD_PRES	24000	(null)
2	Kochhar	AD_VP	17000	(null)
3	De Haan	AD_VP	17000	(null)
...				
12	Zlotkey	SA_MAN	10500	0.2
13	Abel	SA_REP	11000	0.3
14	Taylor	SA_REP	8600	0.2
15	Grant	SA_REP	7000	0.15
...				
18	Fay	MK_REP	6000	(null)
19	Higgins	AC_MGR	12008	(null)
20	Gietz	AC_ACCOUNT	8300	(null)

1 - 59

## Definición de un valor nulo

- Las columnas con valor NULL se pueden seleccionar en una consulta SELECT
- Estas columnas pueden ser parte de una expresión aritmética.
- Cualquier expresión aritmética usando los valores NULL se traduce en NULL.
- Columnas de cualquier tipo de datos pueden contener valores nulos. Sin embargo, algunas limitaciones (NOT NULL y PRIMARY KEY) impiden que los nulos se utilicen en la columna.

1 - 60

## Valores Nulos en Expresiones Aritméticas

- Las expresiones aritméticas que contengan un valor nulo se evalúan como nulo.

```
SELECT last_name, 12*salary*commission_pct  
FROM employees;
```

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)

...

12	Zlotkey	25200
13	Abel	39600
14	Taylor	20640
15	Grant	12600

...

17	Hartstein	(null)
18	Fay	(null)
19	Higgins	(null)
20	Gietz	(null)

1 - 61

## Agenda

- Capacidades de la sentencia SQL SELECT
- Expresiones aritméticas y valores NULL en la instrucción SELECT
- Alias de columna
- El uso del operador de concatenación, cadenas de caracteres literales, operador comilla alternativa, y la palabra clave DISTINCT
- Operador DESCRIBE

1 - 62

## Definiendo un Alias de Columna

- Cuando se muestra el resultado de una consulta, SQL Developer normalmente utiliza el nombre de la columna seleccionada como el encabezado de la columna.
- Podemos cambiar un encabezado de columna utilizando un alias de columna.

Alias de Columna:

- Cambia el nombre de un encabezado de columna
- Es útil con cálculos
- Sigue inmediatamente al nombre de la columna (también puede ser la palabra clave opcional `AS` entre el nombre de la columna y el alias)
- Requiere comillas dobles si contiene espacios o caracteres especiales, o si se trata de mayúsculas y minúsculas

1 - 63

## Usando un Alias de Columna

```
SELECT last_name AS name, commission_pct comm  
FROM employees;
```

	NAME	COMM
1	King	(null)
2	Kochhar	(null)
3	De Haan	(null)
4	Hunold	(null)

...

```
SELECT last_name "Name", salary*12 "Annual Salary"  
FROM employees;
```

	Name	Annual Salary
1	King	288000
2	Kochhar	204000
3	De Haan	204000
4	Hunold	108000

...

1 - 64

## Agenda

- Capacidades de la sentencia SQL `SELECT`
- Expresiones aritméticas y valores `NULL` en la instrucción `SELECT`
- Alias de columna
- El uso del operador de concatenación, cadenas de caracteres literales, operador comilla alternativa, y la palabra clave `DISTINCT`
- Operador `DESCRIBE`

1 - 65

## Operador Concatenación

- Puede enlazar columnas a otras columnas, expresiones aritméticas, o valores constantes para crear una expresión de caracteres usando el operador de concatenación `(||)`.

```
SELECT last_name||job_id AS "Employees"
FROM employees;
```

Employees
1 AbetSA_REP
2 DaviesST_CLERK
3 De HaanAD_VP
4 ErnstIT_PROG
5 FayMK_REP
6 GietzAC_ACCOUNT
7 GrantSA_REP
8 HartsteinMK_MAN

...

### El operador de concatenación:

- Enlaza columnas o cadenas de caracteres a otras columnas
- Está representado por dos barras verticales `(||)`
- Crea una columna resultante que es una expresión de caracteres

1 - 66



## Literales

- Un literal es un carácter, un número o una fecha que se incluye en la instrucción `SELECT`.
  - No es un nombre de columna o un alias de columna
- Los valores literales de Fecha y carácter deben estar encerrados entre comillas simples.
  - Los literales NUMERICOS no deben ir entre comillas simples
- Cada literal dentro de una `SELECT` se muestra por cada fila devuelta.

1 - 67

## Usando Literales

```
SELECT last_name || ' is a ' || job_id  
       AS "Employee Details"  
FROM   employees;
```

Employee Details	
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG
11	King is a AD_PRES

...

1 - 68

## Operador Alternativo Quote (q)

- Muchas sentencias SQL utilizan el carácter comilla simple, dentro del Literal.
- Para poder representarlo, es necesario utilizar el operador **Quote(q)** junto con un delimitador como [], {}, etc.. `q' [ . . . ] '`

```
SELECT department_name || q'[ Department's Manager Id: ]'
      || manager_id
      AS "Department and Manager"
FROM departments;
```

	Department and Manager
1	Administration Department's Manager Id: 200
2	Marketing Department's Manager Id: 201
3	Shipping Department's Manager Id: 124
4	IT Department's Manager Id: 103
5	Sales Department's Manager Id: 149
6	Executive Department's Manager Id: 100
7	Accounting Department's Manager Id: 205
8	Contracting Department's Manager Id:

### Quote(q)

- Especificar su propio delimitador de comilla.
- Seleccione cualquier delimitador.
- Aumentar la legibilidad y facilidad de uso.

1 - 69

## Filas duplicadas

- Por defecto para todas las consultas, se incluyen las filas duplicadas
- Para eliminar las filas duplicadas en el resultado, incluya la palabra clave **DISTINCT** en la cláusula **SELECT** inmediatamente después de la palabra clave **SELECT**.

1

```
SELECT department_id
FROM employees;
```

	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60
6	60
7	50
8	50

...

2

```
SELECT DISTINCT department_id
FROM employees;
```

	DEPARTMENT_ID
1	(null)
2	90
3	20
4	110
5	50
6	80
7	60
8	10

Puede especificar varias columnas después de que el calificador **DISTINCT**.  
El calificador **DISTINCT** afecta a todas las columnas seleccionadas, y el resultado es cada combinación distinta de las columnas.

1 - 70

## Agenda

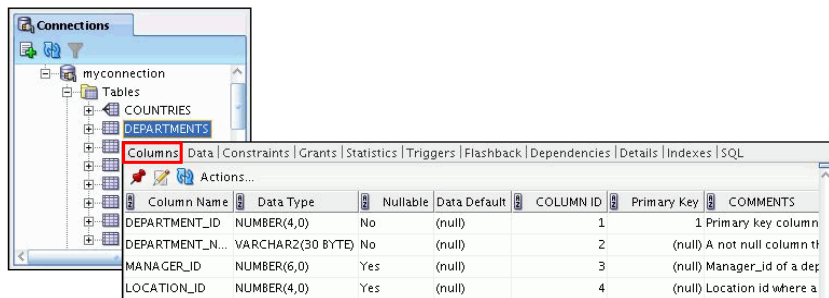
- Capacidades de la sentencia SQL `SELECT`
- Expresiones aritméticas y valores `NULL` en la instrucción `SELECT`
- Alias de columna
- El uso del operador de concatenación, cadenas de caracteres literales, operador comilla alternativa, y la palabra clave `DISTINCT`
- Comando `DESCRIBE`

1 - 71

## Visualizando la Estructura de una Tabla

- Podemos utilizar el comando `DESCRIBE tabla` para visualizar la estructura de una tabla.
- También podemos seleccionar la tabla en el árbol de conexiones y mirar en la pestaña correspondiente

```
DESC[RIBE] tablename
```



Column Name	Data Type	Nullable	Data Default	COLUMN ID	Primary Key	COMMENTS
DEPARTMENT_ID	NUMBER(4,0)	No	(null)	1		1 Primary key column
DEPARTMENT_N...	VARCHAR2(30 BYTE)	No	(null)	2		(null) A not null column th
MANAGER_ID	NUMBER(6,0)	Yes	(null)	3		(null) Manager_id of a dep
LOCATION_ID	NUMBER(4,0)	Yes	(null)	4		(null) Location id where a

1 - 72

## Usando el comando DESCRIBE

```
DESCRIBE employees
```

```
DESCRIBE Employees
Name          Null    Type
-----
EMPLOYEE_ID   NOT NULL NUMBER(6)
FIRST_NAME    NOT NULL VARCHAR2(20)
LAST_NAME     NOT NULL VARCHAR2(25)
EMAIL         NOT NULL VARCHAR2(25)
PHONE_NUMBER  NOT NULL VARCHAR2(20)
HIRE_DATE     NOT NULL DATE
JOB_ID        NOT NULL VARCHAR2(10)
SALARY        NOT NULL NUMBER(8,2)
COMMISSION_PCT          NUMBER(2,2)
MANAGER_ID     NUMBER(6)
DEPARTMENT_ID  NUMBER(4)
```

1 - 73

## Resumen

En esta lección, usted debe haber aprendido a escribir una instrucción `SELECT` que:

- Devuelve todas las filas y columnas de una tabla
- Devuelve columnas especificadas de una tabla
- Utiliza los alias de columna para mostrar los encabezados de columna más descriptivos

1 - 75

## Práctica 2:

Esta práctica se abordan los siguientes temas:

- Selección de todos los datos de diferentes tablas
- Describir la estructura de las tablas
- Realizar cálculos aritméticos y especificar nombres de columna

1 - 76

# 3

## Restringir y ordenar datos



ORACLE®

## Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Limitar las filas recuperadas por una consulta
- Ordenar las filas que se recuperan mediante una consulta
- Utilizar la sustitución de ampersand para restringir y ordenar la salida en tiempo de ejecución

1 - 78

## Agenda

- Limitando las filas con:
  - Clausula `WHERE`
  - Operadores de comparación `=`, `<=`, `BETWEEN`, `IN`, `LIKE`, y condiciones `NULL`
  - Condiciones lógicas usando los operadores `AND`, `OR`, y `NOT`
- Reglas de precedencia para los operadores en una expresión
- Ordenación de filas utilizando la cláusula `ORDER BY`
- Limitando filas en consultas `SQL`
- Las variables de sustitución
- Comandos `DEFINE` y `VERIFY`

1 - 79


## La limitación de filas mediante una Selección

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100 King	AD_PRES	90
2	101 Kochhar	AD_VP	90
3	102 De Haan	AD_VP	90
4	103 HunaId	IT_PROG	60
5	104 Ernst	IT_PROG	60
6	107 Lorentz	IT_PROG	60

...

"queremos devolver todos  
los empleado del  
departamento 90"



EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100 King	AD_PRES	90
2	101 Kochhar	AD_VP	90
3	102 De Haan	AD_VP	90

1 - 80

## La limitación de filas mediante una Selección

- La restricción de filas devueltas puede ser conseguida mediante la cláusula WHERE:

```
SELECT *|{[DISTINCT] column [alias],...}  
FROM table  
[WHERE logical expression(s)];
```

- Una cláusula WHERE contiene una condición que debe cumplirse.
- WHERE debe de seguir obligatoriamente a la cláusula FROM.
  - Si la condición es verdadera, se devuelve las filas que cumplen esa condición

1 - 81

## Usando la cláusula WHERE

- En el ejemplo, la instrucción SELECT recupera employee\_id, last\_name, job\_id, department\_id de todos los empleados que se encuentran en el departamento 90.

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

No se puede utilizar **alias de columna** en la cláusula WHERE.

1 - 82

## Cadenas de Caracteres y Fechas

- Las cadenas de caracteres y fechas deben de ser encerrados entre comillas simples. (números NO)
- Los valores de caracteres son *case-sensitive*
- El formato de fecha por defecto es DD-MON-RR.
  - Oracle almacenan las fechas en un formato numérico interno, que representa los siglo, año, mes, día, horas, minutos y segundos

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID
1 Whalen	AD_ASST	10

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-OCT-03' ;
```

LAST_NAME
1 Rajs

1 - 83



## Operadores de Comparación

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

1 - 84

## Usando Operadores de Comparación

- SELECT recupera el apellido y el salario de la tabla de empleados de cualquier empleado cuyo salario es inferior o igual a \$ 3.000

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

1 - 85

## Condiciones rango usando el operador **BETWEEN**

- Puede mostrar filas en función de un rango de valores utilizando el operador **BETWEEN**
- El rango tiene que especificar un límite inferior y un límite superior.

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

Lower limit

Upper limit

Los valores que se especifican con el operador **BETWEEN** son **inclusivos**

1 - 86

## Condiciones rango usando el operador **[NOT] BETWEEN**

- El operador **BETWEEN** también puede ser utilizado para indicar un rango con cadenas de caracteres.
- Se utiliza orden de diccionario.

```
SELECT last_name
FROM   employees
WHERE  last_name BETWEEN 'King' AND 'Whalen'
```

Lower limit

Upper limit

1 - 87

## Usando el operador [NOT] IN

- El operador **IN** permite testear un valor dentro de un conjunto de valores.
- El conjunto de valores se puede especificar en cualquier orden

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12008	101
8	202	Fay	6000	201

El operador **IN** se puede utilizar con cualquier tipo de datos

```
:
WHERE last_name IN
('Hartstein', 'Vargas');
```

1 - 88

## Búsqueda de patrones usando el operador LIKE

- Use el operador **LIKE** para realizar una búsqueda de un patrón dentro de una cadena o columna.
- La condición de búsqueda puede contener literales, números y los caracteres comodines:
  - **%** indica cero o mas caracteres
  - **\_** indica un solo carácter.

```
SELECT first_name
FROM   employees
WHERE  first_name LIKE 'S%';
```

	FIRST_NAME
1	Shelley
2	Steven

1 - 89

## Caracteres comodín

- Se pueden combinar los dos caracteres comodín ( %, \_ ) con caracteres literales para la coincidencia de patrones:

```
SELECT last_name
FROM employees
WHERE last_name LIKE '_o%';
```

LAST_NAME
1 Kochhar
2 Lorentz
3 Mourgos

- Si necesitamos buscar uno de esos caracteres, deberemos utilizar el identificador `ESCAPE` `character` para definir un carácter especial

```
SELECT last_name
FROM employees
WHERE last_name LIKE '$_o%' ESCAPE '$';
```

1 - 90

## Usando NULL en condiciones

- Para trabajar de forma correcta con los nulos `NULL`, deberemos utilizar las cláusulas `IS NULL` y `IS NOT NULL` en condiciones.
  - Un valor nulo significa que el valor no está disponible, sin asignar, es desconocido o inaplicable
- Los nulos no pueden utilizar operadores como `=` `<>` `>=`, etc

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL;
```

LAST_NAME	MANAGER_ID
1 King	(null)

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id = NULL;
```

1 - 91

## Definiendo condiciones con operadores lógicos

- Los operadores lógicos combinan los resultados de dos o mas condiciones para producir un único resultado lógico.
- Se puede utilizar varias condiciones en una sola cláusula WHERE con el AND y OR operadores.

Operator	Meaning
AND	Devuelve TRUE si ambos componentes son TRUE
OR	Devuelve TRUE si alguno de los compontes es TRUE
NOT	Devuelve TRUE si la condición es FALSE

1 - 92

## Usando el operador AND

- En el ejemplo, las 2 condiciones deben de ser cierto para que el registro que se seleccione.
  - Sólo aquellos empleados que tienen un puesto de trabajo que contiene la cadena 'MAN' y ganar \$ 10,000 o más se seleccionan.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	149	Zlotkey	SA_MAN	10500
2	201	Hartstein	MK_MAN	13000

1 - 93

## Usando el operador OR

- En el ejemplo, si una de las condiciones es TRUE el registro se selecciona.
  - Sólo aquellos empleados que tienen un puesto de trabajo que contiene la cadena 'MAN' o que ganan \$ 10,000 o más se seleccionan.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	100	King	AD_PRES	24000
2	101	Kochhar	AD_VP	17000
3	102	De Haan	AD_VP	17000
4	124	Mourgos	ST_MAN	5800
5	149	Zlotkey	SA_MAN	10500
6	174	Abel	SA_REP	11000
7	201	Hartstein	MK_MAN	13000
8	205	Higgins	AC_MGR	12008

1 - 94

## Usando el operador OR

- En el ejemplo
  - Muestra last\_name y job\_id de todos los empleados cuyo identificador de trabajo no es IT\_PROG, ST\_CLERK, o SA\_REP.

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

1 - 95

## Agenda

- Limitando las filas con:
  - Clausula WHERE
  - Operadores de comparación =, <=, BETWEEN, IN, LIKE, y condiciones NULL
  - Condiciones lógicas usando los operadores AND, OR, y NOT
- Reglas de prioridad para los operadores en una expresión
- Ordenación de filas utilizando la cláusula ORDER BY
- Limitando filas en consultas SQL
- Las variables de sustitución
- Comandos DEFINE y VERIFY

1 - 96

## Reglas de Prioridad

Operator	Meaning
1	Operadores Aritméticos
2	Operador de Concatenación
3	Condiciones de Comparación
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Distinto que
7	Operador Lógico NOT
8	Operador Lógico AND
9	Operador Lógico OR

Se pueden utilizar los paréntesis para sustituir las reglas de prioridad

1 - 97

## Reglas de Prioridad

```
SELECT last_name, department_id, salary
FROM employees
WHERE department_id = 60
OR department_id = 80
AND salary > 10000;
```

1

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Hunold	60	9000
2	Ernst	60	6000
3	Lorentz	60	4200
4	Zlotkey	80	10500
5	Abel	80	11000

```
SELECT last_name, department_id, salary
FROM employees
WHERE (department_id = 60
OR department_id = 80)
AND salary > 10000;
```

2

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Zlotkey	80	10500
2	Abel	80	11000

1 - 98

## Agenda

- Limitando las filas con:
  - Clausula WHERE
  - Operadores de comparación =, <=, BETWEEN, IN, LIKE, y condiciones NULL
  - Condiciones lógicas usando los operadores AND, OR, y NOT
- Reglas de precedencia para los operadores en una expresión
- Ordenación de filas utilizando la cláusula ORDER BY
- Limitando filas en consultas SQL
- Las variables de sustitución
- Comandos DEFINE y VERIFY

1 - 99



## Usando la cláusula ORDER BY

- El orden de las filas que se devuelven en un resultado de la consulta no está definido.
- La cláusula ORDER BY se puede utilizar para ordenar las filas según los criterios requeridos.

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	De Haan	AD_VP	90	13-JAN-01
2	Gietz	AC_ACCOUNT	110	07-JUN-02
3	Higgins	AC_MGR	110	07-JUN-02
4	King	AD_PRES	90	17-JUN-03
5	Whalen	AD_ASST	10	17-SEP-03
6	Rajs	ST_CLERK	50	17-OCT-03

1 - 100

## Ordenación

- Ordenando en orden descendente:

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY department_id DESC ;
```

1

- Ordenando por un alias de columna:

```
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;
```

2

Números (ASC) → 1 .. 9  
Fechas (ASC) → 01-ENE-92 antes que 01-ENE-95  
Caracteres (ASC) → "A" primero y "Z" última  
Los valores NULOS se muestran al final en ASC y los primeros en DESC

NULLS FIRST  
NULLS LAST

1 - 101

## Ordenación

- Ordenación usando posiciones de columnas

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY 3;
```

3

- Ordenación por múltiples columnas:

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```

4

1 - 102

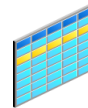
## Agenda

- Limitando las filas con:
  - Clausula WHERE
  - Operadores de comparación =, <=, BETWEEN, IN, LIKE, y condiciones NULL
  - Condiciones lógicas usando los operadores AND, OR, y NOT
- Reglas de precedencia para los operadores en una expresión
- Ordenación de filas utilizando la cláusula ORDER BY
- **Limitando filas en consultas SQL**
- Las variables de sustitución
- Comandos DEFINE y VERIFY

1 - 103

## Clausula para limitar filas en SQL

- En Oracle Database 12c Release 1, el `SELECT` se ha mejorado para clausulas que limiten el el número de filas que se devuelven en el conjunto de resultados.
- Las consultas que ordenan los datos y luego limitar el número de filas son ampliamente utilizados y se refieren a menudo como consultas `Top-N`.
- Las consultas `Top-N` obtienen un resultado y luego devolver sólo las primeras `n` filas.



1 - 104

## Clausula para limitar filas en SQL

- Se puede especificar el número de filas o porcentaje de filas a devolver con la clausula `FETCH FIRST [opciones]`
- Esta clausula se pone al final de la sentencia `SELECT` después de la clausula `ORDER BY`

```
SELECT ...  
  FROM ...  
[ WHERE ... ]  
[ ORDER BY ... ]  
[OFFSET offset { ROW | ROWS }]  
[FETCH { FIRST | NEXT } [{ row_count | percent PERCENT  
}] { ROW | ROWS }  
  { ONLY | WITH TIES }]
```

1 - 105

## Clausula para limitar filas en SQL

- `OFFSET n ROW | ROWS`
  - Para especificar a partir de qué fila hay que devolver los resultados.
  - El valor de desplazamiento debe ser un número.
  - Si especifica un número negativo, el `offset` se trata como 0.
  - Si especifica `NULL` se trata como 0..
- `FETCH ... FIRST | NEXT`
  - Utilice esta cláusula para especificar el número de filas o porcentaje de filas que se devolverán.

```
[FETCH { FIRST | NEXT } [{ row_count | percent PERCENT  
}] { ROW | ROWS }
```

<b>Row_count</b>	Número de Filas a devolver
<b>Percent N</b>	Porcentaje del resultado

1 - 106

## Clausula para limitar filas en SQL

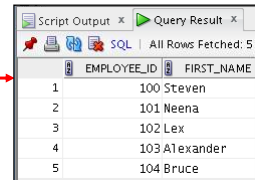
- `ONLY`
  - Especificar sólo para devolver exactamente el número especificado de filas o porcentaje de filas
- `WITH TIES`
  - Devuelve filas adicionales con la misma clave de ordenación como la última fila recuperada.
  - Si especifica `WITH TIES`, es obligatorio especificar `GROUP BY`
    - Ssino se usa Group by, esta opción no hará nada

```
{ ONLY | WITH TIES }
```

1 - 107

## Clausula para limitar filas en SQL: Ejemplo

```
SELECT employee_id, first_name  
FROM employees  
ORDER BY employee_id  
FETCH FIRST 5 ROWS ONLY;
```

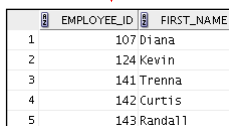


Script Output x Query Result x

SQL | All Rows Fetched: 5

	EMPLOYEE_ID	FIRST_NAME
1	100	Steven
2	101	Neena
3	102	Lex
4	103	Alexander
5	104	Bruce

```
SELECT employee_id, first_name  
FROM employees  
ORDER BY employee_id  
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```



	EMPLOYEE_ID	FIRST_NAME
1	107	Diana
2	124	Kevin
3	141	Trenna
4	142	Curtis
5	143	Randal

1 - 108

## Agenda

- Limitando las filas con:
  - Clausula WHERE
  - Operadores de comparación =, <=, BETWEEN, IN, LIKE, y condiciones NULL
  - Condiciones lógicas usando los operadores AND, OR, y NOT
- Reglas de precedencia para los operadores en una expresión
- Ordenación de filas utilizando la cláusula ORDER BY
- Limitando filas en consultas SQL
- Las variables de sustitución
- Comandos DEFINE y VERIFY

1 - 109

## SUSTITUCION DE VARIABLES

- Hasta ahora las sentencias SQL se han ejecutado con columnas y condiciones predeterminadas y sus valores.
- Puede editar la cláusula WHERE para proporcionar un valor diferente cada vez que ejecute el comando, pero existe también una forma más sencilla.
- Si se utiliza una variable de sustitución en lugar de los valores exactos en la cláusula WHERE, puede ejecutar la misma consulta para diferentes valores.



1 - 110

## SUSTITUCION DE VARIABLES

Utilizar variables de sustitución para:

- Almacenar valores temporalmente con una sustitución de un solo ampersand (&) y de dos ampersands (&&)

Utilizar las variables de sustitución para complementar:

- Condiciones WHERE
- Cláusulas ORDER BY
- Expresiones de columna
- Nombres de tabla
- Sentencias SELECT completas

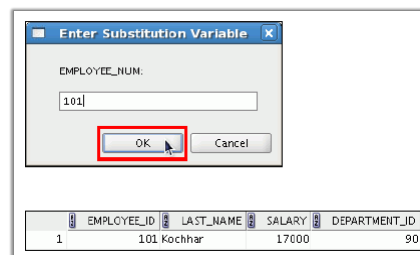
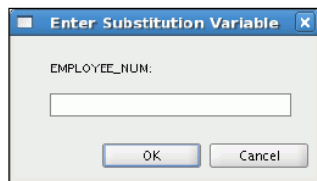
1 - 111

## Uso de la Variable de Sustitución

### Un Solo Ampersand

- Utilizar una variable prefijada con un ampersand (&) para solicitar al usuario un valor:

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num ;
```



	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

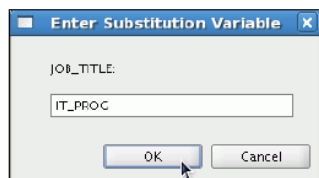
1 - 112

## Uso de la Variable de Sustitución

### Valores de Fecha y Carácter

- Utilizar las comillas simples para los valores de fecha y carácter

```
SELECT last_name, department_id, salary*12
FROM employees
WHERE job_id = '&job_title' ;
```



	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

1 - 113

## Uso de la Variable de Sustitución

Nombres de Columna, Expresiones y Texto

```
SELECT employee_id, last_name, job_id, &column_name  
FROM employees  
WHERE &condition  
ORDER BY &order_column ;
```

Enter Substitution Variable

COLUMN\_NAME:  
salary

OK

Enter Substitution Variable

CONDITION:  
salary > 15000

OK

Enter Substitution Variable

ORDER\_COLUMN:  
last\_name

OK Cancel

1 - 114

## Uso de la Variable de Sustitución

DOS Ampersand

- Usar dos ampersands (&&) si se desea reutilizar el valor de la variable sin preguntar siempre al usuario:

```
SELECT employee_id, last_name, job_id, &&column_name  
FROM employees  
ORDER BY &column_name ;
```

Enter Substitution Variable

COLUMN\_NAME:  
department\_id

OK Cancel

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

1 - 115



## Agenda

- Limitando las filas con:
  - Clausula WHERE
  - Operadores de comparación =, <=, BETWEEN, IN, LIKE, y condiciones NULL
  - Condiciones lógicas usando los operadores AND, OR, y NOT
- Reglas de precedencia para los operadores en una expresión
- Ordenación de filas utilizando la cláusula ORDER BY
- Limitando filas en consultas SQL
- Las variables de sustitución
- Comandos DEFINE y VERIFY

1 - 116

## Uso del Comando DEFINE

Usar el comando DEFINE para crear y asignar un valor a una variable.

Usar el comando UNDEFINE de iSQL\*Plus para eliminar una variable.

```
DEFINE employee_num = 200
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num ;
UNDEFINE employee_num
```

1 - 117

## Uso del Comando VERIFY

Usar el comando VERIFY para cambiar la visualización de la variable de sustitución, antes y después de que SQL Developer sustituya las variables de sustitución con los valores

The screenshot shows the SQL Developer interface. On the left, a code editor displays the following SQL query:

```
SET VERIFY ON
SELECT employee_id, last_name, salary
FROM employees
WHERE employee_id = &employee_num;
```

Below the code editor, the 'Enter Substitution Variable' dialog box is open, showing 'EMPLOYEE\_NUM:' with the value '200' entered. The 'OK' button is highlighted.

On the right, the 'Script Output' tab is active, showing the execution results. The query is displayed with the substitution variable replaced by its value:

```
SELECT employee_id, last_name, salary
FROM employees
WHERE employee_id = 200
```

Below the query, the results are shown in a table:

EMPLOYEE_ID	LAST_NAME	SALARY
200	Whalen	4400

At the bottom, it indicates '1 rows selected'.

1 - 118

## Usando la clausula ORDER BY

```
SELECT      expr
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expr, numeric_position} [ASC|DESC]];
```

- Permite ordenar las filas utilizando los criterios siguientes:
  - Orden ascendente (valor por defecto) ASC
  - Orden descendente (DESC) DESC
  - Por múltiples columnas (la columna más a la izquierda es por la que primero se clasifica)
  - Con valores nulos
- Puede especificar una expresión, un alias, o una posición de la columna como la condición de clasificación.

1 - 119

## Resumen

En esta lección, usted debe haber aprendido a :

- Limitar las filas recuperadas por una consulta
- Ordenar las filas que se recuperan mediante una consulta
- Utilizar la sustitución de ampersand para restringir y ordenar la salida en tiempo de ejecución

1 - 121

## Practica 3:

Esta práctica se abordan los siguientes temas:

- Selección de datos y cambiar el orden de las filas que se muestran
- La restricción de las filas mediante la cláusula `WHERE`
- Ordenando las filas mediante la cláusula `ORDER BY`
- El uso de variables de sustitución para facilitar la flexibilidad de sus sentencias `SELECT`

1 - 122

# 4

## Funciones de una sólo Fila



ORACLE®

### Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente :

- Describir los diferentes tipos de funciones disponibles en SQL
- Utilizar las funciones de carácter, número y de fecha en las instrucciones SELECT

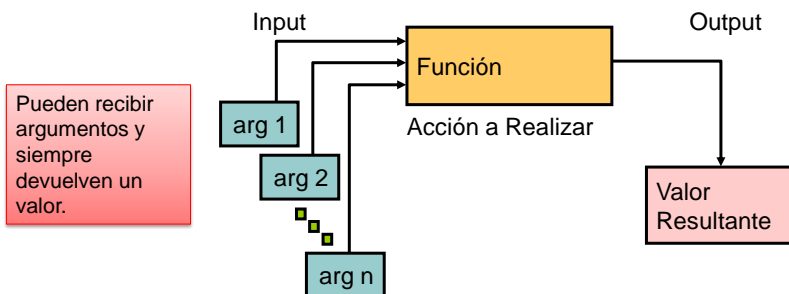
## Agenda

- Las funciones de SQL de una sola fila
  - Funciones de caracteres
  - Funciones de anidación
  - Funciones de números
  - Trabajando con fechas
  - Funciones de fecha

1 - 125

## Funciones SQL

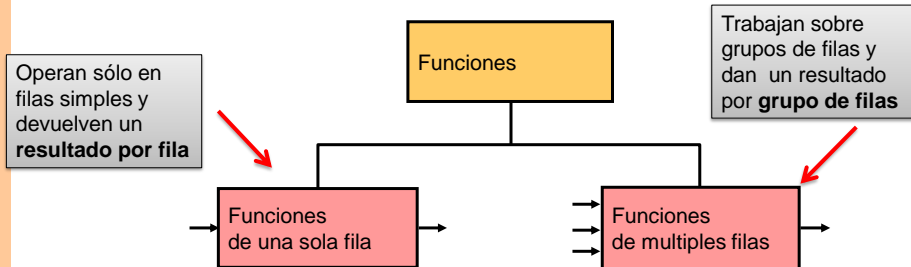
- Las funciones son una característica muy potente de SQL.
- Pueden ser utilizados para hacer lo siguiente:
  - Realizar cálculos sobre los datos
  - Modificar elementos de datos individuales
  - Manipular de salida para grupos de filas
  - Cambiar formatos de fechas y números



1 - 126

## Dos tipos de funciones de SQL

- Hay 2 tipos de funciones:
  - Funciones **de una Sola Fila**
  - Funciones **de Varias Filas (Grupo)**



1 - 127

## Funciones de una Sola Fila

Funciones de una sola fila, permiten:

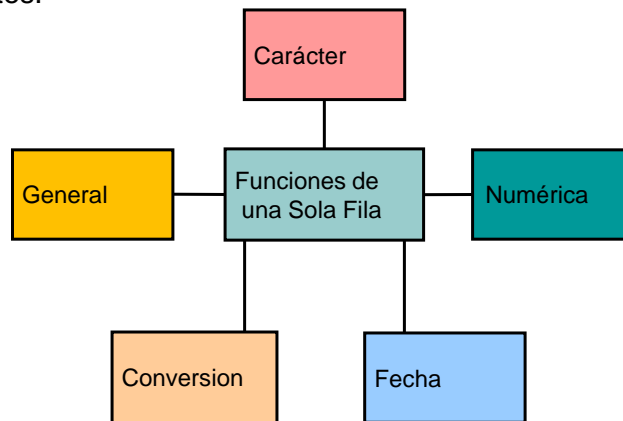
- Manipular los elementos de datos
- Aceptar los argumentos y devolver un valor
- Actuar en cada fila que se devuelve
- Devolver un resultado por fila
- Puede modificar el tipo de datos
- Se pueden anidar
- Aceptar los argumentos que pueden ser una columna o una expresión
- Puede ser utilizado en SELECT, WHERE y ORDER BY

```
function_name [(arg1, arg2,...)]
```

1 - 128

## Funciones de una Sola Fila

En esta lección cubre las siguientes funciones de una sola fila siguientes:



1 - 129

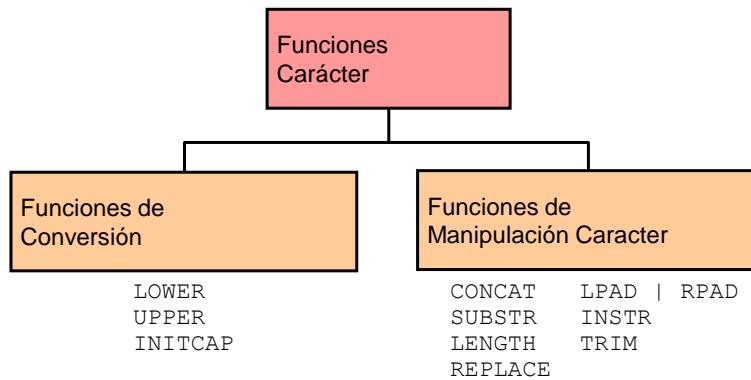
## Agenda

- Las funciones de SQL de una sola fila
- **Funciones de caracteres**
- Funciones de anidación
- Funciones de números
- Trabajando con fechas
- Funciones de fecha

1 - 130

## Funciones de Caracter

- Las funciones de caracteres de una sola fila aceptan datos de caracteres como entrada y pueden devolver valores numéricos tanto carácter y. funciones de caracteres



1 - 131

## Funciones de Conversión

Estas funciones convierten de cadenas de caracteres:

Function	Result
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

1 - 132



## Funciones de Conversión

Visualizar el número de empleado, nombre y número de departamento para los empleados apellidados higgins:

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE last_name = 'higgins';
```

0 rows selected

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205 Higgins	110

- El primer caso, no aparece ninguna fila porque los datos de la tabla EMPLEADOS se almacena en MAYUSCULAS

1 - 133

## Funciones de Manipulación de Caracteres

Estas funciones manipulan cadenas de caracteres:

Function	Result
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,12, '*')	*****24000
RPAD(salary, 12, '*')	24000*****

1 - 134

## Uso: Funciones de Manipulación de Caracteres

**1**

```
SELECT CONCAT(CONCAT(last_name, ''s job category is '), job_id)
"Job" FROM employees
WHERE SUBSTR(job_id, 4) = 'REP';
```

Job
1 Abel's job category is SA_REP
2 Fay's job category is MK_REP
3 Grant's job category is SA_REP
4 Taylor's job category is SA_REP

**2**

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM employees
WHERE SUBSTR(last_name, -1, 1) = 'n';
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
1	102 Lex De Haan	7	5
2	200 Jennifer Whalen	6	3
3	201 Michael Hartstein	9	2

empleados cuyos apellidos terminan con la letra "n".

1 - 135

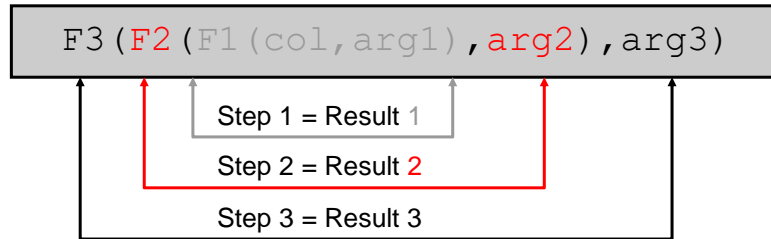
## Agenda

- Las funciones de SQL de una sola fila
- Funciones de caracteres
- **Funciones de anidación**
- Funciones de números
- Trabajando con fechas
- Funciones de fecha

1 - 136

## Funciones Anidadas

- Las funciones de una sola fila se pueden anidar a cualquier nivel.  
Funciones anidadas se evalúan desde el nivel más profundo al nivel menos profundo.



1 - 137

## Funciones Anidadas: Ejemplos

```
SELECT last_name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
FROM   employees  
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

Evaluación de la instrucción SQL:

1. SUBSTR (LAST\_NAME, 1, 8)
2. CONCAT(Result1, '\_US')
3. UPPER( )

Recupera los ocho primeros caracteres del apellido  
concatena el resultado con \_us.  
Convierte los resultados a las mayúsculas

1 - 138

## Agenda

- Las funciones de SQL de una sola fila
- Funciones de caracteres
- Funciones de anidación
- **Funciones de números**
- Trabajando con fechas
- Funciones de fecha

1 - 139

## Funciones numéricas

- **ROUND:** Redondea el valor a un decimal indicado
- **TRUNC:** Trunca el valor a un decimal indicado
- **CEIL:** Devuelve el número entero más pequeño mayor o igual a un número especificado
- **FLOOR:** Devuelve el mayor número entero igual o menor que un número especificado
- **MOD:** Devuelve el resto de la división

Function	Result
ROUND (45.926, 2)	45.93
TRUNC (45.926, 2)	45.92
CEIL (2.83)	3
FLOOR (2.83)	2
MOD (1600, 300)	100

1 - 140

## Usando la función ROUND

```
SELECT ROUND(45.923, 2), ROUND(45.923, 0),  
       ROUND(45.923, -1)  
FROM   DUAL;
```

	ROUND(45.923,2)	ROUND(45.923,0)	ROUND(45.923,-1)
1	45.92	46	50

DUAL es una tabla pública que se puede utilizar para ver los resultados de las funciones y cálculos.

1 - 141

## Usando la función TRUNC

```
SELECT TRUNC(45.923, 2), TRUNC(45.923),  
       TRUNC(45.923, -1)  
FROM   DUAL;
```

	TRUNC(45.923,2)	TRUNC(45.923)	TRUNC(45.923,-1)
1	45.92	45	40

1 - 142

## Usando la función MOD

Mostrar los registros de los empleados donde el `employee_id` es un número par

```
SELECT employee_id as "Even Numbers", last_name  
FROM employees  
WHERE MOD(employee_id,2) = 0;
```

	Even Numbers	LAST_NAME
1	174	Abel
2	142	Davies
3	102	De Haan
4	104	Ernst
5	202	Fay
6	206	Gietz
7	178	Grant
8	100	King
9	124	Mourgos
10	176	Taylor
11	144	Vargas
12	200	Whalen

1 - 143

## Agenda

- Las funciones de SQL de una sola fila
- Funciones de caracteres
- Funciones de anidación
- Funciones de números
- **Trabajando con fechas**
- Funciones de fecha

1 - 144

## Trabajando con Fechas

- En Oracle, las fechas es almacenada en formato numérico interno: **siglo, año, mes, día, horas, minutos y segundos**.
- El formato de fecha por defecto es DD-MON-RR. (en 12c)
  - Le permite almacenar fechas del siglo 21 en el siglo 20, especificando sólo los dos últimos dígitos del año
  - Le permite almacenar fechas del siglo 20 en el siglo 21 de la misma manera

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date < '01-FEB-2008';
```

	LAST_NAME	HIRE_DATE
1	King	17-JUN-03
2	Kochhar	21-SEP-05

...

1 - 145

## Formato de Fecha RR

Current Year	Specified Date	RR Format	YY Format
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		Si el año de dos dígitos especificado es:	
		0–49	50–99
Si dos dígitos del año en curso son los siguientes:	0–49	La fecha devuelta es en el siglo <b>actual</b>	La fecha devuelta es en el siglo <b>antes</b> del actual
	50–99	La fecha devuelta es en el siglo <b>después</b> del actual	La fecha devuelta es en el siglo <b>actual</b>

1 - 146

## Usando la función SYSDATE

- SYSDATE devuelve la fecha y hora actual definida para el sistema operativo en el que reside la base de datos

SYSDATE es una función que devuelve:

- Date
- Time

```
SELECT sysdate
FROM dual;
```

	SYSDATE
1	24-AUG-12

### NOTA:

Si estamos en España y conectado a una base de datos remota Estados Unidos (EE.UU.), la función sysdate devolverá la fecha y la hora EE.UU

**CURRENT\_DATE** (devuelve la fecha actual en la zona de tiempo de la sesión)

1 - 148

## Usando las funciones CURRENT\_DATE y CURRENT\_TIMESTAMP

- CURRENT\_DATE devuelve la fecha actual de la sesión de usuario.

```
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

	SESSIONTIMEZONE	CURRENT_DATE
1	Etc/Universal	26-MAY-14

- CURRENT\_TIMESTAMP Devuelve la fecha actual y la hora de la sesión de usuario.

```
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;
```

	SESSIONTIMEZONE	CURRENT_TIMESTAMP
1	Etc/Universal	26-MAY-14 12.25.34.401622000 AM ETC/UNIVERSAL

1 - 149



## Aritmética con fechas

- Añadir o restar un número (**días**) a una fecha para obtener un valor **de fecha resultante**.
- Restas dos fechas para encontrar el **número de días** entre esas fechas.
- Añadir horas a una fecha.
- **NO** se pueden **sumar 2 fechas**

1 - 150

## Usando operadores aritméticos con fechas

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM employees  
WHERE department_id = 90;
```

	LAST_NAME	WEEKS
1	King	478.871917989417989417989417989418
2	Kochhar	360.729060846560846560846560846561
3	De Haan	605.300489417989417989417989417989

El ejemplo de la diapositiva muestra el apellido y el número de semanas que los trabajadores del departamento 90 llevan en la empresa.

1 - 151

## Agenda

- Las funciones de SQL de una sola fila
- Funciones de caracteres
- Funciones de anidación
- Funciones de números
- Trabajando con fechas
- Funciones de fecha

1 - 152

## Funciones de Manipulación de Fechas

Function	Resultado
MONTHS_BETWEEN	Número de meses entre dos fechas
ADD_MONTHS	Añadir meses a una fecha
NEXT_DAY	Día de la semana de la fecha especificada
LAST_DAY	Último día del mes
ROUND (fecha [, 'fmt'])	Fecha redondeada a la unidad específica da en <b>fmt</b> . Si se omite el modelo de formato <b>fmt</b> , la fecha se redondea al día más cercano
TRUNC (fecha [, 'fmt'])	Devuelve fecha con la parte de hora del día truncado según <b>fmt</b> . Si se omite el modelo de formato <b>fmt</b> , la fecha se redondea al día más cercano

1 - 153

## Usando Funciones de Fecha

Function	Result
MONTHS_BETWEEN ( '01-SEP-05', '11-JAN-04' )	19.6774194
ADD_MONTHS ( '31-JAN-04', 1 )	'29-FEB-04'
NEXT_DAY ( '01-SEP-05', 'FRIDAY' )	'08-SEP-05'
LAST_DAY ( '01-FEB-05' )	'28-FEB-05'

1 - 154

## Usando las funciones ROUND y TRUNC con fechas

- Las funciones ROUND y TRUNC usadas con con fechas, se truncan o redondean al modelo de formato especificado.
- Se puede redondear a las fechas del año o el mes próximo.
  - Si el modelo es mensual,
    - Fechas 1-15 resultado en el primer día del mes en curso.
    - Fechas 16-31 resultado en el primer día del siguiente mes.
  - Si el modelo de formato es el año
    - Mes 1-6 resultado en el 1 de enero del año en curso.
    - Mes 7-12 meses como resultado 1 de enero del próximo año.

Function	Result
ROUND (SYSDATE, 'MONTH' )	01-AUG-03
ROUND (SYSDATE , 'YEAR' )	01-JAN-04
TRUNC (SYSDATE , 'MONTH' )	01-JUL-03
TRUNC (SYSDATE , 'YEAR' )	01-JAN-03

1 - 155

## Resumen

En esta lección, usted debe haber aprendido a:

- Describir los diferentes tipos de funciones disponibles en SQL
- Utilizar las funciones de carácter, número y de fecha en las instrucciones SELECT

1 - 157

## Practica 4:

Esta práctica se abordan los siguientes temas

- Escribir una consulta que muestra el `SYSDATE`
- Creación de consultas que requieren el uso de funciones numéricas, de caracteres y de fecha
- La realización de los cálculos de años y meses de servicio para un empleado

1 - 158

# 5

## Uso de las funciones de conversión y las expresiones condicionales



ORACLE®

### Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente :

- Describir los diferentes tipos de funciones de conversión que están disponibles en SQL
- Utilice las funciones de conversión `TO_CHAR`, `TO_NUMBER`, y `TO_DATE`
- Aplicar las expresiones condicionales en una instrucción `SELECT`

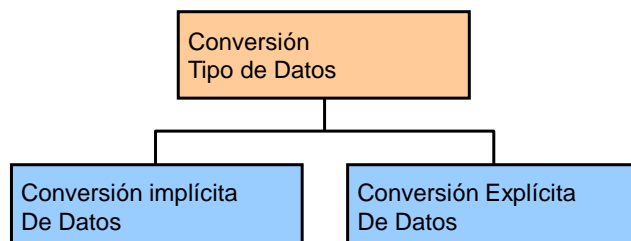
## Agenda

- Conversión implícita y explícita de tipo de datos
- Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Funciones Generales:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Expresiones Condicionales:
  - `CASE`
  - `Searched CASE`
  - `DECODE`

1 - 161

## Conversion Functions

- En algunos casos, el servidor Oracle recibe datos de un tipo no esperado y los puede convertir automáticamente los datos al tipo de datos esperado.
- Esta conversión de tipo de datos pueden ser de forma implícita o explícita



1 - 162

## Conversion Functions

- **IMPLICITAS:**
  - Las conversiones `IMPLICITAS` de tipo de datos son aplicadas de forma automática por Oracle y se explica en las siguientes diapositivas.
- **EXPLICITAS**
  - Las conversiones `EXPLICITAS`, son realizadas por el usuario y se realizan mediante el uso de las funciones de conversión.

### Nota:

- Aunque la conversión implícita de tipo de datos está disponible, se recomienda que lo haga la conversión de tipo de datos explícito para garantizar la fiabilidad de las sentencias SQL.

1 - 163

## Conversión Implícita de Datos

El servidor de Oracle puede realizar automáticamente la conversión de tipos de datos en una expresión

- La expresión `hire_date > '01-JAN-90'` convierte internamente '01-JAN-90' a una fecha y evalúa la condición
- Por lo tanto, un valor `VARCHAR2` o `CHAR` se puede convertir implícitamente a un número o fecha de tipo de datos en una expresión.

From	To
<code>VARCHAR2</code> or <code>CHAR</code>	<code>NUMBER</code>
<code>VARCHAR2</code> or <code>CHAR</code>	<code>DATE</code>

1 - 164

## Conversión Implícita de Datos

En general, el servidor Oracle utiliza la regla para las expresiones cuando se necesita una conversión de tipos de datos.

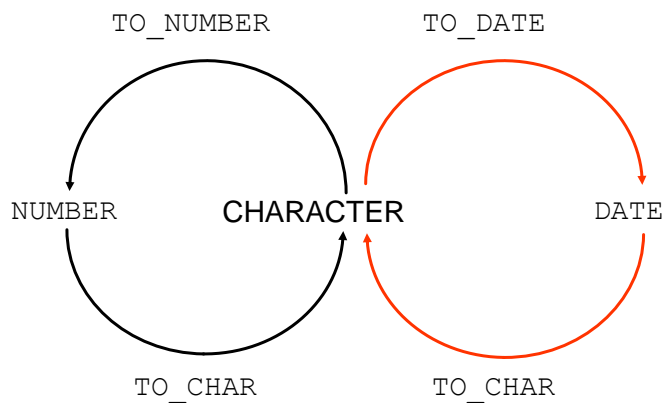
- Por ejemplo, la expresión `job_id = 2` da como resultado la conversión implícita del número 2 de la cadena "2" porque es un `job_id` (2) la columna `VARCHAR`.

From	To
NUMBER	VARCHAR2 or CHAR
DATE	VARCHAR2 or CHAR

1 - 165

## Conversión Explícita de Datos

SQL proporciona tres funciones para convertir un valor de un tipo de datos a otro:



1 - 166



## Agenda

- Conversión implícita y explícita de tipo de datos
- **Funciones** TO\_CHAR, TO\_DATE, TO\_NUMBER
- Funciones Generales:
  - NVL
  - NVL2
  - NULLIF
  - COALESCE
- Expresiones Condicionales:
  - CASE
  - Searched CASE
  - DECODE

1 - 167

## Usando la funcion TO\_CHAR con fechas

TO\_CHAR convierte un tipo de datos de fecha y hora a un valor de tipo de datos VARCHAR2 con un formato especificado

- Un modelo de formato es un literal de caracteres que describe el formato de fecha y hora

```
TO_CHAR(date[, 'format_model'])
```

### Modelo de Formato

- Debe estar encerrado entre comillas simples
- Distingue entre mayúsculas y minúsculas
- Puede incluir cualquier formato de fecha válida
- Se separa del valor de fecha por una coma

```
SELECT employee_id,  
       TO_CHAR(hire_date, 'MM/YY')  
FROM   employees  
WHERE  last_name = 'Higgins';
```

1 - 168

## Elementos disponibles en el Modelo de Formato

Elemento	Resultado
YYYY	Año completo en números
YEAR	Año
MM	Valor de dos dígitos para el me
MONTH	Nombre completo del Mes
MON	Tres letras de la abreviatura del mes
DY	Tres letras de la abreviatura del día de la semana
DAY	Nombre completo del día de la semana
DD	Día de la Semana

1 - 169

## Elementos disponibles en el Modelo de Formato: Ejemplos

- Formato 24h de horas, minutos y segundos

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

- Añadir cadenas de caracteres encerrándolas entre comillas dobles:

DD "of" MONTH	12 of OCTOBER
---------------	---------------

1 - 170

## Using la función TO\_CHAR con fechas

```
SELECT last name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
       AS HIREDATE  
FROM   employees;
```

	LAST_NAME	HIREDATE
1	King	17 June 2003
2	Kochhar	21 September 2005
3	De Haan	13 January 2001
4	Hunold	3 January 2006
5	Ernst	21 May 2007
6	Lorentz	7 February 2007
7	Mourgos	16 November 2007
8	Rajs	17 October 2003

...

- Muestra los apellidos y las fechas de contratación para todos los empleados.
- El formato `fm` elimina todos los CEROS no válidos del elemento

1 - 171

## Usando la función TO\_CHAR con números

Estos son algunos de los elementos de formato que se pueden utilizar con la función TO\_CHAR para mostrar un valor numérico

```
TO_CHAR(number[, 'format_model'])
```

Elemento	Resultado
9	Representa un número
0	Obliga a que aparezcan ceros a la izquierda
\$	Coloca un signo de dólar \$
L	Utiliza el símbolo de moneda local
.	Imprime el punto decimal
,	Imprime el separador de miles (,)

1 - 172

## Usando la función TO\_CHAR con números

### NOTAS

- El servidor de Oracle muestra una cadena de signos (#) cuando el número de dígitos superar el número de dígitos previstos en el formato.
- El servidor de Oracle redondea el valor decimal almacenado con el número de cifras decimales previstas en el modelo de formato.

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM   employees  
WHERE  last_name = 'Ernst';
```

	SALARY
1	\$6,000.00

1 - 173

## Usando las funciones TO\_NUMBER y TO\_DATE

- Convertir una cadena de caracteres a un formato de número utilizando la función TO\_NUMBER :

```
TO_NUMBER(char[, 'format_model'])
```

- Convertir una cadena de caracteres a un formato de fecha utilizando la función TO\_DATE :

```
TO_DATE(char[, 'format_model'])
```

1 - 174

## Usando las funciones TO\_NUMBER y TO\_DATE

- Estas funciones tienen un modificador de `fx` que permite buscar, exclusivamente, la coincidencia indicada sin tener en cuenta los blancos existente

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May 24, 2007',
'fxMonth DD, YYYY');
```

- Hay dos espacios después del mes de mayo y ante el número 24 en el ejemplo.
- Debido a que se utiliza el modificador `fx`, se requiere una coincidencia exacta y los espacios después de la palabra no se reconocen

1 - 175

## Usando las funciones TO\_CHAR y TO\_DATE con el formato de Fecha RR

- Encontrar a los empleados contratados antes de 1990.
- Utilizaremos el formato de fecha RR.
  - Debido a que el año en curso es mayor que 1999, el formato RR interpreta la parte de año de la fecha 1950-1999.

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM   employees
WHERE  hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1 Popp	03-Feb-1989

1 - 176

## Agenda

- Conversión implícita y explícita de tipo de datos
- Funciones TO\_CHAR, TO\_DATE, TO\_NUMBER
- **Funciones Generales:**
  - NVL
  - NVL2
  - NULLIF
  - COALESCE
- **Expresiones Condicionales:**
  - CASE
  - Searched CASE
  - DECODE

1 - 177

## Funciones Generales

Las siguientes funciones se pueden utilizar con cualquier tipo de datos y permiten la utilización de los nulos:

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

1 - 178

## Funcion NVL

Convierte un valor nulo a un valor real::

- Los tipos de datos que se pueden utilizar son la fecha, carácter, y el número.
- Sintaxis:
  - `expr1` es el valor de la fuente o expresión que puede contener un valor nulo
  - `expr2` es el valor objetivo para la conversión del null
- Ejemplos
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

1 - 179

## Usando la función NVL

```
SELECT last name, salary, NVL(commission_pct, 0)
      (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	King	24000	0	288000
2	Kochhar	17000	0	204000
3	De Haan	17000	0	204000
4	Hunold	9000	0	108000
5	Ernst	6000	0	72000
6	Lorentz	4200	0	50400
7	Mourgos	5800	0	69600
8	Rajs	3500	0	42000
9	Davies	3100	0	37200
10	Matos	2600	0	31200

...

En el ejemplo se ha utilizado NVL para convertir NULL a ceros

1 - 180

## Usando la función NVL2

### NVL2(expr1, expr2, expr3)

- La función NVL2 examina la primera expresión.
  - Si la primera expresión no es nulo, la función devuelve el NVL2 **segunda expresión**.
  - Si la primera expresión es nulo, se devuelve la **tercera expresión**.

```
SELECT last_name, salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

1 - 181

## Usando la función NULLIF

### NULLIF(expr1, expr2)

- La función NULLIF compara dos expresiones:
  - Si son iguales la función devuelve **NULL**.
  - Si no son iguales la función devuelve **expr1**.

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name, LENGTH(last_name) "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```

	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
8	Michael	7	Hartstein	9	7
9	Shelley	7	Higgins	7	(null)

1 - 182



## Usando la función COALESCE

### COALESCE(expr1, expr2, ... exprn)

- La función COALESCE es una ampliación de la función NVL
- La ventaja de la función COALESCE sobre la función NVL es que la función COALESCE puede tomar varios valores alternativos.
- Si la primera expresión no es nulo, la función COALESCE devuelve esa expresión; de lo contrario, hace un COALESCE de las expresiones restantes.

1 - 183

## Usando la función COALESCE

```
SELECT last name, salary, commission_pct,  
COALESCE((salary+(commission_pct*salary)), salary+2000) "New Salary"  
FROM employees;
```

	LAST_NAME	SALARY	COMMISSION_PCT	New Salary
1	King	24000	(null)	26000
2	Kochhar	17000	(null)	19000
3	De Haan	17000	(null)	19000
4	Hunold	9000	(null)	11000
5	Ernst	6000	(null)	8000
6	Lorentz	4200	(null)	6200
7	Mourgos	5800	(null)	7800
8	Rajs	3500	(null)	5500
9	Davies	3100	(null)	5100
10	Matos	2600	(null)	4600
11	Vargas	2500	(null)	4500
12	Zlotkey	10500	0.2	12600
13	Abel	11000	0.3	14300
14	Taylor	8600	0.2	10320
15	Grant	7000	0.15	8050
16	Whalen	4400	(null)	6400
17	Hartstein	13000	(null)	15000
18	Fay	6000	(null)	8000
19	Higgins	12008	(null)	14008
20	Gietz	8300	(null)	10300

- Para los empleados que no reciben ningún tipo de comisión, la columna de nuevo sueldo muestra el salario se incrementa en \$ 2.000
- Para los empleados que reciben comisión, la columna de nuevo sueldo muestra el monto de la comisión calculada añadido al salario

1 - 184

## Agenda

- Conversión implícita y explícita de tipo de datos
- Funciones TO\_CHAR, TO\_DATE, TO\_NUMBER
- Funciones Generales:
  - NVL
  - NVL2
  - NULLIF
  - COALESCE
- Expresiones Condicionales:
  - CASE
  - Searched CASE
  - DECODE

1 - 185

## Expresiones condicionales

- Proporcionar el uso de la lógica IF-THEN-ELSE dentro de una instrucción SQL
- Utilice los métodos siguientes:
  - expresión CASE
  - Búsqueda expresión CASE
  - Función DECODE

1 - 186

## Expresión CASE

- Facilita las consultas condicionales haciendo el trabajo de una instrucción IF-THEN-ELSE

```
CASE expr WHEN comparison_expr1 THEN return_expr1
          [WHEN comparison_expr2 THEN return_expr2
           WHEN comparison_exprn THEN return_exprn
           ELSE else_expr]
END
```

WHEN . . . . THEN return . . . .

Diferentes opciones de comparación

ELSE

Si no se cumple ninguna condición y existe una clausula ELSE, se ejecuta;  
de lo contrario devuelve NULL

NOTA:

Las expresiones expr y comparison\_expr deben ser del mismo tipo

1 - 187

## Usando la Expresión CASE

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary
                   WHEN 'ST_CLERK' THEN 1.15*salary
                   WHEN 'SA_REP' THEN 1.20*salary
                   ELSE salary END "REVISED_SALARY"
FROM employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
1	King	AD_PRES	24000	24000
...				
4	Hunold	IT_PROG	9000	9900
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
10	Matos	ST_CLERK	2600	2990
11	Vargas	ST_CLERK	2500	2875
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
15	Grant	SA_REP	7000	8400

1 - 188

## Búsqueda de expresión CASE

- La búsqueda se produce de izquierda a derecha hasta que la aparición de las condiciones enumeradas se encuentra, y luego se devuelve la expresión de retorno.
- Si no se encuentra ninguna condición para ser verdad, y si existe una cláusula ELSE, la expresión de retorno en la cláusula ELSE se devuelve; de lo contrario, se devuelve un NULL
- Permite la utilización de otros comparadores que no son igualdad

```
CASE
  WHEN condition1 THEN use_expression1
  WHEN condition2 THEN use_expression2
  WHEN condition3 THEN use_expression3
  ELSE default_use_expression
END
```

1 - 189

## Búsqueda de expresión CASE

```
SELECT last_name,salary,
(CASE WHEN salary<5000 THEN 'Low'
      WHEN salary<10000 THEN 'Medium'
      WHEN salary<20000 THEN 'Good'
      ELSE 'Excellent'
END) qualified_salary
FROM employees,
```

1 - 190

## Funcion DECODE

- La función DECODE decodifica una expresión de una manera similar a la lógica IF-THEN-ELSE.
- La función DECODE compara la expresión con cada valor posible. Si la expresión es igual a un valor, devuelve el resultado indicado
- Se puede poner un valor por **defecto a devolver**, sino NULL

```
DECODE(col|expression, search1, result1  
      [, search2, result2,...,]  
      [, default])
```

1 - 191

## Usando la función DECODE

```
SELECT last_name, job_id, salary,  
       DECODE(job_id, 'IT_PROG', 1.10*salary,  
                 'ST_CLERK', 1.15*salary,  
                 'SA_REP', 1.20*salary,  
                 salary)  
       REVISED_SALARY  
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
4	Hunold	IT_PROG	9000	9900
5	Ernst	IT_PROG	6000	6600
6	Lorentz	IT_PROG	4200	4620
7	Mourgos	ST_MAN	5800	5800
8	Rajs	ST_CLERK	3500	4025
9	Davies	ST_CLERK	3100	3565
10	Matos	ST_CLERK	2600	2990
11	Vargas	ST_CLERK	2500	2875
12	Zlotkey	SA_MAN	10500	10500
...				
13	Abel	SA_REP	11000	13200
14	Taylor	SA_REP	8600	10320
15	Grant	SA_REP	7000	8400

1 - 192

## Usando la función DECODE

se determina la tasa de impuestos para cada empleado en el departamento 80 con base en el salario mensual, en base a una tasa de impuestos:

```
SELECT last name, salary,  
       DECODE (TRUNC(salary/2000, 0),  
               0, 0.00,  
               1, 0.09,  
               2, 0.20,  
               3, 0.30,  
               4, 0.40,  
               5, 0.42,  
               6, 0.44,  
               0.45) TAX_RATE  
FROM   employees  
WHERE  department_id = 80;
```

1 - 193

## Resumen

Esta práctica se abordan los siguientes temas:

- Modificar los formatos de fecha para su visualización utilizando funciones
- Convertir datos de columnas usando funciones.
- Utilice las funciones NVL
- Usa la lógica IF-THEN-ELSE y otras expresiones condicionales en una instrucción SELECT

1 - 195

## Práctica 5:

Esta práctica se abordan los siguientes temas:

- Creación de consultas que utilizan `TO_CHAR`, `TO_DATE`, y otras funciones de fecha
- La creación de consultas que utilizan expresiones condicionales como `CASE`, `CASE` de búsqueda, y `DECODE`

1 - 196

# 6

## Informes agregados utilizando las funciones de grupo



ORACLE®

## Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Identificar las funciones de grupo disponibles
- Describir el uso de las funciones de grupo
- Grupo de los datos mediante el uso de la cláusula `GROUP BY`
- Incluir o excluir filas agrupadas mediante el uso de la cláusula `HAVING`

1 - 198

## Agenda

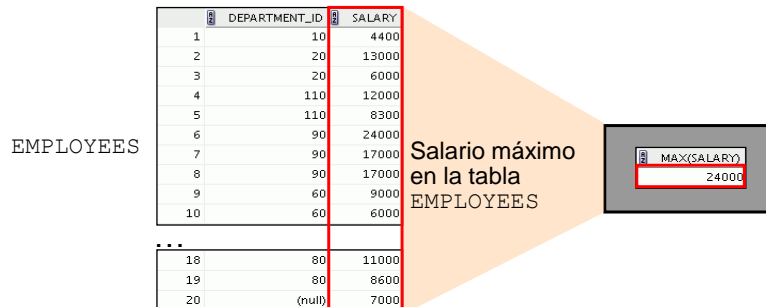
- Funciones de Grupo:
  - Tipos y sintaxis
  - Uso de `AVG`, `SUM`, `MIN`, `MAX`, `COUNT`
  - Uso de la cláusula `DISTINCT` con funciones de grupo
  - Valores `NULL` en funciones de grupo
- Filas agrupadas:
  - Cláusula `GROUP BY`
  - Cláusula `HAVING`
- Funciones de Grupo Anidadas

1 - 199



## Funciones de Grupo

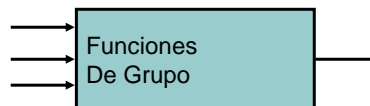
- A diferencia de las funciones de una sola fila, funciones de grupo operan en grupos de filas para dar un resultado por grupo.
- Estos conjuntos pueden comprender toda la tabla o la tabla dividida en grupos.



1 - 200

## Tipos de funciones de Grupo

- AVG
- COUNT
- MAX
- MIN
- SUM
- LISTAGG
- STDDEV
- VARIANCE



1 - 201

## Funciones de Grupo: Sintaxis

- La función de grupo se coloca después de la palabra clave `SELECT`.
- Usted puede tener múltiples funciones de grupo separados por comas.

```
SELECT  group_function([DISTINCT|ALL] column), ...  
FROM    table  
[WHERE  condition];
```

- `DISTINCT`
    - Hace que la función de considerar sólo los valores no duplicados;
  - `ALL`
    - Considera todos los valores, incluyendo los duplicados ( POR DEFECTO)
- Todas las funciones de grupo ignoran los valores nulos

1 - 202

## Usando las funciones `AVG` y `SUM`

- Puede utilizar las funciones `AVG`, `SUM`, `MIN` y `MAX` en columnas numéricas.
- El ejemplo de la diapositiva muestra:
  - El Salario MEDIO, MAXIMO, MINIMO y SUMA de todos los empleados de ventas.

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

1 - 203

## Usando las funciones MIN y MAX

- Puede utilizar las funciones MAX y MIN para los siguientes tipos de datos: numérico, carácter, y de fecha.
- Se realiza una ordenación por diccionario en los tipos carácter
- El ejemplo de la diapositiva muestra:
  - Los empleados más jóvenes y de más veteranos.

```
SELECT MIN(hire date), MAX(hire date)
FROM employees;
```

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	13-JAN-01	29-JAN-08

MAX y MIN no se pueden utilizar con tipos LOB o tipos de datos LONG.

1 - 204

## Usando la función COUNT

COUNT (\*) devuelve el número de filas en una tabla:

1 

```
SELECT COUNT(*)
FROM employees
WHERE department_id = 50;
```

	COUNT(*)
1	5

COUNT(column) devuelve el número de filas con valores no nulos para column

2 

```
SELECT COUNT(commission_pct)
FROM employees
WHERE department_id = 50;
```

	COUNT(COMMISSION_PCT)
1	0

1 - 205

## Usando la Clausula DISTINCT

- `COUNT (DISTINCT expr)` devuelve el número de valores no nulos distintos en `expr`
- Para mostrar el número de valores distintos de departamento en la tabla `EMPLOYEES`:

```
SELECT COUNT(DISTINCT department_id)
FROM employees;
```

COUNT(DISTINCTDEPARTMENT_ID)	
1	7

1 - 206

## Funciones de grupo y valores nulos

- Todas las funciones de grupo ignoran los valores nulos en la columna.

1 

```
SELECT AVG(commission_pct)
FROM employees;
```

AVG(COMMISSION_PCT)	
1	0.2125

- La utilización de la función `NVL` fuerza a la inclusión de los valores NULOS en la agrupación y en su cambio

2 

```
SELECT AVG(NVL(commission_pct, 0))
FROM employees;
```

AVG(NVL(COMMISSION_PCT,0))	
1	0.0425

1 - 207

## Agenda

- Funciones de Grupo:
  - Tipos y sintaxis
  - Uso de AVG, SUM, MIN, MAX, COUNT
  - Uso de la cláusula DISTINCT con funciones de grupo
  - Valores NULL en funciones de grupo
- Filas agrupadas:
  - Clausula GROUP BY
  - Clausula HAVING
- Funciones de Grupo Anidadas

1 - 208

## Creación de Grupos de Datos

- A veces, es necesario dividir la tabla en grupos mas pequeños para obtener información de esos grupos.
- Esto se puede hacer mediante el uso de la cláusula GROUP BY.

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
18	110	8300
19	110	12000
20	(null)	7000

4400  
9500  
3500  
6400  
10033

salario medio en la tabla  
EMPLOYEES de cada  
departamento

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.333333333333...
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	10	4400
8	60	6400

1 - 209

## Creación de Grupos de Datos: Sintaxis de GROUP BY

- Puede utilizar la cláusula GROUP BY para dividir las filas de una tabla en grupos.
- A continuación, puede utilizar las funciones de grupo para devolver información de resumen para cada grupo.

*group\_by\_expression*

Especifica las columnas cuyos valores determinan la base para agrupar filas

```
SELECT    column, group_function(column)
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

1 - 210

## Creación de Grupos de Datos: Sintaxis de GROUP BY

### Directrices

- Toda columna que aparezca en la cláusula GROUP BY debe de aparecer en la cláusula SELECT.
  - Recibe un mensaje de error si no incluye en la lista de columnas en la cláusula GROUP BY.
- El uso de una cláusula WHERE, puede excluir filas antes de dividirlos en grupos.
- Puede sustituir la columna con una expresión en la instrucción SELECT.
- No se puede utilizar un alias de columna en la cláusula GROUP BY.

1 - 211

## Usando la Clausula GROUP BY

Todas las columnas de la lista `SELECT` que no están en funciones de grupo deben estar en la cláusula `GROUP BY`.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id;
```

[illegible]

1 - 212

## Usando la Clausula GROUP BY

La columna `GROUP BY` no tiene que estar en la lista `SELECT`.

```
SELECT    AVG(salary)
FROM      employees
GROUP BY  department id ;
```

[illegible]

1 - 213

## Agrupando por mas de una columna

- A veces, es necesario agrupar por mas de una columna, con el objeto de obtener información más concreta de datos.
- La diapositiva muestra un informe que muestra el salario total que se paga a cada puesto de trabajo en cada departamento.

```
SELECT department_id, job_id,  
sum(salary)  
FROM employees  
GROUP BY department_id, job_id  
ORDER BY job_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	110	AC_ACCOUNT	8300
2	110	AC_MGR	12008
3	10	AD_ASST	4400
4	90	AD PRES	24000
5	90	AD_VP	34000
6	60	IT_PROG	19200
7	20	MK_MAN	13000
8	20	MK_REP	6000
9	80	SA_MAN	10500
10	80	SA_REP	19600
11	(null)	SA_REP	7000
12	50	ST_CLERK	11700
13	50	ST_MAN	5800

1 - 214

## Using the GROUP BY Clause on Multiple Columns

- GROUP BY grupos de filas, pero no garantiza el orden de las operaciones lógicas.
- Para ordenar las agrupaciones, utilice la cláusula ORDER BY.

```
SELECT department_id, job_id, SUM(salary)  
FROM employees  
WHERE department_id > 40  
GROUP BY department_id, job_id  
ORDER BY department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	50	ST_CLERK	11700
2	50	ST_MAN	5800
3	60	IT_PROG	19200
4	80	SA_MAN	10500
5	80	SA_REP	19600
6	90	AD PRES	24000
7	90	AD_VP	34000
8	110	AC_ACCOUNT	8300
9	110	AC_MGR	12008

1 - 215



## Consulta Ilegales en funciones de Grupo

Cualquier columna o expresión en la lista `SELECT` que no es una función de grupo deben estar en la cláusula `GROUP BY`:

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

ORA-00937: not a single-group group function  
00937. 00000 - "not a single-group group function"

La cláusula `GROUP BY` debe de aparecer para aplicar la función de grupo

```
SELECT department_id, job_id, COUNT(last_name)
FROM employees
GROUP BY department_id;
```

ORA-00979: not a GROUP BY expression  
00979. 00000 - "not a GROUP BY expression"

La columna `job_id` debe de ser agregada a la cláusula `GROUP BY` o eliminada de la `SELECT`.

1 - 216

## Consulta Ilegales en funciones de Grupo

- No se puede utilizar la cláusula `WHERE` para restringir grupos.
- Se utiliza la cláusula `HAVING` para restringir grupos.
- No se pueden utilizar las funciones de grupo en la cláusula `WHERE`.

```
SELECT department_id, AVG(salary)
FROM employees
WHERE AVG(salary) > 8000
GROUP BY department_id;
```

ORA-00934: group function is not allowed here  
00934. 00000 - "group function is not allowed here"  
\*Cause:  
\*Action:  
Error at Line: 3 Column: 9

No se puede utilizar la cláusula `WHERE` para restringir los grupos

1 - 217

## Restricción del Grupo de Resultados

- Se puede utilizar la cláusula `HAVING` para restringir grupos de la misma forma en que se utiliza la cláusula `WHERE` para restringir las filas.

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
18	110	8300
19	110	12000
20	(null)	7000

Mostrar el salario máximo por departamento cuando es mayor de \$ 10,000

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

1 - 218

## Restricción del Grupo de Resultados con `HAVING`

Cuando se utiliza la cláusula `HAVING`, el servidor Oracle restringe los grupos de la siguiente manera:

- Las filas se agrupan.
- Se aplica la función de grupo.
- Se muestran los grupos que concuerden con la cláusula `HAVING`.

```
SELECT    column, group_function
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[HAVING   group_condition]
[ORDER BY column];
```

1 - 219

## Usando la clausula HAVING

- Se muestra los números de departamento y salarios máximos para los departamentos con un sueldo máximo mayor a \$10.000.

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary) > 10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	90	24000
2	20	13000
3	110	12008
4	80	11000

1 - 220

## Usando la clausula HAVING

- Se muestra el JOB\_ID y salario mensual total para cada puesto de trabajo que tiene una nómina total superior a \$13.000

```
SELECT job_id, SUM(salary) PAYROLL
FROM employees
WHERE job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING SUM(salary) > 13000
ORDER BY SUM(salary) ;
```

	JOB_ID	PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

1 - 221

# Agenda

- Funciones de Grupo:
  - Tipos y sintaxis
  - Uso de `AVG`, `SUM`, `MIN`, `MAX`, `COUNT`
  - Uso de la clausula `DISTINCT` con funciones de grupo
  - Valores `NULL` en funciones de grupo
- Filas agrupadas:
  - Clausula `GROUP BY`
  - Clausula `HAVING`
- Funciones de Grupo Anidadas

[illegible]

## Resumen

En esta lección, usted debe haber aprendido a :

- Uso de las funciones de grupo `COUNT`, `MAX`, `MIN`, `SUM`, `AVG`, `LISTAGG`, `DESVEST`, y `VARIANCE`
- Escribir consultas que utilizan la cláusula `GROUP BY`
- Escribir consultas que utilizan la cláusula `HAVING`

1 - 225

## Práctica 6:

Esta práctica se abordan los siguientes temas:

- Escribir consultas que utilizan funciones de grupo
- La agrupación por filas para lograr más de un resultado
- La restricción de los grupos mediante el uso de la cláusula `HAVING`

1 - 226

# 7

## Viendo los datos de las multiples tablas usando JOIN



ORACLE®

### Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Escribir sentencias `SELECT` para acceder a los datos de más de una tabla utilizando `equijoins` y `nonequijoins`
- Realizar de `JOINS` de una misma tabla mediante el uso de una `self-join`
- Ver los datos obtenidos mediante `OUTER joins`
- Generar un producto cartesiano de todas las filas de dos o más tablas

## Agenda

- Tipos de JOINS y su sintaxis
  - Natural join
  - Join mediante la clausula USING
  - Join mediante la clausula ON
  - Self-join
  - Nonequijoins
  - OUTER join:
    - LEFT OUTER join
    - RIGHT OUTER join
    - FULL OUTER join
  - Producto Cartesiano
    - Cross join

1 - 229

## Obteniendo datos de múltiples tablas

- A veces es necesario utilizar los datos de más de una tabla.
- En un modelo de datos, las diferentes entidades suelen estar normalizadas y en tablas separadas
  - Los empleados en su tabla con sus identificadores correspondiente
  - Lo trabajos identificados en otra tabla y asociado a los empleados como FK
  - etc
- Para obtener todos estos datos, es necesario vincular ambas tablas de una forma correcta.

1 - 230

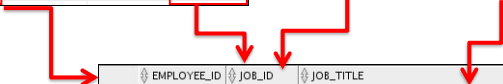
## Obteniendo datos de múltiples tablas

EMPLOYEES

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	JOB_ID
1	100	Steven	King	AD_PRES
2	101	Neena	Kochhar	AD_VP
3	102	Lex	De Haan	AD_VP
4	103	Alexander	Hunold	IT_PROG
5	104	Bruce	Ernst	IT_PROG
6	105	David	Austin	IT_PROG
7	106	Valli	Pataballa	IT_PROG
8	107	Diana	Lorentz	IT_PROG
9	108	Nancy	Greenberg	FI_MGR
10	109	Daniel	Faviet	FI_ACCOUNT

JOBS

	JOB_ID	JOB_TITLE
1	AD_PRES	President
2	AD_VP	Administration Vice President
3	AD_ASST	Administration Assistant
4	FI_MGR	Finance Manager
5	FI_ACCOUNT	Accountant
6	AC_MGR	Accounting Manager
7	AC_ACCOUNT	Public Accountant
8	SA_MAN	Sales Manager
9	SA_REP	Sales Representative



	EMPLOYEE_ID	JOB_ID	JOB_TITLE
1	206	AC_ACCOUNT	Public Accountant
2	205	AC_MGR	Accounting Manager
3	200	AD_ASST	Administration Assistant
4	100	AD_PRES	President
5	101	AD_VP	Administration Vice President
6	102	AD_VP	Administration Vice President
7	109	FI_ACCOUNT	Accountant

...

1 - 231

## Tipos de Joins

- Para unir tablas, mediante JOIN, se puede utilizar una sintaxis de la combinación que es compatible con el estándar SQL: 1999.
- Antes del lanzamiento de Oracle9i, la sintaxis utilizada por Oracle para los JOINS era diferente de las normas American National Standards Institute (ANSI).
- La estandarización del SQL 1999 ofrece una sintaxis única para todo SQL, pero no ofrece ventajas de rendimiento sobre BBDD de Oracle

1 - 232



## Tipos de Joins

Los JOIN que cumplan con el estándar SQL: 1999 son los siguientes:

- NATURAL JOIN
- JOIN con la cláusula USING
- JOIN con la cláusula ON
- OUTER joins:
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
- CROSS JOIN

1 - 233

## Uniando Tabla con la Sintaxis SQL:1999

Utilice un JOIN para consultar los datos de más de una tabla:

```
SELECT  table1.column, table2.column
FROM    table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2 ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

A continuación veremos las diferentes opciones de JOIN

1 - 234

## Agenda

- Tipos de JOINS y su sintaxis
- **Natural join**
- Join mediante la cláusula USING
- Join mediante la cláusula ON
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 235

## Creando Natural Joins

- La cláusula NATURAL JOIN es utilizada cuando existen columnas con el mismo nombre en dos tablas.
- NATURAL JOIN selecciona las filas de las dos tablas que tienen valores iguales en todas las columnas coincidentes.
- Si las columnas que tienen los mismos nombres tienen diferentes tipos de datos, se devuelve un error.

```
SELECT *  
FROM table1 NATURAL JOIN table2;
```

1 - 236

## Obteniendo Registros mediante Natural Joins

- Las Tablas JOBS y EMPLOYEES con unidas mediante la columna JOB\_ID (que es la única columna del mismo nombre en ambas tablas)

```
SELECT employee_id, first_name, job_id, job_title  
from employees NATURAL JOIN jobs;
```

	EMPLOYEE_ID	FIRST_NAME	JOB_ID	JOB_TITLE
1	100	Steven	AD_PRES	President
2	101	Neena	AD_VP	Administration Vice President
3	102	Lex	AD_VP	Administration Vice President
4	103	Alexander	IT_PROG	Programmer
5	104	Bruce	IT_PROG	Programmer
6	105	David	IT_PROG	Programmer
7	106	Valli	IT_PROG	Programmer
8	107	Diana	IT_PROG	Programmer
9	108	Nancy	FI_MGR	Finance Manager
10	109	Daniel	FI_ACCOUNT	Accountant
11	110	John	FI_ACCOUNT	Accountant

Podemos añadir una clausula **WHERE** para hacer un filtrado de filas obtenidas

*NATURAL JOIN jobs WHERE department\_id IN (20, 50);*

1 - 237

## Agenda

- Tipos de JOINS y su sintaxis
- Natural join
- Join mediante la clausula USING
- Join mediante la clausula ON
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 238

## Creando JOIN con la clausula USING

- `NATURAL JOIN` une todas las columnas con nombres que coinciden con los tipos de datos y para unir las tablas.
- La cláusula `USING` se puede utilizar para especificar únicamente las columnas que se deben utilizar para una combinación de igualdad.

```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
       USING (department_id);
```

1 - 239

## Uniendo Nombre de Columna

EMPLOYEES

	EMPLOYEE_ID	DEPARTMENT_ID
1	200	10
2	201	20
3	202	20
4	205	110
5	206	110
6	100	90
7	101	90
8	102	90
9	103	60
10	104	60

...

Foreign key

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

Primary key

- Debemos elegir las columnas apropiadas para poder utilizar un `equijoin` (igualdad).
- De forma habitual se utiliza una relación PK – FK.

1 - 240

## Obteniendo Registros mediante la clausula USING

```
SELECT employee_id, last_name,  
       location_id, department_id  
FROM   employees JOIN departments  
USING (department_id);
```

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	144	Vargas	1500	50
5	143	Matos	1500	50
6	142	Davies	1500	50
7	141	Rajs	1500	50
8	124	Mourgos	1500	50
...				
18	206	Gietz	1700	110
19	205	Higgins	1700	110

La columna **department\_id** debe de existir en ambas tablas

1 - 241

## Cualificar nombres de columna ambiguas

- Cuando se unen dos o más tablas, a veces, es necesario cualificar los nombres de las columnas iguales para evitar la ambigüedad.
  - La columna `DEPARTMENT_ID` en la `SELECT` puede ser de cualquiera de la tabla departamentos o la tabla empleados.
- Es recomendable añadir el **alias de tabla** para ejecutar la consulta de forma correcta. (**tabla.columna**)
- Si no hay nombres de columna en común entre las dos tablas, no hay necesidad de cualificar las columnas.
  - Sin embargo, utilizando el prefijo de la tabla aumenta la velocidad de análisis sintáctico de la declaración.

1 - 242

## Usando Alias de Table con la clausula USING

- Los alias de tabla se crean añadiendo un literal justo después de la tabla a utilizar
- Restricciones
  - Al unirse con la cláusula USING, no se puede calificar una columna que se utiliza en la cláusula USING en s
  - No califique una columna que se utiliza en un NATURAL
  - Si la misma columna se utiliza en otras partes de la instrucción SQL, no se puede utilizar un alias a ella.

```
SELECT l.city, d.department_name  
FROM   locations l JOIN departments d  
USING (location_id)  
WHERE  d.location_id = 1400;
```

ORA-25154: column part of USING clause cannot have qualifier  
25154. 00000 - "column part of USING clause cannot have qualifier"  
\*Cause: Columns that are used for a named-join (either a NATURAL join  
or a join with a USING clause) cannot have an explicit qualifier.  
\*Action: Remove the qualifier.  
Error at Line: 4 Column: 6

1 - 244

## Agenda

- Tipos de JOINS y su sintaxis
- Natural join
- Join mediante la clausula USING
- Join mediante la clausula ON
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 245

## Creando JOIN con la cláusula ON

- La condición de JOIN para los NATURAL JOIN es básicamente una combinación de igualdad de todas las columnas con el mismo nombre.
- Utilice la cláusula ON se utiliza para especificar las condiciones específicas de las columnas al unirse.
- Estas condiciones de unión pueden ser diferentes a la igualdad.
- La cláusula ON hace que el código fácil de entender.

1 - 246

## Recuperando Registros con la cláusula ON

- Las columnas DEPARTMENT\_ID en EMPLOYEES y en la tabla DEPARTMENTS se unen utilizando la cláusula ON

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	124	Mourgos	50	50	1500
5	144	Vargas	50	50	1500
6	143	Matos	50	50	1500
7	142	Davies	50	50	1500
8	141	Rajs	50	50	1500
9	107	Lorentz	60	60	1400
10	104	Ernst	60	60	1400
11	103	Hunold	60	60	1400

1 - 247

## Creando Tres vias de Union

- Podemos unir 3 Tablas mediante 2 condiciones de unión mediante la ON, según aparece en la transparencia

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100 Seattle	Executive
2	101 Seattle	Executive
3	102 Seattle	Executive
4	103 Southlake	IT
5	104 Southlake	IT
6	107 Southlake	IT
7	124 South San Francisco	Shipping
8	141 South San Francisco	Shipping
9	142 South San Francisco	Shipping

1. Employees - Departments
2. Resultado - Locations

1 - 248

## Aplicando condiciones adicionales a un JOIN

Utilice la cláusula AND o la cláusula WHERE para aplicar condiciones adicionales:

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149;
```

O

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
WHERE  e.manager_id = 149;
```

1 - 249



## Agenda

- Tipos de JOINS y su sintaxis
- Natural join
- Join mediante la clausula USING
- Join mediante la clausula ON
- **Self-join**
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 250

## Uniendo una Tabla consigo misma

A veces es necesario combinar una tabla consigo misma para encontrar datos que están en la misma tabla (nacimiento-sexo)

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...

Los empleado tienen un número de trabajador EMPLOYEE\_ID y también un número de jefe MANAGER\_ID

1 - 251

## Self-Joins usando la clausula ON

- La cláusula ON también se puede utilizar para unir columnas que tienen nombres diferentes, dentro de la misma Tabla o en una tabla diferente.
  - El ejemplo que se muestra es un self-join de las tablas EMPLEADOS, sobre la base de las columnas y EMPLOYEE\_ID MANAGER\_ID.

```
SELECT worker.last_name emp, manager.last_name mgr  
FROM employees worker JOIN employees manager  
ON (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King
8	Kochhar	King

1 - 252

## Agenda

- Tipos de JOINS y su sintaxis
- Natural join
- Join mediante la clausula USING
- Join mediante la clausula ON
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 253

## Nonequijoins

- Un nonequijoin es una condición de unión que contiene algo que no sea un operador de igualdad.

EMPLOYEES

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	Fay	6000
4	Higgins	12000
5	Gietz	8300
6	King	24000
7	Kochhar	17000
8	De Haan	17000
9	Hunold	9000
10	Ernst	6000
...		
19	Taylor	8600
20	Grant	7000

JOB\_GRADES

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

La relación entre la tabla empleados y la tabla JOB\_GRADES es un ejemplo de un nonequijoin

SALARY en la tabla EMPLOYEES oscila entre los valores de las columnas y LOWEST\_SAL HIGHEST\_SAL de la tabla JOB\_GRADES

1 - 254

## Recuperando Registros con Nonequijoins

```
SELECT e.last_name, e.salary, j.grade_level
FROM employees e JOIN job_grades j
ON e.salary
   BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

A la cláusula ON se añade una condición de filtrado BETWEEN

1 - 255

## Agenda

- Tipos de JOINS y su sintaxis
- Natural join
- Join mediante la clausula USING
- Join mediante la clausula ON
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 256

## Recuperando Registros que no coinciden directo usando OUTER Joins

- Si una fila no satisface una condición de unión, la fila no aparece en el resultado de la consulta.
- Un OUTER join devuelve todas las filas que satisfacen la condición de unión y también devuelve parte o la totalidad de las filas de una tabla que son huérfanas
  - Empleados que no tienen departamento
  - Departamentos que no tienen empleados

1 - 257

## Recuperando Registros que no coinciden directo usando OUTER Joins

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

No hay empleados en el departamento 190.

El empleado "Grant" no ha sido asignado a ningún departamento

Equijoin with EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst

...

18	80	Abel
19	80	Taylor

1 - 258

## LEFT OUTER JOIN

Esta consulta recupera todas las filas de la tabla EMPLOYEES que es la tabla de la izquierda, incluso si no hay ninguna coincidencia en la tabla DEPARTMENTS.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping
...			
16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

1 - 259

## RIGHT OUTER JOIN

Esta consulta recupera todas las filas de la tabla `DEPARTMENTS` que es la tabla de la izquierda, incluso si no hay ninguna coincidencia en la tabla `EMPLOYEES`.

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

1 - 260

## FULL OUTER JOIN

Esta consulta recupera todas las filas de la tabla `EMPLOYEES`, incluso si no hay ninguna coincidencia en la tabla `DEPARTMENTS`.

También recupera todas las filas de la tabla `DEPARTMENTS`, incluso si no hay ninguna coincidencia en la tabla `EMPLOYEES`.

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

...

17	Zlotkey	80	Sales
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

1 - 261

## Agenda

- Tipos de JOINS y su sintaxis
- Natural join
- Join mediante la clausula USING
- Join mediante la clausula ON
- Self-join
- Nonequijoins
- OUTER join:
  - LEFT OUTER join
  - RIGHT OUTER join
  - FULL OUTER join
- Producto Cartesiano
  - Cross join

1 - 262

## Productos Cartesianos

Un producto cartesiano se forma cuando:

- Se omite una condición de unión
- Una condición de unión no es válida
- Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla

Se incluye siempre una condición de unión válida si desea evitar un producto cartesiano.

**EMPLOYEES (20 filas)**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	Whalen	10
2	Marstein	20
3	Fay	20
4	Higgins	110
...	...	...
19	Tucker	90
20	DeSilva	90

**DEPARTMENTS (8 filas)**

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	Administration	1700
2	Marketing	1800
3	Sales	1500
4	IT	1400
5	Sales	2500
6	Executive	1700
...	...	...
11	Support	1700
12	Support	1700

**Producto cartesiano:**  
20 x 8 = 160 filas

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	20	1800
2	20	1800
...	...	...
19	90	1700
20	90	1700

1 - 263

## Creando Cross Joins

- A `CROSS JOIN` es una operación `JOIN` que produce el producto cartesiano de dos tablas.
- Para crear un producto cartesiano, especifique el `CROSS JOIN` en la instrucción `SELECT`.

```
SELECT last_name, department_name  
FROM employees  
CROSS JOIN departments ;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
158	Vargas	Contracting
159	Whalen	Contracting
160	Zlotkey	Contracting

Es una buena práctica para establecer explícitamente `CROSS JOIN` en su `SELECT` cuando tiene la intención de crear un producto cartesiano

1 - 264

## Resumen

En esta lección, usted debe haber aprendido a:

- Escribir sentencias `SELECT` para acceder a los datos de más de una tabla utilizando `equijoins` y `nonequijoins`
- Realizar de `JOINS` de una misma tabla mediante el uso de una `self-join`
- Ver los datos obtenidos mediante `OUTER joins`
- Generar un producto cartesiano de todas las filas de dos o más tablas

1 - 266



## Practica 7

Esta práctica se abordan los siguientes temas:

- Unir tablas mediante una combinación de igualdad (equijoin)
- Realización outer-joins y self-joins
- Adición de condiciones

1 - 267

# 8

## Usando Subconsultas para Solventar Consultar



ORACLE®

## Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Definir subConsultas
- Describir los tipos de problemas que las subConsultas pueden resolver
- Enumerar los tipos de subConsultas
- SubConsultas de una sola fila, de varias filas, de varias columnas

1 - 269

## Agenda

- Subconsulta: Tipos, la sintaxis y directrices
- Subconsultas de una sola fila:
  - Funciones de Grupo en en SubConsulta
  - Clausula `HAVING` en las subConsultas
- SubConsultas de múltiples filas
  - Operador `ALL` o `ANY`
- SuConsultas de múltiples columnas
- Valores `NULL` en SubConsultas

1 - 270

## Usando una subconsulta para resolver un problema

¿Quién tiene un salario mayor que el de Abel?

Consulta principal:



¿Qué empleados tienen un salario mayor que el de Abel?

Subconsulta:



¿Cuál es el salario de Abel?

1 - 271

## Sintaxis de las SubConsultas

- Una subconsulta es una instrucción `SELECT` que se incrusta dentro de otra instrucción `SELECT`.
- Se puede incluir una Subconsulta en:
  - `WHERE`, `HAVING`, `FROM`
- La subselect se pone entre paréntesis en lugar del valor requerido

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT      select_list
         FROM         table);
```

1 - 272

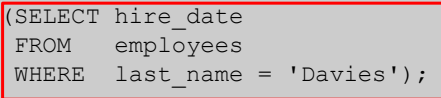
## Sintaxis de las SubConsultas

- Pueden ser muy útil cuando se necesita para seleccionar filas de una tabla con una condición que depende de los datos de la tabla en sí.
- La subconsulta (`inner query`) realiza antes de la consulta principal (`outer query`).
- El resultado de la subconsulta es utilizado por la consulta principal.

1 - 273

## Usando SubConsultas

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date > (SELECT hire_date
                   FROM   employees
                   WHERE  last_name = 'Davies');
```



- En la diapositiva, la consulta interna determina la fecha de contratación del empleado **Davies**.
- La consulta externa toma el resultado de la consulta interna y utiliza este resultado para mostrar todos los empleados que fueron contratados después Davies.

1 - 274

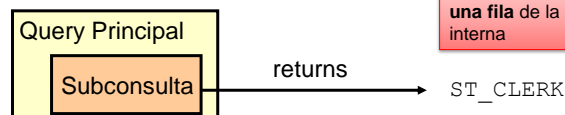
## Normas y directrices para el uso de subconsultas

- Subconsultas encerrar entre paréntesis.
- Las subconsultas deben de ir en el lado derecho de la condición de comparación. (la subconsulta puede aparecer en cualquiera de los lados del operador de comparación).
- Utilice los **operadores de una sola fila** con subconsultas de una sola fila y **operadores de múltiples filas** con subconsultas de varias filas

1 - 275

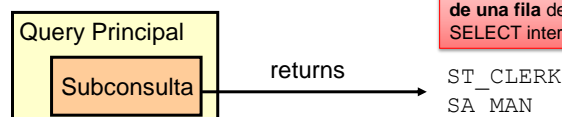
## Tipos de Subconsultas

- Subconsulta de una Fila



Las consultas que devuelven **sólo una fila** de la instrucción SELECT interna

- Subconsulta de múltiples filas



las consultas que devuelven **más de una fila** de la instrucción SELECT interna

1 - 276

## Agenda

- Subconsulta: Tipos, la sintaxis y directrices
- Subconsultas de una sola fila:
  - Funciones de Grupo en en SubConsulta
  - Clausula HAVING en las subConsultas
- SubConsultas de múltiples filas
  - Operador ALL o ANY
- SubConsultas de múltiples columnas
- Valores NULL en SubConsultas

1 - 277

## Subconsultas de una sólo fila

- Devolver una sola fila
- Utilizar operadores de comparación habituales

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  employee_id = 141);
```

1 - 278

## Usando Subconsultas de una sola fila

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Taylor')
AND    salary > (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Taylor');
```

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000

Empleados que realizan el mismo trabajo que "Taylor", pero ganan más salario que él.

- 2 Subconsultas
- 1 Consulta

1 - 279

## Uso de las funciones de grupo en una subconsulta

- Se pueden obtener datos en una consulta principal mediante el uso de una **función de grupo** en una subconsulta para devolver una sola fila.
- La subconsulta está entre paréntesis y se coloca después de la condición de comparación.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary = (SELECT MIN(salary)
                 FROM   employees);
```

	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

1 - 280

## Clausula HAVING con Subconsultas

- El servidor de Oracle ejecuta las subconsultas en primer lugar.
- Oracle devuelve el resultado y hace la comparación con la cláusula HAVING de la consulta principal.

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) > (SELECT MIN(salary)
FROM employees
WHERE department_id = 30);
```

	DEPARTMENT_ID	MIN(SALARY)
1	100	6900
2	(null)	7000
3	90	17000
4	20	6000
5	70	10000
6	110	8300
7	80	6100
8	40	6500
9	60	4200
10	10	4400

1 - 281

## ¿Qué está mal en esta orden?

```
SELECT employee_id, last_name
FROM employees
WHERE salary = (SELECT MIN(salary)
FROM employees
GROUP BY department_id);
```

ORA-01427: single-row subquery returns more than one row  
01427. 00000 - "single-row subquery returns more than one row"  
\*Cause:  
\*Action:

Operador de una sola fila utilizada con multiples filas en la subconsulta

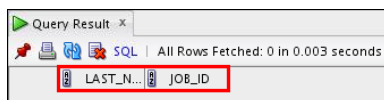
1 - 282



## No hay filas devueltas por la Subconsulta

- Otro problema común con subconsultas se produce cuando no hay filas a devolver en la subconsulta.
- La consulta externa toma los resultados de la subconsulta como NULL

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  last_name = 'Haas');
```



The screenshot shows the 'Query Result' window in SQL Developer. The title bar says 'Query Result x'. Below the title bar, there are icons for file operations and a status bar that reads 'All Rows Fetched: 0 in 0.003 seconds'. The main area of the window shows a table with two columns: 'LAST\_NAME' and 'JOB\_ID'. The table is empty, indicating that no rows were returned by the query.

Subconsulta no devuelve ninguna fila porque no hay ningún empleado llamado "Haas".

1 - 283

## Agenda

- Subconsulta: Tipos, la sintaxis y directrices
- Subconsultas de una sola fila:
  - Funciones de Grupo en en SubConsulta
  - Clausula HAVING en las subConsultas
- SubConsultas de múltiples filas
  - Operador ALL o ANY
- SubConsultas de múltiples columnas
- Valores NULL en SubConsultas

1 - 284

## Subconsultas de Múltiples Filas

- Devuelve más de una fila
- Utilizar operadores de comparación de múltiples filas

Operador	Significado
IN	Igual a cualquier miembro de la lista
ANY	Debe ir precedida de =, !=, >, <, <=, > =. Devuelve TRUE si existe al menos un elemento del conjunto de resultados de la subconsulta para el que la relación es TRUE.
ALL	Debe ir precedida de =, !=, >, <, <=, > =. Devuelve TRUE si la relación es cierta para todos los elementos en el conjunto de resultados de la subconsulta.

1 - 285

## Usando el operador ANY en Subconsultas de Múltiples Filas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

El operador ANY compara un valor con cada valor devuelto por una subconsulta

< ANY → menor que cualquiera  
 > ANY → mayor que cualquiera  
 = ANY → es equivalente a IN.

1 - 286

## Usando el operador ALL en Subconsultas de de Multiples Filas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees          9000, 6000, 4200
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

El operador ALL compara un valor con cada valor devuelto por una subconsulta

< ALL → menor que todos

> ALL → mayor que todos

1 - 287

## Agenda

- Subconsulta: Tipos, la sintaxis y directrices
- Subconsultas de una sola fila:
  - Funciones de Grupo en en SubConsulta
  - Clausula HAVING en las subConsultas
- SubConsultas de múltiples filas
  - Operador ALL o ANY
- SubConsultas de múltiples columnas
- Valores NULL en SubConsultas

1 - 288

## Subconsultas de múltiples columnas

- Una subconsulta de múltiples columnas devuelve más de una columna a la consulta externa.
- Comparaciones de columna en múltiples comparaciones por pares, trios, etc de columnas.
  - Debemos de comparar tantas columnas como columnas devuelva la subconsulta
- La subconsulta debe de ser referenciada con las columnas a comparar entre paréntesis.
- Una subconsulta de múltiples columnas puede aparece en la consulta externa en el FROM, WHERE o HAVING

1 - 289

## Subconsultas de múltiples columnas: Ejemplo

Mostrar todos los empleados con el salario más bajo en cada departamento

```
SELECT first_name, department_id, salary
FROM employees
WHERE (salary, department_id) IN
      (SELECT min(salary), department_id
       FROM employees
       GROUP BY department_id)
ORDER BY department_id;
```

	FIRST_NAME	DEPARTMENT_ID	SALARY
1	Jennifer	10	4400
2	Pat	20	6000
3	Peter	50	2500
4	Diana	60	4200
5	Jonathon	80	8600
6	Neena	90	17000
7	Lex	90	17000
8	William	110	8300

Como la subconsulta devuelve varias columnas, las columnas de la comparación entre paréntesis

```
WHERE (column, column, ...) IN
      (SELECT column, column, ...
```

1 - 290

## Agenda

- Subconsulta: Tipos, la sintaxis y directrices
- Subconsultas de una sola fila:
  - Funciones de Grupo en en SubConsulta
  - Clausula HAVING en las subConsultas
- SubConsultas de múltiples filas
  - Operador ALL o ANY
- SubConsultas de múltiples columnas
- Valores NULL en SubConsultas

1 - 291

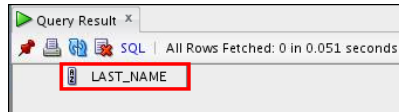
## Valores NULL en una subconsulta

- Hay determinados tipos de subconsultas, que cuando la subconsulta devuelve algún valor nulo, Oracle utiliza el valor `NULL` para la respuesta.
- Todas las condiciones que comparan resultado valor nulo en un valor nulo.
- En subconsultas con `NOT IN` es equivalente a `<> ALL` y como uno de ellos es `NULL`  $\rightarrow$  `NULL`
  - El valor `NULL` como parte de los resultados de una subconsulta **no es un problema si se utiliza el operador `IN`** pues equivale a **`= ANY`**.

1 - 292

## Valores NULL en una subconsulta

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```



LAST_NAME
-----------

Subconsulta no devuelve ninguna fila porque uno de los valores devueltos por una sub consulta es nula.

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

1 - 293

## Resumen

En esta lección, usted debe haber aprendido a:

- Definir subConsultas
- Describir los tipos de problemas que las subConsultas pueden resolver
- Enumerar los tipos de subConsultas
- SubConsultas de una sola fila, de varias filas, de varias columnas

1 - 296

## Practica 8

Esta práctica se abordan los siguientes temas:

- Creación de subconsultas para consultar los valores sobre la base de criterios desconocidos
- El uso de subconsultas para averiguar los valores que existen en un conjunto de datos y no en otro

1 - 297

# 9

## Usando operadores SET



ORACLE®

## Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Describir los operadores SET
- Usar los operadores SET para combinar múltiples consultas en una sola query
- Controlar el orden de devolución de las filas devueltas

1 - 299

## Agenda

- Operadores SET: Tipos y Directivas
  - Tablas usadas en esta lección
  - Operadores UNION y UNION ALL
  - Operador INTERSECT
  - Operador MINUS
  - Coincidencias en sentencias SELECT
  - Usando la cláusula ORDER BY en operadores SET

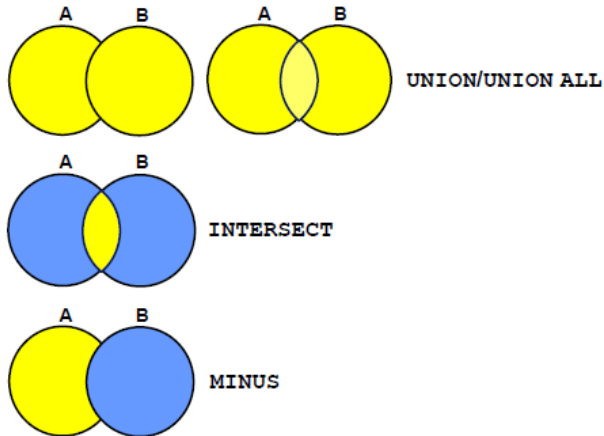
1 - 300



## Operadores SET

Existen tres tipos de relaciones

- UNION
- INTERSECT
- MINUS



1 - 301

## Operadores SET

### UNION

- Combinación de todas las filas del primer conjunto con todas las filas del segundo. Cuando se duplique una fila por existencia de la misma en los dos conjuntos solo aparecerá una

### INTERSECT

- Combinación de las filas de los dos conjuntos, cuando la fila exista en ambos conjuntos

### MINUS

- Combinación de las filas del primer conjunto que no estén en el segundo

1 - 302

## Instrucciones de los Operadores de Definición

- Las expresiones de las listas `SELECT` debe coincidir en el número de columnas seleccionadas.
- Los tipos de dato para cada columna de la segunda consulta deben coincidir con los tipos de dato de su columna correspondiente en la primera consulta.
- Los paréntesis se pueden utilizar para modificar la secuencia de ejecución.  

```
Select ..... union select ..... intersect....
```

Los conjuntos son evaluados de izquierda a derecha
- La sentencia `ORDER BY` puede aparecer sólo una vez al final de la sentencia.

1 - 303

## Servidor de Oracle y Operadores de Definición

- Las filas duplicadas se eliminan automáticamente excepto en `UNION ALL`.
- Los nombres de columna de la primera consulta aparecen en el resultado.
- Por defecto, la salida se ordena en orden ascendente, excepto en `UNION ALL`.

1 - 304

## Agenda

- Operadores SET: Tipos y Directivas
- **Tablas usadas en esta lección**
- Operadores UNION y UNION ALL
- Operador INTERSECT
- Operador MINUS
- Coincidencias en sentencias SELECT
- Usando la cláusula ORDER BY en operadores SET

1 - 305

## Tablas usando en esta Lección

Las tablas utilizadas en esta lección son:

- **EMPLOYEES:** Proporciona detalles con respecto a todos los empleados actuales
- **RETIRED\_EMPLOYEES:** Proporcionar detalles con respecto a todos los empleados anteriores

1 - 306

## Agenda

- Operadores SET: Tipos y Directivas
- Tablas usadas en esta lección
- **Operadores UNION y UNION ALL**
- Operador INTERSECT
- Operador MINUS
- Coincidencias en sentencias SELECT
- Usando la cláusula ORDER BY en operadores SET

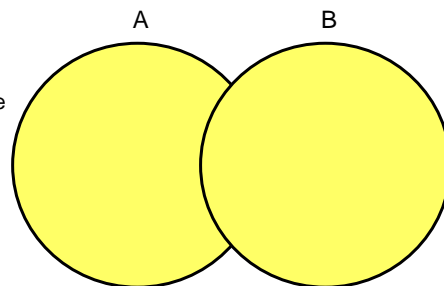
1 - 307

## UNION Operator

El operador UNION devuelve todas las filas seleccionadas en cualquier consulta.

Utilice el operador UNION para devolver todas las filas de varias tablas y eliminar las filas duplicadas.

El operador UNION devuelve las filas de ambas consultas después de la eliminación de duplicados.



1 - 308

## Usando el Operador UNION

Mostrar los detalles actuales y anteriores del puesto de todos los empleados. Mostrar cada empleado sólo una vez.

```
SELECT job_id
FROM employees
UNION
SELECT job_id
FROM retired_employees
```

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	FI_ACCOUNT
7	FI_MGR
8	IT_PROG
9	MK_MAN
10	MK_REP
11	PU_CLERK
12	PU_MAN
13	SA_MAN
14	SA_REP
15	ST_CLERK
16	ST_MAN

### NOTAS

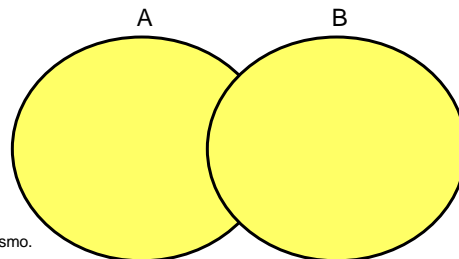
- El número de columnas seleccionadas debe ser el mismo.
- Los tipos de dato de las columnas seleccionadas deben pertenecer al mismo grupo de tipo de dato (por ejemplo, numérico o de caracteres).
- El operador UNION funciona en todas las columnas seleccionadas.
- Los valores NULL no se ignoran durante la comprobación de duplicados.
- Por defecto

1 - 309

## Operador UNION ALL

Utilice el operador `UNION ALL` para devolver todas las filas de varias consultas

El operador `UNION ALL` devuelve las filas de ambas consultas, incluyendo todas las duplicaciones.



### NOTAS

- El número de columnas seleccionadas debe ser el mismo.
- Los tipos de dato de las columnas seleccionadas deben pertenecer al mismo grupo de tipo de dato (por ejemplo, numérico o de caracteres).
- El operador UNION funciona en todas las columnas seleccionadas.
- Los valores NULL no se ignoran durante la comprobación de duplicados.
- Devuelve los registros duplicados.

1 - 310

## Usando el Operador UNION ALL

Mostrar los departamentos actuales y anteriores de todos los empleados.

```
SELECT job_id, department_id
FROM   employees
UNION ALL
SELECT job_id, department_id
FROM   retired_employees
ORDER BY job_id;
```

JOB_ID	DEPARTMENT_ID
1 AC_ACCOUNT	110
2 AC_MGR	110
3 AD_ASST	10
4 AD_PRES	90
5 AD_PRES	90
6 AD_VP	90
7 AD_VP	80
8 AD_VP	90
9 AD_VP	90

...

28 SA_REP	80
29 SA_REP	80
30 SA_REP	(null)
31 ST_CLERK	50
32 ST_CLERK	50
33 ST_CLERK	50
34 ST_CLERK	50
35 ST_MAN	50

1 - 311

## Agenda

- Operadores SET: Tipos y Directivas
- Tablas usadas en esta lección
- Operadores UNION y UNION ALL
- **Operador INTERSECT**
- Operador MINUS
- Coincidencias en sentencias SELECT
- Usando la clausula ORDER BY en operadores SET

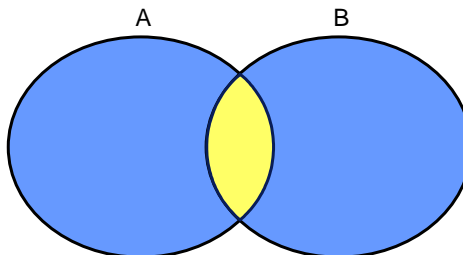
1 - 312

## Operador INTERSECT

El operador `INTERSECT` devuelve filas comunes a ambas consultas.

.

El operador `INTERSECT` devuelve filas que son comunes a ambas consultas.



### NOTAS

- El número de columnas y los tipos de dato de las columnas seleccionadas por las sentencias `SELECT` en las consultas deben ser idénticos en todas las sentencias `SELECT` utilizadas en la consulta.
- No es necesario, sin embargo, que los nombres de las columnas sean idénticos.
- Si se invierte el orden de las tablas intersectadas no se alterará el resultado.
- `INTERSECT` no ignora los valores `NULL`.

1 - 313

## Usando el Operador INTERSECT

Mostrar los ID de empleado y de cargo de los empleados que actualmente tienen el mismo puesto que anteriormente (es decir, han cambiado de cargo pero ahora han vuelto a realizar el mismo trabajo que realizaban anteriormente).

```
SELECT manager_id, department_id
FROM employees
INTERSECT
SELECT manager_id, department_id
FROM retired_employees
```

	MANAGER_ID	DEPARTMENT_ID
1	149	80

1 - 314

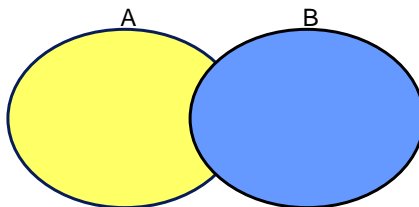
## Agenda

- Operadores SET: Tipos y Directivas
- Tablas usadas en esta lección
- Operadores `UNION` y `UNION ALL`
- Operador `INTERSECT`
- **Operador `MINUS`**
- Coincidencias en sentencias `SELECT`
- Usando la cláusula `ORDER BY` en operadores SET

1 - 315

## Operador `MINUS`

El operador `MINUS` devuelve todas las filas distintas seleccionadas por la primera consulta, pero que no están presentes en el juego de resultados de la segunda consulta.



### NOTAS

- El número de columnas y los tipos de dato de las columnas seleccionadas por las sentencias `SELECT` de las consultas deben pertenecer al mismo grupo de tipo de dato en todas las sentencias `SELECT` utilizadas en la consulta.
- No es necesario, sin embargo, que los nombres de las columnas sean idénticos.
- `MINUS` no ignora los valores `NULL`.

1 - 316



## Usando el Operador MINUS

Mostrar los identificadores de empleado cuyos empleados no han cambiado sus puestos ni una vez.

```
SELECT employee_id, job_id
FROM employees
WHERE department_id = 80
MINUS
SELECT employee_id, job_id
FROM retired_employees
WHERE department_id = 90;
```

	EMPLOYEE_ID	JOB_ID
1	149	SA_MAN
2	174	SA_REP
3	176	SA_REP

1 - 317

## Agenda

- Operadores SET: Tipos y Directivas
- Tablas usadas en esta lección
- Operadores UNION y UNION ALL
- Operador INTERSECT
- Operador MINUS
- Coincidencias en sentencias SELECT
- Usando la cláusula ORDER BY en operadores SET

1 - 318

## Coincidencia de las Sentencias SELECT

- Debido a que las expresiones de las listas SELECT de las consultas deben coincidir en número, puede utilizar columnas ficticias y funciones de conversión de tipos de dato para cumplir con esta regla.
- Podemos utilizar funciones como TO\_CHAR ( o cualquier otra función de conversión) cuando las columnas no existan en una tabla o en la otra.

```
SELECT location_id, department_name "Department",  
       TO_CHAR(NULL) "Warehouse location"  
FROM departments  
UNION  
SELECT location_id, TO_CHAR(NULL) "Department",  
       state_province  
FROM locations;
```

1 - 319

## Coincidencia de las Sentencias SELECT: Ejemplo

Utilizar el operador UNION, mostrar el ID de empleado, ID de cargo y salario de todos los empleados.

```
SELECT FIRST_NAME, JOB_ID, TO_DATE(hire_date) "HIRE_DATE"  
FROM employees  
UNION  
SELECT FIRST_NAME, JOB_ID, TO_DATE(NULL) "HIRE_DATE"  
FROM retired_employees;
```

	FIRST_NAME	JOB_ID	HIRE_DATE
1	Alex	PU_CLERK	(null)
2	Alexander	IT_PROG	03-JAN-06
3	Alexandra	IT_PROG	(null)
4	Bruce	IT_PROG	21-MAY-07
5	Bruce	IT_PROG	(null)
6	Curtis	ST_CLERK	29-JAN-05
7	Dany	FI_ACCOUNT	(null)
8	Dei	PU_MAN	(null)
9	Diana	IT_PROG	07-FEB-07

...

1 - 320

## Agenda

- Operadores SET: Tipos y Directivas
- Tablas usadas en esta lección
- Operadores UNION y UNION ALL
- Operador INTERSECT
- Operador MINUS
- Coincidencias en sentencias SELECT
- Usando la cláusula ORDER BY en operadores SET

1 - 321

## Usando la cláusula ORDER BY en operadores SET

- La cláusula ORDER BY sólo puede aparecer una vez al final de la consulta compuesta.
- Las consultas de componente no pueden tener cláusulas ORDER BY individuales.
- La cláusula ORDER BY reconoce sólo las columnas de la primera consulta SELECT.
- Por defecto, la primera columna de la primera consulta SELECT se utiliza para ordenar la salida en orden ascendente

1 - 322

## Resumen

En esta lección, usted debe haber aprendido a:

- uso `UNION` de para devolver todas las filas distintas
- uso `UNION ALL` de para devolver todas las filas con duplicados
- uso de `INTERSECT` para devolver las filas que estén en ambos conjuntos
- uso de `MINUS` para devolver las filas que sólo están en el conjunto inicial y no en el segundo conjunto
- uso de `ORDER BY` sólo par ordenar la salida final de resultados

1 - 324

## Practica 9

Esta práctica se abordan los siguientes temas:

- Operador `UNION`
- Operador `INTERSECT`
- Operador `MINUS`

1 - 325

# 10

## Gestión de tablas usando DML



ORACLE®

### Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Describir los comandos de Manipulación de Datos (DML)
- Control de Transacciones

## Agenda

- Añadir nuevas filas en una tabla
  - Comando `INSERT`
- Cambiando datos en una tabla
- Eliminando datos de una tabla:
  - Comando `DELETE`
  - Comando `TRUNCATE`
- Trabajando con transacciones mediante `COMMIT`, `ROLLBACK`, y `SAVEPOINT`
- Consistencia de Lectura
- Uso de `FOR UPDATE` en la sentencia `SELECT`

1 - 328

## Lenguaje de Manipulación de Datos

- DML es una parte fundamental de SQL, cuando se desea añadir, actualizar o borrar los datos, se ejecuta una instrucción DML
- Una colección de instrucciones DML que forman una unidad lógica de trabajo se llama una `transacción`
  - Ejemplo:
    - Cuando un banco cliente transfiere dinero de una cuenta de ahorros en una cuenta corriente, la transacción podría consistir en tres operaciones separadas:
      - La disminución de la cuenta de ahorros
      - El aumento de la cuenta corriente, y
      - El registro de la transacción en el diario de transacciones
    - El servidor de Oracle debe garantizar que todas las tres sentencias

1 - 329

## Añadiendo una nueva fila a la Tabla

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Inserte una nueva fila en la tabla DEPARTAMENTOS.

Para insertar filas en una tabla se utiliza el comando **INSERT**

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	70	Public Relations	100	1700
2	10	Administration	200	1700
3	20	Marketing	201	1800
4	50	Shipping	124	1500
5	60	IT	103	1400
6	80	Sales	149	2500
7	90	Executive	100	1700
8	110	Accounting	205	1700
9	190	Contracting	(null)	1700

1 - 330

## Sintaxis del comando INSERT

- Utilizaremos el comando **INSERT** para añadir filas a una tabla, con la siguiente sintaxis:

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

- Con esta sintaxis, se introduce sólo 1 fila en la tabla
- Las columnas se identifican por su nombre.
- La asociación de columna y su valor es posicional.
- Los valores deben cumplir con el tipo de datos de la columna
- Los valores constantes de tipo carácter o fecha deben ir encerrados entre comillas simples ( ' ' )

Las columnas de la tabla no aparecerán si indicamos todos los valores para todas las columnas

1 - 331

## Insertando nuevas Filas: Ejemplo

- Inserción de una fila indicando las columnas de la tabla

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

- Inserción de una fila sin indicar las columnas de la tabla

```
INSERT INTO departments  
VALUES (80, 'Public Relations2', 101, 1700);  
1 rows inserted
```

1 - 332

## Insertando Filas con Valores NULL

- Método implícito: omitir la columna de la lista de columnas.

```
INSERT INTO departments (department_id,  
                        department_name)  
VALUES (30, 'Purchasing');  
1 rows inserted
```

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

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);  
1 rows inserted
```

Asegúrese de que puede utilizar valores **nulos** en la columna indicadas, sino daría erros de inserción. Utilizar el comando DESCRIBE.

1 - 333



## Insertando valores Especiales

- Puede utilizar funciones especiales para introducir los valores de la tabla.
- La función SYSDATE registra la fecha y la hora actuales.
  - Utiliza la función CURRENT\_DATE para obtener la fecha actual en la zona de tiempo de la sesión

```
INSERT INTO employees (employee_id,  
                        first_name, last_name,  
                        email, phone_number,  
                        hire date, job_id, salary,  
                        commission_pct, manager_id,  
                        department_id)  
VALUES  
    (113,  
     'Louis', 'Popp',  
     'LPOPP', '515.124.4567',  
     CURRENT_DATE, 'AC_ACCOUNT', 6900,  
     NULL, 205, 110);
```

1 rows inserted

1 - 334

## Insertando valores de Fecha y Hora

- El formato DD-MON-RR se utiliza generalmente para insertar un valor de fecha
  - Con el formato de RR, el sistema proporciona el siglo correcta de forma automática
  - O también el formato de fecha DD-MM-YYYY
  - O utilizando la función TO\_DATE
- Añadiendo un nuevo empleado:

```
INSERT INTO employees  
VALUES  
    (114,  
     'Den', 'Raphealy',  
     'DRAPHEAL', '515.127.4561',  
     TO_DATE('FEB 3, 2003', 'MON DD, YYYY'),  
     'SA_REP', 11000, 0.2, 100, 60);
```

1 rows inserted

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-03	SA_REP	11000	0.2	100

1 - 335

## Creando un Script

- Podemos crear ficheros de texto con variables de intercambio (&) para posteriormente ejecutarlo y que pida los valores a insertar

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

Enter Substitution Variable

DEPARTMENT\_ID:

40

OK Cancel

Enter Substitution Variable

DEPARTMENT\_NAME:

Human Resources

OK Cancel

Enter Substitution Variable

LOCATION:

2500

OK Cancel

1 - 336

## Insertando filas desde otra tabla

- Puede utilizar la instrucción `INSERT` para añadir filas a una tabla en la que los valores se derivan de las tablas existentes
- El comando `INSERT` utilizará una subconsulta:

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

5 rows inserted.

- NO se debe de utilizar la cláusula `VALUES`.
- La subconsulta debe de devolver el mismo número de columnas y tipo que las columnas a insertar.
- Inserta todas las filas devueltas por la subconsulta en la tabla.

1 - 337

## Agenda

- Añadir nuevas filas en una tabla
  - Comando `INSERT`
- Cambiando datos en una tabla
- Eliminando datos de una tabla:
  - Comando `DELETE`
  - Comando `TRUNCATE`
- Trabajando con transacciones mediante `COMMIT`, `ROLLBACK`, y `SAVEPOINT`
- Consistencia de Lectura
- Uso de `FOR UPDATE` en la sentencia `SELECT`

1 - 338

## Cambiando datos en una Tabla

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Filas Actualizadas en la tabla EMPLOYEES

Cambio de departamento para algunos empleados

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

1 - 339

## Sintaxis del comando UPDATE

- La modificación de los datos ya insertados en una tabla se realiza con el comando UPDATE:

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

<i>Table</i>	<b>Nombre de la tabla a actualizar datos</b>
<i>Column=value</i>	<b>columna y valor a modificar</b>
<i>WHERE condición</i>	<b>filtro de filas donde aplicar la actualización</b>

- Este comando puede actualizar mas de una fila si se cumple la condición o no existe la condición `WHERE`

**Nota:** En general, utilice la columna de clave principal en la cláusula `WHERE` para identificar una sola fila para la actualización

1 - 340

## Actualizando filas en una tabla

- Mediante la clausula `WHERE` indicamos la fila o filas que se van a modificar:

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
```

1 rows updated

- Si queremos realizar una modificación en todas las filas, omitiremos la clausula `WHERE` :

```
UPDATE copy_emp
SET    department_id = 110;
```

22 rows updated

- Especificar `SET column_name= NULL` para actualizar una columna a `NULL`.

1 - 341

## Actualización de dos columnas con una subconsulta

Actualizar el trabajo y el salario del empleado 103 a los mismos datos que el empleado 205

```
UPDATE employees
SET (job_id,salary) = (SELECT job_id,salary
                       FROM employees
                       WHERE employee_id = 205)
WHERE employee_id = 103;
```

1 rows updated

```
UPDATE employees
SET column = ( . . . ) ,
    column = ( . . . )
WHERE employee_id = 103;
```

1 - 342

## Actualización de filas en función de otra tabla

Utilice las subconsultas en las sentencias UPDATE para actualizar valores de fila en una tabla basada en los valores de otra tabla:

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

1 rows updated

1 - 343

## Agenda

- Añadir nuevas filas en una tabla
  - Comando INSERT
- Cambiando datos en una tabla
- Eliminando datos de una tabla:
  - Comando DELETE
  - Comando TRUNCATE
- Trabajando con transacciones mediante COMMIT, ROLLBACK, y SAVEPOINT
- Consistencia de Lectura
- Uso de FOR UPDATE en la sentencia SELECT

1 - 344

## Eliminando filas de una tabla

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Eliminando una fila de la tabla DEPARTMENTS:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

1 - 345

## Comando DELETE

Puede eliminar filas existentes de una tabla con la sentencia `DELETE` :

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

*Table*                      **Nombre de la tabla a borrar datos**  
*WHERE condición*   **filtro de filas donde aplicar el borrado**

**Nota:** Si no se elimina ningún filas, se devuelve el mensaje "0 filas eliminadas"

1 - 346

## Borrando filas de una Tabla

- Mediante la clausula `WHERE` indicamos la fila o filas que se van a borrar:

```
DELETE FROM departments
WHERE department_name = 'Finance';
```

1 rows deleted

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

- Todas las filas de la tabla son borradas si omitimos la clausula `WHERE` :

```
DELETE FROM copy_emp;
```

22 rows deleted

1 - 347

## Borrando filas en función de otra tabla

Utilice las subconsultas en las instrucciones `DELETE` para eliminar filas de una tabla en función de los valores de otra tabla:

```
DELETE FROM employees
WHERE department_id IN
  (SELECT department_id
   FROM departments
   WHERE department_name
     LIKE '%Public%');
1 rows deleted
```

1 - 348

## Comando TRUNCATE

- Un método más eficiente de vaciar una tabla es mediante el uso de la instrucción `TRUNCATE`.
- La instrucción `TRUNCATE` elimina rápidamente todas las filas de una tabla o clúster y deja intacta su estructura
  - Instrucción DDL y no genera `Rollbacks`
  - Elimina todos los bloques asociados a la tabla
  - No permite deshacer la operación

- Sintaxis:

```
TRUNCATE TABLE table_name;
```

- Ejemplo:

```
TRUNCATE TABLE copy_emp;
```

1 - 349



## Agenda

- Añadir nuevas filas en una tabla
  - Comando `INSERT`
- Cambiando datos en una tabla
- Eliminando datos de una tabla:
  - Comando `DELETE`
  - Comando `TRUNCATE`
- Trabajando con transacciones mediante `COMMIT`, `ROLLBACK`, y `SAVEPOINT`
- Consistencia de Lectura
- Uso de `FOR UPDATE` en la sentencia `SELECT`

1 - 350

## Transacciones en la Base de Datos

- El servidor de Oracle garantiza la coherencia de datos basado en transacciones
- Las transacciones:
  - Dan mayor flexibilidad y control en el cambio de datos
  - Permiten asegurar la consistencia de datos en caso de fallo de proceso de usuario o fallo del sistema.
- Una transacción de base de datos consta de una de las siguientes:
  - Conjunto de 1 a N sentencias DML
  - 1 sola sentencia DDL
  - 1 sola sentencia DCL

1 - 351

## Transacciones en la Base de Datos: Inicio y Fin

- Una transacción comienza cuando:
  - Se ejecuta la primera instrucción DML.
  - Finaliza la transacción anterior
- Una transacción finaliza con los siguientes eventos:
  - Comando `COMMIT` o `ROLLBACK`
  - Operaciones DDL or DCL (Commit automático).
  - El usuario sale de `SQL Developer` o `SQL*Plus`.
  - El sistema se bloquea.

1 - 352

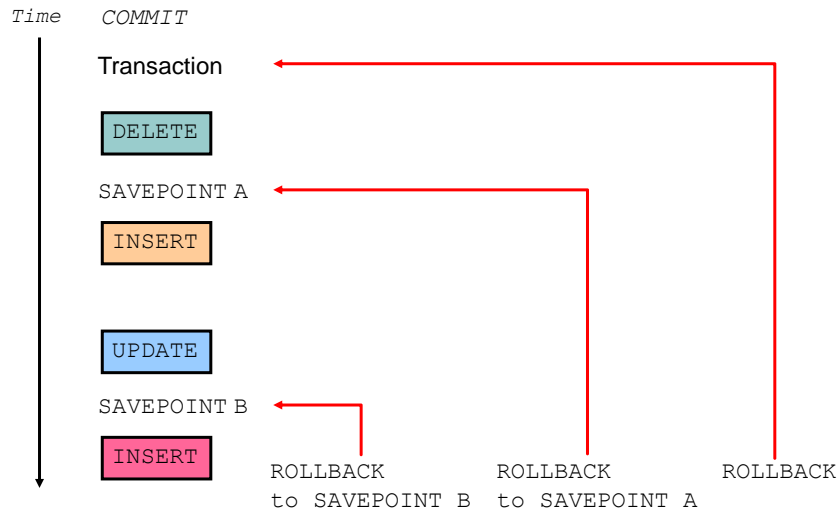
## Ventajas de `COMMIT` y `ROLLBACK`

Mediante los comandos `COMMIT` y `ROLLBACK` podemos:

- Garantizar la coherencia de datos
- Realizar cambios en los datos y posteriormente validarlos o deshacerlos
- Agrupar operaciones a realizar en la base de datos de forma lógica

1 - 353

## Control de transacción explícita



1 - 354

## Revertir los cambios a un marcador

- Se puede crear un marcador en la transacción actual mediante la instrucción `SAVEPOINT`, con el objetivo de dividir la transacción en secciones más pequeñas
- Posteriormente podemos deshacer la transacción completa (`ROLLBACK`) o deshacerla hasta el marcado anterior (`ROLLBACK TO SAVEPOINT`)

```
UPDATE...  
SAVEPOINT update_done;  
SAVEPOINT update_done succeeded.  
INSERT...  
ROLLBACK TO update_done;  
ROLLBACK TO succeeded.
```

1 - 355

## Procesamiento de Transacciones implícitas

- Una confirmación automática se produce en las siguientes circunstancias:
  - Sentencia DDL
  - Sentencia DCL
  - Salida normal de SQL Developer o SQL\*Plus, sin emitir explícitamente COMMIT o ROLLBACK
- Un rollback automático se produce cuando hay una terminación anormal de SQL Developer o SQL\*Plus o un fallo del sistema.

**Nota:** En SQL \* Plus, el comando AUTOCOMMIT se puede activar o desactivar  
`Set autocommit on/off` ///  
Preferences / Database / Worksheet Parameters/ Autocommit in SQL Worksheet

1 - 356

## Estado de datos Antes de COMMIT o ROLLBACK

- Cuando se hace una operación DML, los datos ´son modificados **sólo en la sesión** donde se lanza la operación DML.
- El estado anterior de los datos se puede recuperar mediante la operación ROLLBACK.
- La sesión actual puede revisar los resultados de las operaciones DML mediante el uso de la instrucción SELECT.
- Otras sesiones no pueden ver los resultados de las sentencias DML emitidos por la sesión actual, hasta ser validados.
- Las filas afectadas están bloqueados; otra sesión no puede cambiar los datos de las filas afectadas.

1 - 357

## Estado de datos después del COMMIT

- Los cambios en los datos se guardan en la base de datos.
- El estado anterior de los datos se sobrescribe.
- Todas las sesiones pueden ver los resultados.
- Los bloqueos de las filas afectadas son liberados; esas filas están disponibles para otras sesiones de manipular.
- Todos los `savepoints` se borran.

1 - 358

## Validando datos

- Hacer los cambios:

```
DELETE FROM EMPLOYEES
WHERE employee_id=113;
1 rows deleted
INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- Validando los cambios:

```
COMMIT;
Committed.
```

1 - 359

## Estado de datos después de ROLLBACK

Los datos pendientes son deshechos Descartar mediante la instrucción ROLLBACK :

- Cambios en los datos se deshacen.
- Estado anterior de los datos se restaura.
- Los bloqueos de las filas afectadas son liberados.

```
DELETE FROM copy_emp;  
ROLLBACK ;
```

1 - 360

## Estado de datos después de ROLLBACK: Ejemplo

```
DELETE FROM test;  
4 rows deleted.  
  
ROLLBACK;  
Rollback complete.  
  
DELETE FROM test WHERE id = 100;  
1 row deleted.  
  
SELECT * FROM test WHERE id = 100;  
No rows selected.  
  
COMMIT;  
Commit complete.
```

1 - 361

## Rollback Statement-Level Rollback

- Si una sola instrucción `DML` falla durante la ejecución, solamente esta declaración se deshace.
- Oracle implementa de forma implícita un `savepoint`
  - Todos los demás cambios se conservan
- El usuario debe poner fin a las transacciones de forma explícita mediante la ejecución de una sentencia `COMMIT` o `ROLLBACK`.
- El servidor de Oracle emite un `COMMIT` implícito antes y después de cualquier sentencia `DDL`
  - Incluso si su sentencia `DDL` no se ejecuta satisfactoriamente, no se puede deshacer la declaración anterior porque el servidor emite una confirmación.

1 - 362

## Agenda

- Añadir nuevas filas en una tabla
  - Comando `INSERT`
- Cambiando datos en una tabla
- Eliminando datos de una tabla:
  - Comando `DELETE`
  - Comando `TRUNCATE`
- Trabajando con transacciones mediante `COMMIT`, `ROLLBACK`, y `SAVEPOINT`
- **Consistencia de Lectura**
- Uso de `FOR UPDATE` en la sentencia `SELECT`

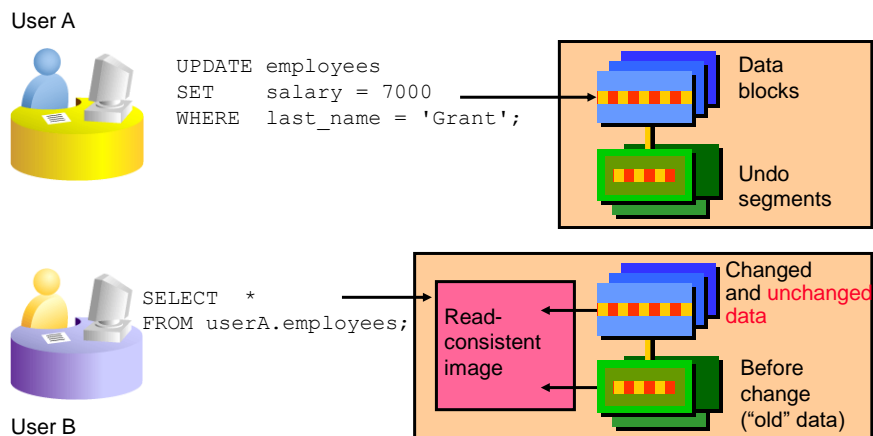
1 - 363

## Consistencia de Lectura

- Leer consistencia garantiza una visión consistente de los datos en todo momento.
- Permite que:
  - Los lectores no vean los datos que se encuentra en proceso de ser cambiado.
  - Escritores se aseguran de que los cambios en la base de datos se realizan de una manera coherente.
- La Consistencia de Lectura permite, en los mismos datos:
  - Los lectores no esperan a que los escritores
  - Escritores no esperan a los lectores
  - Escritores esperan para los escritores

1 - 364

## Implementando la consistencia de lectura



1 - 365



## Agenda

- Añadir nuevas filas en una tabla
  - Comando `INSERT`
- Cambiando datos en una tabla
- Eliminando datos de una tabla:
  - Comando `DELETE`
  - Comando `TRUNCATE`
- Trabajando con transacciones mediante `COMMIT`, `ROLLBACK`, y `SAVEPOINT`
- Consistencia de Lectura
- **Uso de `FOR UPDATE` en la sentencia `SELECT`**

1 - 366

## Uso de `FOR UPDATE` en la sentencia `SELECT`

- Cuando se emite una sentencia `SELECT` en la base de datos para consultar algunos registros, no se coloca ningún bloqueo en las filas seleccionadas.
- Sólo se bloquean aquellos registros que se han cambiado, pero que aún no se han confirmado.
- Aún así, otros usuarios podrán leer dichos registros tal y como aparecían antes del cambio (la “imagen anterior” de los datos).
- Hay ocasiones, sin embargo, en las que puede que desee bloquear un juego de registros incluso antes de cambiarlos en el programa.

1 - 367

## Uso de FOR UPDATE en la sentencia SELECT

- Oracle ofrece la cláusula `FOR UPDATE` de la sentencia `SELECT` para realizar este bloqueo.
- Cuando emite una sentencia `SELECT...FOR UPDATE`, Oracle obtiene automáticamente los bloqueos a nivel de fila exclusivos de todas las filas identificadas por la sentencia `SELECT`.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

1 - 368

## FOR UPDATE Clause: Examples

- Puede adjuntar la palabra clave opcional `NOWAIT` a la cláusula `FOR UPDATE` para indicar al servidor de Oracle que no espere si otro usuario ha bloqueado la tabla.
- En este caso, el control se devolverá inmediatamente al para que pueda realizar otro trabajo o simplemente esperar un período de tiempo antes de volver a intentarlo.
  - Sin la cláusula `NOWAIT`, el proceso se bloqueará hasta que la tabla esté disponible, cuando otro usuario libere los bloqueos a través de la emisión un comando `COMMIT` o `ROLLBACK`.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE WAIT 5
ORDER BY employee_id;
```

1 - 369

## Quiz

The following statements produce the same results:

```
DELETE FROM copy_emp;
```

```
TRUNCATE TABLE copy_emp;
```

- a. True
- b. False

1 - 370

## Resumen

En esta lección, usted debe haber aprendido los siguientes comando:

Function	Description
INSERT	Añadir una nueva fila en la tabla
UPDATE	Modificar filas existentes en la tabla
DELETE	Eliminar filas existentes en la tabla
TRUNCATE	Eliminar todas las filas de una tabla
COMMIT	Validar los cambios pendientes
SAVEPOINT	Punto de SALVAGUARDIA para operaciones rollback
ROLLBACK	Deshacer las operaciones pendientes
FOR UPDATE clause in SELECT	Bloquear todas las filas devueltas de una sentencias SELECT

1 - 371

## Practica 10

Esta práctica se abordan los siguientes temas:

- Insertando filas en tablas
- Actualizando y borrando filas en tablas
- Controlando Transacciones

1 - 372

# 11

## Introducción al Lenguaje de Definición de Datos



ORACLE®

## Objetivos

Después de completar esta lección, usted debería ser capaz de hacer lo siguiente:

- Categorizar los objetos principales de una BBDD
- Revisar la estructura de una tabla
- Listar los tipos de datos disponibles en las columnas
- Crear una tabla simple
- Explicar como son creadas la restricciones en tiempo de creación de la tabla
- Describir como se trabajan con los objetos de esquema

1 - 374

## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- Comando `CREATE TABLE`
- Tipos de Datos
- Restricciones: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creando una tabla usando una subconsulta
- Comando `ALTER TABLE`
- Comando `DROP TABLE`

1 - 375

## Objetos de la Base de Datos

- La base de datos Oracle puede contener varias estructuras de datos.
- Cada estructura deberá indicarse en el diseño de base de datos de manera que se puede crear durante la etapa de construcción de desarrollo de base de datos.

Objeto	Descripción
Table	Es la unidad básica de almacenamiento
View	Representación lógica de un subconjuntos de datos de una o más tablas
Sequence	Valores numéricos incrementales
Index	Estructura que mejora el rendimiento de consultas
Synonym	nombre alternativo para un objeto

1 - 376

## Reglas de Nombres

Los nombres de tabla y nombres de columna deben:

- Comenzar con una letra
- Entre 1-30 caracteres de largo
- Constituido únicamente por A-Z, a-z, 0-9, \_ \$ y #
- Nombre de objeto único dentro del mismo usuario.
- No puede ser una palabra reservada servidor Oracle

### Nota:

- No se distingue entre mayúsculas y minúsculas en los nombres.

1 - 377

## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- **Comando CREATE TABLE**
- Tipos de Datos
- Restricciones: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Creando una tabla usando una subconsulta
- Comando ALTER TABLE
- Comando DROP TABLE

1 - 378

## Comando CREATE TABLE

- El comando `CREATE TABLE` se utiliza para crear tablas y almacenar datos.
- Este comando tienen un efecto inmediato en la base de datos y también grabar información en el diccionario de datos.
- Para crear una tabla, un usuario debe tener el privilegio `CREATE TABLE` y un área de almacenamiento en el que crear objetos (`TABSPACE`)



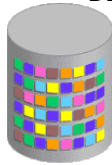
1 - 379

## Comando CREATE TABLE

- Sintaxis:

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

Schema	Esquema donde crear la tabla. Por defecto el del usuario que ejecuta la sentencia
Table	Nombre de la Tabla a crear
Column	Nombre de la columna
Datatype	Tipo de dato de la columna
DEFAULT expr	Especifica un valor por defecto si se omite un valor en la instrucción INSERT



**Nota:** Para crear tablas en cualquier esquema, es necesario el privilegio `CREATE ANY TABLE`

1 - 380

## Creando Tablas

- Creando la tabla:

```
CREATE TABLE dept  
    (deptno      NUMBER(2),  
     dname       VARCHAR2(14),  
     loc         VARCHAR2(13),  
     create_date DATE DEFAULT SYSDATE);
```

table DEPT created.

- Confirmando la creación:

```
DESCRIBE dept
```

NAME	Null	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE

Debido a que `CREATE` es una sentencia DDL, se realiza un `commit` de forma automática

1 - 381



## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- Comando CREATE TABLE
- Tipos de Datos
- Restricciones: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Creando una tabla usando una subconsulta
- Comando ALTER TABLE
- Comando DROP TABLE

1 - 382

## Tipos de Datos

Tipo de Dato	Descripcion
VARCHAR2( <i>size</i> )	Datos de caracteres de longitud variable
CHAR( <i>size</i> )	Datos de caracteres de longitud fija
NUMBER( <i>p</i> , <i>s</i> )	Datos numéricos de longitud variable
DATE	Valores de fecha y hora
LONG	Datos de caracteres de longitud variable (hasta 2 GB)
CLOB	Datos de caracteres de longitud variable. Tamaño máximo(4 Gb - 1) * (DB_BLOCK_SIZE).
RAW and LONG RAW	Datos binarios de longitud variable
BLOB	Datos binarios de longitud variable. Tamaño máximo(4 Gb - 1) * (DB_BLOCK_SIZE)
BFILE	Datos binarios almacenados en un archivo externo (hasta 4 GB)
ROWID	Número interno de Oracle, representado en base-64 que representa una dirección única

1 - 383

## Tipos de Datos. LONG

### Restricciones

- Una columna `LONG` no se copia cuando se crea una tabla con una subconsulta.
- Una columna `LONG` no se puede incluir en un `GROUP BY` o una cláusula `ORDER BY`.
- Sólo puede existir una columna `LONG` por tabla.
- No se pueden definir `CONSTRAINTS` en columnas `LONG`.
- Oracle recomienda la utilización de columnas `CLOB` en lugar de una columna `LONG`.

1 - 384

## Tipos de fecha y hora

Se pueden utilizar diferentes tipos de datos de fecha y hora:

Tipo de Dato	Descripción
<code>TIMESTAMP</code>	Fecha con fracciones de segundo
<code>INTERVAL YEAR TO MONTH</code>	Se almacenan los datos como un intervalo de años y meses
<code>INTERVAL DAY TO SECOND</code>	Se almacenan los datos como un intervalo de días, horas, minutos y segundos

### Nota:

- Estos tipos de datos de fecha y hora están disponibles desde Oracle 9i y versiones posteriores.
- Los tipos de datos de fecha y hora se discuten en detalle en SQL Workshop II



1 - 385

## Opción DEFAULT

- Cuando se define una tabla, puede especificar que una columna se debe dar un valor predeterminado mediante la opción por defecto.
- Esta opción evita que haya valores `NULL` en dichas columnas cuando se inserta una fila sin un valor para la columna.
- El valor por defecto puede ser una:
  - Una expresión literal
  - Una función de SQL (como `SYSDATE` o `USERS`)
- El Valor por defecto no puede ser:
  - El nombre de otra columna
  - Una pseudo-columna (como `NEXTVAL` o `CURRVAL`).

1 - 386

## Opción DEFAULT

- Especificar un valor predeterminado para una columna en la tabla `CREATE`.

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

- Los valores literales, expresiones o funciones de SQL son **valores legales**.
- El nombre de otra columna o una pseudo-columna son valores **no válidos**.
- El tipo de datos predeterminado debe coincidir con el tipo de la columna.

```
CREATE TABLE hire_dates  
  (id          NUMBER(8),  
   hire_date DATE DEFAULT SYSDATE);  
table HIRE_DATES created.
```

1 - 387

## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- Comando `CREATE TABLE`
- Tipos de Datos
- **Restricciones:** NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Creando una tabla usando una subconsulta
- Comando `ALTER TABLE`
- Comando `DROP TABLE`

1 - 388

## Incluyendo Restricciones (Constraints)

- El servidor Oracle utiliza restricciones para evitar la entrada de datos no válidos en tablas.
- Las restricciones son aplicadas a nivel de tabla.
- Todas las restricciones se almacenan en el diccionario de datos
- Los siguientes tipos de restricciones son válidas:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



1 - 389

## Directrices de las Restricciones (Constraints)

- Debemos darle un nombre significativo a cada restricción, sino Oracle genera un nombre propio usando el formato `SYS_Cn`.
- La creación de la restricción puede ser creada:
  - Al mismo tiempo que la creación de la tabla
  - Después de la creación de la tabla
- Se puede definir una restricción a nivel de columna o tabla
- Podemos ver las restricciones que tenemos en el diccionario de datos (`USER_CONSTRAINTS`)

1 - 390

## Definiendo Restricciones

- Sintaxis:

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

- Sintaxis a nivel de Columna:

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

- Sintaxis a nivel de Tabla:

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

1 - 391

## Definiendo Restricciones

- Ejemplo de restricción a nivel de columna:

```
CREATE TABLE employees (
  employee_id NUMBER(6)
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,
  first_name VARCHAR2(20),
  ...);
```

1

- Ejemplo de restricción a nivel de Tabla:

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

2

1 - 392

## Restricción NOT NULL

Asegura que los valores NULL no estén en la columna:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathan	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIEZT	515.123.8181	07-JUN-94

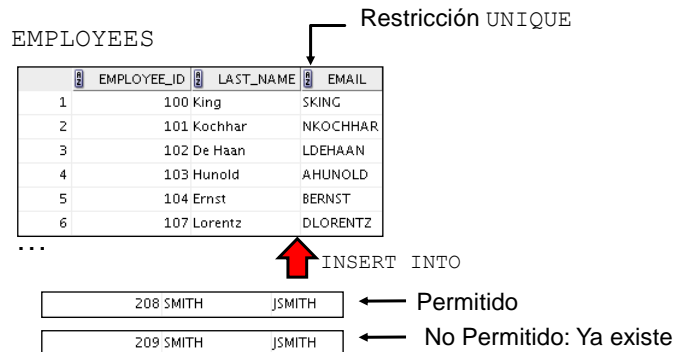
↑  
restricción NOT NULL  
(Primary Key impone  
restricción NO NULL.)

↑  
La ausencia de restricción NOT NULL  
(Cualquier fila puede contener un valor  
nulo para esta columna).  
↑  
Restricción NOT NULL

1 - 393

## Restricción UNIQUE

- Una restricción **UNIQUE** requiere que todos los valores de una columna o un conjunto de columnas (clave) sean únicos.
- **UNIQUE** permiten la entrada de los nulos a menos que también define las restricciones **NOT NULL**



1 - 394

## Restricción UNIQUE

Definiendo la restricción a nivel de tabla

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

Restricción **UNIQUE** a la columna de correo electrónico de la tabla empleados.  
El nombre de la restricción es **EMP\_EMAIL\_UK**.

1 - 395

## Restricción PRIMARY KEY

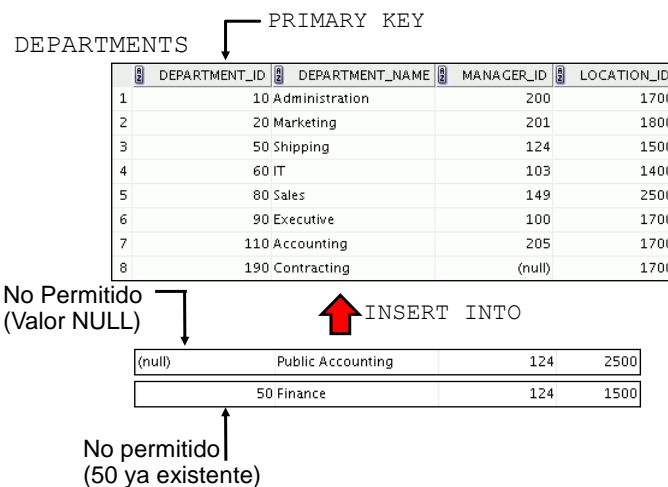
- Una restricción `PRIMARY KEY` crea una clave principal de la tabla.
- Sólo una clave primaria puede ser creado para cada tabla.
- La restricción `PRIMARY KEY` es una columna o un conjunto de columnas que identifica de forma única cada fila de una tabla.
- Esta restricción hace cumplir la unicidad de una columna o conjuntos de columnas, y asegura que ninguna columna que forma parte de la clave principal puede contener un valor nulo.

### Nota:

- Debido a la unicidades parte de la definición de la restricción de clave principal, el servidor Oracle refuerza la unicidad de forma implícita mediante la creación de un **índice único** en la columna o columnas de la clave primaria.

1 - 396

## Restricción PRIMARY KEY



1 - 397

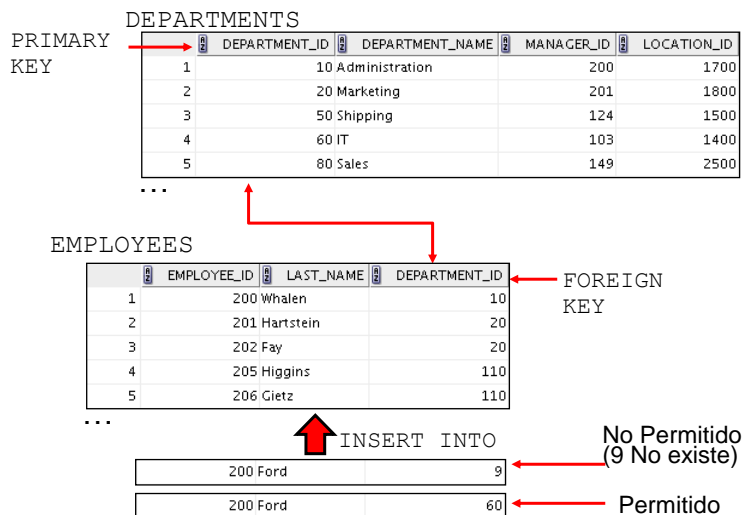


## Restricción FOREIGN KEY

- Una restricción FOREIGN KEY (restricción de integridad) designa una columna o columnas como clave externa, y establece una relación con una clave primaria o una clave única en la misma tabla o una tabla diferente.
- Un valor de clave externa debe coincidir con un valor existente en la tabla padre (PRIMARY KEY) o ser nulo.
- Las claves externas se basan en los valores de los datos y son conexiones lógicas; a nivel físico, se implementan mediante punteros.

1 - 398

## Restricción FOREIGN KEY



1 - 399

## Restricción FOREIGN KEY

La restricción se puede definir a nivel de fila o de columna:

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

1 - 400

## Restricción FOREIGN KEY : Palabras claves

- **FOREIGN KEY:** se utiliza para definir la restricción en la columna en la tabla secundaria.
- **REFERENCES:** identifica la tabla y la columna de la tabla primaria
- **ON DELETE CASCADE:** indica que cuando se elimina una fila de la tabla padre, también se eliminan las filas dependientes de la tabla secundaria
- **ON DELETE SET NULL:** indica que cuando se elimina una fila de la tabla padre, los valores de clave externa se establecen en NULL.

1 - 401

## Restricción CHECK

- La restricción `CHECK` define una condición que debe satisfacer cada fila.
- Para satisfacer la restricción, cada fila de la tabla debe hacer la condición `TRUE`
- Condiciones:
  - La condición puede usar las mismas construcciones como las condiciones de la consulta, excepto las consultas que hacen referencia a otros valores en otras filas
  - Una sola columna puede tener múltiples restricciones

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

1 - 402

## CREATE TABLE: Ejemplo

```
CREATE TABLE teach_emp (  
    empno      NUMBER(5) PRIMARY KEY,  
    ename      VARCHAR2(15) NOT NULL,  
    job        VARCHAR2(10),  
    mgr        NUMBER(5),  
    hiredate   DATE DEFAULT (sysdate),  
    photo      BLOB,  
    sal        NUMBER(7,2),  
    deptno     NUMBER(3) NOT NULL  
    CONSTRAINT admin_dept_fkey REFERENCES  
        departments(department_id));
```

1 - 403

## Violando Restricciones

- Cuando tenemos restricciones en columnas o tablas, se devuelve un error si se intenta violar dicha restricción

"parent key not found" violation ORA-02291.

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

Error starting at line 1 in command: UPDATE employees SET   department_id = 55 WHERE department_id = 110	
Error report: SQL Error: ORA-02291: integrity constraint (ORA1.EMP_DEPT_FK) violated - parent key not found 02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found" *Cause:   A foreign key value has no matching primary key value. *Action:  Delete the foreign key or add a matching primary key.	

Departamento 55 no existe.

1 - 404

## Violando Restricciones

- No se puede eliminar una fila que contiene una clave principal que se utiliza como una clave externa de otra tabla.

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

Error starting at line 1 in command: DELETE FROM departments WHERE department_id = 60	
Error report: SQL Error: ORA-02292: integrity constraint (ORA1.JHIST_DEPT_FK) violated - child record found 02292. 00000 - "integrity constraint (%s.%s) violated - child record found" *Cause:   attempted to delete a parent key value that had a foreign dependency. *Action:  delete dependencies first then parent or disable constraint.	

- Se puede realizar dicho borrado sino existe ningún elemento en la Clave Foranea

1 - 405

## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- Comando `CREATE TABLE`
- Tipos de Datos
- Restricciones: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- **Creando una tabla usando una subconsulta**
- Comando `ALTER TABLE`
- Comando `DROP TABLE`

1 - 406

## Creación de una tabla mediante una subconsulta

- Crear una tabla e insertar filas mediante la combinación de la sentencia `CREATE TABLE` y la opción `AS subconsulta`.

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

- El número de columnas debe de coincidir con las columnas que queremos crear.

Table	Nombre de la tabla
Column	Es el nombre de la columnas a crear
Subquery	Subconsulta que genera las filas y la estructura de tabla

1 - 407

## Creación de una tabla mediante una subconsulta

crea una tabla denominada DEPT80, que contiene detalles de todos los empleados que trabajan en el departamento de 80

```
CREATE TABLE dept80
AS
SELECT employee_id, last_name,
       salary*12 ANNSAL,
       hire_date
FROM   employees
WHERE  department id = 80;
```

table DEPT80 created.

```
DESCRIBE dept80
```

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

1 - 408

## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- Comando CREATE TABLE
- Tipos de Datos
- Restricciones: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK
- Creando una tabla usando una subconsulta
- **Comando ALTER TABLE**
- Comando DROP TABLE

1 - 409

## Comando ALTER TABLE

Después de crear una tabla, puede que tenga que cambiar la estructura de la tabla para cualquiera de las siguientes razones:

- Añadir una nueva columna
- Modificar una definición de columna existente
- Definir un valor predeterminado para la nueva columna
- Quitar una columna
- Cambiar el nombre de una columna
- Cambiar la tabla de estado de sólo lectura

1 - 410

## Comando ALTER TABLE

Utilice la sentencia `ALTER TABLE` para añadir, modificar o eliminar columnas:

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

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

```
ALTER TABLE table
DROP (column [, column] ...);
```

1 - 411

## Añadiendo una Columna

- Se utiliza la cláusula `ADD` para agregar columnas::

```
ALTER TABLE dept80
ADD      (job_id VARCHAR2(9)) ;
```

table DEPT80 altered.

- Directrices:

- No se puede especificar la ubicación en la columna debe aparecer. La nueva columna se convierte en la última columna.

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	149	Zlotkey	10500	29-JAN-08	(null)
2	174	Abel	11000	11-MAY-04	(null)
3	176	Taylor	8600	24-MAR-06	(null)

- Si una tabla ya contiene filas cuando se añade una columna, la nueva columna es inicialmente nulo o toma el valor por defecto para todas las filas
- Se puede agregar una columna `NOT NULL` a una tabla vacía y sin el valor por defecto

1 - 412

## Modificando una Columna

- Puede cambiar el tipo de datos, el tamaño y el valor predeterminado de una columna.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2(30)) ;
```

table DEPT80 altered.

- Directrices:

- El cambio de una columna, sólo afecta a las inserciones posteriores en la tabla.
- Puede aumentar la anchura o la precisión de una columna numérica.
- Puede aumentar el ancho de las columnas de caracteres.
- Puede disminuir el ancho de una columna si:
  - La columna contiene sólo valores nulos
  - La tabla no tiene ninguna fila
  - La disminución no debe de ser menor a los valores existentes
- Puede cambiar el tipo de datos si la columna contiene sólo valores nulos (excepto con familias de caracteres `CHAR` - `VARCHAR`)

1 - 413



## Borrando un columna

Podemos eliminar una columna de una tabla utilizando la sentencia `ALTER TABLE` con la cláusula `DROP COLUMN`.

```
ALTER TABLE dept80  
DROP (job_id);
```

table DEPT80 altered.

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	149	Zlotkey	10500	29-JAN-08
2	174	Abel	11000	11-MAY-04
3	176	Taylor	8600	24-MAR-06

- Directrices:
  - La columna puede o no puede contener datos.
  - la sentencia `ALTER DROP`, sólo puede ser eliminada una columna a la vez.
  - En la tabla debe de quedar como mínimo una columna.
  - Una vez borrada la columna, ésta no puede ser recuperada.
  - El borrado de una columna puede tomar un tiempo si la columna tiene un gran número de valores

1 - 414

## Opción SET UNUSED

- La opción `SET UNUSED` marca una o más columnas como no utilizada de modo que se pueden descartar cuando la demanda de recursos del sistema es menor.
- La especificación de esta cláusula en realidad **no eliminar las columnas de destino** de cada fila de la tabla, las marca para ser borradas.
- Las columnas **no utilizadas** se tratan como si se hubiesen borrado, a pesar de que sus datos de la columna se mantiene en las filas del mismo.
- Después de una columna se ha marcado **como sin usar**, usted no tiene acceso a esa columna

1 - 415

## Opción SET UNUSED

- La información de columnas no utilizadas se almacena en la vista de diccionario USER\_UNUSED\_COL\_TABS
- Se puede utilizar la DROP UNUSED COLUMNS para eliminar las columnas que están marcados como no utilizados.
- Se puede especificar la palabra clave ONLINE para indicar que se permitirán las operaciones de DML en la tabla mientras se marca la columna o columnas UNUSED.

```
ALTER TABLE <table_name>  
SET UNUSED (<column_name> [, <column_name>]);  
OR  
ALTER TABLE <table_name>  
SET UNUSED COLUMN <column_name> [, <column_name>];
```

```
ALTER TABLE <table_name>  
DROP UNUSED COLUMNS;
```

1 - 416

## Tablas Read-Only

- Podemos especificar la sentencia ALTER TABLE con la clausula READ ONLY para poner una tabla en modo de solo lectura.
- Cuando la tabla está en el modo Lectura, no se puede emitir ninguna declaración DML que afectan a la tabla o la sentencia SELECT ... FOR UPDATE

```
ALTER TABLE employees READ ONLY;  
  
-- perform table maintenance and then  
-- return table back to read/write mode  
  
ALTER TABLE employees READ WRITE;
```

1 - 417

## Tablas Read-Only

Podemos utilizar esta clausula para:

- Impedir que se realicen sentencias DDL o DML durante el mantenimiento de tablas
  - Puede emitir sentencias DDL, siempre y cuando no modifiquen los datos en la tabla.
- Las operaciones en los índices asociados a la tabla se permiten cuando la tabla está en modo de sólo lectura.
- Podemos especificar los valores `READ/WRITE` para cambiar una tabla de sólo lectura a modo lectura / escritura.

**Nota:**

Puede eliminar una tabla que está en modo de sólo lectura. El comando `DROP` sólo se ejecuta en el diccionario de datos, por lo que no se requiere el acceso a los contenidos de la tabla

1 - 418

## Agenda

- Objetos de Base de Datos
  - Reglas de Nombres
- Comando `CREATE TABLE`
- Tipos de Datos
- Restricciones: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creando una tabla usando una subconsulta
- Comando `ALTER TABLE`
- Comando `DROP TABLE`

1 - 419

## Borrando una Tabla

- La sentencia `DROP TABLE` mueve una tabla a la **papelera de reciclaje** o elimina la tablas y todos sus datos de la base de datos por completo (`PURGE`)
- Eliminar una tabla hace que se invaliden los objetos dependientes de la misma y elimina los privilegios de objeto sobre la tabla.
- Cuando se elimina una tabla, la base de datos pierde todos los datos de la tabla y todos los **índices** asociados a ella.
- Sintaxis:

```
DROP TABLE dept80;
```

```
table DEPT80 dropped.
```

1 - 420

## Borrando una Tabla

### Directivas

- Todos los datos se elimina de la tabla.
- Cualquier VISTA y sinónimos permanecen, pero no son válidos.
- Ninguna transacción pendiente es validada.
- Sólo el creador de la tabla o un usuario con el privilegio `DROP ANY TABLE` puede borrar una tabla.
- Sino se ha utiliza la clausula `PURGE`, podemos utilizar la sentencia `FLASHBACK TABLE` para restaurar una tabla eliminada de la papelera de reciclaje

1 - 421

## Resumen

En esta lección, usted debe haber aprendido a:

- Categorizar los objetos principales de una BBDD
- Revisar la estructura de una tabla
- Listar los tipos de datos disponibles en las columnas
- Crear una tabla simple
- Explicar como son creadas la restricciones en tiempo de creación de la tabla
- Describir como se trabajan con los objetos de esquema

1 - 423

## Practica 11

Esta práctica se abordan los siguientes temas

- Creación de nuevas tablas
- Crear tablas usando la sintaxis `CREATE TABLE AS`
- Verificar que las tablas existen
- Modificar Tablas
- Añadir columnas
- Borrar columnas
- Modificar una tabla a read-only
- Borrar tablas

1 - 424