

# **ORACLE**

## Academy

# Database Programming with SQL

1-1

Oracle Application Express

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Distinguir entre software de aplicaciones y software de sistema y dar un ejemplo de cada
  - Conectarse al entorno de prácticas Oracle Application Express
  - Ejecutar una consulta simple para recuperar información de la base de datos
  - Aplicar las reglas de SQL para mostrar todas las columnas y un subjuego de columnas especificado por criterios

# Objetivo

- Cada día, de un modo u otro, se entra en contacto con aplicaciones informáticas
- Si hoy ha comprobado el correo electrónico, probablemente lo haya hecho mediante una aplicación
- Si ha comprado un artículo en una tienda de comestibles, el dependiente ha escaneado dicho artículo utilizando una aplicación que calcula su cuenta y actualiza el inventario de la tienda
- En este curso, aprenderá la sintaxis de SQL con la aplicación denominada Oracle Application Express

# Programas de Aplicación

- Aunque las computadoras han existido desde hace mucho tiempo (posiblemente antes de que naciera), su uso para la informática empresarial y personal no tuvo lugar hasta que se desarrollaron los programas de aplicaciones
- Los programas de aplicación permiten al usuario final, personas como usted y yo, comprar programas completamente desarrollados, listos para usar
- Ya no era necesario saber cómo funcionaba el programa, tan solo que lo hacía y realizaba lo que queríamos que hiciera

ORACLE

Academy

DP 1-1  
Oracle Application Express

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

5

# Programas de Aplicación

- El software de programas de aplicación es diferente del software de sistema
- El software de sistema consta de programas de bajo nivel diseñados para interactuar con el hardware informático
- Los sistemas operativos, los compiladores y las utilidades del sistema son ejemplos de software de sistema
- Por el contrario, el software de aplicación incluye programas para procesamiento de texto, bases de datos, juegos, correo electrónico y gráficos

Software de aplicación: programa que proporciona las instrucciones informáticas que otorgan al usuario las herramientas para realizar una tarea.

Software de sistema: programas que interactúan con el hardware informático.

# Programas de Aplicación

- Yahoo.com utiliza la base de datos Oracle para almacenar datos
- En lugar de hacer que todos los usuarios que deseen buscar en la base de datos o recuperar correo electrónico aprendan SQL, una aplicación tiene todo el SQL (y otros lenguajes de codificación) programado en ella
- Con unos clics del mouse, los usuarios tienen acceso a toda la información que necesitan

# Uso de Aplicaciones

- Una aplicación es como un vehículo
- Para conducir un vehículo, es necesario saber lo suficiente para hacerlo funcionar
- Tiene una especie de "shell" para ocultar todas las cosas que no necesita saber, como el modo en que funciona la transmisión o cómo se usa el combustible como la gasolina o el diésel para alimentar el motor
- ¿Podría haber obtenido el carnet de conducir si hubiera tenido que demostrar la comprensión de cada sistema (eléctrico, de transmisión, hidráulico, de combustible, etc.) que se utiliza para que funcione un vehículo?

# Oracle Application Express

- En este curso, utilizará Oracle Application Express
- Esta aplicación permite a muchos de los desarrolladores crear y acceder a las aplicaciones como si estuvieran en ejecución en bases de datos independientes
- Con funciones integradas como temas de diseño, controles de navegación, manejadores de formularios e informes flexibles, Oracle Application Express acelera el proceso de desarrollo de la aplicación

# Oracle Application Express

- Dos componentes en Oracle Application Express son:
  - Taller de SQL
  - Creador de Aplicaciones
- Para obtener más información sobre SQL, utilizará el componente Taller de SQL
- Para diseñar una aplicación, se utiliza Creador de Aplicaciones

# Oracle Application Express

- Oracle Application Express (APEX) es la herramienta que utilizaremos para que pueda crear tablas y recuperar información de una base de datos Oracle
- Al recuperar información de una base de datos, a menudo deberá buscar un subjuego de los datos según unos criterios de búsqueda específicos
- Familiarizarse con SQL le ayudará a encontrar más rápidamente la información que necesita

"Oracle Application Express User Guide" se encuentra en la sección 0, Recursos del Curso, del curso Programación de Bases de Datos con SQL. Este documento le ayudará a familiarizarse con el uso de Oracle Application Express y cada uno de sus componentes desde la perspectiva de un usuario final.

# Oracle Application Express

- Las cuentas de Oracle Application Express (APEX) se suministran sin tablas ni datos
- Puede encontrar un archivo de secuencia de comandos e instrucciones sobre cómo ejecutar la secuencia de comandos en los Recursos para estudiantes del Member Hub
- Al ejecutar el script, las tablas y los datos utilizados en el curso se agregarán a su esquema
- Para obtener más información sobre el uso de APEX, consulte las Guías para el alumno y el instructor de APEX iAcademy

"Oracle Application Express User Guide" se encuentra en la sección 0, Recursos del Curso, del curso Programación de Bases de Datos con SQL. Este documento le ayudará a familiarizarse con el uso de Oracle Application Express y cada uno de sus componentes desde la perspectiva de un usuario final.

## Sentencia SELECT básica

- El comando SELECT \* devuelve todas las filas de una tabla
- La sintaxis es la siguiente:

```
SELECT *
FROM <table name>;
```

- Por ejemplo:

```
SELECT *
FROM employees;
```



DP 1-1  
Oracle Application Express

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

13

Sintaxis: las reglas que rigen la formación de sentencias en un lenguaje de programación.

Los alumnos deben introducir el ejemplo anterior en APEX para ver la salida de resultados.

# Sentencia SELECT con una Condición

- Para devolver un subjuego de los datos, modifique la sentencia SELECT
- La sintaxis es la siguiente:

```
SELECT <column name 1, column name 2, etc.>
FROM <table name>
WHERE <condition>;
```

- Por ejemplo:

```
SELECT first_name, last_name, job_id
FROM employees
WHERE job_id = 'SA_REP';
```

La <condition> comienza con la palabra WHERE seguida de un <column name> y un operador de comparación (=, >, <, etc.) seguido de un valor o IS NULL, IS NOT NULL.

Subjuego: una parte de un grupo mayor de cosas relacionadas.

Operador de comparación: se utiliza en condiciones que comparan una expresión con otra expresión o valor.

La sintaxis de las sentencias SELECT y los operadores se tratan con más detalle en las siguientes lecciones.

## Corrección de Errores.

- Al introducir comandos SQL, es importante utilizar la ortografía correcta, de lo contrario, recibirá un mensaje de error
- Por ejemplo (SELECT: ortografía incorrecta):

```
SELECT *  
FROM employees;
```

- Daría como resultado el mensaje de error:

ORA-00900: invalid SQL statement

- Para rectificar, solo tiene que corregir la ortografía y volver a ejecutarlo

Al aprender a escribir las sentencias SQL, es normal que se cometan errores al principio. Con práctica y experiencia, cometerá menos errores, y podrá rectificar errores más fácilmente.

## Corrección de Errores.

- También es importante utilizar los nombres y la ortografía correctos de las columnas y tablas
- Por ejemplo (nombre de la tabla employees: ortografía incorrecta):

```
SELECT *  
FROM employee;
```

- Daría como resultado el mensaje de error:



ORA-00942: table or view does not exist

- Para rectificar, solo tiene que corregir la ortografía y volver a ejecutarlo

El profesor mostrará las tablas que se han creado en el esquema de su base de datos mediante el Explorador de objetos.

## Corrección de Errores.

- Por ejemplo, (columna first\_name introducida de forma incorrecta):

```
SELECT name  
FROM employees;
```

- Daría como resultado el mensaje de error:



ORA-00904: "NAME": invalid identifier

- Para rectificar, simplemente introduzca el nombre correcto de la columna y vuelva a ejecutarlo

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Software de la aplicación
  - Software de sistema
  - Oracle Application Express
  - Sintaxis
  - Subjuego
  - Operador de comparación

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Distinguir entre software de aplicaciones y software de sistema y dar un ejemplo de cada
  - Conectarse al entorno de prácticas Oracle Application Express
  - Ejecutar una consulta simple para recuperar información de la base de datos
  - Aplicar las reglas de SQL para mostrar todas las columnas y un subjuego de columnas especificado por criterios

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

1-2

Tecnología de Base de Datos Relacional

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Definir y dar un ejemplo de una base de datos relacional
  - Identificar los términos clave de la tabla, incluidos fila, columna, campo, clave primaria y clave ajena
  - Relacionar la importancia de las bases de datos con la vida diaria



# Objetivo

- Las bases de datos son parte de nuestra vida diaria, aunque la mayoría del tiempo ni siquiera pensemos en ellas
- Si alguna vez ha realizado una reserva en una línea aérea, ha utilizado un cajero automático, o ha realizado una llamada de teléfono móvil, ha utilizado una base de datos
- De hecho, muchas ciudades usan bases de datos de sistemas de dirección de tráfico inteligentes para controlar los semáforos
- Así que, la próxima vez que esté esperando en un semáforo en rojo puede que una base de datos sea la responsable de su retraso
- En esta lección, aprenderá más sobre bases de datos y cómo se crean y organizan

**ORACLE**

Academy

DP 1-2  
Tecnología de Base de Datos Relacional

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

4

Algunas bases de datos interesantes encontradas en Internet:

[www.rcdb.com](http://www.rcdb.com) (bases de datos de montaña rusa, mucha información y apta para la búsqueda)

[www.classical.net](http://www.classical.net) (base de datos de música en línea que enlaza con muchas otras bases de datos)

[www.rangercentral.com](http://www.rangercentral.com) (base de datos de los Power Rangers)

[www.museumstuff.com](http://www.museumstuff.com) (lista de 1000 museos, gran exploración)

[www.drewsullivan.com](http://www.drewsullivan.com) (la base de datos de bases de datos del periodista, súper interesante)

# Bases de Datos Relacionales

- Una base de datos relacional permite relacionar tablas por medio de un campo común
- Con tan solo dos tablas se puede considerar como una base de datos relacional si comparten un campo común

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

Base de datos relacional: recopilaciones de objetos o relaciones, juego de operadores que actúan en esas relaciones, e integridad de datos para su precisión y consistencia.

Campo: intersección de una fila y una columna.

# Bases de Datos Relacionales

- En realidad, las bases de datos utilizadas en negocios tienen muchas tablas, y cada tabla comparte un campo común con otra tabla
- La tabla "países" que se muestra es una de las diversas tablas en la base de datos Employees y solo un ejemplo de las muchas tablas que se usarán en este curso

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

La columna "region\_id" columna de esta tabla es un campo común con la tabla "regiones"

# Bases de Datos Relacionales

- Para comprender el nivel de importancia de las bases de datos en el mundo actual, tenga en cuenta las siguientes estadísticas:
  - Actualmente, el 20 % de los datos del mundo reside en RDBMS
  - En los próximos dos años, se espera que las bases de datos crezcan hasta un tamaño superior a 100 terabytes
  - Una base de datos así de grande podría almacenar 100 000 ejemplares de la Enciclopedia Británica, 200 000 horas de música o unos 10 000 millones de páginas web

RDBMS: un sistema de gestión de bases de datos relacionales.

# Bases de Datos Relacionales

- Algunas de las 10 principales bases de datos más grandes del mundo que usan el RDBMS de Oracle son:
  - France Telecom, 29,2 TB, una empresa de comunicaciones (un TB es un terabyte, equivalente a 1000 gigabytes)
  - Amazon.com, 13 TB, venta de libros y mercancías
  - The Claria Corporation, 12 TB, empresa de marketing por comportamiento en Internet que sigue el comportamiento en Internet del usuario



# Revisión de Términos Clave

- Revise los siguientes términos clave:
  - tabla: estructura básica de almacenamiento
  - columna: un tipo de dato de una tabla
  - fila: datos de una instancia de tabla
  - campo: valor de la intersección de una fila y una columna
  - clave primaria: identificador único para cada fila
  - clave ajena: columna que hace referencia a una columna de clave primaria de otra tabla

# Propiedades de Tablas

- Hay seis propiedades de tablas en una base de datos relacional:
  - Propiedad 1: las entradas en las columnas tienen un valor único
  - Propiedad 2: las entradas en las columnas son del mismo tipo
  - Propiedad 3: cada fila es única
  - Propiedad 4: la secuencia de columnas no es significativa
  - Propiedad 5: la secuencia de filas no es significativa
  - Propiedad 6: cada columna tiene un nombre único

## Acceso a los Datos en un RDBMS

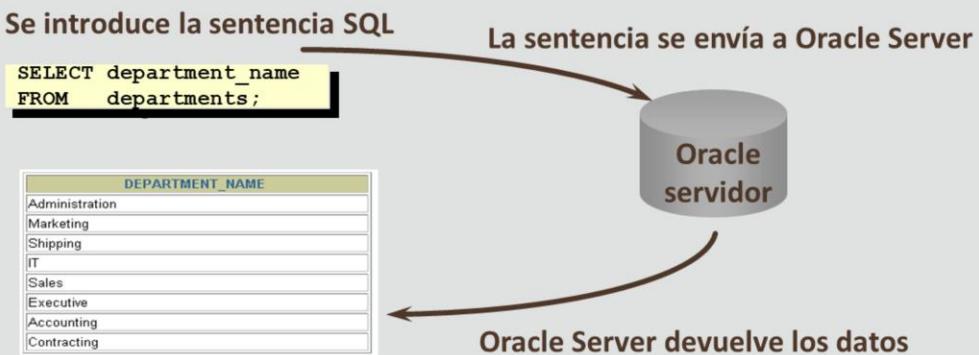
- Un sistema de administración de una base de datos relacional (RDBMS) organiza los datos en filas y columnas relacionadas
- Para acceder a los datos de una base de datos, no tiene que saber dónde se ubican los datos físicamente, ni debe especificar una ruta de acceso a las tablas
- Solo tiene que utilizar sentencias de lenguaje de consulta estructurado (SQL) y operadores

# Comunicación con Bases de Datos

- Trabajar con la base de datos es muy similar a llamar y hablar con un amigo por teléfono
  - En primer lugar, debe elegir un método para comunicarse (el teléfono)
  - Una vez conectado, le hace una pregunta a su amigo (una consulta)
  - En respuesta a su pregunta, su amigo le contesta (devolución de datos).
- Es muy simple, la mayoría de nosotros somos expertos en esto
- En esta clase, nuestro método de comunicación con la base de datos será a través de Oracle Application Express
- Al formular una pregunta con SQL, la aplicación devolverá una respuesta

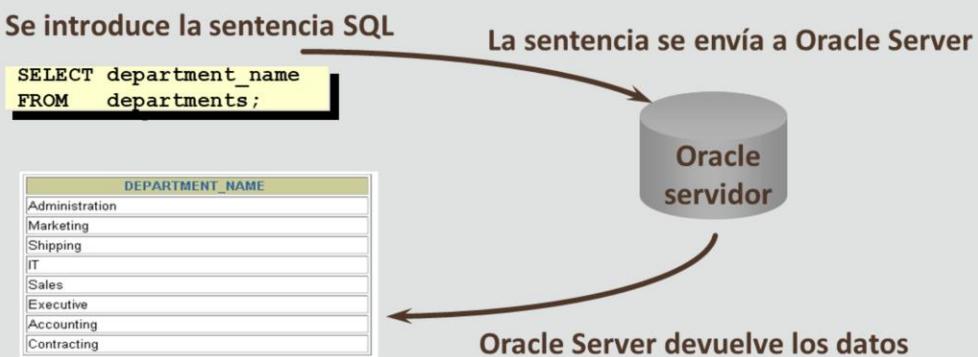
# Comunicación con Bases de Datos

- Como se muestra en el diagrama, comunicarse con un RDBMS se consigue introduciendo una sentencia SQL en Oracle Application Express



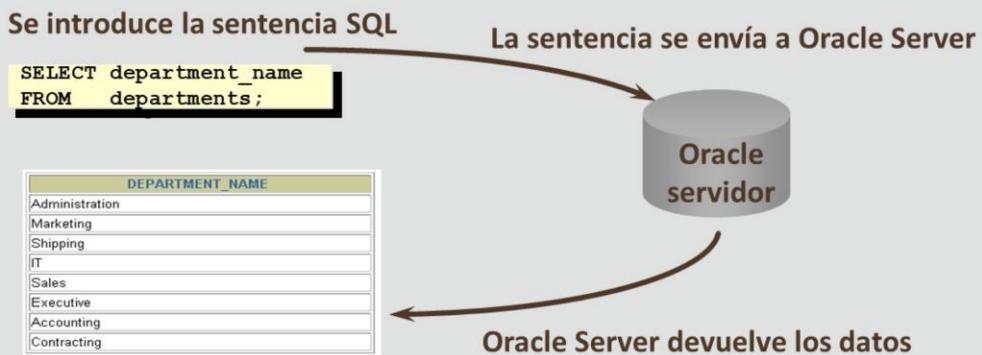
# Comunicación con Bases de Datos

- A continuación, se enviará la solicitud a Oracle Server (una base de datos que se ejecuta en una computadora), la solicitud se procesa y se muestran los datos obtenidos



# Comunicación con Bases de Datos

- En sistemas de base de datos de gran tamaño, con muchos usuarios, servidores y tablas forman el RDBMS



## Categorías de Sentencias SQL

- Las sentencias SQL se agrupan en varias categorías dependiendo de las funciones que realizan
- En este curso, aprenderá a utilizar SQL para ejecutar estas sentencias
- La sentencia de recuperación de datos recupera datos de la base de datos mediante la palabra clave SELECT

# Categorías de Sentencias SQL

- Hay cuatro categorías principales de sentencias SQL:
  - Lenguaje de manipulación de datos (DML)
  - Lenguaje de definición de datos (DDL)
  - Lenguaje de control de transacciones (TCL)
  - Lenguaje de control de datos (DCL)



**ORACLE**  
Academy

DP 1-2  
Tecnología de Base de Datos Relacional

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

17

# Categorías de Sentencias SQL

- Lenguaje de manipulación de datos (DML)

- Las sentencias DML comienzan con INSERT, UPDATE, DELETE o MERGE y se utilizan para modificar los datos de la tabla introduciendo nuevas filas, cambiando las filas existentes o eliminando las filas existentes

- Lenguaje de definición de datos (DDL)

- Las sentencias DDL crean, cambian y eliminan las estructuras de datos de la base de datos
  - Las palabras clave CREATE, ALTER, DROP, RENAME y TRUNCATE comienzan las sentencias DDL

La sentencia SELECT es una forma limitada de sentencia DML ya que solo puede acceder a los datos de la base de datos. No puede manipular los datos de la base de datos, aunque puede operar en los datos a los que accede antes de devolver los resultados de la consulta.

# Categorías de Sentencias SQL

- Lenguaje de control de transacciones (TCL)

- Las sentencias TCL se utilizan para gestionar los cambios realizados por las sentencias DML
- Los cambios de los datos se ejecutan mediante COMMIT, ROLLBACK y SAVEPOINT
- Los cambios del TCL se pueden agrupar en transacciones lógicas

- Lenguaje de control de datos (DCL)

- Las palabras clave del DCL GRANT y REVOKE se utilizan para proporcionar o eliminar derechos de acceso a la base de datos y las estructuras dentro de ella

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Lenguaje de control de datos (DCL)
  - Lenguaje de definición de datos (DDL)
  - Lenguaje de manipulación de datos (DML)
  - Field
  - Clave ajena
  - RDBMS

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Clave primaria
  - Base de datos relacional
  - Fila
  - Tabla
  - Control de transacciones (TCL)

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Definir y dar un ejemplo de una base de datos relacional
  - Identificar los términos clave de la tabla, incluidos fila, columna, campo, clave primaria y clave ajena
  - Relacionar la importancia de las bases de datos con la vida diaria



# **ORACLE**

## Academy



# ORACLE

## Academy



# Programación de bases de datos con SQL

1-3

## Anatomía de una Sentencia SQL

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Emparejar proyección y selección con sus capacidades correctas
  - Crear una sentencia SELECT básica
  - Utilizar la sintaxis correcta para mostrar todas las filas de una tabla
  - Utilizar la sintaxis correcta para seleccionar columnas específicas de una tabla, modificar la forma en que se muestran los datos y realizar cálculos utilizando expresiones aritméticas y operadores

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Formular consultas mediante la prioridad de operador correcta para mostrar los resultados deseados
  - Definir un valor nulo
  - Demostrar el efecto que los valores nulos crean en las expresiones aritméticas
  - Construir una consulta con un alias de columna

## Palabra clave SELECT

- SELECT es una de las palabras clave más importantes, si no la más importante, en SQL
- SELECT se utiliza para recuperar información de la base de datos. Cuando aprenda cómo utilizar SELECT, habrá abierto la puerta de la base de datos
- Imagine una base de datos que contiene información sobre películas como, por ejemplo, el título, el género, el estudio, el productor, la fecha de estreno, las series, el país, el idioma, la puntuación, la duración y demás
- ¿Qué pasaría si solo necesitan los títulos de las películas creadas en India?
- La sentencia SELECT le permite buscar datos específicos

# Sentencia SELECT

- La sentencia SELECT recupera información de la base de datos
- La sintaxis de la sentencia SELECT es la siguiente:

```
SELECT <column_name(s)>
FROM <table_name>;
```

- En su formato más simple, una sentencia SELECT debe incluir lo siguiente:
  - Una cláusula SELECT, que especifica las columnas que se van a mostrar
  - Una cláusula FROM, que especifica la tabla que contiene las columnas enumeradas en la cláusula SELECT

Cláusula SELECT: especifica las columnas que se van a mostrar (o utilice el símbolo \* para mostrar todas las columnas).

Una cláusula FROM: especifica la tabla que contiene las columnas enumeradas en la cláusula SELECT.

Columna: una implantación de un atributo o relación en una tabla.

# Convenciones

- En este curso, se utilizará lo siguiente:

```
SELECT last_name  
FROM employees;
```

- Una palabra clave hace referencia a un comando SQL individual
- Por ejemplo, SELECT y FROM son palabras claves
- Una cláusula es una parte de una sentencia SQL
- Por ejemplo, SELECT employee\_id,last\_name, es una cláusula
- Una sentencia es una combinación de dos o más cláusulas
- Por ejemplo, SELECT last\_name FROM employees es una sentencia

En este curso se utilizan las siguientes convenciones de estilo SQL:

Palabras clave SQL en MAYÚSCULAS: por ejemplo, SELECT, FROM, WHERE.

Los nombres de la tabla y la columna en minúsculas: por ejemplo, first\_name, employees.

Cada cláusula en una nueva línea: por ejemplo,

```
SELECT last_name
```

```
FROM employees
```

```
WHERE employee_id = 101;
```

# Capacidades de las Sentencias SELECT

- Proyección: se utiliza para seleccionar columnas de una tabla
- Selección: se utiliza para seleccionar filas de una tabla

Tabla 2: proyección


Tabla 2: selección


Proyección: puede utilizar la capacidad de proyección en SQL para elegir las columnas en una tabla que desee que le devuelva su consulta. Puede seleccionar el número de columnas de la tabla que deseé.

Selección: puede utilizar la capacidad de selección en SQL para elegir las filas en una tabla que desee que le devuelva una consulta. Puede utilizar varios criterios para restringir las filas que ve

# Proyección y Selección

ID	FIRST_NAME	LAST_NAME	SALARY
10	John	Doe	4000
20	Jane	Jones	3000
30	Sylvia	Smith	5000
40	Hai	Nguyen	6000



**ORACLE**  
Academy

DP 1-3  
Anatomía de una Sentencia SQL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

9

La cláusula SELECT determina la proyección.

La cláusula WHERE determina la selección.

La ejecución de esta consulta daría como resultado la visualización de la columna de salario de cualquier empleado con el last\_name (apellido) Smith.

# Selección de Todas las Columnas

- Puede mostrar todas las columnas de datos de una tabla mediante el uso de un símbolo de asterisco (\*) en lugar de un nombre de columna en la cláusula SELECT
- En el ejemplo, están seleccionadas todas las columnas de la tabla de países

```
SELECT *
FROM countries;
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

A veces, realizar una tabla SELECT \* FROM puede tardar bastante tiempo en devolver los datos. Todo depende del número de filas almacenadas en esa tabla concreta. Recuerde que las tablas de Oracle pueden almacenar millones de filas de datos.

## Selección de Todas las Columnas

- También puede mostrar todas las columnas en una tabla enumerándolas individualmente

```
SELECT country_id, country_name, region_id  
FROM countries;
```

COUNTRY_ID	COUNTRY_NAME	REGION_ID
CA	Canada	2
DE	Germany	1
UK	United Kingdom	1
US	United States of America	2

Todos los ejemplos de las diapositivas tienen un punto y coma después de cada sentencia. Oracle Application Express no necesita esta sintaxis, pero las demás interfaces de Oracle SQL sí, por lo que se han incluido en los cursos.

# Proyección de Columnas Concretas

- Si desea PROYECTAR para que solo se muestren las columnas específicas de una tabla, simplemente enumere cada uno de los nombres de las columnas que desee y separe cada nombre con una coma en la cláusula SELECT

```
SELECT location_id, city, state_province  
FROM locations;
```

LOCATION_ID	CITY	STATE_PROVINCE
1800	Toronto	Ontario
2500	Oxford	Oxford
1400	Southlake	Texas
1500	South San Francisco	California
1700	Seattle	Washington



Academy

DP 1-3  
Anatomía de una Sentencia SQL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

12

# Uso de Operadores Aritméticos

- Con unas sencillas reglas y directrices, puede construir sentencias SQL que sean fáciles de leer y de editar
- Conocer las reglas hará que el aprendizaje de SQL sea más fácil
- Puede que necesite modificar la forma en la que se muestran los datos, realizar cálculos o consultar supuestos de posibilidades
- Por ejemplo, "¿qué pasaría si a cada empleado se le elevara el sueldo un 5 %?
- ¿Cómo afectaría eso a sus cifras de beneficio anuales?"

# Uso de Operadores Aritméticos

- Estos tipos de cálculos son posibles mediante las expresiones aritméticas
- Ya está familiarizado con las expresiones aritméticas de matemáticas:
  - sumar (+), restar (-) , multiplicar (\*) y dividir (/)
- Tenga en cuenta que este ejemplo no crea nuevas columnas en las tablas ni cambia los valores reales de los datos
- Los resultados de los cálculos solo aparecerán en la salida

Expresión aritmética: una expresión que da como resultado un valor numérico.

# Uso de Operadores Aritméticos

- El ejemplo que se muestra utiliza el operador de suma para calcular un aumento de sueldo de 300 para todos los empleados y muestra una nueva columna, SALARY+300, en la salida
- Colocar espacios en blanco antes y después de un operador aritmético no afectará a la salida

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

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Whalen	4400	4700
Higgins	12000	12300
Gietz	8300	8600
Zlotkey	10500	10800
Abel	11000	11300
Taylor	8600	8900
Grant	7000	7300

Operador aritmético: símbolo que se utiliza para realizar una operación en algunos valores.

## Prioridad de los Operadores Aritméticos

- La prioridad es el orden en el que Oracle evalúa los diferentes operadores en la misma expresión
- Al evaluar una expresión con varios operadores, Oracle evalúa los operadores con mayor prioridad antes de evaluar los de menor prioridad
- Oracle evalúa los operadores con la misma prioridad de izquierda a derecha dentro de una expresión

# Prioridad de los Operadores Aritméticos

- Los operadores aritméticos realizan las operaciones matemáticas de multiplicación, división, suma y resta
- Si estos operadores aparecen juntos en una expresión, la multiplicación y la división se evaluarán primero.
- Por lo que el orden es: \* / + -
- Una forma fácil recordar la prioridad de los operadores es la fórmula mnemotécnica: Mi Deuda Se Reduce

## Prioridad de los Operadores Aritméticos

- Si los operadores en una expresión tienen la misma prioridad, la evaluación se realiza de izquierda a derecha
- Siempre puede utilizar los paréntesis para forzar que la expresión entre paréntesis se evalúe primero
- En las tablas de ejemplo que aparecen en la siguiente diapositiva, tenga en cuenta las diferencias en la salida entre la consulta que utilizó los paréntesis y la que no

# Prioridad de los Operadores Aritméticos

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

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Whalen	4400	52900
Higgins	12000	144100
Gietz	8300	99700

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

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Whalen	4400	54000
Higgins	12000	145200
Gietz	8300	100800



El ejemplo de la izquierda utiliza la prioridad de los operadores para determinar el orden en el que se realizan las operaciones. Como \* (multiplicación) tiene mayor prioridad, el salario se multiplica primero por 12 y, a continuación, se agrega 100 al resultado de la multiplicación. Por lo tanto, para el empleado King (salario, 24 000), se calcula  $12 * 24\ 000 = 288\ 000$ , después se añade 100 para dar la respuesta 288 100.

El ejemplo de la derecha utiliza paréntesis para forzar la adición que se llevará a cabo en primer lugar, el resultado de la adición, entonces, se multiplica por 12. Por lo tanto, King tiene  $24\ 000 + 100 = 24\ 100$ , después se multiplica por 12 para proporcionar la respuesta 289 200.

## Valores NULL

- En SQL, NULL es una palabra interesante
- Para comprender el concepto NULL, debe saber qué es NULL y qué no es NULL
- NULL es un valor que no está disponible, sin asignar, desconocido o que no es aplicable
- Un valor NULL no es lo mismo que un cero o un espacio
- En SQL, un cero es un número y un espacio es un carácter

NULL: un valor que no está disponible, sin asignar, desconocido o que no es aplicable.

## Valores NULL

- A veces, no conoce el valor de una columna
- En una base de datos, puede almacenar valores desconocidos en sus bases de datos
- Las bases de datos relacionales utilizan un marcador llamado NULL o null para representar dichos valores desconocidos

NULL: un valor que no está disponible, sin asignar, desconocido o que no es aplicable.

## Valores NULL

- Si cualquier valor de columna en una expresión aritmética es null, el resultado es nulo o desconocido
- Si intenta dividir un número entre un valor null, el resultado será nulo o desconocido
- Sin embargo, si se intenta una división entre cero, aparecerá un mensaje de error

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	-
Kochhar	AD_VP	17000	-
De Haan	AD_VP	17000	-
Whalen	AD_ASST	4400	-
Higgins	AC_MGR	12000	-
Gietz	AC_ACCOUNT	8300	-
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3

Salarios y Comisiones

## Valores NULL

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

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT	SALARY*COMMISSION_PCT
King	AD_PRES	24000	-	-
Kochhar	AD_VP	17000	-	-
De Haan	AD_VP	17000	-	-
Whalen	AD_ASST	4400	-	-
Higgins	AC_MGR	12000	-	-
Gietz	AC_ACCOUNT	8300	-	-
Zlotkey	SA_MAN	10500	.2	2100
Abel	SA_REP	11000	.3	3300
Taylor	SA_REP	8600	.2	1720

En el ejemplo, una fila con un valor null en la columna commission\_pct devolverá un valor nulo para la columna salary\*commission\_pct en esta consulta.

Los valores NULL aparecen en APEX utilizando el símbolo – (guión).

## Alias

- Un alias es una forma de cambiar el nombre de una cabecera de columna en la salida
- Sin alias, cuando se muestra el resultado de una sentencia SQL, el nombre de las columnas que se muestran será el mismo que los nombres de columna de la tabla o un nombre que muestre una operación aritmética como  $12*(SALARY + 100)$
- Es probable que desee que la salida le muestre un nombre que sea más fácil de comprender, un nombre más "descriptivo"
- Los alias de columna le permiten cambiar el nombre de las columnas en la salida



Alias de columna: cambia el nombre de una cabecera de columna para fines de visualización. El nombre de la columna original en la tabla permanece sin cambios.

# Alias

- Hay varias reglas al utilizar los alias de columna para dar formato a la salida
- Un alias de columna:
  - Cambia el nombre de una cabecera de columna
  - Es útil para realizar cálculos
  - Sigue de forma inmediata al nombre de columna
  - Puede tener la palabra clave opcional AS entre el nombre de columna y el alias
  - Necesita comillas dobles si el alias contiene espacios o caracteres especiales o es sensible a mayúsculas/minúsculas

# Uso de Alias de Columna

- La sintaxis de los alias es la siguiente:

```
SELECT * |column|expr [ AS alias], ....  
FROM      table;
```

- Ejemplos:

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

NAME	COMM
King	-
Kochhar	-
De Haan	-

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

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000



Academy

DP 1-3  
Anatomía de una Sentencia SQL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

26

Las comillas dobles son necesarias para alias con más de una palabra o cuando el formato del alias es distinto del nombre en mayúsculas predeterminadas (p. ej., Nombre del Departamento, departamento o dept.)

El primer ejemplo se utiliza la palabra clave AS, pero no tiene "comillas dobles", por lo que los alias se muestran con las MAYÚSCULAS predeterminadas.

El segundo ejemplo omite AS, pero como el alias es de más de una palabra, utiliza las comillas.

El uso de la palabra clave opcional AS facilita la lectura de las sentencias SQL.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Expresión aritmética
  - Operador aritmético
  - Cláusula
  - Columna
  - Alias de columna
  - Cláusula FROM
  - NULL

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Proyección
  - Cláusula SELECT
  - Selección
  - Sentencia SELECT
  - Sentencia
  - Cláusula WHERE
  - \* (Asterisco)

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Emparejar proyección y selección con sus capacidades correctas
  - Crear una sentencia SELECT básica
  - Utilizar la sintaxis correcta para mostrar todas las filas de una tabla
  - Utilizar la sintaxis correcta para seleccionar columnas específicas de una tabla, modificar la forma en que se muestran los datos y realizar cálculos utilizando expresiones aritméticas y operadores

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Formular consultas mediante la prioridad de operador correcta para mostrar los resultados deseados
  - Definir un valor nulo
  - Demostrar el efecto que los valores nulos crean en las expresiones aritméticas
  - Construir una consulta con un alias de columna

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

2-1

Columnas, Caracteres y Filas

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Aplicar el operador de concatenación para enlazar columnas a otras columnas, expresiones aritméticas o valores constantes para crear una expresión de caracteres
  - Utilizar los alias de columna para cambiar el nombre de las columnas en el resultado de la consulta
  - Introducir valores literales de tipo de carácter, número o fecha en una sentencia SELECT
  - Definir y utilizar DISTINCT para eliminar las filas duplicadas
  - Editar, ejecutar y guardar las sentencias SQL en Oracle Application Express



Concatenación: el hecho de estar enlazadas entre sí como en una cadena; unión en una serie enlazada.

# Objetivo

- En esta lección se abordan los siguientes objetivos:
  - Si está escribiendo un artículo sobre los Juegos Olímpicos, puede que desee saber cuántos países diferentes y cuántos atletas diferentes de cada país fueron representados
  - Tener que consultar listas y listas de nombres de participantes puede ser muy tedioso
  - Afortunadamente, con SQL, el trabajo puede tardar menos de un minuto
  - Además, puede formatear la salida para que se lea como una frase
  - Encontrará muy útiles estas funciones de SQL

## DESCRIBE

- Utilice el comando DESCRIBE (DESC) para mostrar la estructura de una tabla

```
DESCRIBE <table_name>;
```

- DESC devuelve el nombre de la tabla, los tipos de dato, las claves primarias y ajenas y las columnas con valores nulos, así como otros detalles del objeto que se tratarán más adelante en el curso
- En la siguiente diapositiva aparece un ejemplo de cómo utilizar el comando DESCRIBE

# DESCRIBE

```
DESC departments;
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENTS	DEPARTMENT_ID	NUMBER	-	4	0	1	-	-	-
	DEPARTMENT_NAME	VARCHAR2	30	-	-	-	-	-	-
	MANAGER_ID	NUMBER	-	6	0	-	-	-	-
	LOCATION_ID	NUMBER	-	4	0	-	-	-	-

En general, las palabras clave de SQL no se pueden abbreviar. DESCRIBE es un comando propiedad de Oracle, que es el motivo por el que se puede abbreviar a DESC.

## DESCRIBE

- Esta información es importante al insertar nuevas filas en una tabla, ya que debe conocer el tipo de dato que acepta cada columna y si la columna puede o no quedarse vacía



ORACLE  
Academy

DP 2-1  
Columnas, Caracteres y Filas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

7

# Operador de Concatenación

- Concatenación significa conectar o enlazar juntos en una serie
- El símbolo de concatenación es 2 barras verticales a veces denominadas "líneas verticales"
- Los valores a ambos lados del operador || se combinan para crear una sola columna de salida
- La sintaxis es la siguiente:

```
string1 || string2 || string_n
```

- Cuando los valores se concatenan, el valor resultante es una cadena de caracteres

# Operador de Concatenación

- En SQL, el operador de concatenación puede enlazar columnas a otras columnas, expresiones aritméticas o valores constantes para crear una expresión de caracteres
- El operador de concatenación se utiliza para crear salidas de texto legible
- En el siguiente ejemplo, el valor de department\_id se concatena al department\_name

```
SELECT department_id ||  
department_name  
FROM departments;
```



DP 2-1  
Columnas, Caracteres y Filas

DEPARTMENT_ID    DEPARTMENT_NAME
Administration
Marketing
Shipping
IT
...

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

9

# Operador de Concatenación

- En esta variante del ejemplo anterior, `||'''||` se utiliza para introducir un espacio entre `department_id` y `department_name`
- El carácter 'espacio' en comillas simples crea un espacio entre los valores de la columna

```
SELECT department_id ||' '|| department_name  
FROM departments;
```

DEPARTMENT_ID  ' '  DEPARTMENT_NAME
10 Administration
20 Marketing
50 Shipping
60 IT
...

# Concatenación y Alias de Columna

- Los alias de columna son útiles cuando se utiliza el operador de concatenación para que la línea SELECT por defecto no aparezca como la cabecera de la columna

```
SELECT department_id ||' '||
      department_name AS "Department Info"
FROM departments;
```

Department Info
10 Administration
20 Marketing
50 Shipping
60 IT
...

```
SELECT first_name ||' '||
      last_name AS "Employee Name"
FROM employees;
```

Employee Name
Ellen Abel
Curtis Davies
Lex De Haan

ORACLE

Academy

DP 2-1  
Columnas, Caracteres y Filas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

11

# Concatenación y Valores Literales

- Un valor literal es un valor de datos fijo, por ejemplo, un carácter, número o fecha
- A continuación, se muestran ejemplos de valores literales:
  - 'dollars'
  - 1000
  - 'January 1, 2009'
- Mediante la concatenación y los valores literales, puede crear una salida que parezca una frase o una sentencia

## Concatenación y Valores Literales

- Los valores literales se pueden incluir en la lista SELECT con el operador de concatenación
- Los caracteres y las fechas se deben estar entre comillas simples "
- Cada fila devuelta de una consulta con valores literales tendrá la misma cadena de caracteres en ella

Los literales de número no tienen que estar entre comillas simples.

# Concatenación y Valores Literales

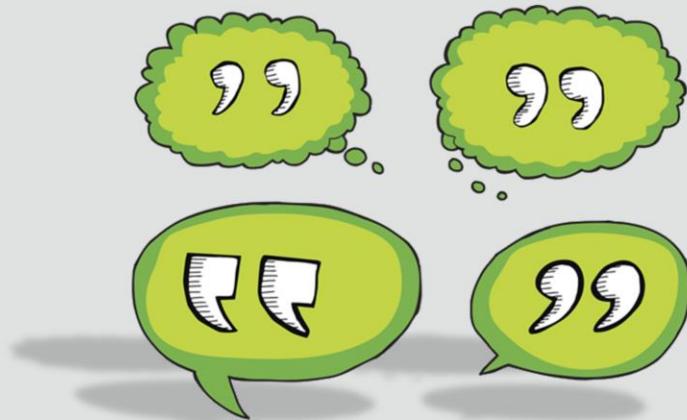
- En el siguiente ejemplo, King gana 24 000 dólares al mes
- 'has a monthly salary of ' y 'dollars.', son ejemplos de literales
- Si desea crear una sentencia SQL para generar salidas en este formato, se escribiría de la siguiente forma:

PAY
King has a monthly salary of 24000 dollars.
Kochhar has a monthly salary of 17000 dollars.
De Haan has a monthly salary of 17000 dollars.
Whalen has a monthly salary of 4400 dollars.
Higgins has a monthly salary of 12000 dollars.
Gietz has a monthly salary of 8300 dollars.
...

```
SELECT last_name || ' has a monthly
    salary of ' || salary || '
    dollars.' AS Pay
FROM employees;
```

# Concatenación y Valores Literales

- Tenga en cuenta el carácter de espacio tras la comilla de apertura y antes de la comilla de cierre
- ¿Qué sucede si elimina los espacios?



**ORACLE**  
Academy

DP 2-1  
Columnas, Caracteres y Filas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

15

# Concatenación y Valores Literales

- También puede incluir números como valores literales
- En el siguiente ejemplo, el número 1 se concatena a las cadenas, 'has a' y 'year salary of '

```
SELECT last_name ||' has a '|| 1 ||' year salary of '||  
      salary*12 || ' dollars.' AS Pay  
FROM employees;
```

PAY
King has a 1 year salary of 288000 dollars.
Kochhar has a 1 year salary of 204000 dollars.
De Haan has a 1 year salary of 204000 dollars.
Whalen has a 1 year salary of 52800 dollars.
Higgins has a 1 year salary of 144000 dollars.
...

## Uso de DISTINCT para Eliminar las Filas Duplicadas

- Muchas veces, deseará saber cuántas instancias únicas de algo existen
- Por ejemplo, si deseara una lista de todos los departamentos para los que hay empleados
- Podría escribir una consulta para seleccionar los valores department\_id de la tabla employees:

```
SELECT department_id  
FROM employees;
```

DEPARTMENT_ID
90
90
90
10
110
110
80
80
80
...

## Uso de DISTINCT para Eliminar las Filas Duplicadas

- Observe todas las filas duplicadas
- ¿Cómo se puede modificar la sentencia para que elimine las filas duplicadas?

```
SELECT department_id  
FROM employees;
```

DEPARTMENT_ID
90
90
90
10
110
110
80
80
80
...

## Uso de DISTINCT para Eliminar las Filas Duplicadas

- A menos que se indique lo contrario, la salida de una consulta SQL mostrará el resultado sin eliminar las filas duplicadas
- En SQL, se utiliza la palabra clave DISTINCT para eliminar las filas duplicadas

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

- El cualificador DISTINCT afecta a todas las columnas enumeradas y devuelve todas las combinaciones diferentes de las columnas de la cláusula SELECT
- La palabra clave DISTINCT debe aparecer directamente después de la palabra clave SELECT

DEPARTMENT_ID
-
90
20
110
80
50
10
60

## EXECUTE, SAVE y EDIT en Oracle Application Express

- Ahora que ha estado utilizando Oracle Application Express para crear y ejecutar sentencias sería bueno poder guardar dichas sentencias para más adelante a fin de poder ejecutarlas de nuevo o, quizás, editarlas ligeramente y, a continuación, guardar una nueva copia de la sentencia
- Oracle Application Express tiene utilidades para hacerlo
- El profesor le demostrará estas utilidades y podrá encontrar más información en Oracle Application Express User Guide

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - DESCRIBE
  - Operador de concatenación
  - Valores literales
  - DISTINCT

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Aplicar el operador de concatenación para enlazar columnas a otras columnas, expresiones aritméticas o valores constantes para crear una expresión de caracteres
  - Utilizar los alias de columna para cambiar el nombre de las columnas en el resultado de la consulta
  - Introducir valores literales de tipo de carácter, número o fecha en una sentencia SELECT
  - Definir y utilizar DISTINCT para eliminar las filas duplicadas
  - Editar, ejecutar y guardar las sentencias SQL en Oracle Application Express



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

2-2

## Limitación de Filas Seleccionadas

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Aplicar sintaxis SQL para restringir las filas devueltas de una consulta
  - Demostrar la aplicación de la sintaxis de la cláusula WHERE
  - Explicar el motivo por el que es importante, desde una perspectiva de negocio, poder limitar fácilmente los datos recuperados de una tabla
  - Crear y generar salidas mediante una consulta SQL que contenga cadenas de caracteres y valores de fecha
  - ¿Ha tenido alguna vez "sobrecarga de información"?



# Objetivo

- En esta lección se abordan los siguientes objetivos:
  - La televisión está encendida, su madre le pregunta cómo le ha ido en clase, suena el teléfono y el perro está ladrando
  - ¿No sería fantástico poder limitar la cantidad de información tiene que procesar a la vez?
  - En SQL, este es el trabajo de la cláusula WHERE
  - Es importante poder seleccionar la información que necesita ver de una tabla
  - Las tablas pueden tener millones de filas de datos, y es un desperdicio de recursos buscar y que devuelvan datos que no necesite o deseé

## Sentencia SELECT

- SELECT se utiliza para recuperar información de la base de datos
- Una sentencia SELECT debe incluir como mínimo una cláusula SELECT y una cláusula FROM
- La cláusula WHERE es opcional

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

Sintaxis general de la cláusula SELECT: el uso de [ ] es opcional. El | significa "o".

## Cláusula WHERE

- Al recuperar datos de la base de datos, puede que necesite limitar las filas de datos que se muestran
- Esto se consigue con la cláusula WHERE
- Una cláusula WHERE contiene una condición que se debe cumplir e, inmediatamente después, le sigue la cláusula FROM en una sentencia SQL
- La sintaxis de la cláusula WHERE es:

```
WHERE column_name comparison_condition comparison_value
```

- Nota: No se puede utilizar un alias en la cláusula WHERE

## Cláusula WHERE

- Examine la siguiente sentencia SQL de la base de datos Employees:

```
SELECT employee_id, first_name,  
last_name  
FROM employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan

- Mediante la adición de una cláusula WHERE, las filas se limitan a aquellas filas en las que el valor de employee\_id es 101

```
SELECT employee_id, first_name,  
last_name  
FROM employees  
WHERE employee_id = 101;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
101	Neena	Kochhar

ORACLE

Academy

DP 2-2  
Limitación de Filas Seleccionadas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

7

## Operadores de Comparación en la Cláusula WHERE

- Como ha visto en la diapositiva anterior, el signo = se puede utilizar en la cláusula WHERE
- Además del operador "igual que" ( = ), se pueden utilizar otros operadores de comparación para comparar una expresión con otra:
  - = igual que
  - > mayor que
  - >= mayor o igual que
  - < menor que
  - <= menor o igual que
  - <> no es igual que (o != o ^=)

## Operadores de Comparación en la Cláusula WHERE

- En el siguiente ejemplo, la columna department\_id se utiliza en la cláusula WHERE, con el operador de comparación =
- Se devuelven todos los empleados con un department\_id de 90

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

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90

## Cadenas de Caracteres y de Fecha en la Cláusula WHERE

- Las fechas y cadenas de caracteres de la cláusula WHERE se deben incluir entre comillas simples (' ')
- Sin embargo, los números no se deben incluir entre comillas simples

## Cadenas de Caracteres y de Fecha en la Cláusula WHERE

- Observe el siguiente ejemplo de la base de datos Employees
- La cláusula WHERE contiene una cadena y se incluye entre comillas simples

```
SELECT first_name, last_name  
FROM employees  
WHERE last_name = 'Taylor';
```

## Cadenas de Caracteres y de Fecha en la Cláusula WHERE

- Qué cree que pasará si la cláusula WHERE se escribiera así:

```
WHERE last_name = 'jones';
```

- Todas las búsquedas de caracteres son sensibles a mayúsculas/minúsculas
- Debido a que la tabla Employees almacena todos los apellidos con el uso de mayúsculas adecuado, no se devolverán filas en este ejemplo

## Cadenas de Caracteres y de Fecha en la Cláusula WHERE

- Es importante recordar este aspecto
- En otra lección, aprenderá a utilizar otras palabras clave SQL como UPPER, LOWER e INITCAP que facilitarán evitar los errores de la sensibilidad a mayúsculas/minúsculas

## Operadores de Comparación en la Cláusula WHERE

- Los operadores de comparación se pueden utilizar en todas las formas siguientes en la cláusula WHERE:

```
WHERE hire_date < '01-Jan-2000'
```

```
WHERE salary >= 6000
```

```
WHERE job_id = 'IT_PROG'
```

- En el siguiente ejemplo de la base de datos Employees, ¿qué filas se seleccionarán?
- ¿Se incluirán los salarios de 3000 en el juego de resultados?

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



DP 2-2  
Limitación de Filas Seleccionadas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

14

Como las cadenas de caracteres, los valores de fecha deben estar entre comillas simples.

Los formatos aceptables para los valores de fecha son:

Utilizar un guión o una barra inclinada adelante para separar día, mes y año, p. ej., '20-MAR-1999', '20/MAR/1999'.

Utilizar 3 caracteres para el mes, ya sea en mayúsculas, minúsculas o con mayúscula inicial, p. ej., '20/MAR/1999', '20/mar/1999', '20/Mar/1999'.

Utilizar valores año de 4 dígitos.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Cláusula WHERE
  - Operadores de comparación

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Aplicar sintaxis SQL para restringir las filas devueltas de una consulta
  - Demostrar la aplicación de la sintaxis de la cláusula WHERE
  - Explicar el motivo por el que es importante, desde una perspectiva de negocio, poder limitar fácilmente los datos recuperados de una tabla
  - Crear y generar salidas mediante una consulta SQL que contenga cadenas de caracteres y valores de fecha



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

2-3

## Operadores de Comparación

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Aplicar el operador de comparación adecuado para devolver un resultado deseado
  - Demostrar un uso adecuado de las condiciones BETWEEN, IN y LIKE para devolver un resultado deseado
  - Distinguir entre cero y NULL, el cual significa que no está disponible, que está sin asignar, que es desconocido o que no es aplicable
  - Explicar el uso de las condiciones de comparación y NULL

# Objetivo

- Las comparaciones se utilizan en las conversaciones diarias sin realmente pensar en ello
  - Las comparaciones se utilizan en las conversaciones diarias sin realmente pensar en ello
  - Las comparaciones se utilizan en las conversaciones diarias sin realmente pensar en ello
  - Las comparaciones se utilizan en las conversaciones diarias sin realmente pensar en ello



# Objetivo

- La necesidad de expresar estos tipos de comparaciones también existe en SQL
- Las condiciones de comparación se utilizan para buscar datos en una tabla que cumplan determinadas condiciones
- Poder formular una cláusula SELECT para devolver datos específicos es una función potente de SQL

# Operadores de Comparación

- Ya está familiarizado con los operadores de comparación, como igual que (=), menor que (<) y mayor que (>)
- SQL tiene otros operadores que agregan funcionalidad para recuperar juegos específicos de los datos
- Son los siguientes:
  - BETWEEN...AND
  - IN
  - LIKE

## BETWEEN...AND

- El operador BETWEEN...AND se utiliza para seleccionar y mostrar las filas según un rango de valores
- Cuando se utiliza con la cláusula WHERE, la condición BETWEEN...AND devolverá un rango de valores entre los límites inferior y superior, incluyendo estos

Incluyendo significa que también se devolverán los valores coincidentes con los límites inferior y superior.

## BETWEEN...AND

- Tenga en cuenta que en el ejemplo de la base de datos Employees, los valores devueltos incluyen el valor del límite inferior y el valor del límite superior
- Los valores especificados con la condición BETWEEN se incluyen
- Tenga en cuenta también que el valor de límite inferior se debe enumerar en primer lugar

```
SELECT last_name, salary  
FROM employees  
WHERE salary BETWEEN 9000 AND 11000;
```

- Tenga en cuenta también que el valor de límite inferior se debe enumerar en primer lugar

LAST_NAME	SALARY
Zlotkey	10500
Abel	11000
Hunold	9000

## BETWEEN...AND

- Usar BETWEEN...AND es lo mismo que utilizar la siguiente expresión:

```
WHERE salary >= 9000 AND salary <=11000;
```

- De hecho, no hay ningún beneficio en el rendimiento al usar una expresión u otra
- Para mayor simplicidad al leer código, utilizamos BETWEEN...AND

# IN

- La condición IN también se conoce como la "condición de miembro"
- Se utiliza para probar si un valor está en un juego de valores especificado
- Por ejemplo, IN se podría utilizar para identificar a los alumnos cuyos números de identificación sean 2349, 7354 o 4333, o a las personas cuyos código de llamada por teléfono internacional sean 1735, 82 o 10

```
SELECT city, state_province,  
country_id  
FROM locations  
WHERE country_id IN('UK', 'CA');
```

CITY	STATE_PROVINCE	COUNTRY_ID
Toronto	Ontario	CA
Oxford	Oxford	UK

ORACLE

Academy

DP 2-3  
Operadores de Comparación

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

10

## IN

- En este ejemplo, la cláusula WHERE también se puede escribir como un juego de condiciones OR:

```
SELECT city, state_province, country_id  
FROM locations  
WHERE country_id IN('UK', 'CA');  
...  
WHERE country_id = 'UK' OR country_id = 'CA';
```

- Al igual que con BETWEEN...AND, la condición IN se puede escribir con sintaxis igual de eficaz

## LIKE

- ¿Alguna vez ha ido de compras buscando algo que ha visto en una revista o en televisión, pero no estaba seguro del artículo exacto?
- Con las búsquedas en las bases de datos pasa más o menos lo mismo
- Un administrador puede saber que el apellido de un empleado empieza por "S", pero no saber el nombre completo del empleado
- Afortunadamente, en SQL, la condición LIKE permite seleccionar las filas que coincidan con caracteres, fechas o patrones de números

## LIKE

- Dos símbolos, el (%) y el guión bajo (\_), llamados caracteres comodín, se pueden utilizar para crear una cadena de búsqueda
- El símbolo del porcentaje (%) se utiliza para representar cualquier secuencia de cero o más caracteres

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

LAST_NAME
Kochhar
Lorentz
Mourgos

## LIKE

- El símbolo del carácter de subrayado (\_) se utiliza para representar un único carácter
- En el ejemplo que se muestra a continuación, se devolverán todos los empleados cuyos apellidos empiecen por cualquier letra seguida de una "o" y, a continuación, seguidos de cualquier otro número de letras

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

LAST_NAME
Kochhar
Lorentz
Mourgos

## LIKE

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

- ¿Cuáles de los siguientes apellidos se podría haber devuelto con la consulta anterior?
  - 1. Sommersmith
  - 2. Oog
  - 3. Fong
  - 4. Mo

Respuesta: Si ha dicho 1, 2, 3, y 4, ¡es lo correcto!

## LIKE

- Una opción adicional que es importante:
  - Cuando necesite tener una coincidencia exacta para una cadena que tenga un carácter % o \_, deberá indicar que dichos caracteres no son comodines sino parte del elemento que está buscando

## LIKE

- La opción ESCAPE se puede utilizar para indicar que los caracteres \_ o % son parte del nombre, no valores comodín
- Por ejemplo, si deseamos recuperar el JOB\_ID de un empleado de la tabla employees que contenga el patrón \_R, tendríamos que utilizar un carácter de escape para indicar que estamos buscando un guión bajo, y no solo cualquier carácter

```
SELECT last_name, job_id  
FROM EMPLOYEES  
WHERE job_id LIKE '%\_R%' ESCAPE '\';
```

- En este ejemplo, se utiliza la barra invertida '\' como carácter de escape, pero se puede utilizar cualquier carácter

# LIKE

- Sin la opción ESCAPE, se devolverían todos los empleados que tuvieran una R en sus JOB\_ID

```
SELECT last_name, job_id  
FROM EMPLOYEES  
WHERE job_id LIKE '%_R%'
```

LAST_NAME	JOB_ID
Abel	SA_REP
Davies	ST_CLERK
Ernst	IT_PROG
Fay	MK_REP
Fay	MK_REP
Grant	SA_REP
Higgins	AC_MGR
Hunold	IT_PROG
King	AD_PRES
Lorentz	IT_PROG
Matos	ST_CLERK
Rajs	ST_CLERK
Taylor	SA_REP
Vargas	ST_CLERK

## IS NULL, IS NOT NULL

- ¿Se acuerda de NULL?
- Es el valor que no está disponible, sin asignar, desconocido o que no es aplicable
- Poder probar el valor NULL es a menudo aconsejable
- Es posible que desee saber todas las fechas en junio en las que, ahora mismo, no tenga programado un concierto
- Es posible que desee saber todos los clientes que no tengan números de teléfono registrados en su base de datos

## IS NULL, IS NOT NULL

- La condición IS NULL prueba los datos que no están disponibles, sin asignar, o datos desconocidos
- IS NOT NULL prueba los datos disponibles en la base de datos
- En el ejemplo de la siguiente diapositiva, la cláusula WHERE se escribe para recuperar todos los apellidos de aquellos empleados que no tengan un jefe

## IS NULL, IS NOT NULL

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

LAST_NAME	MANAGER_ID
King	

- El empleado King es el Presidente de la compañía, por lo que no tiene jefe

```
SELECT last_name, commission_pct  
FROM employees  
WHERE commission_pct IS NOT NULL;
```

LAST_NAME	COMMISSION_PCT
Zlotkey	.2
Abel	.3
Taylor	.2
Grant	.15

- IS NOT NULL devuelve las filas que tienen un valor en la columna commission\_pct

Al comparar los valores NULL, NO PUEDE utilizar los operadores con el signo igual (=) o distinto de (! =). La consulta se ejecutará, pero no devolverá datos, ya que el valor real de NULL es desconocido, por lo que ¿cómo podemos comprobar si algo es igual o distinto a un valor que desconocemos?

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - BETWEEN...AND
  - IN
  - LIKE
  - IS NULL
  - IS NOT NULL

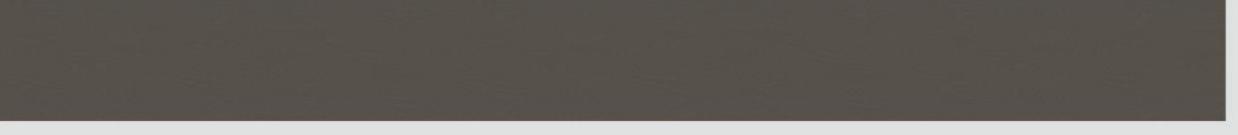
# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Aplicar el operador de comparación adecuado para devolver un resultado deseado
  - Demostrar un uso adecuado de las condiciones BETWEEN, IN y LIKE para devolver un resultado deseado
  - Distinguir entre cero y NULL, este último significa que no está disponible, sin asignar, desconocido o que no es aplicable
  - Explicar el uso de las condiciones de comparación y NULL



# ORACLE

## Academy



# **ORACLE**

## Academy

# Database Programming with SQL

3-1

## Comparaciones Lógicas y Reglas de Prioridad

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Evaluar las comparaciones lógicas para restringir las filas devueltas en función de dos o más condiciones
  - Aplicar las reglas de prioridad para determinar el orden en el que se evalúan y calculan las expresiones

## Objetivo

- No hay muchas cosas en la vida que dependan de una sola condición
- Por ejemplo, si desea ir a la universidad, probablemente necesite buenas notas y dinero para pagarla
- Si tiene dinero adicional, puede guardarlo o gastarlo
- Si desea ir al cine, es posible que no desee ir este fin de semana y puede que no desee sentarse en las primeras 10 filas de la sala

## Objetivo

- En SQL, por lo general, es aconsejable poder restringir las filas devueltas en una consulta en función de dos o más condiciones
- Como jefe de un negocio de comida rápida, puede que necesite conocer los nombres de su los empleados que sean cocineros o los que toman los pedidos
- No necesita ni desea una lista de todo el personal, solo quiere un subjuego de ella
- Los operadores condicionales como AND, NOT y OR facilitan el uso de este tipo de solicitudes

# Condiciones Lógicas

- Las condiciones lógicas combinan el resultado de dos condiciones de componentes para producir un único resultado según dichas condiciones
- Por ejemplo, para asistir a un concierto de rock, debe adquirir una entrada y el transporte para llegar allí
- Si se cumplen las dos condiciones, puede ir al concierto
- ¿Qué hacer si no puede obtener transporte, puede ir?

## Condiciones Lógicas

- Otra condición lógica combina dos condiciones de componentes con OR
- Todos los empleados recibirán un aumento si tienen un registro de asistencia perfecto o cumplen con su cuota de ventas mensual
- Si un empleado cumple cualquiera de estas dos condiciones, el empleado obtiene un aumento

# Operadores Lógicos

- Un operador lógico combina los resultados de dos o más condiciones para producir un único resultado
- Se devuelve un resultado SOLO SI el resultado global de la condición es verdadero
- AND: devuelve TRUE (verdadero) si ambas condiciones son verdaderas
- OR: devuelve TRUE (verdadero) si cualquier condición es verdadera
- NOT: devuelve TRUE (verdadero) si la condición es falsa

## Operador AND

- En la siguiente consulta, los resultados devueltos serán las filas que cumplan ambas condiciones especificadas en la cláusula WHERE

```
SELECT last_name, department_id, salary  
FROM employees  
WHERE department_id > 50 AND salary > 12000;
```

LAST_NAME	DEPARTMENT_ID	SALARY
King	90	24000
Kochhar	90	17000
De Haan	90	17000

# Operador AND

- Otro ejemplo del uso de AND en la cláusula WHERE

```
SELECT last_name, hire_date, job_id  
FROM employees  
WHERE hire_date > '01-Jan-1998' AND job_id LIKE 'SA%';
```

LAST_NAME	HIRE_DATE	JOB_ID
Zlotkey	29-Jan-2000	SA_MAN
Taylor	24-Mar-1998	SA_REP
Grant	24-May-1999	SA_REP

Recordatorio: las fechas se deben introducir entre comillas simples y el formato por defecto es dd/Mes/aaaa

## Operador OR

- Si la cláusula WHERE utiliza la condición OR, los resultados devueltos de una consulta serán las filas que cumplan una de las condiciones de OR
- En otras palabras, todas las filas devueltas tienen un location\_id de 2500 o tienen un manager\_id igual a 124

```
SELECT department_name, manager_id, location_id  
FROM departments  
WHERE location_id = 2500 OR manager_id=124;
```

DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
Shipping	124	1500
Sales	149	2500

## Operador NOT

- El operador NOT devolverá las filas que no cumplan con la condición de la cláusula WHERE

```
SELECT department_name, location_id  
FROM departments  
WHERE location_id NOT IN (1700,1800);
```

DEPARTMENT_NAME	LOCATION_ID
Shipping	1500
IT	1400
Sales	2500

## Reglas de Prioridad o Qué Ocurre en Primer Lugar

- Considere la siguiente sentencia SELECT.
- ¿En qué orden se evalúan las expresiones calculadas?

```
SELECT last_name||' '||salary*1.05 AS "Employee Raise"  
FROM employees  
WHERE department_id IN(50,80) AND first_name LIKE 'C%'  
    OR last_name LIKE '%s%';
```

- Afortunadamente, cuando las cosas se complican de esta manera, SQL cuenta con algunas reglas básicas fáciles de seguir

## Reglas de Prioridad o Qué Ocurre en Primer Lugar

- Tenga en cuenta que el operador AND se evalúa antes que el operador OR
- Esto significa que, para que el ejemplo en la diapositiva anterior, si no se cumple alguna de las condiciones de la sentencia AND, se utilizará el operador OR para seleccionar las filas
- Es importante recordar este concepto

ORDER	OPERATOR
1	Arithmetic + - * /
2	Concatenation
3	Comparison <, <=, >, >=, <>
4	IS (NOT) NULL, LIKE, (NOT) IN
5	(NOT) BETWEEN
6	NOT
7	AND
8	OR

## Reglas de Prioridad o Qué Ocurre en Primer Lugar

- En primer lugar, se evalúa la condición AND, por lo que se devolverán todos los empleados que trabajen en el departamento 80 o 50, y tengan un nombre que empiece por "C"
- La cláusula OR se evalúa a continuación y se devuelven los empleados cuyo apellido contenga "s"

```
SELECT last_name||' '||salary*1.05
      AS "Employee Raise", department_id,
first_name
FROM employees
WHERE department_id IN(50,80)
      AND first_name LIKE 'C%'
      OR last_name LIKE '%s%';
```

ORACLE

Academy

DP 3-1  
Comparaciones Lógicas y Reglas de Prioridad

Employee Raise	DEPARTMENT_ID	FIRST_NAME
Higgins 12600	110	Shelley
Mourgos 6090	50	Kevin
Rajs 3675	50	Trenna
Davies 3255	50	Curtis
Matos 2730	50	Randall
Vargas 2625	50	Peter
Ernst 6300	60	Bruce
Hartstein 13650	20	Michael

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

15

El empleado Curtis Davis está en el departamento 50 y su nombre empieza por "C", por lo que lo devuelve la cláusula AND.

La cláusula OR devuelve, a continuación, cualquier empleado cuyo apellido contenga la letra "s".

Si alguna de las condiciones no se cumplen en la sentencia AND, la condición OR tendría que evaluar si son verdaderas para devolver alguna fila.

## Reglas de Prioridad o Qué Ocurre en Primer Lugar

- En este ejemplo, el orden de los operadores OR y AND se ha invertido con respecto a la diapositiva anterior

```
SELECT last_name||' '||salary*1.05 AS "Employee Raise",
       department_id,
       first_name
  FROM employees
 WHERE department_id IN(50,80)
   OR first_name LIKE 'C%'
   AND last_name LIKE '%s%';
```

El orden de las operaciones es:

1. first\_name comienza con una "C" y last\_name contiene una "s". Deben cumplirse ambas condiciones para devolverlas.
2. Se devolverá cualquier instancia de los empleados de los departamentos 50 y 80.

## Reglas de Prioridad o Qué Ocurre en Primer Lugar

- Agregar paréntesis cambia la forma en la que se evalúa la cláusula WHERE y las filas devueltas

```
SELECT last_name||' '| salary*1.05 AS "Employee Raise",
       department_id,
       first_name
  FROM employees
 WHERE (department_id IN(50,80) OR first_name LIKE 'C%')
   AND last_name LIKE '%s%';
```

El orden de las operaciones es:

1. Los valores de los paréntesis están seleccionados.
2. Se devolverán todas las instancias de los valores entre los paréntesis que también contengan la letra "s" en su last\_name (apellido).

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - AND
  - OR
  - NOT
  - Reglas de prioridad

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Evaluar las comparaciones lógicas para restringir las filas devueltas en función de dos o más condiciones
  - Aplicar las reglas de prioridad para determinar el orden en el que se evalúan y calculan las expresiones

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

3-2

## Ordenación de Filas

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear una consulta para ordenar un juego de resultados en orden ascendente o descendente
  - Establecer el orden en el que se evalúan y calculan las expresiones en función de las reglas de prioridad
  - Crear una consulta para ordenar un juego de resultados con un alias de columna
  - Crear una consulta para ordenar un juego de resultados para una o varias columnas

# Objetivo

- Por naturaleza, la mayoría de nosotros necesitamos orden en nuestras vidas
- Imagine que cada vez que cenara, tuviera que buscar en cada cajón o armario de la cocina para buscar un cuchillo y un tenedor
- Ordenar, agrupar y clasificar facilita la búsqueda de las cosas
- Los biólogos agrupan a los animales en filos, los astrónomos ordenan el brillo de las estrellas por su magnitud y los programadores de Java organizan el código en clases

# Objetivo

- La vida diaria está ordenada en muchas situaciones:
  - Los libros en la biblioteca
  - Las estanterías del supermercado
  - Los documentos almacenados en archivadores
- Poder ordenar resultados es una función útil en SQL y permite a los programadores mostrar información de muchas formas diferentes
- Para el diseño de bases de datos, las funciones de negocio se ordenan por entidades y atributos; en la información de la base de datos, SQL utiliza la cláusula ORDER BY

## Cláusula ORDER BY

- La información ordenada en orden ascendente nos es conocida
- Es lo que hace que consultar un número en una agenda, buscar una palabra en el diccionario, o localizar una casa por su dirección sea relativamente fácil
- SQL utiliza la cláusula ORDER BY para ordenar datos.
- La cláusula ORDER BY puede especificar varias formas en las que ordenar las filas devueltas en una consulta

## Cláusula ORDER BY

- El orden por defecto es el orden ascendente
- Se muestran los valores numéricos del más bajo al más alto
- Los valores de fecha se muestran con el valor más reciente en primer lugar
- Los valores de caracteres se muestran en orden alfabético
- Los valores nulos se muestran al final para las secuencias ascendentes y al principio para las secuencias descendentes
- NULLS FIRST especifica que los valores nulos se deben devolver antes los valores no nulos
- NULLS LAST especifica que los valores nulos se deben devolver después los valores no nulos



DP 3-2  
Ordenación de Filas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

7

Si utiliza NULLS FIRST o NULLS LAST, deben estar al final de la cláusula ORDER BY, después de los nombres de columna. Por ejemplo:

```
SELECT last_name, hire_date, department_id  
FROM employees  
ORDER BY department_id NULLS LAST;
```

## Cláusula ORDER BY

- El siguiente ejemplo de empleados utiliza la cláusula ORDER BY para ordenar hire\_date en orden ascendente (valor por defecto)
- Nota: La cláusula ORDER BY debe ser la última cláusula de la sentencia SQL

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

LAST_NAME	HIRE_DATE
King	17-Jun-1987
Whalen	17-Sep-1987
Kochhar	21-Sep-1989
Hunold	03-Jan-1990
Ernst	21-May-1991
De Haan	13-Jan-1993
Gietz	07-Jun-1994
Higgins	07-Jun-1994
Rajs	17-Oct-1995
Hartstein	17-Feb-1996

# Ordenación en Orden Descendente

- Puede invertir el orden por defecto en la cláusula ORDER BY para que aparezcan en orden descendente especificando la palabra clave DESC después del nombre de columna en la cláusula ORDER BY

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

LAST_NAME	HIRE_DATE
Zlotkey	29-Jan-2000
Mourgos	16-Nov-1999
Grant	24-May-1999
Lorentz	07-Feb-1999
Vargas	09-Jul-1998
Taylor	24-Mar-1998
Matos	15-Mar-1998
Fay	17-Aug-1997
Davies	29-Jan-1997
Abel	11-May-1996



Si utiliza ASC o DESC en la cláusula ORDER BY influirá la colocación de los valores nulos: los valores nulos se muestran los últimos en orden ascendente y los primeros en orden descendente.

Además, puede utilizar NULLS FIRST para especificar que los valores nulos se deben devolver antes de los valores no nulos. NULLS LAST especifica que los valores nulos se deben devolver después los valores no nulos.

# Uso de Alias de Columna

- Puede ordenar los datos mediante un alias de columna
- El alias utilizado en la sentencia SELECT es al que se hace referencia en la cláusula ORDER BY

```
SELECT last_name, hire_date  
      AS "Date Started"  
  FROM employees  
 ORDER BY "Date Started";
```

LAST_NAME	Date Started
King	17-Jun-1987
Whalen	17-Sep-1987
Kochhar	21-Sep-1989
Hunold	03-Jan-1990
Ernst	21-May-1991
De Haan	13-Jan-1993
Gietz	07-Jun-1994
Higgins	07-Jun-1994
Rajs	17-Oct-1995
Hartstein	17-Feb-1996

## Ordenación con Otras Columnas

- También es posible utilizar la cláusula ORDER BY para ordenar la salida según una columna que no aparezca en la cláusula SELECT
- En el siguiente ejemplo, los datos se ordenan según la columna last\_name, incluso aunque esta columna no aparece en la sentencia SELECT

```
SELECT employee_id,  
first_name  
FROM employees  
WHERE employee_id < 105  
ORDER BY last_name;
```

EMPLOYEE_ID	FIRST_NAME
102	Lex
104	Bruce
103	Alexander
100	Steven
101	Neena



Es difícil verificar los resultados cuando se ordenan según una columna que no está seleccionada. En el mundo real, debería ejecutar la consulta seleccionando la columna last\_name hasta que estuviera seguro de que obtenía los datos correctos. Después, podría eliminar dicha columna en la sentencia SELECT.

# Orden de Ejecución

- El orden de ejecución de una sentencia SELECT es el siguiente:
  - Cláusula FROM: busca la tabla que contiene los datos
  - Cláusula WHERE: restringe las filas que se van a devolver
  - Cláusula SELECT: selecciona en el conjunto de datos reducido las columnas solicitadas
  - Cláusula ORDER BY: ordena el conjunto de resultados



## Ordenación con Varias Columnas

- También se puede ordenar los resultados de la consulta por más de una columna
- De hecho, no hay ningún límite en el número de columnas que se pueden agregar a la cláusula ORDER BY



## Ordenación con Varias Columnas

- A continuación se muestra un ejemplo de ordenación con varias columnas
- Los empleados se ordenan primero por número de departamento (del más bajo al más alto), a continuación, se muestran los apellidos en orden alfabético (A-Z)

```
SELECT department_id, last_name
FROM employees
WHERE department_id <= 50
ORDER BY department_id,
last_name;
```

DEPARTMENT_ID	LAST_NAME
10	Whalen
20	Fay
20	Hartstein
50	Davies
50	Matos
50	Mourgos
50	Rajs
50	Vargas

## Ordenación con Varias Columnas

- Para crear una cláusula ORDER BY a fin de ordenar por varias columnas, especifique las columnas que se van a devolver y separe los nombres de columna mediante comas
- Si desea invertir el orden de ordenación de una columna, agregue DESC después del nombre

```
SELECT department_id,  
last_name  
FROM employees  
WHERE department_id <= 50  
ORDER BY department_id DESC,  
last_name;
```

DEPARTMENT_ID	LAST_NAME
50	Davies
50	Matos
50	Mourgos
50	Rajs
50	Vargas
20	Fay
20	Hartstein
10	Whalen

ORACLE

Academy

DP 3-2  
Ordenación de Filas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

15

Se ha invertido el orden para el valor de department\_id (de la diapositiva anterior) con DESC, por lo que ahora se muestran del más alto al más bajo, el orden del apellido sigue siendo alfabético, de la A a la Z.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Cláusula ORDER BY
  - ASCENDENTE
  - DESCENDENTE
  - Orden de Ejecución

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear una consulta para ordenar un juego de resultados en orden ascendente o descendente
  - Crear una consulta para ordenar un juego de resultados con un alias de columna
  - Crear una consulta para ordenar un juego de resultados para una o varias columnas

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

3-3

## Introducción a las Funciones

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Identificar las aplicaciones adecuadas de funciones de fila única en las sentencias de consulta
  - Clasificar una función como función de una sola fila o de varias filas
  - Diferenciar entre funciones de una sola fila y funciones de varias filas y los resultados devueltos por cada una



# Objetivo

- Al introducir dinero en una máquina de bebidas, ocurre algo entre el momento en el que se deposita el dinero y en el que se dispensa su bebida favorita
- La máquina procesa la transacción internamente
- Su dinero es la entrada y la bebida es la salida
- La máquina realiza una función
- La máquina:
  - Cuenta su dinero
  - Se asegura de que ha seleccionado su elección
  - Devuelve cambio, si es necesario

# Objetivo

- En SQL, existen muchos tipos de funciones que se utilizan para transformar la entrada de una forma en la salida de otra forma
- Estas funciones se utilizan para manipular los valores de datos
- Las funciones son pequeños programas que realizan una acción en un valor o columna y producen algo distinto como salida

¿Cómo puede averiguar si el nombre y apellido de los empleados de la tabla employees se almacena en mayúsculas, minúsculas o mayúsculas y minúsculas? Ejecute una sentencia SELECT para consultar la salida.

```
SELECT first_name, last_name  
FROM employees;
```

# Funciones

- Las funciones tienen tanto entrada como salida. La entrada en una función se conoce como argumento

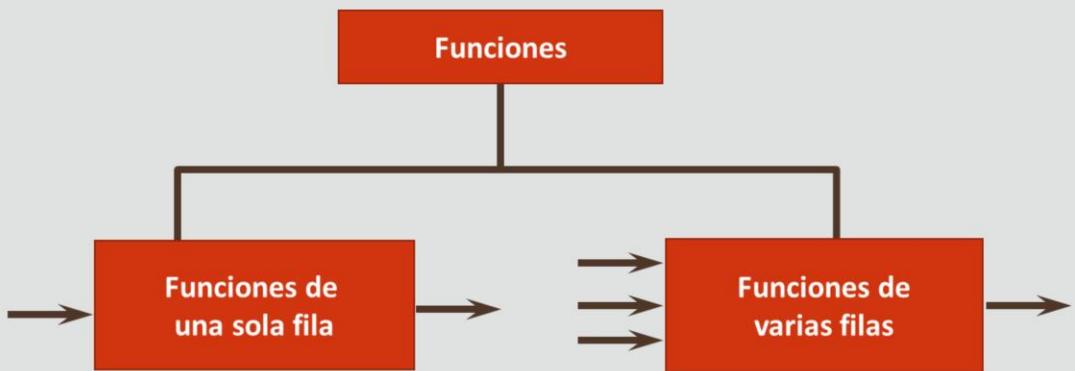


- En el ejemplo de la máquina de bebidas, la entrada es el dinero y la salida es una bebida



# Funciones

- Oracle cuenta con dos tipos de funciones diferentes:
  - De una sola fila
  - De varias filas



Las funciones de una sola fila se tratan en mayor profundidad en la siguiente sección.

Las funciones de varias filas se tratan más adelante en el curso.

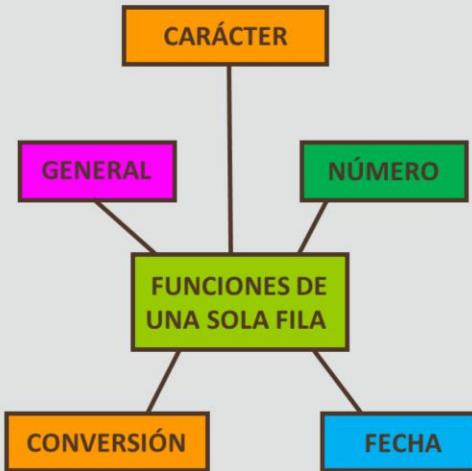
## Funciones de una Sola Fila frente a Funciones de Varias Filas

- Las funciones de una sola fila funcionan solo en filas únicas y devuelven un resultado por fila
- Existen distintos tipos de funciones de una sola fila, como funciones de carácter, número, fecha y conversión
- Las funciones de varias filas pueden manipular grupos de filas para proporcionar un resultado por grupo de filas
- Estas funciones también se conocen como funciones de grupo



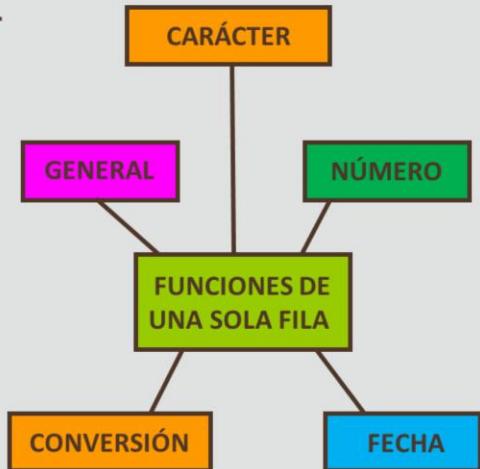
# Funciones de una Sola Fila

- En SQL, las funciones de una sola fila se pueden utilizar para:
  - Realizar cálculos como redondear números para una determinada posición decimal
  - Modificar elementos de datos individuales como la conversión de valores de caracteres de mayúsculas a minúsculas



# Funciones de una Sola Fila

- Dar formato a fechas y números para su visualización como, por ejemplo, convertir el formato de fecha de una base de datos numérico interno en un formato estándar
- Convertir los tipos de dato de columna, como la conversión de una cadena de caracteres en un número o fecha



# Funciones de una Sola Fila

- Las funciones de una sola fila aceptan uno o más argumentos y devolverán un único resultado por fila
- Por lo tanto, si se aplica la función de una sola fila a 12 filas, obtendrá 12 resultados de la función de una sola fila
- En resumen, las funciones de una sola fila hacen lo siguiente:
  - Manipular elementos de datos
  - Aceptar argumentos y devolver un valor
  - Actuar en cada fila devuelta
  - Devolver un resultado por fila
  - Posibilidad de modificar el tipo de dato
  - Posibilidad de anidamiento

Un argumento puede ser uno de los siguientes elementos:

Constante proporcionada por el usuario

Valor de variable

Nombre de la columna

Expresión

Entre las funciones adicionales de una sola fila se incluyen:

Posibilidad de devolver un valor de datos de un tipo diferente al que se hace referencia

Posibilidad de esperar uno o más argumentos

Se pueden utilizar en cláusulas SELECT, WHERE y ORDER BY.

## Funciones de Varias Filas

- Las funciones de varias filas (o de grupo) toman muchas filas como entrada, y devuelven un valor único como salida
- La entrada de filas puede ser toda la tabla o la tabla dividida en grupos más pequeños
- Los ejemplos de las funciones de varias filas (de grupo) incluyen:
  - MAX: busca el valor más alto en un grupo de filas
  - MIN: busca el valor más bajo en un grupo de filas
  - AVG: busca el valor medio de un grupo de filas

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Función de una sola fila
  - Función de varias filas

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Identificar las aplicaciones adecuadas de funciones de fila única en las sentencias de consulta
  - Clasificar una función como función de una sola fila o de varias filas
  - Diferenciar entre funciones de una sola fila y funciones de varias filas y los resultados devueltos por cada una



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

4-1

## Manipulación de Mayúsculas/Minúsculas y de Caracteres

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Seleccionar y aplicar funciones de una sola fila que realicen conversión de mayúsculas/minúsculas y/o la manipulación de caracteres
  - Seleccionar y aplicar las funciones de manipulación de mayúsculas/minúsculas de caracteres LOWER, UPPER e INTCAP en una consulta SQL
  - Seleccionar y aplicar las funciones de manipulación de caracteres CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM y REPLACE en una consulta SQL
  - Escribir consultas flexibles usando
  - variables de sustitución



# Objetivo

- Poder cambiar la forma en la que se presentan los datos es importante cuando se trabaja con datos de una base de datos
- La mayor parte del tiempo en SQL, necesitamos cambiar la forma en la que aparecen los datos según los requisitos de la tarea que estamos intentando lograr
- En esta lección, aprenderá diversas formas con las que transformar los datos para que se ajusten a una situación concreta

## Tabla DUAL

- La tabla DUAL tiene una fila denominada "X" y una columna denominada "DUMMY"

DUMMY
X

- La tabla DUAL se utiliza para crear sentencias SELECT y ejecutar funciones que no estén directamente relacionadas con una tabla de base de datos concreta
- Las consultas que utilizan la tabla DUAL devuelven una fila como resultado. DUAL puede ser útil para realizar cálculos y también para evaluar expresiones que no derivan de una tabla

## Tabla DUAL

- DUAL se utilizará para obtener muchas de las funciones de una sola fila
- En este ejemplo, la tabla DUAL se utiliza para ejecutar una sentencia SELECT que contenga un cálculo
- Como puede ver, la sentencia SELECT devuelve un valor que no existe en la tabla DUAL
- El valor devuelto es el resultado del cálculo ejecutado

```
SELECT (319/29) + 12  
FROM DUAL;
```

(319/29)+12
23



DP 4-1  
Manipulación de Mayúsculas/Mlnúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

6

# Funciones de Caracteres de Una Sola Fila

- Las funciones de caracteres de una sola fila se dividen en dos categorías:
  - Funciones que convierten las mayúsculas/minúsculas de las cadenas de caracteres
  - Funciones que puede unir, extraer, mostrar, encontrar, llenar y recortar cadenas de caracteres
- Las funciones de una sola fila se pueden utilizar en las cláusulas SELECT, WHERE y ORDER BY

Las funciones de una sola fila son un código predefinido y muy potente que acepta argumentos y devuelven un valor. Un argumento se puede definir como un nombre de columna, una expresión o una constante.

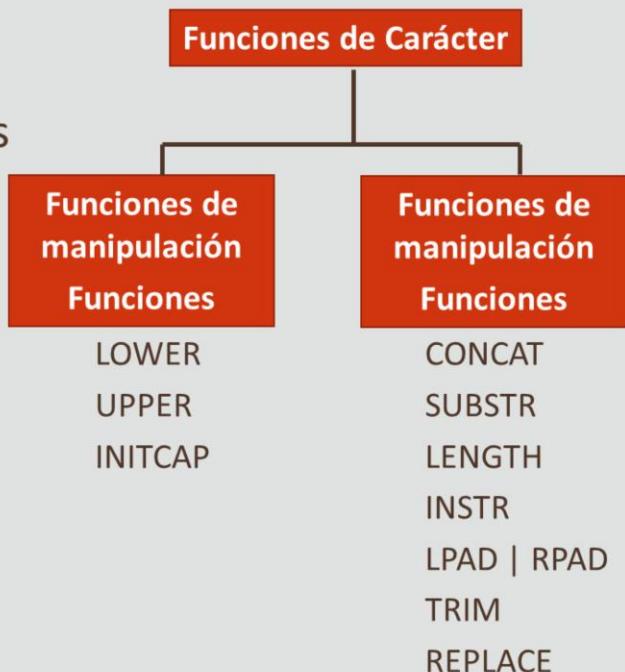
## Funciones de Caracteres de Una Sola Fila

- Las funciones de manipulación de mayúsculas/minúsculas son importantes porque puede que no siempre sepa cómo se han escrito (mayúscula, minúscula o mayúsculas y minúsculas) los datos almacenados en la base de datos
- La manipulación de mayúsculas/minúsculas le permite convertir temporalmente los datos de la base de datos en mayúsculas y minúsculas, según desee
- Se evita el hecho de que no coincidan el almacenamiento en mayúsculas/minúsculas en la base de datos y la redacción de mayúsculas/minúsculas de las solicitudes

# Funciones de Manipulación de Mayúsculas/Minúsculas

- Las funciones de manipulación de mayúsculas/minúsculas se utilizan para convertir los datos del estado en el que se almacenan en una tabla en mayúsculas, minúsculas o mayúsculas y minúsculas

## CHARACTER FUNCTIONS



**ORACLE**  
Academy

DP 4-1  
Manipulación de Mayúsculas/Minúsculas y de Caracteres

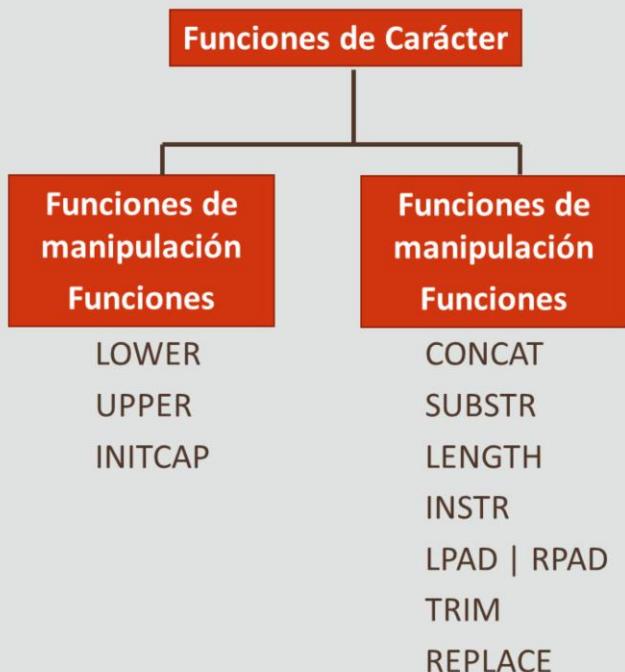
Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

9

# Funciones de Manipulación de Mayúsculas/Minúsculas

- Estas conversiones se puede utilizar para aplicar formato a la salida y también se pueden usar para buscar cadenas específicas

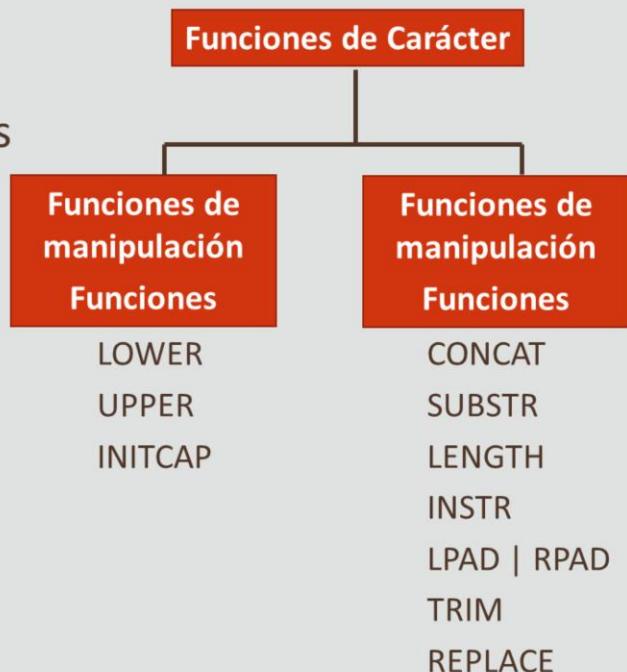
## CHARACTER FUNCTIONS



# Funciones de Manipulación de Mayúsculas/Minúsculas

- Las funciones de manipulación de mayúsculas/minúsculas se pueden utilizar en la mayoría de las partes de una sentencia SQL.

## CHARACTER FUNCTIONS



# Funciones de Manipulación de Mayúsculas/Minúsculas

- Las funciones de manipulación de mayúsculas/minúsculas suelen ser útiles cuando se está realizando una búsqueda de datos y no sabe si los datos que está buscando están en mayúsculas o minúsculas
- Desde el punto de vista de la base de datos, 'V' y 'v' NO son el mismo carácter y, como tal, necesita buscar utilizando las mayúsculas/minúsculas adecuadas
- LOWER(columna | expresión) convierte los caracteres alfabéticos en minúscula

```
SELECT last_name  
FROM employees  
WHERE LOWER(last_name) = 'abel';
```



Academy

DP 4-1  
Manipulación de Mayúsculas/MInúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

12

El motivo por el que Oracle diferencia entre 'V' y 'v' se debe al modo en el que almacena los caracteres. No almacena los caracteres directamente, sino sus correspondientes valores binarios, dependiendo del juego de caracteres de la base de datos. En la mayoría del mundo occidental, se habrá utilizado un juego de caracteres ACSII como juego de caracteres de la base de datos, y los códigos binarios para 'V' y 'v' son números distintos; por lo tanto, Oracle no considera que sean iguales.

# Funciones de Manipulación de Mayúsculas/Minúsculas

- **UPPER(columna | expresión)** convierte los caracteres alfabéticos en mayúscula

```
SELECT last_name  
FROM emplo oyees  
WHERE UPPER(last_name) = 'ABEL';
```

- **INITCAP(columna | expresión)** convierte los valores de caracteres alfabéticos en mayúscula para la primera letra de cada palabra

```
SELECT last_name  
FROM employees  
WHERE INITCAP(last_name) = 'Abel';
```



Academy

DP 4-1  
Manipulación de Mayúsculas/MInúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

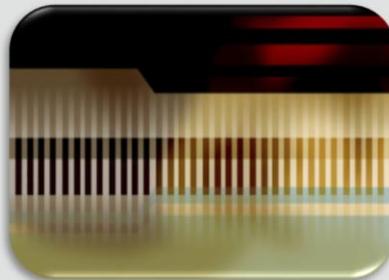
13

El uso de las funciones de manipulación de mayúsculas/minúsculas en la cláusula WHERE le permite recuperar filas, independientemente de si están almacenadas en la tabla en mayúscula o minúscula.

El uso de las funciones de manipulación de mayúsculas/minúsculas en la cláusula SELECT modifica la forma en la que se muestran los resultados de la consulta.

# Funciones de Manipulación de Caracteres

- Las funciones de manipulación de caracteres se utilizan para extraer, cambiar, formatear o modificar de alguna forma una cadena de caracteres
- A la función se pasan uno o más caracteres o palabras, entonces, esta realiza sus funciones en las cadenas de caracteres de entrada y devuelve el valor cambiado, extraído, contado o alterado



# Funciones de Manipulación de Caracteres

- CONCAT: une dos valores
- Toma 2 argumentos de cadena de caracteres y une la segunda cadena a la primera. También se puede escribir mediante el operador de concatenación: 'Hello' || 'World'

Ejemplos:	resultado
<pre>SELECT CONCAT('Hello', 'World') FROM DUAL;</pre>	HelloWorld
<pre>SELECT CONCAT(first_name, last_name) FROM employees;</pre>	EllenAbel CurtisDavies ...

# Funciones de Manipulación de Caracteres

- SUBSTR: extrae una cadena de una longitud determinada
- Los argumentos son (cadena de caracteres, posición inicial, Length)
- El argumento Length es opcional y, si se omite, devuelve todos los caracteres al final de la cadena

Ejemplos:	resultado
SELECT SUBSTR('HelloWorld',1,5) FROM DUAL;	Hello
SELECT SUBSTR('HelloWorld', 6) FROM DUAL;	Mundo
SELECT SUBSTR(last_name,1,3) FROM employees;	Abe Dav

# Funciones de Manipulación de Caracteres

- LENGTH: muestra la longitud de una cadena como un valor numérico
- La función toma una cadena de caracteres como argumento y devuelve el número de caracteres de esa cadena de caracteres

Ejemplos:	resultado
SELECT LENGTH('HelloWorld') FROM DUAL;	10
SELECT LENGTH(last_name) FROM employees;	4 6 ...



DP 4-1  
Manipulación de Mayúsculas/MInúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

17

El segundo ejemplo devuelve el número de caracteres en cada apellido de los empleados.

# Funciones de Manipulación de Caracteres

- INSTR: encuentra la posición numérica de los caracteres especificados
- INSTR busca la primera incidencia de una subcadena dentro de una cadena de caracteres y devuelve la posición como un número
- Si no se encuentra la subcadena, se devuelve el número cero

Ejemplos:	resultado
<pre>SELECT INSTR('HelloWorld', 'W') FROM DUAL;</pre>	6
<pre>SELECT last_name, INSTR(last_name, 'a') FROM employees;</pre>	Abel 0 Davies 2 ...

# Funciones de Manipulación de Caracteres

- LPAD: rellena la parte izquierda de una cadena de caracteres, dando lugar a un valor justificado a la derecha
- LPAD necesita 3 argumentos: una cadena de caracteres, el número total de caracteres en la cadena rellena y el carácter con el que llenarla

Ejemplos:	resultado
SELECT LPAD('HelloWorld',15, '-') FROM DUAL;	-----HelloWorld
SELECT LPAD(last_name, 10,'*') FROM employees;	*****Abel ****Davies ...



Academy

DP 4-1  
Manipulación de Mayúsculas/MInúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

19

# Funciones de Manipulación de Caracteres

- RPAD: rellena la parte derecha de una cadena de caracteres, dando lugar a un valor justificado a la izquierda

Ejemplos:	resultado
SELECT RPAD('HelloWorld',15, '-') FROM DUAL;	HelloWorld-----
SELECT RPAD(last_name, 10,'*') FROM employees;	Abel***** Davies**** ...

# Funciones de Manipulación de Caracteres

- TRIM: elimina todos los caracteres especificados, ya sea del principio, del final, o de ambos de una cadena
- La sintaxis de la función TRIM es:

Ejemplos:	resultado
SELECT TRIM(LEADING 'a' FROM 'abcba') FROM DUAL;	bcba
SELECT TRIM(TRAILING 'a' FROM 'abcba') FROM DUAL;	abcb
SELECT TRIM(BOTH 'a' FROM 'abcba') FROM DUAL;	bcb



Ejemplo 1: elimina la primera 'a' del inicio de la cadena 'abcba'

Ejemplo 2: elimina la última 'a' del final de la cadena 'abcba'

Ejemplo 3: elimina tanto la primera 'a' como la última 'a' de la cadena 'abcba'.

Si se omite LEADING, TRAILING o BOTH, la función devuelve BOTH (ambos).

Si el carácter especificado no es el primer (o último) carácter de la cadena, no se recorta, por ejemplo TRIM (LEADING 'a' FROM 'xyz') devolvería 'xyz'

# Funciones de Manipulación de Caracteres

- REPLACE: sustituye una secuencia de caracteres de una cadena por otro juego de caracteres
- La sintaxis de la función REPLACE es:

```
REPLACE (string1, string_to_replace, [replacement_string] )
```

- string1 es la cadena que cuyos caracteres serán sustituidos
- String\_to\_replace es la cadena que se buscará y se sacará de string1
- [replacement\_string] es la nueva cadena que se va a insertar en string1

# Funciones de Manipulación de Caracteres

Ejemplos:	resultado
SELECT REPLACE('JACK and JUE','J','BL') FROM DUAL;	BLACK and BLUE
SELECT REPLACE('JACK and JUE','J') FROM DUAL;	ACK and UE
SELECT REPLACE(last_name,'a','*') FROM employees;	Abel D*vies De H**n



Ejemplo 1: todas las instancias de 'J' en la cadena 'JACK and JUE' se sustituye por 'BL'.

Ejemplo 2: si el argumento de cadena de sustitución se omite, se suprime string\_to\_replace.

Por lo tanto todas las instancias de 'J' en la cadena 'JACK and JUE' se sustituye por 'BL'.

Ejemplo 3: cada instancia del carácter 'a' en los apellidos del empleado se sustituye por un carácter '\*'.

# Uso de Alias de Columna con Funciones

- Todas las funciones funcionan en los valores que aparecen entre paréntesis y cada nombre de función indica su finalidad, recordar esto es útil al construir una consulta
- Un alias de columna se utiliza a menudo para asignar un nombre a una función
- Cuando se utiliza un alias de columna, dicho alias aparece en la salida en lugar de la sintaxis de la función real

## Uso de Alias de Columna con Funciones

- En los siguientes ejemplos, el alias "User Name" ha sustituido la sintaxis de la función de la primera consulta
- Por defecto, el nombre de columna de una sentencia SELECT aparece como la cabecera de la columna
- En el segundo ejemplo de consulta, sin embargo, no hay ninguna columna en la tabla para los resultados producidos, por lo que en su lugar se usa la sintaxis de la consulta

# Uso de Alias de Columna con Funciones

```
SELECT LOWER(last_name) ||  
LOWER(SUBSTR(first_name,1,1))  
AS "User Name"  
FROM employees;
```

User Name
abele
daviesc
de haanl

```
SELECT LOWER (last_name) || LOWER(SUBSTR(first_name,1,1))  
FROM f_staffs;
```

LOWER(LAST_NAME)  LOWER(SUBSTR(FIRST_NAME,1,1))
---

does
------

millerb
---------

tuttlem
---------



Academy

DP 4-1  
Manipulación de Mayúsculas/MInúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

26

El primer ejemplo utiliza un alias para la cabecera de columna, y es más legible y fácil de recordar que el segundo ejemplo que no tiene ningún alias de columna.

## Variables de Sustitución

- En ocasiones, es posible que necesite ejecutar la misma consulta con muchos valores diferentes para obtener conjuntos de resultados diferentes
- Imagine, por ejemplo, que tuviera que escribir un informe de los empleados y sus departamentos, pero la consulta solo debe devolver los datos de un departamento a la vez
- Sin el uso de variables de sustitución, esta solicitud significaría que tendría que editar varias veces la misma sentencia para cambiar la cláusula WHERE

## Variables de Sustitución

- Afortunadamente para nosotros, Oracle Application Express soporta las variables de sustitución
- Para usarlas, todo lo que tiene que hacer es sustituir el valor codificado en la sentencia con un :named\_variable
- Oracle Application Express le pedirá un valor al ejecutar la sentencia

# Variables de Sustitución

- Si esta fuera la consulta original:

```
SELECT first_name, last_name, salary, department_id  
FROM employees  
WHERE department_id= 10;
```

– Vuelva a ejecutarla con valores diferentes: 20, 30, 40, etc.

- Se puede reescribir como:

```
SELECT first_name, last_name, salary, department_id  
FROM employees  
WHERE department_id=:enter_dept_id;
```

- Observe el uso de : delante de enter\_dept\_id
- El signo de dos puntos es el bit mágico y hace que Oracle Application Express reconozca el texto que le sigue como una variable



Academy

DP 4-1  
Manipulación de Mayúsculas/MInúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

29

Una variable de sustitución la define el usuario en el momento de la ejecución.

## Variables de Sustitución

- Al hacer clic en Run, Oracle Application Express muestra una ventana emergente como la siguiente:

		<b>Submit</b>
Bind Variable	Value	
:ENTER_DEPT_ID	<input type="text"/>	

- NOTA: los bloqueadores de elementos emergentes deben estar desactivados; de lo contrario APEX no podrá preguntar por el valor de la variable, ya que esto se introduce mediante una ventana emergente

## Variables de Sustitución

- Las variables de sustitución se tratan como cadenas de caracteres en Oracle Application Express, lo que significa que al transferir caracteres o valores de fecha, no necesita las comillas simples que se suelen utilizar para delimitar las cadenas
- Por lo tanto, una cláusula WHERE debería tener un aspecto similar al siguiente:

```
SELECT *
FROM employees
WHERE last_name = :1_name;
```

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Funciones de carácter
  - CONCAT
  - DUAL
  - Expresión
  - Formato
  - INITCAP
  - Input
  - INSTR



DP 4-1  
Manipulación de Mayúsculas/Minúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

32

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - LENGTH
  - LOWER
  - LPAD
  - Salida
  - REPLACE
  - RPAD
  - Funciones de una sola fila
  - SUBSTR

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - TRIM
  - UPPER
  - Variable de sustitución

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Seleccionar y aplicar funciones de una sola fila que realicen conversión de mayúsculas/minúsculas y/o la manipulación de caracteres
  - Seleccionar y aplicar las funciones de manipulación de mayúsculas/minúsculas de caracteres LOWER, UPPER e INTCAP en una consulta SQL
  - Seleccionar y aplicar las funciones de manipulación de caracteres CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM y REPLACE en una consulta SQL
  - Escribir consultas flexibles
  - usando variables de sustitución



**ORACLE**

Academy

DP 4-1  
Manipulación de Mayúsculas/Minúsculas y de  
Caracteres

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

35

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

4-2

## Funciones Numéricas

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Seleccionar y aplicar las funciones numéricas de una sola fila ROUND TRUNC y MOD en una consulta SQL
  - Distinguir entre los resultados obtenidos al aplicar TRUNC a un valor numérico y aplicar ROUND a un valor numérico
  - Las consecuencias de negocio al aplicar los valores numéricos TRUNC y ROUND



## Objetivo

- Uno de los motivos por los que ponemos nuestro dinero en un banco es para aprovechar el interés que acumula a lo largo del tiempo
- Los bancos ajustan el tipo de interés con diversos indicadores económicos como la inflación y la bolsa
- Normalmente, los tipos de interés se expresan en forma de porcentaje, como 3,45 %



# Objetivo

- Si un banco decidiera redondear la tasa de porcentaje al 3,5 %, ¿representaría una ventaja para usted?
- Si decidiera tan solo borrar los valores decimales y calcular el interés a un 3 %, ¿le haría feliz?
- El truncado y el redondeo de números juegan una parte importante en los negocios y, a su vez, en las bases de datos que soportan dichos negocios, ya que almacenan y acceden a datos numéricos



# Funciones Numéricas

- Las tres funciones numéricas son:

- ROUND
- TRUNC
- MOD



**ORACLE**  
Academy

DP 4-2  
Funciones Numéricas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

6

# ROUND

- ROUND se puede utilizar tanto con números como con fechas
- Se utiliza principalmente para redondear números a un número especificado de posiciones decimales, pero también se puede utilizar para redondear números a la izquierda de la coma decimal
- Sintaxis:

```
ROUND(column|expression, decimal places)
```

- Tenga en cuenta que si el número de posiciones decimales no está especificado o es cero, el número se redondeará sin decimales
- `ROUND(45.926) 46`
- `ROUND(45.926, 0) 46`

El uso de la función ROUND con fechas se tratará en una lección posterior.

## ROUND

- Si el número de posiciones decimales es un número positivo, el número se redondea a ese número de posiciones decimales a la derecha de la coma decimal
- $\text{ROUND}(45.926, 2)$  45.93
- Si el número de posiciones decimales es un número negativo, el número se redondea a ese número de posiciones decimales a la izquierda de la coma decimal
- $\text{ROUND}(45.926, -1)$  50

# TRUNC

- La función TRUNC se puede utilizar tanto con números como con fechas. Principalmente se utiliza para terminar la columna, la expresión o el valor en un número especificado de posiciones decimales
- Cuando se usa TRUNC, si el número de posiciones decimales no se ha especificado, entonces, tal como ocurre con ROUND, el número especificado se define por defecto en cero
- Sintaxis:

```
TRUNC(column|expression, decimal places)
```

- TRUNC (45.926, 2) 45.92

El uso de la función TRUNC con fechas se tratará en una lección posterior.

# TRUNC

- Al igual que con ROUND, si la expresión TRUNC no especifica el número de posiciones decimales o especifica un cero, el número se trunca en cero posiciones decimales
- $\text{TRUNC}(45.926, 0)$  45
- $\text{TRUNC}(45.926)$  45
- Recuerde que TRUNC no redondea el número.
- Simplemente termina el número en un punto determinado

## MOD

- La función MOD encuentra el resto después de que un valor se divide entre otro valor
- Por ejemplo, el MOD de 5 dividido entre 2 es 1
- MOD se puede utilizar para determinar si un valor es par o impar. Si se divide un valor entre 2 y no hay ningún resto, el número debe ser un número par
- Por ejemplo, si el MOD de  $x$  dividido entre 2 es 0, entonces,  $x$  debe ser un número par

Nota: el operador de módulo se utiliza en muchos lenguajes de programación, y devuelve el resto de una operación de división.

## MOD

- La columna "Mod Demo" mostrará si el número de aeropuertos de cada país es un número par o impar

```
SELECT country_name, MOD(airports,2)
AS "Mod Demo"
FROM wf_countries;
```

- 1 Significa que el número es impar, y cero significa que es par

COUNTRY_NAME	Mod Demo
Canada	1
Republic of Costa Rica	0
Republic of Cape Verde	1
Greenland	0
Dominican Republic	0
State of Eritrea	1
...	

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Funciones de número
  - MOD
  - ROUND
  - TRUNC

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Seleccionar y aplicar las funciones numéricas de una sola fila ROUND TRUNC y MOD en una consulta SQL
  - Distinguir entre los resultados obtenidos al aplicar TRUNC a un valor numérico y aplicar ROUND a un valor numérico
  - Las consecuencias de negocio al aplicar los valores numéricos TRUNC y ROUND



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

4-3

## Funciones de Fecha

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Mostrar el uso de SYSDATE y las funciones de fecha
  - Establecer las consecuencias de negocio mundial a fin de poder manipular fácilmente los datos almacenados en formato de fecha

# Objetivo

- ¿Se ha preguntado alguna vez cuántos días quedan del año escolar o el número de semanas que falta hasta la graduación?
- Debido a que la base de datos de Oracle almacena las fechas como números, puede realizar cálculos en las fechas con los operadores matemáticos de suma y resta, entre otros
- Las empresas dependen de poder utilizar funciones de fecha para planificar las nóminas y los pagos, realizar un seguimiento de las revisiones de rendimiento de los empleados y años de servicio, o realizar un seguimiento de los pedidos y envíos
- Todas estas necesidades de negocio se realizan fácilmente mediante sencillas funciones de fecha de SQL

## Visualización de Fechas

- La visualización por defecto y el formato de entrada de las fechas es DD-Mon-YYYY
- Por ejemplo: 02-Dec-2014
- Sin embargo, las bases de datos Oracle almacena fechas de modo interno en un formato numérico que representa el siglo, el año, el mes, el día, la hora, el minuto y el segundo
- Las fechas de Oracle válidas son del 1 de enero de 4712 A.C. y el 31 de diciembre de 9999 D.C.
- Esto representa el rango de fechas que puede almacenar correctamente en una base de datos Oracle

## SYSDATE

- SYSDATE es una función de fecha que devuelve la fecha y hora actuales del servidor de base de datos
- Utilice SYSDATE para mostrar la fecha actual, utilice la tabla DUAL

```
SELECT SYSDATE  
FROM dual;
```

SYSDATE
01-Jul-2017

## Tipo de Dato DATE

- El tipo de dato DATE siempre almacena internamente la información del año como un número de cuatro dígitos: dos dígitos para el siglo y dos dígitos para el año
- Por ejemplo, Oracle Database almacena el año como 1996 o 2004, no solo como 96 o 04
- En las versiones anteriores, el componente de siglo no se mostraba por defecto
- Sin embargo, debido a las cambiantes necesidades de los negocios en todo el mundo, el año de 4 dígitos es ahora la visualización por defecto

# Trabajar con fechas

Ejemplos:	resultado
<pre>SELECT last_name, hire_date + 60 FROM employees;</pre>	Agrega 60 días a hire_date
<pre>SELECT last_name, (SYSDATE - hire_date)/7 FROM employees;</pre>	Muestra el número de semanas desde que se contrató al empleado
<pre>SELECT employee_id, (end_date - start_date)/365 AS "Tenure in last job" FROM job_history;</pre>	Busca el número de días que mantuvo trabajo un empleado y, a continuación, lo divide entre 365 para mostrarlo en años

Estos ejemplos muestran el uso de las operaciones aritméticas con fechas.

# Funciones de Fecha

- Las funciones de fecha que se muestran en la tabla funcionan en fechas de Oracle
- Todas las funciones de fecha devuelven un valor del tipo de dato DATE excepto la función MONTHS\_BETWEEN, que devuelve un valor de tipo de dato numérico

Función	Descripción
MONTHS_BETWEEN	Número de meses entre dos fechas
ADD_MONTHS	Agregar meses de calendario a fecha
NEXT_DAY	Fecha de la siguiente incidencia del día de la semana especificado
LAST_DAY	Último día del mes
ROUND	Redondear fecha
TRUNC	Truncar fecha

Para los ejemplos que figuran a continuación, SYSDATE es 01-Jul-2015

## Funciones de Fecha

- **MONTHS\_BETWEEN:** toma 2 argumentos DATE y devuelve el número de meses de calendario entre las 2 fechas
- Si el primer argumento es una fecha anterior a la segunda, el número devuelto es negativo

Ejemplos de Función de Fecha:	Resultado	
SELECT last_name, hire_date FROM employees WHERE MONTHS_BETWEEN (SYSDATE, hire_date)>240;	King Kochhar De Haan ...	17-Jun-1987 21-Sep-1989 13-Jan-1993 ...

Este ejemplo devuelve los empleados contratados hace más de 240 meses (20 años).

## Funciones de Fecha

- ADD\_MONTHS: toma 2 argumentos, una fecha y un número. Devuelve un valor de fecha con el argumento numérico agregado al componente mensual de la fecha
- Si el número proporcionado es negativo, la función restará ese número de meses del argumento de fecha

Ejemplos de Función de Fecha:	Resultado
SELECT ADD_MONTHS (SYSDATE, 12) AS "Next Year" FROM dual;	01-Jul-2016

Este ejemplo agrega 12 meses a la fecha actual.

# Funciones de Fecha

- **NEXT\_DAY:** toma 2 argumentos, una fecha y un día de la semana y devuelve la fecha de la siguiente incidencia de ese día de la semana después del argumento DATE

Ejemplos de Función de Fecha:	Resultado
SELECT NEXT_DAY (SYSDATE, 'Saturday') AS "Next Saturday" FROM dual;	08-Jul-2017

# Funciones de Fecha

- **LAST\_DAY:** toma un argumento DATE y devuelve la fecha del último día del mes del argumento DATE

Ejemplos de Función de Fecha:	Resultado
SELECT LAST_DAY (SYSDATE) AS "End of the Month" FROM dual;	31-Jul-2015

# Funciones de Fecha

- ROUND: devuelve una fecha redondeada a la unidad especificada en el segundo argumento

Ejemplos de Función de Fecha:	Resultado		
SELECT hire_date, ROUND (hire_date, 'Month') FROM employees WHERE department_id=50;	16-Nov-1999 17-Oct-1995 29-Jan-1997 ...	01-Dec-1999 01-Nov-1995 01-Feb-1997 ...	
SELECT hire_date, ROUND (hire_date, 'Year') FROM employees WHERE department_id=50;	16-Nov-1999 17-Oct-1995 29-Jan-1997 ...	01-Jan-2000 01-Jan-1996 01-Jan-1997 ...	

# Funciones de Fecha

- TRUNC: devuelve una fecha truncada a la unidad especificada en el segundo argumento

Ejemplos de Función de Fecha:	Resultado	
SELECT hire_date, TRUNC(hire_date, 'Month') FROM employees WHERE department_id=50;	16-Nov-1999 17-Oct-1995 29-Jan-1997 ...	01-Nov-1999 01-Oct-1995 01-Jan-1997 ...
SELECT hire_date, TRUNC(hire_date, 'Year') FROM employees WHERE department_id=50;	16-Nov-1999 17-Oct-1995 29-Jan-1997 ...	01-Jan-1999 01-Jan-1995 01-Jan-1997 ...

Si 'month' es la unidad especificada, la fecha devuelta será el primer día del mes del argumento DATE.

Si 'Year' es la unidad especificada, TRUNC devolverá el primer día del año del argumento DATE.

# Funciones de Fecha

- A continuación se muestra un ejemplo de una consulta con varias funciones de fecha
- La salida se muestra en la siguiente diapositiva

```
SELECT employee_id, hire_date,
       ROUND(MONTHS_BETWEEN(SYSDATE, hire_date)) AS TENURE,
       ADD_MONTHS(hire_date, 6) AS REVIEW,
       NEXT_DAY(hire_date, 'FRIDAY'), LAST_DAY(hire_date)
  FROM employees
 WHERE MONTHS_BETWEEN (SYSDATE, hire_date) > 36;
```

# Funciones de Fecha

- El juego de resultados de esta consulta devuelve 20 filas en las que se incluye:

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(HIRE_DATE,'FRIDAY')	LAST_DAY(HIRE_DATE)
100	17-Jun-1987	348	17-Dec-1987	19-Jun-1987	30-Jun-1987
101	21-Sep-1989	321	21-Mar-1990	22-Sep-1989	30-Sep-1989
102	13-Jan-1993	281	13-Jul-1993	15-Jan-1993	31-Jan-1993
200	17-Sep-1987	345	17-Mar-1988	18-Sep-1987	30-Sep-1987
205	07-Jun-1994	265	07-Dec-1994	10-Jun-1994	30-Jun-1994
206	07-Jun-1994	265	07-Dec-1994	10-Jun-1994	30-Jun-1994
149	29-Jan-2000	197	29-Jul-2000	04-Feb-2000	31-Jan-2000
174	11-May-1996	241	11-Nov-1996	17-May-1996	31-May-1996
176	24-Mar-1998	219	24-Sep-1998	27-Mar-1998	31-Mar-1998
178	24-May-1999	205	24-Nov-1999	28-May-1999	31-May-1999

More than 10 rows available. Increase rows selector to view more rows.



Academy

DP 4-3  
Funciones de Fecha

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

17

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - ADD\_MONTHS
  - LAST\_DAY
  - MONTHS\_BETWEEN
  - NEXT\_DAY
  - SYSDATE
  - ROUND
  - TRUNC

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Seleccionar y aplicar las funciones de una sola fila MONTHS\_BETWEEN, ADD\_MONTHS, NEXT\_DAY, LAST\_DAY, ROUND y TRUNC que funcionan en los datos de fecha
  - Explicar cómo transforman las funciones de fecha las fechas de Oracle en datos de fecha o en valores numéricos
  - Mostrar un uso adecuado de los operadores aritméticos con fechas

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Mostrar el uso de SYSDATE y las funciones de fecha
  - Establecer las consecuencias de negocio mundial a fin de poder manipular fácilmente los datos almacenados en formato de fecha



DP 4-3  
Funciones de Fecha

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

5-1

## Funciones de Conversión

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Proporcionar un ejemplo de conversión de tipo de dato implícita y explícita y una conversión de tipo de dato implícita
  - Explicar el motivo de la importancia, desde una perspectiva de negocio, de que un lenguaje tenga capacidades de conversión de datos incorporadas
  - Crear una consulta SQL que aplique correctamente las funciones de una sola fila TO\_CHAR, TO\_NUMBER y TO\_DATE para obtener el resultado deseado

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Aplicar el modelo de formato de fecha y/o caracteres adecuado para producir una salida deseada
  - Explicar y aplicar el uso de YY y RR para devolver el año correcto como se almacenó en la base de datos

# Objetivo

- Imagine que tuviera que leer todos los libros de texto en archivos de texto sin párrafos y sin mayúsculas
- Sería difícil leerlos
- Afortunadamente, existen programas de software disponibles que permiten poner en mayúsculas y aplicar color al texto, subrayarlos, ponerlos en negrita, centrarlos y agregar gráficos
- En el caso de bases de datos, los cambios de formato y visualización se realizan utilizando funciones de conversión
- Estas funciones permiten mostrar los números en la moneda local, aplicar una gran variedad de formatos a las fechas, mostrar la hora incluyendo hasta los segundos y hacer un seguimiento del siglo al que hace referencia una fecha

# Tipos de Dato

- Cuando se crea una tabla para una base de datos, el programador SQL debe definir qué tipo de dato se almacenará en cada uno de los campos de la tabla
- En SQL, hay diferentes tipos de dato. Estos tipos de dato definen el dominio de valores que puede incluir cada columna
- Para esta lección, utilizará:
  - VARCHAR2
  - CHAR
  - NUMBER
  - DATE



En la mayoría de los lenguajes de programación se necesita que el programador declare el tipo de dato de cada objeto de datos. Para los datos almacenados en una base de datos, el programador SQL define un tipo de dato para todas las columnas de la base de datos. Los tipos de dato se tratarán con mayor detalle más adelante en el curso, pero por ahora utilizaremos VARCHAR2, CHAR, NUMBER y DATE.

# Descripción de Tipos de Dato

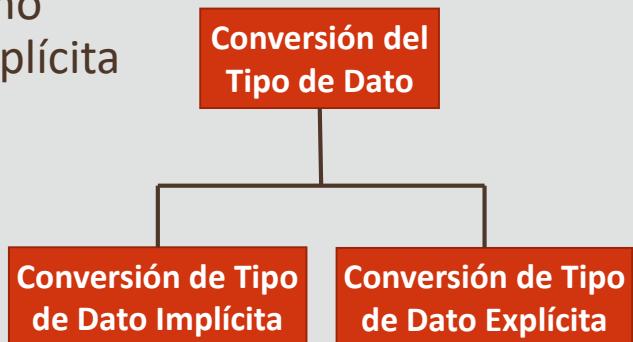
- VARCHAR2: se utiliza para datos de caracteres de longitud variable, incluidos los números, los guiones y los caracteres especiales
- CHAR: se utiliza para datos de texto y de caracteres de longitud fija, incluidos los números, los guiones y los caracteres especiales

# Descripción de Tipos de Dato

- NUMBER: se utiliza para almacenar datos numéricos de longitud variable. No se permiten guiones, texto u otros datos no numéricos. La moneda se almacena como tipo de dato numérico
- DATE: se utiliza para valores de fecha y hora. Internamente, Oracle almacena las fechas como números y, por defecto, se muestra información de DATE como DD-Mes-YYYY (por ejemplo, 23-Oct-2013)

# Conversión de Tipo

- Oracle Server puede convertir automáticamente datos VARCHAR2 y CHAR en tipos de dato NUMBER y DATE
- Puede convertir datos de tipo NUMBER y DATE de nuevo en un tipo de dato CHARACTER
- A esto se le conoce como conversión de datos implícita



**ORACLE**  
Academy

DP 5-1  
Funciones de Conversión

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

9

Implícito: algo que está sobreentendido, pero que no se expresa directamente.

# Conversión de Tipo

- Aunque esto es una característica útil, siempre es mejor realizar conversiones de tipo de dato explícitas para garantizar la fiabilidad en las sentencias SQL

## Conversiones de tipos de dato implícitas

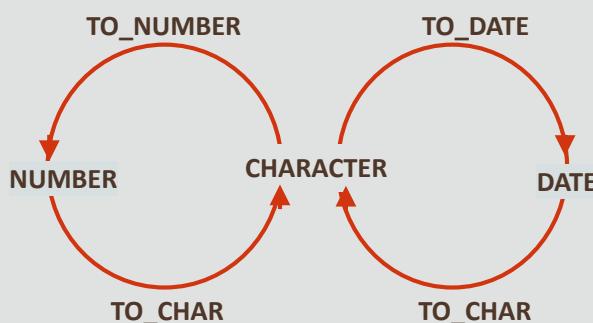
FROM	TO
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Explícito: claramente formulado o definido.

# Conversión de Tipo

- Las cuatro funciones de conversión de tipos de dato que aprenderá son:
  - Convertir un tipo de dato de fecha en tipo de dato de carácter
  - Convertir un tipo de dato numérico en tipo de dato de carácter

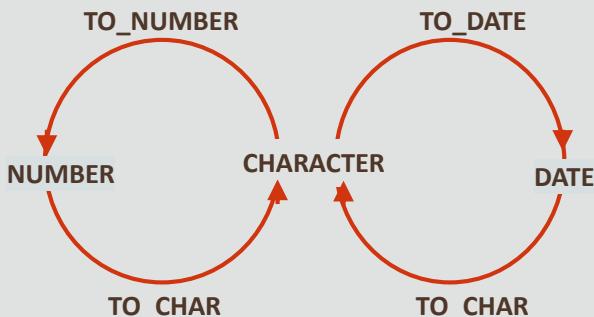
CONVERSIÓN DE TIPO DE DATO EXPLÍCITA



# Conversión de Tipo

- Las cuatro funciones de conversión de tipos de dato que aprenderá son:
  - Convertir un tipo de dato de caracteres en tipo de dato numérico
  - Convertir un tipo de dato de caracteres en tipos de dato de fecha

CONVERSIÓN DE TIPO DE DATO EXPLÍCITA



# Conversión de Datos de Fecha en Datos de Caracteres

- Por lo general, es aconsejable que un formato de fecha se convierta de su formato DD-Mes-YYYY por defecto en otro formato especificado por el usuario
- La función que permite llevar esto a cabo es:

```
TO_CHAR (date column name, 'format model you specify')
```

- El 'modelo de formato' debe estar entre comillas simples y es sensible a mayúsculas/minúsculas
- Separe el valor de fecha del modelo de formato con una coma

## Conversión de Datos de Fecha en Datos de Caracteres

- Se puede incluir cualquier elemento de formato de fecha válido
- Utilice sp para deletrear un número
- Utilice th para que el número aparezca como un ordinal (1º, 2º, 3º y así sucesivamente)
- Utilice un elemento fm para eliminar los espacios en blanco o eliminar los ceros iniciales de la salida

# Conversión de Datos de Fecha en Datos de Caracteres

YYYY	Año completo en números
YEAR	Año en letra
MM	Valor de dos dígitos del mes
MONTH	Nombre completo del mes
MON	Abreviatura de tres letras del mes
DY	Abreviatura de tres letras del día de la semana
DAY	Nombre completo del día de la semana
DD	Día numérico del mes
DDspth	FOURTEENTH
Ddspth	Fourteenth
ddspth	fourteenth
DDD o DD o D	Día del año, mes o semana
HH24:MI:SS AM	15:45:32 PM
DD "of" MONTH	12 de octubre

- Las tablas muestran los diferentes modelos de formato que se pueden utilizar
- Al especificar elementos temporales, tenga en cuenta que también se puede aplicar formato a las horas (HH), los minutos (MI), los segundos (SS) y AM o PM

# Conversión de Datos de Fecha en Datos de Caracteres

- Ejemplos de salida con diferentes modelos de formato:

Ejemplos:	Salida
<pre>SELECT TO_CHAR(hire_date, 'Month dd, YYYY') FROM employees;</pre>	martes, 07 de junio de 1994
<pre>SELECT TO_CHAR(hire_date, 'fmMonth dd, YYYY') FROM employees;</pre>	martes, 07 de junio de 1994
<pre>SELECT TO_CHAR(hire_date, 'fmMonth ddth, YYYY') FROM employees;</pre>	7 de junio de 1994 3 de enero de 1990



En el ejemplo 2 se utiliza "fm" para suprimir los ceros iniciales de la fecha.

En el ejemplo 3 se agrega "th", etc. para que el día aparezca como un número ordinal

# Conversión de Datos de Fecha en Datos de Caracteres

- Ejemplos de salida con diferentes modelos de formato:

Ejemplos:	Salida
SELECT TO_CHAR(hire_date, 'fmDay ddth Mon, YYYY') FROM employees;	Martes 7 de jun de 1994
SELECT TO_CHAR(hire_date, 'fmDay ddthsp Mon, YYYY') FROM employees;	Martes, siete de junio de 1994
SELECT TO_CHAR(hire_date, 'fmDay, ddthsp "of" Month, Year') FROM employees;	Martes, siete de junio, mil novecientos noventa y cuatro



En el ejemplo 2 se agrega "SP" al formato de día para especificar la parte de día de la fecha.

En el ejemplo 3 se muestra el año deletread y el texto "de" entre el día y el mes. Tenga en cuenta que las comillas dobles son necesarias alrededor del literal de texto que se va a incluir.

# Conversión de Datos de Fecha en Datos de Caracteres

- Ejemplos de salida con diferentes modelos de formato para el tiempo:

Ejemplos:	Salida
SELECT TO_CHAR(SYSDATE, 'hh:mm') FROM dual;	2:07 AM
SELECT TO_CHAR(SYSDATE, 'hh:mm pm') FROM dual;	2:07 AM
SELECT TO_CHAR(SYSDATE, 'hh:mm:ss pm') FROM dual;	2:07:23 AM

# Conversión de Datos Numéricos en Datos de Caracteres (VARCHAR2)

- Los números almacenados en la base de datos no tienen formato
- Esto significa que no tienen ningún signo/símbolo de moneda, comas, decimales, ni ningún otro formato
- Para agregar formato, en primer lugar debe convertir el número en un formato de caracteres

```
TO_CHAR(number, 'format model')
```

- La función SQL que se utiliza para convertir un número en un formato de carácter deseado es:

# Conversión de Datos Numéricos en Datos de Caracteres (VARCHAR2)

- En la tabla se muestran algunos de los elementos de formato que se pueden utilizar con las funciones TO\_CHAR

```
SELECT TO_CHAR(salary,  
  '$99,999') AS "Salary"  
FROM employees;
```

Salary
\$24,000
\$17,000

ELEMENT	DESCRIPTION	EXAMPLE	RESULT
9	Numeric position (# of 9's determine width)	999999	1234
0	Display leading zeros	099999	001234
\$	Floating dollar sign	\$999999	\$1234
L	Floating local currency symbol	L999999	FF1234
.	Decimal point in position specified	999999.99	1234.00
,	Comma in position specified	999,999	1,234
MI	Minus signs to right (negative values)	999999MI	1234-
PR	Parenthesize negative numbers	999999PR	<1234>
EEEE	Scientific notation ( must have four EEEE)	99.999EEEE	1,23E+03
V	Multiply by 10 n times (n= number of 9's after V)	9999V99	9999V99
B	Display zero values as blank, not 0	B9999.99	1234.00



Academy

DP 5-1  
Funciones de Conversión

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

20

NOTA: si el número de "9" en el modelo de formato es menor que el número de dígitos en el número, se muestra #####, por ejemplo, SELECT TO\_CHAR(24000, '\$9,999') devolvería #####, ya que el modelo de formato tiene cuatro "9", pero el número proporcionado tiene cinco dígitos.

# Conversión de Datos Numéricos en Datos de Caracteres (VARCHAR2)

- ¿Puede identificar los modelos de formato utilizados para generar la siguiente salida?
  - 3000,00 \$
  - 4.500
  - 9.000,00
  - 0004422

ELEMENT	DESCRIPTION	EXAMPLE	RESULT
9	Numeric position (# of 9's determine width)	999999	1234
0	Display leading zeros	099999	001234
\$	Floating dollar sign	\$999999	\$1234
L	Floating local currency symbol	L999999	FF1234
.	Decimal point in position specified	999999.99	1234.00
,	Comma in position specified	999,999	1,234
MI	Minus signs to right (negative values)	999999MI	1234-
PR	Parenthesize negative numbers	999999PR	<1234>
EEEE	Scientific notation ( must have four EEEE)	99.999EEEE	1,23E+03
V	Multiply by 10 n times (n= number of 9's after V)	9999V99	9999V99
B	Display zero values as blank, not 0	B9999.99	1234.00

Las respuestas aparecerán en la siguiente diapositiva.

# Conversión de Datos Numéricos en Datos de Caracteres (VARCHAR2)

- Respuestas:

SQL:	Salida
SELECT TO_CHAR(3000, '\$99999.99') FROM dual;	3000,00 \$
SELECT TO_CHAR(4500, '99,999') FROM dual;	4.500
SELECT TO_CHAR(9000, '99,999.99') FROM dual;	9.000,00
SELECT TO_CHAR(4422, '0,009,999') FROM dual;	0004422



# Conversión de Caracteres en Números

- Por lo general, es aconsejable convertir una cadena de caracteres en un número. La función necesaria para realizar esta conversión es:

```
TO_NUMBER(character string, 'format model')
```

- El modelo de formato es opcional, pero se debería incluir si la cadena de caracteres que se va a convertir contiene cualquier carácter que no sean números
- No puede realizar cálculos de forma fiable con datos de caracteres

```
SELECT TO_NUMBER('5,320', '9,999')  
AS "Number"  
FROM dual;
```

Number
5320

# Conversión de Caracteres en Números

- La columna bonus incluye los datos que contienen 4 caracteres y el modelo de formato especifica 3 caracteres, por lo que se devuelve un error

```
SELECT last_name, TO_NUMBER(bonus,  
'999')  
FROM employees  
WHERE department_id = 80;
```



ORA-01722: invalid number

```
SELECT last_name, TO_NUMBER(bonus,  
'9999')  
AS "Bonus"  
FROM employees  
WHERE department_id = 80;
```

LAST_NAME	Bonus
Zlotkey	1500
Abel	1700
Taylor	1250

Oracle Application Express devolverá un error de Oracle (Número no válido) si el modelo de formato no coincide con el número devuelto por la base de datos.

# Conversión de Caracteres en Fechas

- Para convertir una cadena de caracteres en un formato de fecha, utilice:

```
TO_DATE('character string', 'format model')
```

- Esta conversión toma una cadena de caracteres que no sea un valor de fecha como, por ejemplo, "3 noviembre de 2001" y lo convierte en un valor de fecha
- El modelo de formato indica al servidor que la cadena de caracteres "se parece a":

```
TO_DATE('November 3, 2001', 'Month dd, yyyy')
```

– Devolverá 03-Nov-2001

## Reglas del Modificador fx

- Cuando realiza una conversión de caracteres en fecha, el modificador fx (formato exacto) especifica la coincidencia exacta entre el argumento de carácter y el modelo de formato de fecha
- En el siguiente ejemplo, tenga en cuenta que en "May10" no hay ningún espacio entre "May" y "10"
- El modelo de formato fx busca la coincidencia con el argumento de caracteres, ya que tampoco tiene ningún espacio entre "Mon" y "DD"

```
SELECT TO_DATE('May10,1989', 'fxMonDD,YYYY')  
AS "Convert"  
FROM DUAL;
```

CONVERT

10-May-1989

ORACLE

Academy

DP 5-1  
Funciones de Conversión

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

26

# Reglas del Modificador fx

- Las reglas del modificador fx son:
  - La puntuación y el texto entre comillas en el argumento de caracteres deben coincidir con las partes correspondientes del modelo de formato exactamente (excepto en lo que respecta a mayúsculas/minúsculas)
  - El argumento de carácter no puede tener espacios en blanco adicionales
    - Sin fx, Oracle Server ignora los espacios en blanco adicionales
  - Los datos numéricos del argumento de carácter deben tener el mismo número de dígitos que el elemento correspondiente en el modelo de formato
    - Sin fx, los números del argumento de carácter no pueden omitir los ceros iniciales

# Reglas del Modificador fx

Examples:	Output
SELECT TO_DATE('Sep 07, 1965', 'fxMon dd, YYYY') AS "Date" FROM dual;	07-Sep-1965
SELECT TO_DATE('July312004', 'fxMonthDDYYYY') AS "Date" FROM DUAL;	31-Jul-2004
SELECT TO_DATE('June 19, 1990','fxMonth dd, YYYY') AS "Date" FROM DUAL;	19-Jun-1990

## Formato de Fecha RR y Formato de Fecha YY

- Todos los datos de fecha hora se almacenan utilizando años de cuatro dígitos (YYYY)
- No obstante, algunas bases de datos de legado pueden seguir utilizando el formato de dos dígitos (YY)
- No hace tanto tiempo que hemos cambiado de siglo, de 1900 a 2000
- Este cambio vino acompañado de una gran confusión en función de si una fecha escrita como 02-Jan-98 se interpretaría como 2 de enero de 1998 o 2 de enero de 2098

## Formato de Fecha RR y Formato de Fecha YY

- Si los datos que se van a convertir de datos de caracteres a datos de fecha solo contienen un año de dos dígitos, Oracle tiene una forma de interpretar estas fechas con el siglo correcto
- Por ejemplo: '27-Oct-95'

```
SELECT TO_DATE ('27-Oct-95' , 'DD-Mon-YY')  
AS "Date"  
FROM dual;
```

Date
27-Oct-2095

- El año de dos dígitos se interpreta como 2095, que puede que no sea lo previsto

## Formato de Fecha YY y Formato de Fecha RR

- Si YY se utiliza en el modelo de formato, se asume que el año pertenece al siglo actual
- Si el año de dos dígitos no está en el siglo actual, utilizamos RR

```
SELECT TO_DATE ('27-Oct-95' , 'DD-Mon-RR')
  AS "Date"
FROM dual;
```

Date
27-Oct-1995

- El año de dos dígitos ahora se interpreta como 1995

## Algunas Sencillas Reglas

- Si el formato de fecha se especifica con el formato RR, el valor de retorno tiene dos posibilidades, en función del año actual
- Si el año actual está entre 00-49:

- Fechas a partir de 0-49:  
la fecha estará en el siglo actual
- Fechas a partir de 50-99:  
la fecha estará en el siglo pasado

Si dos de los dígitos del año actual son:	Si el año de dos dígitos especificado es:	
	0-49	50-99
0-49	La fecha de devolución está en el siglo actual	La fecha de devolución está en el siglo anterior al actual
50-99	La fecha de devolución está en el siglo posterior al actual	La fecha de devolución está en el siglo actual

# Algunas Sencillas Reglas

- Si el año actual está entre 50-99:
  - Fechas a partir de 0-49: la fecha estará en el siglo siguiente
  - Fechas a partir de 50-99: la fecha estará en el siglo actual

Si dos de los dígitos del año actual son:	Si el año de dos dígitos especificado es:		
	0-49	50-99	
	0-49	La fecha de devolución está en el siglo actual	La fecha de devolución está en el siglo anterior al actual
	50-99	La fecha de devolución está en el siglo posterior al actual	La fecha de devolución está en el siglo actual

## Algunas Sencillas Reglas

- En la siguiente tabla se proporcionan algunos ejemplos de cómo se interpretan YY y RR y, en función del año actual

Año Actual	Fecha Especificada	Formato RR	Formato YY
1995	27-Oct-95	1995	1995
1995	27-Oct-17	2017	1917
2015	27-Oct-17	2017	2017
2015	27-Oct-95	1995	2095

## Algunas Sencillas Reglas

- Cuando consulto mi base de datos de empleados utilizando la siguiente sentencia, devuelve todas las filas de la tabla
- Sé que solo existen unos pocos empleados que se contrataron antes de 1990

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

- Como el modelo de formato en la cláusula WHERE utiliza YY, y el año actual es 2015, la consulta devuelve filas con un valor hire\_date inferior a 2090

Esto se puede solucionar mediante la sustitución de YY en el modelo de formato con RR o mediante años de cuatro dígitos , con YYYY.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - CHAR
  - DATE
  - Formato de fecha DD
  - Funciones de conversión
  - fm
  - NUMBER

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Formato de fecha RR
  - TO\_CHAR
  - TO\_DATE
  - TO\_NUMBER
  - VARCHAR2
  - Modificador fx

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Proporcionar un ejemplo de conversión de tipo de dato implícita y explícita y una conversión de tipo de dato implícita
  - Explicar el motivo de la importancia, desde una perspectiva de negocio, de que un lenguaje tenga capacidades de conversión de datos incorporadas
  - Crear una consulta SQL que aplique correctamente las funciones de una sola fila TO\_CHAR, TO\_NUMBER y TO\_DATE para obtener el resultado deseado

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Aplicar el modelo de formato de fecha y/o caracteres adecuado para producir una salida deseada
  - Explicar y aplicar el uso de YY y RR para devolver el año correcto como se almacenó en la base de datos

# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

5-2

## Funciones NULL

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Mostrar y explicar la evaluación de una función anidada
  - Enumerar al menos cuatro funciones generales que funcionan con cualquier tipo de dato y están relacionadas con el tratamiento de valores nulos
  - Explicar el uso de las funciones COALESCE y NVL
  - Explicar el uso de funciones generales para tratar con valores nulos en los datos
  - Crear y ejecutar una consulta SQL que aplica correctamente las funciones NVL NVL2, NULLIF y
  - COALESCE de una sola fila



# Objetivo

- Además de las funciones que controlan cómo se formatean los datos o se convierten en otro tipo, SQL utiliza un juego de funciones generales diseñado específicamente para tratar valores nulos
- Puede que se pregunte cómo un valor que no está disponible, sin asignar, desconocido o no aplicable pueda merecer tanta atención
- Lo nulo puede ser "nada", pero puede afectar a la forma en que las expresiones se evalúan, ¿cómo se calculan las medias y dónde aparece un valor en una lista ordenada
- Esta lección trata sobre el manejo de valores nulos



DP 5-2  
Funciones NULL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

4

Gran parte de la potencia de SQL se dedica a emular las decisiones que tenemos que tomar cada día. Necesitamos el formato de fecha cambiado para cumplir con el formato de escritura de otro país. Es necesario emitir cheques de nóminas que incluyan signos de monedas en lugar de solo números.

# Método de Evaluación de las Funciones

- Hasta ahora, ha aplicado funciones de una sola fila en sentencias simples
- Sin embargo, es posible anidar funciones a cualquier profundidad
- Es importante saber cómo se evalúan las funciones anidadas
- "Anidación" hace referencia a una cosa que está incluida en otra cosa (como un huevo dentro de un nido)
- El siguiente ejemplo muestra una función anidada
- El proceso de evaluación empieza desde el nivel más profundo hasta el nivel menos profundo

# Método de Evaluación de las Funciones

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'),  
'fmDay, Month ddth, YYYY') AS "Next Evaluation"  
FROM employees  
WHERE employee_id = 100;
```

- Los resultados son:
  - Friday, December 18th, 1987



# Método de Evaluación de las Funciones

```
SELECT TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'FRIDAY'),  
    'fmDay Month ddth, YYYY') AS "Next Evaluation"  
FROM employees  
WHERE employee_id = 100;
```

- Paso 1: A la fecha de contratación se le van a agregar seis meses
- Paso 2: Se identificará el primer viernes siguiente al día devuelto en el paso 1
- Paso 3: Al formato de fecha por defecto se le aplicará formato para que muestre la fecha devuelta por el paso 2 en un formato similar a: Viernes, 18 de diciembre de 1987, y aparecerá en la salida bajo el nombre de la columna "Next Evaluation"

Las funciones anidadas se evalúan a partir de la función más profunda, continuando hacia la menos profunda.

## Funciones Relacionadas con los Valores Nulos

- Al principio del curso, se presentó el término "nulo"
- Nulo es el valor que no está disponible, que está sin asignar, es desconocido o que no es aplicable
- Como resultado, no podemos comprobar si es el mismo que otro valor, porque no sabemos qué valor tiene
- No es igual a nada, ni siquiera a cero
- Pero que no sea realmente nada no significa que no sea importante

# Funciones Relacionadas con los Valores Nulos

- Imagine esta pregunta: ¿Es cierto que X = Y?
- Para responderla tiene que conocer los valores de X e Y
- Oracle tiene cuatro funciones generales relacionadas con el uso de valores nulos
- Las cuatro funciones son:
  - NVL
  - NVL2
  - NULLIF
  - COALESCE

## Función NVL

- La función NVL convierte un valor nulo en un valor conocido de un tipo de dato fijo, ya sea de fecha, carácter o numérico
- Los tipos de dato de la columna de valor nulo y el nuevo valor deben ser los mismos
- La función NVL es:

**NVL (expression 1 value that may contain a null, expression 2 value to substitute for null)**

- NVL (valor o columna que pueden contener un valor nulo, el valor para sustituir un valor nulo)

## Función NVL

- En la siguiente consulta se utiliza la función NVL con tipos de dato de carácter:

```
SELECT country_name, NVL(internet_extension, 'None')
      AS "Internet extn"
  FROM wf_countries
 WHERE location = 'Southern Africa'
 ORDER BY internet_extension DESC;
```

- Los valores nulos se sustituyen por el texto 'None'

COUNTRY_NAME	Internet extn
Juan de Nova Island	None
Europa Island	None
Republic of Zimbabwe	.zw
Republic of Zambia	.zm
Republic of South Africa	.za

## Función NVL

- Los tipos de dato de la columna de valor nulo y el nuevo valor deben ser el mismo que se muestra en los ejemplos siguientes:

Ejemplos:	Salida
<pre>SELECT last_name, NVL(commission_pct, 0) FROM employees WHERE department_id IN(80,90);</pre>	Zlotkey Abel Taylor King
<pre>SELECT NVL(date_of_independence, 'No date') FROM wf_countries;</pre>	1-Jul-1867 15-Sep-1821 5-Jul-1975 No date
*Tipo de datos de date_of_independence es Varchar2	

ORACLE

Academy

DP 5-2  
Funciones NULL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

12

Los valores NULL de las tablas se han sustituido mediante la función NVL (resaltada en rojo).

## Función NVL

- Puede utilizar la función NVL para convertir los valores de columna que contengan valores nulos en un número antes de realizar los cálculos
- Cuando se realiza un cálculo aritmético con un valor nulo, el resultado es nulo
- La función NVL puede convertir el valor null en un número antes de que se realicen los cálculos aritméticos para evitar un resultado nulo

## Función NVL

- En el ejemplo, la columna commission\_pct de la tabla employees contiene valores nulos
- La función NVL se utiliza para cambiar el valor nulo a cero antes de los cálculos aritméticos

```
SELECT last_name,  
NVL(commission_pct, 0)*250  
  AS "Commission"  
FROM employees  
WHERE department_id IN(80,90) ;
```

LAST_NAME	Commission
Zlotkey	50
Abel	75
Taylor	50
King	0
Kochhar	0
De Haan	0

## Función NVL2

- La función NVL2 evalúa una expresión con tres valores
- Si el primer valor no es nulo, la función NVL2 devuelve la segunda expresión
- Si el primer valor es nulo, se devolverá la tercera expresión
- Los valores de la expresión 1 pueden tener cualquier tipo de dato
- La expresión 2 y la expresión 3 pueden tener cualquier tipo de dato, excepto LONG
- El tipo de dato del valor devuelto siempre es el mismo que el tipo de dato de la expresión 2, a menos que la expresión 2 sean datos de caracteres, en cuyo caso, el tipo devuelto es VARCHAR2

LONG es un tipo de dato de caracteres de longitud variable de hasta 2 GB de tamaño. Los tipos de dato se tratan más detalladamente más adelante en el curso.

## Función NVL2

- La función NVL2 es:

```
NVL2 (expression 1 value that may contain a null, expression  
2 value to return if expression 1 is not null, expression 3  
value to replace if expression 1 is null)
```

- Una manera fácil de recordar NVL2 es pensar: "si la expresión 1 tiene un valor, sustituir expresión 2; si la expresión 1 es nula, sustituir expresión 3"

## Función NVL2

- La función NVL2 mostrada utiliza tipos de dato numéricos para las expresiones 1, 2 y 3

```
SELECT last_name, salary,  
       NVL2(commission_pct, salary + (salary * commission_pct),  
salary)  
          AS income  
FROM employees  
WHERE department_id IN(80,90);
```

LAST_NAME	SALARY	INCOME
Zlotkey	10500	12600
Abel	11000	14300
Taylor	8600	10320
King	24000	24000
Kochhar	17000	17000
De Haan	17000	17000

NVL2 comprueba si la expresión 1 (commission\_pct) tiene un valor. Si tiene un valor, se devuelve la expresión 2 (salary + (salary \* commission\_pct)). Si la expresión 1 es NULL, se devuelve la expresión 3 (salary).

## Función NULLIF

- La función NULLIF compara dos expresiones
- Si son iguales, la función devuelve un valor nulo
- Si no son iguales, la función devuelve la primera expresión
- La función NULLIF es:

```
NULLIF(expression 1, expression 2)
```

## Función NULLIF

- En este ejemplo, NULLIF compara la longitud de los nombres y apellidos de los empleados
- Si la longitud de ambos elementos es la misma, NULLIF devuelve NULL (como en la fila 2 Curtis Davies); de lo contrario, se devuelve el valor LENGTH de la expresión 1 de first\_name

```
SELECT first_name, LENGTH(first_name) AS "Length FN", last_name,
       LENGTH(last_name) AS "Length LN", NULLIF(LENGTH(first_name),
       LENGTH(last_name)) AS "Compare Them"
FROM employees;
```

FIRST_NAME	Length FN	LAST_NAME	Length LN	Compare Them
Ellen	5	Abel	4	5
Curtis	6	Davies	6	-
Lex	3	De Haan	7	3



Academy

DP 5-2  
Funciones NULL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

19

A menudo se utilizan funciones NULLIF después de realizar proyectos de migración de datos para probar si los datos del sistema de destino son los mismos que en los sistemas de origen iniciales. Por lo tanto, se utiliza NULLIF para buscar excepciones, no coincidencias, normalmente valores nulos como resultado de un valor NULLIF correcto, ya que desea que los datos de los sistemas de origen y destino sean exactamente los mismos.

## Función COALESCE

- La función COALESCE es una extensión de la función NVL, excepto en que COALESCE puede tener varios valores
- La palabra "coalesce" significa literalmente "unir" y eso es lo que ocurre
- Si la primera expresión es nula, la función continúa bajando por la línea hasta que se encuentra una expresión no nula
- Por supuesto, si la primera expresión tiene un valor, la función devuelve la primera expresión y la función se detiene
- La función COALESCE es:

```
COALESCE (expression 1, expression 2, ...expression n)
```



Academy

DP 5-2  
Funciones NULL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

20

## Función COALESCE

- Examine la sentencia SELECT de la tabla employees mostrada a la derecha
- Si un empleado tiene un valor (no NULL) para commission\_pct, este se devuelve; de lo contrario, si el salario tiene un valor, se devuelve el salario.
- Si los valores commission\_pct y salary de los empleados son NULL, devuelve el número 10

```
SELECT last_name,
       COALESCE(commission_pct, salary, 10)
    AS "Comm"
  FROM employees
 ORDER BY commission_pct;
```

LAST_NAME	Comm
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
Higgins	12000
Gietz	8300

ORACLE

Academy

DP 5-2  
Funciones NULL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

21

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - NVL
  - NVL2
  - NULLIF
  - COALESCE

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Mostrar y explicar la evaluación de una función anidada
  - Enumerar al menos cuatro funciones generales que funcionan con cualquier tipo de dato y están relacionadas con el tratamiento de valores nulos
  - Explicar el uso de las funciones COALESCE y NVL
  - Explicar el uso de funciones generales para tratar con valores nulos en los datos
  - Crear y ejecutar una consulta SQL que aplica correctamente las funciones NVL NVL2, NULLIF -y COALESCE de una sola fila



# **ORACLE**

## Academy



# ORACLE

## Academy

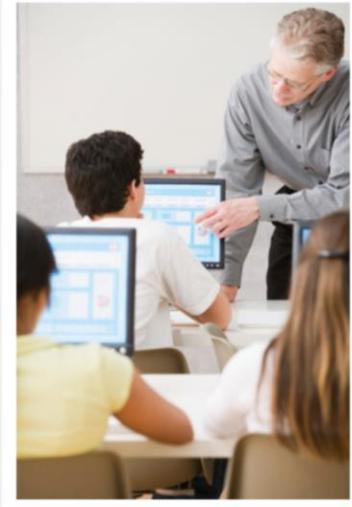


# Database Programming with SQL

5-3

## Expresiones Condicionales

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Comparar y contrastar las funciones DECODE y CASE
  - Crear y ejecutar una consulta SQL que utiliza correctamente las funciones DECODE y CASE
  - Crear y ejecutar dos métodos para implantar la lógica condicional IF-THEN-ELSE



# Objetivo

- Poder tomar decisiones es esencial en el modelado de datos
- Los modeladores tienen que decidir qué funciones de negocio se deben modelar y cuáles no
- En el proceso de modelado de datos es necesario que los diseñadores analicen la información para identificar las entidades, resolver las relaciones y seleccionar los atributos
- Una decisión típica podría ser:
  - (IF) Si una empresa necesita realizar un seguimiento de los datos a lo largo del tiempo, (THEN) En ese caso, el tiempo puede que necesite ser una entidad o (ELSE) En caso contrario, el tiempo debería ser un atributo

# Método de Evaluación de las Funciones

- Este proceso de toma de decisiones en programación no es muy diferente del proceso que se utiliza en la vida diaria
- Piense en la última vez que tuvo que tomar una decisión del tipo if-then-else
- Si (IF) termino mis deberes antes de las 9:00 p.m., puedo ver la televisión; en caso contrario (ELSE), no puedo ver la televisión
- En SQL, estos tipos de elecciones implican métodos de procesamiento condicionales
- Saber cómo utilizar el procesamiento condicional facilita la toma de decisiones para obtener los datos que deseé

# Expresiones Condicionales

- Las dos expresiones condicionales son CASE y DECODE
- Ya ha estudiado NULLIF, que es el equivalente lógico a la expresión CASE, ya que CASE compara dos expresiones
- NULIF compara dos expresiones y, si las dos expresiones son iguales, devuelve null; si no son iguales, devuelve la primera expresión



# Expresiones Condicionales

- Hay dos juegos de comandos o sintaxis que se pueden utilizar para escribir sentencias SQL:
  - Sentencias estándar compatibles con ANSI/ISO SQL 99
  - Sentencias propiedad de Oracle
- Los dos juegos de sintaxis son muy similares, pero hay unas cuantas diferencias
- En este curso, aprenderá a utilizar ambos juegos de sentencias SQL, pero se recomienda utilizar la sintaxis ANSI/ISO SQL 99

Es importante que los alumnos también aprendan sintaxis propiedad de Oracle, ya que es posible que se encuentren con esta sintaxis en bases de datos anteriores.

# Expresiones Condicionales

- CASE y DECODE son ejemplos de una de estas diferencias
- CASE es una sentencia compatible con ANSI/ISO 99 SQL 99
- DECODE es una sentencia propiedad de Oracle
- Ambas sentencias devuelven la misma información utilizando sintaxis diferente



# Expresión CASE

- La expresión CASE básicamente realiza la función de una sentencia IF-THEN-ELSE
- Los tipos de dato de las expresiones CASE, WHEN y ELSE deben ser iguales
- La sintaxis de una expresión CASE es:

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

# Sintaxis de CASE

- La consulta comprueba el valor department\_id
  - Si (IF) es 90, (then) devuelve 'Management'
  - Si (IF) es 80, (then) devuelve 'Sales'
  - Si (IF) es 60, (then) devuelve 'It'
  - En caso contrario (ELSE) devuelve 'Other dept.'

```
SELECT last_name,
CASE department_id
    WHEN 90 THEN 'Management'
    WHEN 80 THEN 'Sales'
    WHEN 60 THEN 'It'
    ELSE 'Other dept.'
END AS "Department"
FROM employees;
```

LAST_NAME	Department
King	Management
Kochhar	Management
De Haan	Management
Whalen	Other dept.
Higgins	Other dept.
Gietz	Other dept.
Zlotkey	Sales
Abel	Sales
Taylor	Sales
Grant	Other dept.
Mourgos	Other dept.
Rajs	Other dept.
Davies	Other dept.
Matos	Other dept.
Vargas	Other dept.
Hunold	It
Ernst	It
Lorentz	It
Hartstein	Other dept.
Fay	Other dept.

## Expresión DECODE

- La función DECODE evalúa una expresión de una forma similar a la lógica IF-THEN-ELSE
- DECODE compara una expresión con cada uno de los valores de búsqueda
- La sintaxis de DECODE es:

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

- Si se omite el valor por defecto, se devuelve un valor nulo donde un valor de búsqueda no coincide con ninguno de los valores

# Expresión DECODE

- Examine el ejemplo:

```
SELECT last_name,
       DECODE(department_id,
              90, 'Management',
              80, 'Sales',
              60, 'It',
              'Other dept.')
  AS "Department"
 FROM employees;
```

- Esta consulta devuelve exactamente los mismos resultados que el ejemplo CASE anterior, pero utilizando sintaxis diferente



LAST_NAME	Department
King	Management
Kochhar	Management
De Haan	Management
Whalen	Other dept.
Higgins	Other dept.
Gietz	Other dept.
Zlotkey	Sales
Abel	Sales
Taylor	Sales
Grant	Other dept.
Mourgos	Other dept.
Rajs	Other dept.
Davies	Other dept.
Matos	Other dept.
Vargas	Other dept.
Hunold	It
Ernst	It
Lorentz	It
Hartstein	Other dept.
Fay	Other dept.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - CASE
  - Expresión condicional
  - DECODE

# Resumen

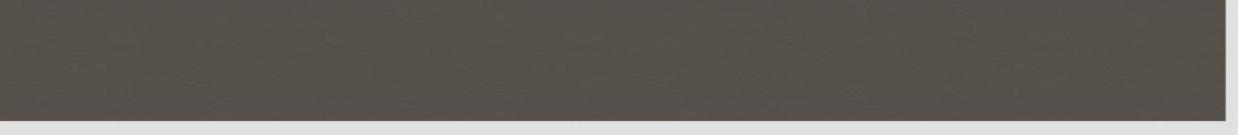
- En esta lección, debe haber aprendido lo siguiente:
  - Comparar y contrastar las funciones DECODE y CASE
  - Crear y ejecutar una consulta SQL que utiliza correctamente las funciones DECODE y CASE
  - Crear y ejecutar dos métodos para implantar la lógica condicional IF-THEN-ELSE





# ORACLE

## Academy



# **ORACLE**

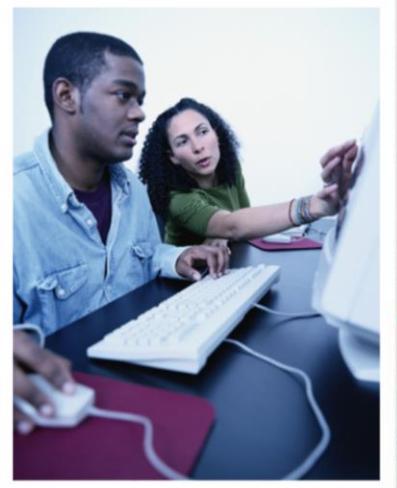
## Academy

# Database Programming with SQL

6-1

## Uniones Cruzadas y Uniones Naturales

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear y ejecutar una unión natural utilizando la sintaxis de unión ANSI-99 SQL
  - Crear una unión cruzada utilizando la sintaxis de unión ANSI-99 SQL
  - Explicar la importancia de tener un estándar para SQL definido por ANSI
  - Describir una necesidad de negocio para combinar la información de varios orígenes de datos



# Objetivo

- Hasta ahora, su experiencia con SQL se ha limitado a consultar y devolver información de una tabla de base de datos a la vez
- Esto no sería un problema si todos los datos de la base de datos estuvieran almacenados en una sola tabla

Obtención de Datos de Varias Tablas

EMPLOYEE_ID	DEPT_ID	DEPT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
...		
110	Accounting	1700
190	Contracting	1700

Algunos nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

# Objetivo

- Sin embargo, sabe del modelado de datos que separar los datos en tablas individuales y ser capaz de asociar las tablas entre sí forma parte de la base del diseño de la base de datos

Obtención de Datos de Varias Tablas

EMPLOYEE_ID	DEPT_ID	DEPT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
...		
110	Accounting	1700
190	Contracting	1700

Algunos nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

# Objetivo

- Afortunadamente, SQL proporciona las condiciones de unión que permiten consultar la información de distintas tablas y combinarlas en un informe

Obtención de Datos de Varias Tablas

EMPLOYEE_ID	DEPT_ID	DEPT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
...		
110	Accounting	1700
190	Contracting	1700

Algunos nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

# Comandos de Unión

- Hay dos juegos de comandos o sintaxis que se pueden utilizar para realizar conexiones entre las tablas de una base de datos:
  - Uniones propiedad de Oracle
  - Uniones estándar compatibles con ANSI/ISO SQL 99
- En este curso, aprenderá a utilizar ambos juegos de comandos de unión
- Las uniones propiedad de Oracle se tratarán en el curso más adelante

# ANSI

- ANSI son las siglas de American National Standards Institute
- Fundada en 1918, ANSI es una organización privada sin ánimo de lucro que administra y coordina el sistema de evaluación de conformidad y estandarización voluntaria de EE. UU
- La misión del Instituto es mejorar tanto la competitividad global de negocios de EE. UU. como la calidad de vida de EE. UU. al promocionar y el facilitar los estándares de conformidad voluntaria y los sistemas de evaluación de conformidad, así como al proteger su integridad

**ORACLE**  
Academy

DP 6-1  
Uniones Cruzadas y Uniones Naturales

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

8

Referencia: <http://www.ansi.org/default.aspx>

# SQL

- El lenguaje de consulta estructurado (SQL) es el lenguaje de estándar del sector de procesamiento de información de los sistemas de gestión de bases de datos relacionales (RDBMS)
- El idioma fue diseñado originalmente por IBM a mediados de 1970, tuvo un uso generalizado a primeros de 1980, y se convirtió en el estándar del sector en 1986, cuando fue adoptado por ANSI



# SQL

- Hasta ahora, ha habido tres estandarizaciones de SQL de ANSI, cada una de ellas basada en la anterior
- Se denominan con el nombre del año en el que se propusieron por primera vez y son ampliamente conocidos por sus abreviaturas: ANSI-86, ANSI-92 y ANSI-99



# UNIÓN NATURAL

- Una cláusula de unión SQL combina campos de 2 (o más) tablas en una base de datos relacional
- Una unión natural se basa en todas las columnas de dos tablas que tengan el mismo nombre y selecciona las filas de las dos tablas que tengan valores equivalentes en todas las columnas coincidentes



**ORACLE**  
Academy

DP 6-1  
Uniones Cruzadas y Uniones Naturales

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

11

# UNIÓN NATURAL

- La tabla employees tiene una columna job\_id
- Esta es una referencia a la columna con el mismo nombre en la tabla jobs

employees					jobs
EMPLOYEE_ID	LAST_NAME	JOB_ID	↔	JOB_ID	JOB_TITLE
100	King	AD_PRES		AD_PRES	President
101	Kochhar	AD_VP		AD_VP	Administration Vice President
...				AD_ASST	Administration Assistant
202	Fay	MK_REP		AC_MGR	Accounting Manager
205	Higgins	AC_MGR		AC_ACCOUNT	Public Accountant
206	Gietz	AC_ACCOUNT		SA_MAN	Sales Manager

# UNIÓN NATURAL

- Como se muestra en el código de ejemplo, cuando se utiliza una unión natural, es posible unir las tablas sin tener que especificar de forma explícita las columnas de la tabla correspondiente
- Sin embargo, los nombres y los tipos de dato de ambas columnas deben ser los mismos

```
SELECT first_name, last_name, job_id, job_title  
FROM employees NATURAL JOIN jobs  
WHERE department_id > 80;
```

- Esta unión devolverá las columnas de la tabla employees y sus valores job\_title relacionados de la tabla jobs en función del valor job\_id de la columna común



DP 6-1  
Uniones Cruzadas y Uniones Naturales

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

13

La cláusula WHERE se ha agregado para aplicar una restricción adicional a una de las tablas, para limitar las filas de la salida.

# UNIÓN NATURAL

```
SELECT first_name, last_name, job_id, job_title  
FROM employees NATURAL JOIN jobs  
WHERE department_id > 80;
```

FIRST_NAME	LAST_NAME	JOB_ID	JOB_TITLE
Steven	King	AD_PRES	President
Neena	Kochhar	AD_VP	Administration Vice President
Lex	De Haan	AD_VP	Administration Vice President
Shelley	Higgins	AC_MGR	Accounting Manager
William	Gietz	AC_ACCOUNT	Public Accountant

# UNIÓN NATURAL

- Aquí se muestra otro ejemplo:

```
SELECT department_name, city  
FROM departments NATURAL JOIN  
locations;
```

- Tanto la tabla departments como la tabla locations tienen una columna, location\_id, que se utiliza para unir las dos tablas
- Observe que la columna de la unión natural no tiene que aparecer en la cláusula SELECT

DEPARTMENT_NAME	CITY
Marketing	Toronto
Sales	Oxford
IT	Southlake
Shipping	South San Francisco
Administration	Seattle
Executive	Seattle
Accounting	Seattle
Contracting	Seattle

## CROSS JOIN

- El valor CROSS JOIN SQL de ANSI/ISO SQL: 1999 une cada fila de una tabla a cada fila de la otra tabla
- El juego de resultados representa todas las posibles combinaciones de filas de las dos tablas
- Esto podría ser muy grande
- Si ejecuta CROSS JOIN en una tabla con 20 filas con una tabla con 100 filas, la consulta devolverá 2000 filas

## Ejemplo de Unión Cruzada

- La tabla employees contiene 20 filas y la tabla departments tiene 8 filas
- Al realizar un CROSS JOIN, devolverá 160 filas

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

LAST_NAME	DEPARTMENT_NAME
Abel	Administration
Davies	Administration
De Haan	Administration
Ernst	Administration
Fay	Administration
Gietz	Administration
Grant	Administration
Hartstein	Administration
Higgins	Administration
Hunold	Administration

El juego de resultados de una CROSS JOIN aúna datos de dos tablas que no están relacionados de forma lógica entre sí.

Las uniones cruzadas son útiles durante las pruebas para crear juegos de datos de gran tamaño para medir el rendimiento de la base de datos.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Unión cruzada
  - Unión Natural

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear y ejecutar una unión natural utilizando la sintaxis de unión ANSI-99 SQL
  - Crear una unión cruzada utilizando la sintaxis de unión ANSI-99 SQL
  - Explicar la importancia de tener un estándar para SQL definido por ANSI
  - Describir una necesidad de negocio para combinar la información de varios orígenes de datos



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

6-2

Cláusulas Join

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear y ejecutar una unión con la cláusula ANSI-99 USING
  - Crear y ejecutar una unión con la cláusula ANSI-99 ON
  - Crear y ejecutar una consulta ANSI-99 que une tres tablas



# Objetivo

- Al agregar más comandos al vocabulario de la base de datos, estará mejor preparado para diseñar consultas que devuelvan el resultado deseado
- El objetivo de una unión es enlazar los datos, entre tablas, sin repetir todos los datos en todas las tablas
- ¿Para qué solicitar más datos de los que realmente necesita?

## Cláusula USING

- En una unión natural, si las tablas tienen columnas con los mismos nombres, pero diferentes tipos de dato, la unión provoca un error
- Para evitar esta situación, la cláusula de unión se puede modificar con una cláusula USING
- La cláusula USING especifica las columnas que se deben utilizar para la unión

Se suele preferir una cláusula USING a una unión natural, incluso aunque las columnas tengan el mismo tipo de dato, así como el mismo nombre, debido a que indica claramente exactamente qué columna de unión se está utilizando.

## Cláusula USING

- La consulta que se muestra es un ejemplo de la cláusula USING
- Las columnas a las que se hace referencia en la cláusula USING no deben tener un cualificador (nombre o alias de la tabla) en ninguna ubicación de la sentencia SQL

```
SELECT first_name, last_name, department_id, department_name  
FROM employees JOIN departments USING (department_id);
```

FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Jennifer	Whalen	10	Administration
Michael	Hartstein	20	Marketing
Pat	Fay	20	Marketing
...	...	...	...



Academy

DP 6-2  
Cláusulas Join

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

6

Si la columna de la cláusula USING tiene un cualificador, se devuelve el siguiente error

ORA-25154: column part of USING clause cannot have qualifier.

Nota: La última línea de la salida de ejemplo utiliza ... .... .... para indicar que se devuelven más datos de los que se muestran en la diapositiva.

## Cláusula USING

- La cláusula USING nos permite utilizar WHERE para limitar las filas de una o de ambas tablas:

```
SELECT first_name, last_name, department_id, department_name  
FROM employees JOIN departments USING (department_id)  
WHERE last_name = 'Higgins';
```

FIRST_NAME	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Shelley	Higgins	110	Accounting

## Cláusula ON

- ¿Qué ocurre si las columnas que se van a unir tienen nombres diferentes, o bien si la unión utiliza operadores de comparación de distinto de, como <, > o BETWEEN?
- No podemos utilizar USING, por lo que en su lugar utilizamos una cláusula ON
- Esto permite especificar una mayor variedad de condiciones de unión
- La cláusula ON también nos permite utilizar WHERE para limitar las filas de una o de ambas tablas

## Ejemplo de Cláusula ON

- En este ejemplo, la cláusula ON se utiliza para unir la tabla employees con la tabla jobs

```
SELECT last_name, job_title  
FROM employees e JOIN jobs j  
ON (e.job_id = j.job_id);
```

- Se necesita una cláusula ON cuando las columnas comunes tengan nombres diferentes en las dos tablas

LAST_NAME	JOB_TITLE
King	President
Kochhar	Administration Vice President
De Haan	Administration Vice President
Whalen	Administration Assistant
Higgins	Accounting Manager
Gietz	Public Accountant
Zlotkey	Sales Manager
Abel	Sales Representative
Taylor	Sales Representative
...	

## Ejemplo de Cláusula ON

- Al usar una cláusula ON en columnas con el mismo nombre en ambas tablas, debe agregar un calificador (ya sea el alias o el nombre de la tabla), de lo contrario se devolverá un error. En el ejemplo, los alias de la tabla se usan como el calificador e.job\_id = j.job\_id, pero también podrían haberse escrito usando los nombres de la tabla  
(employees.job\_id = jobs.job\_id)

```
SELECT last_name, job_title
FROM employees e JOIN jobs j
ON (e.job_id = j.job_id);
```

LAST_NAME	JOB_TITLE
King	President
Kochhar	Administration Vice President
De Haan	Administration Vice President
Whalen	Administration Assistant
Higgins	Accounting Manager
Gietz	Public Accountant
Zlotkey	Sales Manager
Abel	Sales Representative
Taylor	Sales Representative
...	

## Cláusula ON con Cláusula WHERE

- Aquí se muestra la misma consulta con una cláusula WHERE para limitar las filas seleccionadas

```
SELECT last_name, job_title  
FROM employees e JOIN jobs j  
  ON (e.job_id = j.job_id)  
WHERE last_name LIKE 'H%';
```

LAST_NAME	JOB_TITLE
Higgins	Accounting Manager
Hunold	Programmer
Hartstein	Marketing Manager

Nota: Debido a que la columna job\_id de las tablas employees y jobs tienen el mismo nombre y tipo de dato, la consulta anterior también se podría escribir con NATURAL JOIN o JOIN USING.

## Cláusula ON con Operador Distinto de

- En ocasiones puede que tenga que recuperar los datos de una tabla que no tenga ninguna columna correspondiente en otra tabla
- Suponga que desea conocer el valor grade\_level para el salario de cada empleado
- La tabla job\_grades no tiene una columna común con la tabla employees
- Una cláusula ON nos permite unir las dos tablas

tabla job\_grades

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

# Cláusula ON con Operador Distinto de

```
SELECT last_name, salary, grade_level, lowest_sal,  
highest_sal  
FROM employees JOIN job_grades  
ON(salary BETWEEN lowest_sal AND highest_sal);
```

LAST_NAME	SALARY	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
Vargas	2500	A	1000	2999
Matos	2600	A	1000	2999
Davies	3100	B	3000	5999
Rajs	3500	B	3000	5999
Lorentz	4200	B	3000	5999
Whalen	4400	B	3000	5999
Mourgos	5800	B	3000	5999
Fay	6000	C	6000	9999
...				

Con una cláusula JOIN ON, puede utilizar operadores distintos de "=". Por ejemplo, >=, <=, BETWEEN.

## Unión de Tres Tablas

- Tanto USING como ON se pueden utilizar para unir tres o más tablas
- Supongamos que necesitamos un informe de nuestros empleados, su departamento y la ciudad donde está ubicado el departamento
- Necesitamos unir tres tablas: employees, departments y locations



## Ejemplo de la Unión de Tres Tablas

```
SELECT last_name, department_name AS "Department", city  
FROM employees JOIN departments USING (department_id)  
JOIN locations USING (location_id);
```



LAST_NAME	Departemen	CITY
Hartstein	Marketing	Toronto
Fay	Marketing	Toronto
Zlotkey	Sales	Oxford
Abel	Sales	Oxford
Taylor	Sales	Oxford
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
...		

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Cláusula ON
  - Cláusula USING

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear y ejecutar una unión con la cláusula ANSI-99 USING
  - Crear y ejecutar una unión con la cláusula ANSI-99 ON
  - Crear y ejecutar una consulta ANSI-99 que une tres tablas



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

6-3

## Uniones Internas frente a Externas

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Comparar y contrastar una unión interna con una unión externa
  - Crear y ejecutar una consulta para utilizar una unión externa izquierda
  - Crear y ejecutar una consulta para utilizar una unión externa derecha
  - Crear y ejecutar una consulta para utilizar una unión externa completa



## Objetivo

- Hasta ahora, todas las uniones han devuelto datos que coincidían con la condición de unión
- Sin embargo, en ocasiones, deseamos recuperar tanto los datos que cumplen la condición de unión, como los datos que no cumplen la condición de unión
- Las uniones externas en ANSI-99 SQL permiten esta funcionalidad

# Uniones Internas y Externas

- En ANSI-99 SQL, una unión de dos o más tablas que devuelven solo las filas coincidentes se denomina unión interna
- Cuando una unión devuelve las filas no coincidentes, así como las filas coincidentes, se denomina unión externa
- En la sintaxis de las uniones externas se utilizan los términos "izquierda, completa y derecha"
- Estos nombres están asociados con el orden de los nombres de tablas en la cláusula FROM de la sentencia SELECT

NATURAL JOIN, JOIN ON y JOIN USING son tipos de uniones internas.

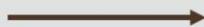
# Uniones Externas Izquierdas y Derechas



- En el ejemplo mostrado de una unión externa izquierda, tenga en cuenta que al nombre de la tabla que aparece a la izquierda de las palabras "left outer join" se le hace referencia como "tabla izquierda"

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

LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Fay	20	Marketing
...		
Zlotkey	80	Sales
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant	-	-



Los nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

# Uniones Externas Izquierdas y Derechas



- Esta consulta devolverá los apellidos de todos los empleados, tanto aquellos que estén asignados a un departamento como los que no

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

LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Fay	20	Marketing
...		
Zlotkey	80	Sales
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant	-	-

Los nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

# Uniones Externas Izquierdas y Derechas



- Esta unión externa derecha devolvería todos los ID de departamento y los nombres de departamento, tanto aquellos que tengan empleados asignados como los que no

```
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	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
...		
King	90	Executive
Kochhar	90	Executive
De Haan	90	Executive
Higgins	110	Accounting
Gietz	110	Accounting
-	190	Contracting

Los nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

## Unión Externa Completa

- Se puede crear una condición de unión para recuperar todas las filas coincidentes y todas las filas no coincidentes de ambas tablas
- Con una unión externa completa se resuelve este problema
- El juego de resultados de una unión externa completa incluye todas las filas de una unión externa izquierda y todas las filas de una unión externa derecha combinadas sin duplicación



## Ejemplo de FULL OUTER JOIN



- En el ejemplo se muestra una unión externa completa.

```
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	DEPT_ID	DEPT_NAME
King	90	Executive
Kochhar	90	Executive
...		
Taylor	80	Sales
Grant	-	-
Mourgos	50	Shipping
...		
Fay	20	Marketing
-	190	Contracting

Los nombres de columna en la salida de ejemplo se han abreviado para que quepan en la diapositiva.

## Caso de Unión

- Cree una unión para mostrar una lista de empleados, su valor job\_id actual y cualquier trabajo anterior que hayan tenido
- La tabla job\_history contiene detalles de los trabajos anteriores de un empleado

```
SELECT last_name, e.job_id AS "Job", jh.job_id AS "Old job",
end_date
FROM employees e LEFT OUTER JOIN job_history jh
ON (e.employee_id = jh.employee_id);
```

LAST_NAME	Job	Old job	END_DATE
King	AD_PRES	-	-
Kochhar	AD_VP	AC_MGR	15-Mar-1997
Kochhar	AD_VP	AC_ACCOUNT	27-Oct-1993
De Haan	AD_VP	IT_PROG	24-Jul-1998
Whalen	AD_ASST	AD_ASST	17-Jun-1993
Whalen	AD_ASST	AC_ACCOUNT	31-Dec-1998
Higgins	AC_MGR	-	-

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - FULL OUTER JOIN
  - Unión interna
  - LEFT OUTER JOIN
  - Unión externa
  - RIGHT OUTER JOIN

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Comparar y contrastar una unión interna con una unión externa
  - Crear y ejecutar una consulta para utilizar una unión externa izquierda
  - Crear y ejecutar una consulta para utilizar una unión externa derecha
  - Crear y ejecutar una consulta para utilizar una unión externa completa



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

6-4

## Autouniones y Consultas Jerárquicas

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear y ejecutar una sentencia SELECT para unir una tabla consigo misma mediante una autounión
  - Interpretar el concepto de una consulta jerárquica
  - Crear un informe con estructura de árbol
  - Aplicar formato a datos jerárquicos
  - Excluir ramas de la estructura de árbol



# Objetivo

- En el modelado de datos, a veces es necesario mostrar una entidad con una relación consigo misma
- Por ejemplo, un empleado también puede ser un jefe
- Mostramos esto con la relación recursiva o de "oreja de cerdo"



# Objetivo

- Una vez que tengamos una verdadera tabla employees, es necesario un tipo especial de unión denominada autounión para acceder a esos datos
- Se utiliza una autounión para unir una tabla a sí misma como si se tratara de dos tablas

```
SELECT worker.last_name || ' works for ' || manager.last_name  
AS "Works for"  
FROM employees worker JOIN employees manager  
ON (worker.manager_id = manager.employee_id);
```

# AUTOUNIÓN

- Para unir una tabla a sí misma, a la tabla se le asignan dos nombres o alias. Esto hará que la base de datos "crea" que hay dos tablas

EMPLOYEES (worker)

employee_id	last_name	manager_id
100	King	
101	Kochar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

EMPLOYEES (manager)

employee_id	last_name
100	King
101	Kochar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

- Manager\_id en la tabla worker es igual a employee\_id en la tabla manager

# AUTOUNIÓN

- Seleccione los nombres de alias relacionados con la asociación de datos con esa tabla

EMPLOYEES (worker)

employee_id	last_name	manager_id
100	King	
101	Kochar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

EMPLOYEES (manager)

employee_id	last_name
100	King
101	Kochar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

- Manager\_id en la tabla worker es igual a employee\_id en la tabla manager

## Ejemplo de SELF-JOIN

```
SELECT worker.last_name, worker.manager_id, manager.last_name  
      AS "Manager name"  
FROM employees worker JOIN employees manager  
  ON (worker.manager_id = manager.employee_id);
```

LAST_NAME	MANAGER_ID	Manager name
Kochhar	100	King
De Haan	100	King
Zlotkey	100	King
Mourgos	100	King
Hartstein	100	King
Whalen	101	Kochhar
Higgins	101	Kochhar
Hunold	102	De Haan
...	...	...

## Consultas Jerárquicas

- Estrechamente relacionadas con las autouniones se encuentran las consultas jerárquicas
- En las páginas anteriores, ha visto cómo puede utilizar las autouniones para ver al jefe directo de cada empleado
- Con las consultas jerárquicas, también podemos ver para quién trabaja ese jefe, y así sucesivamente
- Con este tipo de consulta, podemos crear un diagrama de organización que muestre la estructura de una compañía o de un departamento

# Consultas Jerárquicas

- Imagine un árbol familiar en el que los miembros más mayores de la familia se encuentran cerca de la base o el tronco del árbol y los más jóvenes representan las ramas del árbol
- Las ramas pueden tener sus propias ramas y así sucesivamente



# Uso de Consultas Jerárquicas

- Con las consultas jerárquicas, puede recuperar datos según una relación jerárquica natural entre las filas de una tabla
- Una base de datos relacional no almacena los registros de forma jerárquica
- Sin embargo, cuando existe una relación jerárquica entre las filas de una única tabla, un proceso denominado recorrido por el árbol permite construir la jerarquía
- Una consulta jerárquica es un método de creación de informes, con las ramas de un árbol en un determinado orden

Entre las áreas del mundo real donde se podrían utilizar los recorridos por árboles jerárquicos se incluyen: genealogía humana (árboles familiares), ganado (para crianza), gestión corporativa (jerarquías de gestión), fabricación (ensamblaje de productos).

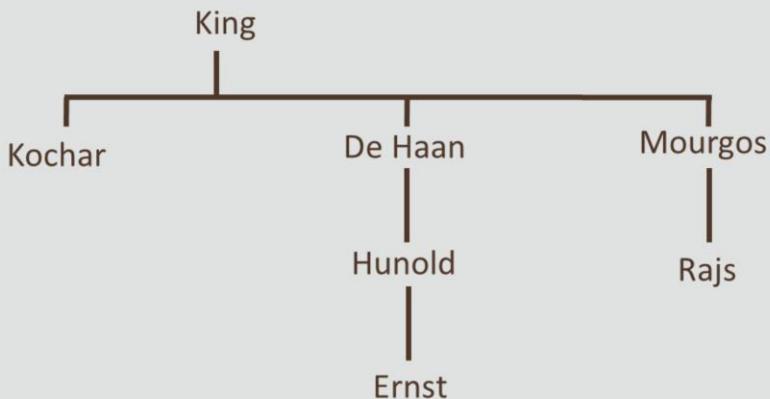
## Datos de Consultas Jerárquicas

- Examine los datos de ejemplo de la tabla EMPLOYEES siguiente y observe cómo puede realizar manualmente las conexiones para ver quién trabaja para quién, empezando por Steven King y desplazándose por el árbol desde ahí

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMM_PCT	MGR_ID	DEPT_ID
100	Steven	King	SKING	515.123.4567	17-Jun-1987	AD_PRES	24000	(null)	(null)	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-Sep-1989	AD_VP	17000	(null)	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-Jan-1993	AD_VP	17000	(null)	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-Jan-1990	IT_PROG	9000	(null)	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-May-1991	IT_PROG	6000	(null)	103	60
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-Nov-1999	ST_MAN	5800	(null)	100	50
141	Trenna	Rajs	TRAJS	650.121.8009	17-Oct-1995	ST_CLERK	3500	(null)	124	50

## Ilustración de Consultas Jerárquicas

- El diagrama de organización que podemos extraer a partir de los datos de la tabla EMPLOYEES será similar a este:



# Palabras Clave de las Consultas Jerárquicas

- Las consultas jerárquicas tienen sus propias palabras clave nuevas: START WITH, CONNECT BY PRIOR y LEVEL
- START WITH identifica qué fila se va a utilizar como raíz del árbol que se está creando; CONNECT BY PRIOR explica cómo realizar las uniones entre filas y LEVEL especifica cuántas ramas de profundidad recorrerá el árbol



**ORACLE**  
Academy

DP 6-4  
Autouniones y Consultas Jerárquicas

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

14

LEVEL: es una pseudocolumna (lo que significa que no existe realmente) que devuelve 1 para la raíz del árbol, 2 para un nivel inferior, 3 para el tercer nivel, y así sucesivamente.

## Ejemplo de Palabras Clave en Consultas Jerárquicas

```
SELECT employee_id, last_name, job_id, manager_id  
FROM employees  
START WITH employee_id = 100  
CONNECT BY PRIOR employee_id = manager_id
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
100	King	AD_PRES	-
101	Kochhar	AD_VP	100
200	Whalen	AD_ASST	101
205	Higgins	AC_MGR	101
206	Gietz	AC_ACCOUNT	205
102	De Haan	AD_VP	100
103	Hunold	IT_PROG	102
104	Ernst	IT_PROG	103

## Otro Ejemplo de Consultas Jerárquicas

```
SELECT last_name ||' reports to ' || PRIOR last_name AS "Walk  
Top Down"  
FROM employees  
START WITH last_name = 'King'  
CONNECT BY PRIOR employee_id = manager_id;
```

Walk Top Down
King reports to
Kochhar reports to King
Whalen reports to Kochhar
Higgins reports to Kochhar
Gietz reports to Higgins
De Haan reports to King
Hunold reports to De Haan
Ernst reports to Hunold

# Ejemplo de LEVEL en Consultas Jerárquicas

- LEVEL es una pseudocolumna que se utiliza con consultas jerárquicas y que cuenta el número de pasos que ha realizado desde la raíz del árbol

```
SELECT LEVEL, last_name ||  
' reports to ' ||  
PRIOR last_name  
    AS "Walk Top Down"  
FROM employees  
START WITH last_name = 'King'  
CONNECT BY PRIOR  
    employee_id = manager_id;
```

LEVEL	Walk Top Down
1	King reports to
2	Kochhar reports to King
3	Whalen reports to Kochhar
3	Higgins reports to Kochhar
4	Gietz reports to Higgins
2	De Haan reports to King
3	Hunold reports to De Haan
4	Ernst reports to Hunold

# Informe de Consulta Jerárquica

- Si deseara crear un informe que mostrara los niveles de dirección de la compañía, empezando por el nivel más alto y sangrando cada uno de los siguientes niveles, sería fácil hacer esto con la pseudocolumna LEVEL y la función LPAD para sangrar los empleados en función de su nivel

```
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')
  AS "Org Chart"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR employee_id = manager_id;
```

# Niveles de Salida de la Consulta Jerárquica

- Como puede ver en el resultado de la derecha, cada fila se sangra con dos caracteres de subrayado por nivel

```
SELECT LPAD(last_name, LENGTH(last_name)+  
          (LEVEL*2)-2, '_') AS "Org_Chart"  
FROM employees  
START WITH last_name = 'King'  
CONNECT BY PRIOR employee_id = manager_id;
```

Org_Chart
King
Kochhar
Whalen
Higgins
Gietz
De Haan
Hunold
Ernst
Lorentz
Rajs
Davies
Matos
Vargas
Zlotkey
Abel
Taylor
Grant
Hartstein
Fay

## Consulta Jerárquica de Abajo a Arriba

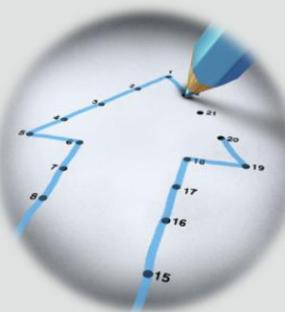
- Como puede ver en el resultado que aparece a continuación, en este ejemplo se muestra cómo crear una consulta jerárquica de abajo a arriba, moviendo la palabra clave PRIOR a la posición posterior al signo igual y mediante el uso de 'Grant' en la cláusula START WITH

```
SELECT LPAD(last_name, LENGTH(last_name) +  
(LEVEL*2)-2, '_') AS ORG_CHART  
FROM employees  
START WITH last_name = 'Grant'  
CONNECT BY employee_id = PRIOR manager_id
```

ORG_CHART
Grant
__Zlotkey
___King

# Depuración de Consultas Jerárquicas

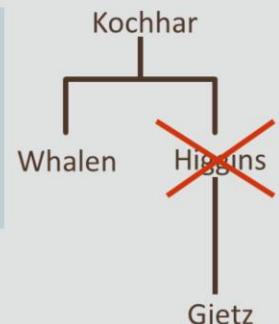
- La depuración de ramas del árbol se puede realizar mediante la cláusula WHERE o la cláusula CONNECT BY PRIOR
- Si se utiliza la cláusula WHERE, solo se excluye la fila especificada en la sentencia; si se utiliza la cláusula CONNECT BY PRIOR, se excluye toda la rama



# Depuración de Consultas Jerárquicas

- Por ejemplo, si desea excluir una única fila de su resultado, debería utilizar la cláusula WHERE para excluir esa fila; sin embargo, en el resultado, parecería que Gietz hubiera trabajado directamente para Kochhar, lo que no refleja la realidad

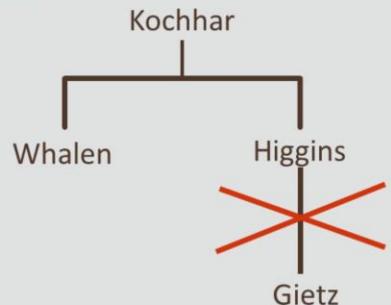
```
SELECT last_name  
FROM employees  
WHERE last_name != 'Higgins'  
START WITH last_name = 'Kochhar'  
CONNECT BY PRIOR employee_id = manager_id;
```



# Depuración de Consultas Jerárquicas

- Sin embargo, si deseara excluir una fila y todas las filas por debajo de ella, debería realizar la parte de exclusión de la sentencia CONNECT BY
- En este ejemplo que excluye a Higgins, también estamos excluyendo a Gietz del resultado

```
SELECT last_name  
FROM employees  
START WITH last_name = 'Kochhar'  
CONNECT BY PRIOR employee_id = manager_id  
AND last_name != 'Higgins';
```



# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Connect By prior
  - Consultas jerárquicas
  - Nivel
  - Autounión
  - Start with
  - Estructura del árbol
  - Recorrido por el árbol
  - Ramas

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear y ejecutar una sentencia SELECT para unir una tabla consigo misma mediante una autounión
  - Interpretar el concepto de una consulta jerárquica
  - Crear un informe con estructura de árbol
  - Aplicar formato a datos jerárquicos
  - Excluir ramas de la estructura de árbol



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

7-1

Unión Igualitaria y  
Producto Cartesiano de Oracle

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección, aprenderá a:

- Nombrar las uniones propiedad de Oracle y sus equivalentes en ANSI/ISO SQL: 99
- Crear y ejecutar una sentencia SELECT que da como resultado un producto cartesiano
- Crear y ejecutar sentencias SELECT para acceder a los datos desde más de una tabla utilizando una unión igualitaria
- Crear y ejecutar sentencias SELECT que agregan condiciones de búsqueda usando el operador AND
- Aplicar la regla para utilizar alias de tabla en una sentencia de unión



## Objetivo

- En la sección anterior se ha tratado la consulta y devolución de los datos de más de una tabla en una base de datos relacional utilizando sintaxis ANSI/ISO SQL: 99
- En las versiones anteriores de las bases de datos Oracle se necesitaba que las uniones utilizaran la sintaxis de unión propiedad de Oracle y muchas de estas bases de datos anteriores aún se utilizan
- En esta lección se presenta la sintaxis de unión propiedad de Oracle para uniones igualitarias y el producto cartesiano y sus homólogos ANSI/ISO SQL: 99



DP 7-1  
Unión Igualitaria y Producto Cartesiano de Oracle

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

4

En las versiones anteriores a Oracle 9i Database se necesita el uso de sintaxis de unión propiedad de Oracle.

# Comandos de Unión

- Los dos juegos de comandos o sintaxis que se pueden utilizar para realizar conexiones entre las tablas de una base de datos:
  - Uniones propiedad de Oracle
  - Uniones estándar compatibles con ANSI/ISO SQL: 99



# Comparación de Unión

- Comparación de las Uniones Propiedad de Oracle con Uniones ANSI/ISO SQL: 1999

Unión Propiedad de Oracle	Equivalente de ANSI/ISO SQL: 1999
Producto cartesiano	Unión cruzada
Unión igualitaria	UNIÓN NATURAL Cláusula JOIN USING Cláusula JOIN ON (si se utiliza el operador de igualdad)
Unión no igualitaria	Cláusula ON

# Uniones Propiedad de ORACLE

- Para consultar datos de más de una tabla con la sintaxis propiedad de Oracle, utilice una condición de unión en la cláusula WHERE
- El formato básico de una sentencia de unión es:

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column1 = table2.column2;
```

## Uniones Propiedad de ORACLE

- Imagine el problema que supondría que dos alumnos de la misma clase tuvieran el mismo apellido
- Cuando sea necesario hablar con "Jackson", el profesor aclara de qué "Jackson" se trata agregando el apellido antes del nombre
- Para que sea más fácil leer una sentencia Join y acelerar el acceso a la base de datos, es una buena práctica agregar el nombre de la tabla delante del nombre de la columna

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column1 = table2.column2;
```

ORACLE

Academy

DP 7-1

Unión Igualitaria y Producto Cartesiano de Oracle

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

8

# Uniones Propiedad de ORACLE

- A esto se le denomina "cualificar sus columnas"
- La combinación del nombre de tabla y el nombre de columna ayuda a eliminar nombres ambiguos cuando dos tablas contienen una columna con el mismo nombre de columna
- Si aparece el mismo nombre de columna en ambas tablas, el nombre de columna debe ir precedido del nombre de la tabla

## Ejemplo de Sintaxis de Unión

- Para cualificar las columnas, utilice la sintaxis nombretabla.nombrecolumna, como se muestra en el siguiente ejemplo

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column1 = table2.column2;
```

# UNIÓN IGUALITARIA

- Algunas veces denominada unión "simple" o "interna", una unión igualitaria es una unión de tabla que combina filas con los mismos valores para las columnas especificadas
- Una unión igualitaria es equivalente a ANSI:
  - NATURAL JOIN
  - JOIN USING
  - JOIN ON (cuando la condición de unión utiliza "=")
- En la siguiente diapositiva se muestran el qué, el dónde y el cómo necesarios para unir las tablas

## UNIÓN IGUALITARIA

- ¿Qué? La cláusula SELECT especifica los nombres de columna que se van a mostrar
- ¿Dónde? La cláusula FROM especifica las tablas a las que debe acceder la base de datos, separadas por comas
- ¿Cómo? La cláusula WHERE especifica cómo se van a unir las tablas
- Una unión igualitaria utiliza el operador Igual que para especificar la condición de unión

# UNIÓN IGUALITARIA

```
SELECT employees.last_name, employees.job_id, jobs.job_title; ¿Qué?  
FROM employees, jobs           ¿Dónde?  
WHERE employees.job_id = jobs.job_id; ¿Cómo?
```

LAST_NAME	JOB_ID	JOB_TITLE
King	AD_PRES	Presiden
Kochhar	AD_VP	Administration Vice President
De Haan	AD_VP	Administration Vice President
Whalen	AD_ASST	Administration Assistant
Higgins	AC_MGR	Accounting Manager
Gietz	AC_ACCOUNT	Public Accountant
Zlotkey	SA_MAN	Sales Manager
Abel	SA_REP	Sales Representative
...	...	...

# UNIÓN IGUALITARIA

- Otro ejemplo:

```
SELECT employees.last_name, departments.department_name  
FROM employees, departments  
WHERE employees.department_id = departments.department_id;
```

LAST_NAME	DEPARTMENT_NAME
Whalen	Administration
Hartstein	Marketing
Fay	Marketing
Mourgos	Shipping
Rajs	Shipping
Davies	Shipping
Matos	Shipping
...	...



Nota: La columna utilizada para unir ambas tablas no es necesario que esté en la lista de columnas SELECT.

## Alias

- Trabajar con nombres de tabla y columna largos puede ser complicado
- Afortunadamente, hay una forma de acortar la sintaxis utilizando alias
- Para distinguir las columnas que tienen nombres idénticos, pero que residen en tablas diferentes, utilice alias de tabla
- Un alias de tabla es similar a un alias de columna; cambia el nombre de un objeto dentro de una sentencia
- Se crea mediante la introducción del nuevo nombre para la tabla, justo después del nombre de tabla en la cláusula `from`

## Alias de Tabla

- Los alias de la tabla se utilizan en la consulta siguiente

```
SELECT last_name, e.job_id, job_title  
FROM employees e, jobs j  
WHERE e.job_id = j.job_id  
AND department_id = 80;
```

LAST_NAME	JOB_ID	JOB_TITLE
Zlotkey	SA_MAN	Sales Manager
Abel	SA_REP	Sales Representative
Taylor	SA_REP	Sales Representative

- Cuando los nombres de columna no están duplicados en dos tablas, no tiene que agregar el alias o nombre de la tabla al nombre de la columna

## Alias de Tabla

- Si se utiliza un alias de tabla en la cláusula FROM, el alias de tabla se deberá sustituir por el nombre de tabla mediante la sentencia SELECT
- Si se utiliza el nombre de una tabla en la cláusula SELECT al que se le haya asignado un alias en la cláusula FROM, se producirá un error

```
SELECT last_name, employees.job_id, job_title  
FROM employees e, jobs j  
WHERE e.job_id = j.job_id  
AND department_id = 80;
```



ORA-00904: "EMPLOYEES"."JOB\_ID": invalid identifier

**ORACLE**  
Academy

DP 7-1  
Unión Igualitaria y Producto Cartesiano de Oracle

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

17

## Unión de Producto Cartesiano

- Si en la cláusula WHERE de dos tablas de una consulta de unión no se ha especificado ninguna condición de unión o la condición de unión no es válida, Oracle Server devuelve el producto cartesiano de las dos tablas
- Esta es una combinación de cada fila de una tabla con cada fila de otra
- Un producto cartesiano es equivalente a un ANSI CROSS JOIN
- Para evitar un producto cartesiano, incluya siempre una condición de unión válida en una cláusula WHERE

# Unión de Producto Cartesiano

- En esta consulta, la condición de unión se ha omitido:

```
SELECT employees.last_name, departments.department_name  
FROM employees, departments;
```

LAST_NAME	DEPARTMENT_NAME
Abel	Administration
Davies	Administration
De Haan	Administration
Ernst	Administration
Fay	Administration
Gietz	Administration
Grant	Administration
...	...

160 rows returned in 0.01 seconds

## Restricción de las Filas de una Unión

- Al igual que ocurre con las consultas de una sola tabla, la cláusula WHERE se puede utilizar para restringir las filas tenidas en cuenta en una o más tablas de la unión
- En la consulta mostrada se utiliza el operador AND para limitar las filas devueltas

```
SELECT employees.last_name, employees.job_id, jobs.job_title  
FROM employees, jobs  
WHERE employees.job_id = jobs.job_id  
      AND employees.department_id = 80;
```

LAST_NAME	JOB_ID	JOB_TITLE
Zlotkey	SA_MAN	Sales Manager
Abel	SA_REP	Perwakilan Penjualan
Taylor	SA_REP	Perwakilan Penjualan

## Ejemplo de Sintaxis de Unión

- Si deseara unir tres tablas juntas, ¿cuántas uniones necesitaría?
- ¿Cuántos puentes se necesitan para unir tres islas?
- Para unir tres tablas, tendrá que agregar otra condición de unión a la cláusula WHERE utilizando el operador AND



## Ejemplo de Sintaxis de Unión

- Supongamos que necesitamos un informe de nuestros empleados y la ciudad donde está ubicado su departamento
- Necesitamos unir tres tablas: employees, departments y locations

```
SELECT last_name, city
FROM employees e, departments d,
     locations l
WHERE e.department_id = d.department_id
    AND d.location_id = l.location_id;
```

LAST_NAME	CITY
Hartstein	Toronto
Fay	Toronto
Zlotkey	Oxford
Abel	Oxford
...	...

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Alias
  - Producto cartesiano
  - Unión igualitaria
  - Condiciones de unión
  - Unión de propiedad

# Resumen

- En esta lección, ha aprendido lo siguiente:
  - Nombrar las uniones propiedad de Oracle y sus equivalentes en ANSI/ISO SQL: 99
  - Crear y ejecutar una sentencia SELECT que da como resultado un producto cartesiano
  - Crear y ejecutar sentencias SELECT para acceder a los datos desde más de una tabla utilizando una unión igualitaria
  - Crear y ejecutar sentencias SELECT que agregan condiciones de búsqueda usando el operador AND
  - Aplicar la regla para utilizar alias de tabla en una sentencia de unión



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

7-2

**Uniones No Igualitarias y  
Uniones Externas de Oracle**

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección, aprenderá a:
  - Crear y ejecutar una sentencia SELECT para acceder a los datos desde más de una tabla utilizando una unión no igualitaria
  - Crear y ejecutar una sentencia SELECT para acceder a los datos desde más de una tabla utilizando una unión externa de Oracle



## Objetivo

- ¿Qué ocurre si desea recuperar los datos de una tabla que no tiene ninguna columna correspondiente en otra tabla?
- Por ejemplo, su nota porcentual en matemáticas (92) está almacenada en la columna GRADES de una tabla; la nota con letra está almacenada en la columna LETTER\_GRADE de otra tabla
- ¿Cómo podemos unir la nota numérica con la nota con letra?
- Cuando los datos se registran utilizando un rango, la recuperación es la tarea de una unión no igualitaria

## Objetivo

- Las uniones de Oracle que ha estudiado hasta el momento han devuelto filas con un valor coincidente en ambas tablas
- Las filas que no cumplían esas condiciones simplemente no se incluyeron
- Sin embargo, en ocasiones, desea que se devuelvan todos los datos de una de las tablas, incluso aunque no haya datos coincidentes en la otra tabla
- En esta lección también se tratarán las uniones externas de Oracle para solucionar este problema

# Unión no igualitaria

- Ejemplo:

- Suponga que desea conocer el valor grade\_level para el salario de cada empleado
- La tabla job\_grades no tiene una columna común con la tabla employees
- Una unión igualitaria nos permite unir las dos tablas

tabla job\_grades

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

## Unión no igualitaria

- Ya que no hay ninguna coincidencia exacta entre las dos columnas de cada tabla, el operador de igualdad = no se puede utilizar
- Aunque se pueden utilizar las condiciones de comparación como  $<=$  y  $>=$ , BETWEEN... AND es una forma más efectiva de ejecutar una unión no igualitaria
- Una unión no igualitaria es equivalente a ANSI JOIN ON (donde la condición utilizada es algo distinto al signo igual)

## Unión no igualitaria

```
SELECT last_name, salary, grade_level, lowest_sal,  
highest_sal  
FROM employees, job_grades  
WHERE (salary BETWEEN lowest_sal AND highest_sal);
```

LAST_NAME	SALARY	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
Vargas	2500	A	1000	2999
Matos	2600	A	1000	2999
Davies	3100	B	3000	5999
Rajs	3500	B	3000	5999
Lorentz	4200	B	3000	5999
Whalen	4400	B	3000	5999
Mourgos	5800	B	3000	5999
Fay	6000	C	6000	9999
...				

## Unión Externa

- Una unión externa se utiliza para ver las filas que tengan un valor correspondiente en otra tabla, más aquellas filas de una de las tablas que no tengan ningún valor coincidente en la otra tabla
- Para indicar qué tabla puede tener datos que faltan mediante la sintaxis de unión de Oracle, agregue un signo más (+) después del nombre de columna de la tabla en la cláusula WHERE de la consulta



# Unión Externa

- Esta consulta devolverá los apellidos de todos los empleados, incluidos aquellos que estén asignados a un departamento y los que no
- Se podrían obtener los mismos resultados mediante ANSI LEFT OUTER JOIN

```
SELECT e.last_name,
       d.department_id,
       d.department_name
  FROM employees e, departments d
 WHERE e.department_id =
       d.department_id(+);
```

LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Vargas	50	Shipping
...		
Higgins	110	Accounting
Grant	-	-

# Unión Externa

- Esta unión externa devolvería todos los ID de departamento y los nombres de departamento, tanto aquellos que tengan empleados asignados como los que no
- Se podrían obtener los mismos resultados mediante ANSI RIGHT OUTER JOIN

```
SELECT e.last_name,
d.department_id,
d.department_name
FROM employees e, departments d
WHERE e.department_id =(+)
      d.department_id(+);
```

LAST_NAME	DEPT_ID	DEPT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
...		
Gietz	110	Accounting
-	190	Contracting



Academy

DP 7-2  
Uniones No Igualitarias y  
Uniones Externas de Oracle

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

11

## Unión Externa

- No es posible tener el equivalente de FULL OUTER JOIN mediante la adición de un signo (+) a ambas columnas de la condición de unión
- Si se intenta realizar esto, se produce un error

```
SELECT e.last_name, d.department_id, d.department_name  
FROM employees e, departments d  
WHERE e.department_id(+) = d.department_id(+);
```



ORA-01468: a predicate may reference only one outer-joined table

Es posible realizar una unión externa completa mediante operadores SET. Estos se tratarán más adelante en este curso.

## Unión Externa

- Se muestran las variaciones de sintaxis de la unión externa

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column = table2.column(+);
```

```
SELECT table1.column, table2.column  
FROM table1, table2  
WHERE table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column  
FROM table1, table2  
NEVER table1.column(+) = table2.column(+);
```



# Unión Externa y Equivalentes ANSI

- En la siguiente tabla se muestran las uniones ANSI/ISO SQL: 99 y sus uniones externas equivalentes de Oracle

ANSI/ISO SQL	Sintaxis de Oracle
LEFT OUTER JOIN departments d ON (e.department_id = d.department_id);	WHERE e.department_id = d.department_id(+);
RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id);	WHERE e.department_id(+) = d.department_id;
FULL OUTER JOIN departments d ON (e.department_id = d.department_id);	Ningún equivalente directo

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Unión no igualitaria
  - BETWEEN...AND
  - Uniones Externas

# Resumen

- En esta lección, ha aprendido lo siguiente:
  - Crear y ejecutar una sentencia SELECT para acceder a los datos desde más de una tabla utilizando una unión no igualitaria
  - Crear y ejecutar una sentencia SELECT para acceder a los datos desde más de una tabla utilizando una unión externa de Oracle



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

8-1

## Funciones de Grupo

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Definir y proporcionar un ejemplo de las siete funciones de grupo: SUM, AVG, COUNT, MIN, MAX, STDDEV y VARIANCE
  - Crear y ejecutar una consulta SQL utilizando funciones de grupo
  - Crear y ejecutar funciones de grupo que solo funcionan con tipos de dato numéricos



## Objetivo

- ¿Qué ocurriría si estuviera escribiendo un artículo para el periódico escolar y, para presentar una idea, deseara conocer la edad media de los alumnos del centro educativo?
- ¿Qué tendría que hacer para obtener esta información?
- Podría preguntar a cada alumno su edad en años, meses y días, sumar todas estas cifras y, a continuación, dividir por el número de alumnos del centro educativo
- Esta sería una forma, muy lenta y difícil, de encontrar esta información

# Objetivo

- ¿Qué sucedería si necesitara conocer esto inmediatamente, como para poder cumplir una entrega a las 15:00 p.m. ?
- Podría tener un problema
- ¿Qué ocurriría si la fecha de nacimiento de cada alumno estuviera en una base de datos del centro educativo en la tabla STUDENT?
- En ese caso, eso sería muy fácil
- En esta lección, va a descubrir el potencial de las funciones de grupo en SQL

# Funciones de Grupo

- En SQL, las siguientes funciones de grupo se pueden utilizar en una tabla completa o en un grupo específico de filas. Cada función devuelve un resultado
- Funciones de Grupo:
  - AVG
  - COUNT
  - MIN
  - MAX
  - SUM
  - VARIANCE
  - STDDEV



La función de grupo COUNT se examinará con más detalle en la siguiente lección.

# Lista de Funciones de Grupo

- MIN: se utiliza con las columnas que almacenan cualquier tipo de dato para devolver el valor mínimo
- MAX: se utiliza con las columnas que almacenan cualquier tipo de dato para devolver el valor máximo

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
...	...
	7000
10	4400

```
SELECT MAX(salary)  
FROM employees;
```

MAX (SALARY)  
24000



Academy

# Lista de Funciones de Grupo

- SUM: Se utiliza con las columnas que almacenan los datos numéricos para buscar el total o la suma de valores
- AVG: se utiliza con las columnas que almacenan los datos numéricos para calcular la media

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
...	...
	7000
10	4400

```
SELECT MAX(salary)  
FROM employees;
```

MAX (SALARY)  
24000

# Lista de Funciones de Grupo

- COUNT: devuelve el número de filas
- VARIANCE: se utiliza con columnas que almacenan datos numéricos para calcular la difusión de datos en torno a la media. Por ejemplo, si la nota media para la clase en la última prueba fue del 82% y las puntuaciones del alumno oscilaron entre el 40% y el 100%, la varianza de las puntuaciones sería mayor que si las puntuaciones del alumno oscilaron entre un 78% y un 88%
- STDDEV: similar a la varianza, la desviación estándar mide la difusión de los datos. Para dos juegos de datos con aproximadamente la misma media, cuanto mayor sea la difusión, mayor será la desviación estándar



Academy

DP 8-1  
Funciones de Grupo

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

9

La desviación estándar y la varianza miden la difusión de los datos alrededor de la media. Para este curso, es importante poderlas reconocer y utilizar como funciones de grupo. Describir su funcionamiento no se aborda en este curso.

# Cláusula SELECT de las Funciones de Grupo

- Las funciones de grupo se escriben en la cláusula SELECT:

```
SELECT column,  
group_function(column),  
..  
FROM table  
WHERE condition  
GROUP BY column;
```

- ¿Qué Son las Funciones de Grupo?
- Las funciones de grupo funcionan en juegos de filas para proporcionar un resultado por grupo

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
60	10500
60	11000
60	8600
	7000
10	4400

El salario mínimo en la tabla EMPLOYEES

MIN (SALARY)

2500

La cláusula WHERE se puede incluir para realizar una función de grupo en un subjuego de la tabla, por ejemplo WHERE department\_id = 90.

La cláusula GROUP BY se tratará en una lección posterior.

# Precauciones de la Función de Grupo

- Cosas importantes que debe saber de las funciones de grupo:
  - Las funciones de grupo no se pueden utilizar en la cláusula WHERE:

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



ORA-00934: group function is not allowed here

Trataremos cómo resolver este problema en una lección posterior.

## Ejemplos de Funciones de Grupo

- MIN: se utiliza con las columnas que almacenan cualquier tipo de dato para devolver el valor mínimo

Ejemplos:	resultado
SELECT MIN(life_expect_at_birth) AS "Lowest Life Exp" FROM wf_countries;	32,62
SELECT MIN(country_name) FROM wf_countries;	Anguilla
SELECT MIN(hire_date) FROM employees;	17-Jun-1987

El ejemplo 1 devuelve el número más bajo en la columna life\_expect\_at\_birth.

El ejemplo 2 utiliza una columna de datos de caracteres y devuelve el condado cuyo nombre sea el primero en la lista alfabética de nombres de países.

El ejemplo 3 utiliza una columna de tipo de dato de fecha y devuelve la primera fecha de contratación.

# Ejemplos de Funciones de Grupo

- MAX: se utiliza con las columnas que almacenan cualquier tipo de dato para devolver el valor máximo

Ejemplos:	resultado
SELECT MAX(life_expect_at_birth) AS "Highest Life Exp" FROM wf_countries;	83,51
SELECT MAX(country_name) FROM wf_countries	Sáhara Occidental
SELECT MAX(hire_date) FROM employees;	29-Jan-2000

El ejemplo 1 devuelve el número más alto en la columna life\_expect\_at\_birth.

El ejemplo 2 utiliza una columna de datos de caracteres y devuelve el condado cuyo nombre sea el último en la lista alfabética de nombres de países.

El ejemplo 3 utiliza una columna de tipo de dato de fecha y devuelve la última fecha de contratación.

# Ejemplos de Funciones de Grupo

- **SUM:** Se utiliza con las columnas que almacenan los datos numéricos para buscar el total o la suma de valores

Ejemplos:	resultado
<pre>SELECT SUM(area) FROM wf_countries WHERE region_id = 29;</pre>	241424
<pre>SELECT SUM(salary) FROM employees WHERE department_id = 90;</pre>	58000

Puede restringir la función de grupo a un subjuego de la tabla utilizando una cláusula WHERE.

El ejemplo 1 devuelve el total (suma) de todas las áreas de los países de la región 29 (Caribe).

El ejemplo 2 devuelve el salario total para los empleados del departamento 90.

## Ejemplos de Funciones de Grupo

- AVG: se utiliza con las columnas que almacenan los datos numéricos para calcular la media

Ejemplos:	resultado
<pre>SELECT AVG(area) FROM wf_countries WHERE region_id = 29;</pre>	9656,96
<pre>SELECT ROUND(AVG(salary), 2) FROM employees WHERE department_id = 90;</pre>	19333,33

El ejemplo 1 devuelve la media de todas las áreas de los países de la región 29 (Caribe).

El ejemplo 2 devuelve el salario medio para los empleados del departamento 90, redondeado a dos decimales.

## Ejemplos de Funciones de Grupo

- VARIANCE: se utiliza con columnas que almacenan datos numéricos para calcular la difusión de datos en torno a la media
- STDDEV: similar a la varianza, la desviación estándar mide la difusión de los datos

Ejemplos:	resultado
SELECT ROUND(VARIANCE(life_expect_at_birth),4) FROM wf_countries;	143,2394
SELECT ROUND(STDDEV(life_expect_at_birth), 4) FROM wf_countries;	11,9683

# Función de Grupo y NULL

- Las funciones de grupo ignoran los valores NULL
- En el siguiente ejemplo, los valores nulos no se han utilizado para buscar el valor commission\_pct medio

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

LAST_NAME	COMMISSION_PCT
King	-
Kochhar	-
De Haan	-
Whalen	-
Higgins	-
Gietz	-
Zlotkey	.2
Abel	.3
Taylor	.2
Grant	.15
Mourgos	-
...	...

AVG(COMMISSION\_PCT)

.2125

DP 8-1  
Funciones de Grupo

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

17

La tabla employees tiene 20 filas. Solo 4 empleados tienen un valor commission\_pct, las otras 16 filas contienen NULL. La media se calcula buscando el valor SUM de las filas no nulas y dividiendo por el valor COUNT de las filas no nulas.

Este tema se tratará en mayor profundidad en la siguiente lección.

## Más de una Función de Grupo

- Puede tener más de una función de grupo en la cláusula SELECT, en la misma columna o en columnas diferentes

```
SELECT MAX(salary) , MIN(salary) , MIN(employee_id)
FROM employees
WHERE department_id = 60;
```

MAX(SALARY)	MIN(SALARY)	MIN(EMPLOYEE_ID)
9000	4200	103

## Reglas para Funciones de Grupo

- Las funciones de grupo ignoran los valores nulos
- Las funciones de grupo no se pueden utilizar en la cláusula WHERE
- MIN, MAX y COUNT se pueden utilizar con cualquier tipo de dato; SUM, AVG, STDDEV y VARIANCE se pueden utilizar solo con tipos de dato numéricos



# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - AVG
  - COUNT
  - Funciones de grupo
  - MAX
  - MIN
  - STDDEV
  - SUM
  - VARIANCE

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Definir y proporcionar un ejemplo de las siete funciones de grupo: SUM, AVG, COUNT, MIN, MAX, STDDEV y VARIANCE
  - Crear y ejecutar una consulta SQL utilizando funciones de grupo
  - Crear y ejecutar funciones de grupo que solo funcionan con tipos de dato numéricos



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

8-2

COUNT, DISTINCT, NVL

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear y ejecutar una consulta SQL utilizando la función de grupo COUNT
  - Utilizar DISTINCT y la función NVL con funciones de grupo



## Objetivo

- Ser capaz de agregar (agrupar) los datos mediante las funciones SQL permite a las empresas realizar cálculos que, de lo contrario, se tendrían que ejecutar manualmente
- ¿Recuerda el ejemplo en el que tenía que contar a todos los alumnos de su centro educativo? Una tarea descomunal
- No hay suficientes manos para realizarla manualmente
- Afortunadamente, las funciones de grupo SQL pueden procesar fácilmente estos tipos de solicitudes

Agregado: algo que está formado por la combinación de varios elementos distintos.

# COUNT

- COUNT (expresión) devuelve el número de valores no nulos de la columna de expresión

```
SELECT COUNT(job_id)  
FROM employees;
```

COUNT(JOB_ID)
20

# Valores COUNT y NULL

- En la tabla employees se muestran veinte filas de empleados y, si selecciona commission\_pct, se devuelven veinte filas
- Al agregar una función count a la consulta, COUNT solo ha devuelto cuatro
- COUNT cuenta específicamente la columna commission\_pct, pero
- ignora los valores nulos en la columna

```
SELECT commission_pct  
FROM employees;
```

20 filas devueltas en 0,01 segundos

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

COUNT(COMMISSION_PCT)
4

## COUNT All Rows

- COUNT(\*) devuelve el número de filas de una tabla
- No especifica la columna (que puede o no incluir nulos) que contar; cuenta el número de filas devueltas en el juego de resultados
- Por ejemplo, para averiguar cuántos empleados fueron contratados antes del 01/Ene/1996, se puede utilizar COUNT en la sentencia SELECT

```
SELECT COUNT(*)  
FROM employees  
WHERE hire_date < '01-Jan-1996';
```

COUNT (*)
9



DP 8-2  
COUNT, DISTINCT, NVL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

7

Si se incluye una cláusula WHERE en la sentencia SELECT, COUNT(\*) devuelve el número de filas que cumpla con la condición de la cláusula WHERE.

## COUNT All Rows

- Utilizamos COUNT(\*) cuando queremos asegurarnos de contar todas las filas (incluidos los duplicados), así como aquellos que pueden tener valores nulos en una o más columnas

```
SELECT COUNT(*)
FROM employees
WHERE hire_date < '01-Jan-1996' ;
```

COUNT (*)
9

Tenemos que utilizar (\*) porque las reglas de sintaxis exigen que cada función tenga al menos un argumento de entrada, entre paréntesis.

# DISTINCT

- La palabra clave DISTINCT se utiliza para devolver solo valores no duplicados o combinaciones de valores no duplicados en una consulta
- Examine la consulta siguiente
- Sin utilizar la palabra clave DISTINCT, la consulta ha devuelto todos los valores job\_id de la tabla employees, incluidos los valores duplicados

```
SELECT job_id  
FROM employees;
```

JOB_ID
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
AD_VP
IT_PROG
...

20 filas devueltas en 0,01 segundos

ORACLE

Academy

DP 8-2  
COUNT, DISTINCT, NVL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

9

## Ejemplo de DISTINCT

- Para eliminar las filas duplicadas, utilice la palabra clave DISTINCT, tal y como se muestra aquí
- Utilizando la palabra clave DISTINCT se han devuelto todos los ID de trabajo exactamente una vez, sin valores duplicados

```
SELECT DISTINCT job_id  
FROM employees;
```

JOB_ID
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
IT_PROG
MK_MAN
...

12 filas devueltas en 0,01 segundos

## Valores DISTINCT no duplicados

- La palabra clave DISTINCT, cuando se utiliza en una consulta en que se seleccione más de una columna, devolverá combinaciones no duplicadas de las columnas seleccionadas
- Examine el juego de resultados que se muestra aquí

```
SELECT DISTINCT job_id,  
    department_id  
FROM employees;
```

JOB_ID	DEPARTMENT_ID
IT_PROG	60
SA_REP	80
ST_MAN	50
AD_VP	90
AD_ASST	10
MK_MAN	20
MK_REP	20
SA_MAN	80
SA_REP	-
...	...

13 filas devueltas en 0,01 segundos

# Valores DISTINCT no duplicados

- Tenga en cuenta que no existen duplicados de la combinación de job\_id y department\_id, incluso aunque existan duplicados en ambas columnas

```
SELECT DISTINCT job_id,  
    department_id  
FROM employees;
```

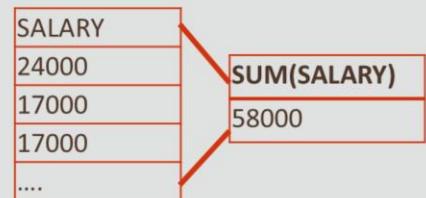
JOB_ID	DEPARTMENT_ID
IT_PROG	60
SA_REP	80
ST_MAN	50
AD_VP	90
AD_ASST	10
MK_MAN	20
MK_REP	20
SA_MAN	80
SA_REP	-
...	...

13 filas devueltas en 0,01 segundos

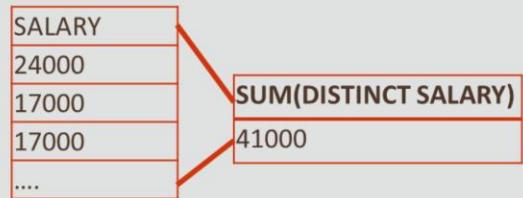
## Uso de DISTINCT

- La palabra clave DISTINCT se puede utilizar con todas las funciones de grupo
- Con DISTINCT, la función solo tiene en cuenta los valores no duplicados
- Con las dos sentencias de la derecha se producen resultados diferentes, ya que la segunda solo tiene en cuenta una incidencia de 17000

```
SELECT SUM(salary)
FROM employees
WHERE department_id = 90;
```



```
SELECT SUM(DISTINCT salary)
FROM employees
WHERE department_id = 90;
```



## DISTINCT y COUNT

- Al utilizar DISTINCT con una función de grupo como COUNT, el juego de resultados devolverá el número de valores de columna no duplicados

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

COUNT (DISTINCT job\_id)

12

¿Cuántos trabajos diferentes tienen asignados los empleados?

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

COUNT (DISTINCT salary)

18

¿Cuántos importes de salarios diferentes se pagan a los empleados?

## NVL

- A veces es preferible incluir valores nulos en funciones de grupo
- Por ejemplo, saber el número medio de pedidos de clientes servidos cada día se podría utilizar para determinar cuánta comida pedir cada mes
- Algunos días el restaurante está cerrado y no se sirve a los clientes, pero el propietario ha descubierto que calcular la media mediante la inclusión de los días que está cerrado es un indicador más fiable que solamente contar los días con clientes

## NVL

- La sentencia SELECT para incluir los valores nulos se podría escribir empezando por:

```
SELECT AVG(NVL(customer_orders, 0))
```

- Otro ejemplo de la tabla employees:

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

AVG(COMMISSION_PCT)
.2125

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

AVG(NVL(COMMISSION_PCT,0))
.0425



Academy

DP 8-2  
COUNT, DISTINCT, NVL

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

16

# NVL

- Compare los resultados de las dos consultas siguientes

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

AVG(COMMISSION_PCT)
.2125

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

AVG(NVL(COMMISSION_PCT,0))
.0425

Como se ha explicado en la lección anterior, la tabla employees tiene 20 filas. Solo 4 empleados tienen un valor commission\_pct, las otras 16 filas contienen NULL. La media se calcula buscando el valor SUM de las filas no nulas y dividiendo por el valor COUNT de las filas no nulas.

En la segunda consulta se sustituye un cero para aquellos empleados que tengan un valor commission\_pct NULL. La media devuelta se calcula mediante la búsqueda del valor SUM de todas las filas (veinte), así como dividiendo el valor COUNT de todos las filas (veinte), por lo que la media es mucho menor.

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Aggregate
  - COUNT (expression)
  - COUNT (DISTINCT expression)
  - DISTINCT

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear y ejecutar una consulta SQL utilizando la función de grupo COUNT
  - Utilizar DISTINCT y la función NVL con funciones de grupo



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

9-1

## Uso de las Cláusulas Group By y Having

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear y ejecutar una consulta SQL utilizando GROUP BY
  - Crear y ejecutar una consulta SQL utilizando GROUP BY ....HAVING
  - Crear y ejecutar GROUP BY en más de una columna
  - Anidar funciones de grupo



# Objetivo

- ¿Si deseara saber la altura media de todos los alumnos?
- Podría escribir una consulta similar a la siguiente:

```
SELECT AVG(height) FROM students;
```



**ORACLE**  
Academy

DP 9-1  
Uso de las Cláusulas Group By y Having

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

4

La tabla students no está disponible en APEX.

## Objetivo

- Y ¿qué ocurriría si deseara saber la altura media de los alumnos en función de su año en el centro educativo?
- Con lo que usted conoce ahora mismo, tendría que escribir una serie de diferentes sentencias SQL para realizar este proceso:

```
SELECT AVG(height) FROM students WHERE year_in_school = 10;
```

```
SELECT AVG(height) FROM students WHERE year_in_school = 11;
```

```
SELECT AVG(height) FROM students WHERE year_in_school = 12;
```

- Y así sucesivamente
- Para simplificar problemas como este con solo una sentencia, utiliza las cláusulas GROUP BY y HAVING

# Uso de GROUP BY

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

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

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333333333333333
90	19333.3333333333333333
110	10150
-	7000

# Uso de GROUP BY

- En la sentencia SELECT mostrada, las filas se están agrupando por department\_id
- A continuación, se aplica la función AVG a cada grupo

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

DEPARTMENT_ID	Avg(Salary)
10	4400
20	9500
50	3500
60	6400
80	10033.3333333333333333
90	19333.3333333333333333
110	10150
-	7000

## Ejemplo de GROUP BY

- ¿Qué sucedería si deseara saber el salario máximo de los empleados de cada departamento?
- Utilizamos una cláusula GROUP BY que indica qué columna utilizar para agrupar las filas

```
SELECT MAX(salary)
FROM employees
GROUP BY department_id;
```

DEPT_ID	SALARY	MAX(SALARY)
90	24000	7000
90	17000	24000
90	17000	13000
60	9000	...
60	6000	
60	4200	
50	5800	
50	3500	
50	3100	
50	2600	
50	2500	
...	...	

## Ejemplo de GROUP BY

- Pero, ¿cómo saber qué salario máximo corresponde a cada departamento?

DEPT_ID	SALARY	MAX(SALARY)
90	24000	
90	17000	
90	17000	
60	9000	7000
60	6000	24000
60	4200	13000
50	5800	
50	3500	
50	3100	
50	2600	
50	2500	
...	...	...

## GROUP BY en SELECT

- Normalmente deseamos incluir la columna GROUP BY en la lista SELECT

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

DEPT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
...	...

DEPT_ID	MAX(SALARY)
-	7000
90	24000
20	13000
...	...

## Cláusula GROUP BY

- Las funciones de grupo requieren que cualquier columna mostrada en la cláusula SELECT que no forme parte de una función de grupo deba aparecer en una cláusula GROUP BY
- ¿Qué es incorrecto en este ejemplo?

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



ORA-00979: not a GROUP BY expression

**ORACLE**  
Academy

DP 9-1  
Uso de las Cláusulas Group By y Having

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

11

Job\_id es aceptable en la lista SELECT, pero last\_name no, ya que cada grupo único de job\_id solo produce una fila de salida (ya que es la columna GROUP BY). Sin embargo, puede haber muchos empleados distintos que tengan ese mismo job\_id, por ejemplo, hay tres empleados con un valor de job\_id SA\_REP.

# COUNT

- En este ejemplo se muestra cuántos países hay en cada región
- Recuerde que las funciones de grupo ignoran los valores nulos, por lo que si algún país no tiene un nombre de país, no se incluirá en el valor COUNT

```
SELECT COUNT(country_name) , region_id  
FROM wf_countries  
GROUP BY region_id  
ORDER BY region_id;
```

COUNT(COUNTRY_NAME)	REGION_ID
15	5
28	9
21	11
8	13
7	14
8	15
5	17
17	18



Academy

DP 9-1  
Uso de las Cláusulas Group By y Having

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

12

## COUNT

- Por supuesto, esto es poco probable, pero, al crear sentencias SQL, tenemos que pensar en todas las posibilidades
- Sería mejor escribir la consulta utilizando COUNT(\*):

```
SELECT COUNT(*), region_id  
FROM wf_countries  
GROUP BY region_id  
ORDER BY region_id;
```

- Esto contaría todas las filas de cada grupo de regiones, sin tener que comprobar las columnas que contuviesen valores NULL

## Cláusula WHERE

- También podemos utilizar una cláusula WHERE para excluir las filas antes de que las filas restantes formen grupos

```
SELECT department_id, MAX(salary)
FROM employees
WHERE last_name != 'King'
GROUP BY department_id;
```

LAST_NAME	DEPT_ID	SALARY
King	90	24000
Kochhar	90	17000
De Haan	90	17000
Hunold	60	9000
Ernst	60	6000
Lorentz	60	4200
...	...	...

DEPT_ID	MAX(SALARY)
-	7000
90	17000
20	13000
...	...

ORACLE

Academy

DP 9-1  
Uso de las Cláusulas Group By y Having

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

14

Como empleado, King está excluido por la cláusula WHERE, el valor MAX(salary) para el departamento 90 se devuelve como 17000.

## Más Ejemplos de GROUP BY

- Muestre la población media de todos los países de cada región
- Redondee la media a un número entero

```
SELECT region_id, ROUND(AVG(population)) AS population
FROM wf_countries
GROUP BY region_id
ORDER BY region_id;
```

- Cuente el número de idiomas hablados para todos los países

```
SELECT country_id, COUNT(language_id) AS "Number of
languages"
FROM wf_spoken_languages
GROUP BY country_id;
```



Academy

DP 9-1  
Uso de las Cláusulas Group By y Having

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

15

## Directrices de GROUP BY

- Las directrices importantes que no se deben olvidar al utilizar una cláusula GROUP BY son:
  - Si incluye una función de grupo (AVG, SUM, COUNT, MAX, MIN, STDDEV, VARIANCE) en una cláusula SELECT junto con cualquier otra columna individual, cada columna individual también debe aparecer en la cláusula GROUP BY
  - No puede utilizar un alias de columna en la cláusula GROUP BY
  - La cláusula WHERE excluye las filas antes de que se dividan en grupos

# Grupos dentro de GRUPOS

- A veces tiene que dividir los grupos en grupos más pequeños
- Por ejemplo, puede que desee agrupar todos los empleados por departamento y, a continuación, dentro de cada departamento, agruparlos por trabajo

```
SELECT department_id, job_id,  
       count(*)  
  FROM employees  
 WHERE department_id > 40  
 GROUP BY department_id, job_id;
```

DEPT_ID	JOB_ID	COUNT(*)
110	AC_ACCOUNT	1
50	ST_CLERK	4
80	SA_REP	2
90	AD_VP	2
50	ST_MAN	1
...	...	...

## Grupos dentro de GRUPOS

- En este ejemplo se muestra el número de empleados que están realizando cada trabajo dentro de cada departamento

```
SELECT department_id, job_id,  
       count(*)  
  FROM employees  
 WHERE department_id > 40  
 GROUP BY department_id, job_id;
```

DEPT_ID	JOB_ID	COUNT(*)
110	AC_ACCOUNT	1
50	ST_CLERK	4
80	SA_REP	2
90	AD_VP	2
50	ST_MAN	1
...	...	...

## Anidamiento de Funciones de Grupo

- Las funciones de grupo se pueden anidar en una profundidad de dos cuando se utilice GROUP BY

```
SELECT max(avg(salary))  
FROM employees  
GROUP by department_id;
```

- ¿Cuántos valores se devuelven con esta consulta?
- La respuesta es uno: la consulta buscará el salario medio de cada departamento y, a continuación, en esa lista, seleccionará el único valor mayor

## HAVING

- Suponga que desea buscar el salario máximo en cada departamento, pero solo para aquellos departamentos que tengan más de un empleado
- ¿Qué es incorrecto en este ejemplo?

```
SELECT department_id, MAX(salary)
FROM employees
WHERE COUNT(*) > 1
GROUP BY department_id;
```



ORA-00934: group function is not allowed here

Se puede utilizar una cláusula WHERE solo para incluir/excluir algunas filas, no grupos de filas. Por lo tanto, no podemos utilizar las funciones de grupo en una cláusula WHERE.

## HAVING

- De la misma forma que ha utilizado la cláusula WHERE para restringir las filas que seleccionó, puede utilizar la cláusula HAVING para restringir los grupos
- En una consulta con una cláusula GROUP BY y HAVING, primero se agrupan las filas, se aplican las funciones de grupo y, a continuación, solo se muestran los grupos que coincidan con la cláusula HAVING



## HAVING

- Se utiliza la cláusula WHERE para restringir las filas; se utiliza la cláusula HAVING para restringir los grupos devueltos por una cláusula GROUP BY

```
SELECT department_id,MAX(salary)
FROM employees
GROUP BY department_id
HAVING COUNT(*)>1
ORDER BY department_id;
```



DEPARTMENT_ID	MAX(SALARY)
20	13000
50	5800
60	9000
80	11000
90	24000
110	12000

La consulta encuentra primero el salario MAX de cada departamento en la tabla employees. A continuación, la cláusula HAVING restringe los grupos devueltos a los departamentos que tengan más de 1 empleado.

## HAVING

- Esta consulta busca la población media de los países de cada región
- A continuación, solo devuelve los grupos de regiones con una población inferior superior a trescientos mil

```
SELECT region_id,
       ROUND (AVG(population))
  FROM wf_countries
 GROUP BY region_id
 HAVING MIN(population)>300000
 ORDER BY region_id;
```

REGION_ID	ROUND(AVG(POPULATION))
14	27037687
17	18729285
30	193332379
34	173268273
143	12023602
145	8522790
151	28343051



Las cláusulas HAVING y GROUP BY pueden utilizar diferentes columnas. En el ejemplo de la diapositiva se realizan GROUP BY en los valores region\_id, pero la cláusula HAVING restringe los grupos según la población.

# HAVING

- Aunque la cláusula HAVING puede preceder a la cláusula GROUP BY en una sentencia SELECT, se recomienda que coloque cada cláusula en el orden que se muestra
- La cláusula ORDER BY (si se utiliza) es siempre la última

```
SELECT column, group_function  
FROM table  
WHERE  
GROUP BY  
HAVING  
ORDER BY
```

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - GROUP BY
  - HAVING

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Crear y ejecutar una consulta SQL utilizando GROUP BY
  - Crear y ejecutar una consulta SQL utilizando GROUP BY ... HAVING
  - Crear y ejecutar GROUP BY en más de una columna
  - Anidar funciones de grupo



# **ORACLE**

## Academy

# **ORACLE**

## Academy

# Database Programming with SQL

9-2

## Uso de las Operaciones Rollup y Cube, y Grouping Sets

**ORACLE**  
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Utilizar ROLLUP para generar valores subtotales
  - Utilizar CUBE para generar valores de tabulación cruzada
  - Utilizar GROUPING SETS para generar un juego de resultados único
  - Utilizar la función GROUPING para identificar los valores de fila adicionales creados por una operación ROLLUP o CUBE



## Objetivo

- Vamos a profundizar un poco más en el problema que se le presentó en la última lección
- Para buscar la altura media de todos los alumnos, utilice esta consulta:

```
SELECT AVG(height) FROM students;
```

- Si desea saber la altura media de los alumnos según sus años en el centro educativo, podría escribir una serie de sentencias SQL distintas, como esta:

```
SELECT AVG(height) FROM students WHERE year_in_school = 10;  
SELECT AVG(height) FROM students WHERE year_in_school = 11;  
SELECT AVG(height) FROM students WHERE year_in_school = 12;
```



Academy

DP 9-2

Uso de las Operaciones Rollup y Cube, y  
Grouping Sets

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

4

## Objetivo

- O bien, podría simplificar el problema mediante la escritura de una sola sentencia que contenga las cláusulas GROUP BY y HAVING
- ¿Qué ocurre si, una vez que haya seleccionado sus grupos y calculado los valores agregados en esos grupos, también deseara los subtotales por grupo y una suma total de todas las filas seleccionadas?

# Objetivo

- Podría importar los resultados en una aplicación de hoja de cálculo, sacar la calculadora, o bien, calcular los totales manualmente en papel con la aritmética
- Pero, mejor aún, podría utilizar algunas de las extensiones de la cláusula GROUP BY, creadas específicamente para este fin: ROLLUP, CUBE y GROUPING SETS
- Al utilizar estas extensiones se necesita menos trabajo por su parte. Además, todas ellas tienen un uso muy eficaz, desde el punto de vista de la base de datos

## ROLLUP

- En las consultas con GROUP BY, a menudo se deben producir subtotales y totales, y la operación ROLLUP puede realizar esta acción por usted
- Sin utilizar el operador ROLLUP, ese tipo de requisito significaría escribir varias consultas y, a continuación, introducir los resultados, por ejemplo, en una hoja de cálculo para calcular y aplicar formato a los resultados
- ROLLUP crea subtotales que se acumulan desde el nivel más detallado hasta la suma total, siguiendo la lista de agrupamiento especificada en la cláusula GROUP BY

Es bastante normal que los jefes no solo deseen la suma de los salarios de un rol de trabajo por departamento, sino que probablemente también desearán el total por departamento y el total de todos los departamentos.

## ROLLUP

- La acción de ROLLUP es directa: crea subtotales que se acumulan desde el nivel más detallado hasta la suma total
- ROLLUP utiliza una lista ordenada de las columnas del agrupamiento en su lista de argumentos
- En primer lugar, calcula los valores de agregación estándar especificados en la cláusula GROUP BY
- A continuación, crea subtotales de nivel superior de forma progresiva, de derecha a izquierda a través de la lista de columnas del agrupamiento
- Por último, crea una suma total

## Tabla de Resultados de ROLLUP

- En la tabla de resultados siguiente, las filas resaltadas en rojo las generan la operación ROLLUP:

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10	-	4400
20	MK_MAN	13000
20	MK_REP	6000
20	-	19000
-	-	23400

← Subtotal de dept\_id 10  
← Subtotal de dept\_id 20  
← Suma total del informe

## Fórmula de Resultado de ROLLUP

- El número de columnas o expresiones que aparecen en la lista de argumentos de ROLLUP determina el número de agrupamientos
- La fórmula es (número de columnas) + 1, donde el número de columnas es el número de columnas que se indica en la lista de argumentos de ROLLUP



## Fórmula de Resultado de ROLLUP

- En la siguiente consulta de ejemplo, se muestran dos columnas en la lista de argumentos de ROLLUP y, por lo tanto, verá que se generan tres valores automáticamente

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP (department_id, job_id);
```

## Sin ROLLUP

- Si utiliza GROUP BY sin ROLLUP para la misma consulta, ¿qué aspecto tendría el resultado?

```
SELECT department_id, job_id, SUM(salary)
FROM   employees
WHERE  department_id < 50
GROUP BY (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
20	MK_MAN	13000
10	AD_ASST	4400
20	MK_REP	6000

- Tendría que ejecutar varias consultas para obtener los subtotales obtenidos con ROLLUP

## CUBE

- CUBE, al igual que ROLLUP, es una extensión de la cláusula GROUP BY
- Genera informes de tabulación cruzada
- Se puede aplicar a todas las funciones de agregación, incluidas AVG, SUM, MIN, MAX y COUNT
- Las columnas que se muestran en la cláusula GROUP BY están incluidas en referencias cruzadas para crear un superjuego de grupos
- Las funciones de agregación especificadas en la lista SELECT se aplican a estos grupos para crear valores de resumen para las filas superagregadas adicionales

## CUBE

- Todas las combinaciones posibles de filas se agregan con CUBE
- Si tiene  $n$  columnas en la cláusula GROUP BY, habrá  $2^n$  posibles combinaciones de superagregados
- Matemáticamente, estas combinaciones forman un cubo de  $n$  dimensiones, que es de donde procede el nombre del operador



## CUBE

- CUBE se utiliza a menudo en las consultas que utilizan columnas de tablas independientes, en lugar de distintas columnas de una sola tabla
- Imagine, por ejemplo, un usuario que consulta la tabla Sales para una compañía como AMAZON.COM
- Un informe de tabulación cruzada normalmente solicitado podría incluir subtotales para todas las combinaciones posibles de las ventas de un mes, región y producto

# CUBE

- En la siguiente sentencia, las filas en rojo las genera la operación CUBE:

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY CUBE (department_id, job_id);
```

Total del informe  
Subtotal de MK\_MAN  
Subtotal de MK\_REP  
Subtotal de AD\_ASST  
Subtotal de departamento 10  
Subtotal de departamento 20



DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_MAN	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_MAN	13000
20	MK_REP	6000

El ejemplo anterior de ROLLUP generó subtotales para cada departamento y un total para el informe. Con CUBE se proporciona dicha información, pero se agregan subtotales para cada trabajo en todos los departamentos.

# GROUPING SETS

- GROUPING SETS es otra extensión de la cláusula GROUP BY
- Se utiliza para especificar varias agrupaciones de datos
- Esto le proporciona la funcionalidad de tener varias cláusulas GROUP BY en la misma sentencia SELECT, lo cual no está permitido en la sintaxis normal



# GROUPING SETS

- Si desea ver los datos de la tabla EMPLOYEES agrupados por (department\_id, job\_id, manager\_id)
- Pero también agrupados por (department\_id, manager\_id)
- Y también agrupados por (job\_id, manager\_id), normalmente tendría que escribir tres sentencias select distintas siendo la única diferencia entre ellas las cláusulas GROUP BY



## GROUPING SETS

- Para la base de datos, esto significa recuperar los mismos datos tres veces distintas, lo que podría suponer una gran sobrecarga
- Imagine que su compañía tiene 3.000.000 de empleados
- Estaría pidiendo a la base de datos que recuperase 9 millones de filas en lugar de solo 3 millones de filas, una gran diferencia
- Por lo tanto, el uso de GROUPING SETS es mucho más eficiente al escribir informes complejos

# GROUPING SETS

- En la siguiente sentencia, las filas resaltadas en color las genera la operación GROUPING SETS:

```
SELECT department_id, job_id, manager_id, SUM(salary)
FROM employees
WHERE department_id < 50
GROUP BY GROUPING SETS
((job_id, manager_id), (department_id, job_id),
(department_id, manager_id));
```

DEPARTMENT_ID	JOB_ID	MANAGER_ID	SUM(SALARY)
-	MK_MAN	100	13000
-	MK_MAN	201	6000
-	AD_ASST	101	4400
10	AD_ASST	-	4400
20	MK_MAN	-	13000
20	MK_REP	-	6000
10	-	101	19000
20	-	100	13000
20	-	201	6000

# Funciones GROUPING

- Al utilizar ROLLUP o CUBE para crear informes con subtotales, muy a menudo también tiene que poder saber qué filas de la salida son filas reales devueltas de la base de datos y qué filas son filas de subtotal calculado resultantes de las operaciones ROLLUP o CUBE

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_MAN	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_MAN	13000
20	MK_REP	6000

## Funciones GROUPING

- Si observa el informe de la derecha, ¿cómo podría distinguir entre las filas reales de la base de datos y las filas calculadas?
- ¿Cómo puede notar la diferencia entre un valor NULL almacenado devuelto por la consulta y los valores NULL creados mediante ROLLUP o CUBE

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
-	-	23400
-	MK_MAN	13000
-	MK_REP	6000
-	AD_ASST	4400
10	-	4400
10	AD_ASST	4400
20	-	19000
20	MK_MAN	13000
20	MK_REP	6000

# Funciones GROUPING

- La función GROUPING resuelve estos problemas
- Utilizando una sola columna de la consulta como argumento, la función GROUPING devolverá un 1 para una fila agregada (calculada) y un 0 para una fila no agregada (devuelta)
- La sintaxis de GROUPING es simplemente GROUPING (nombre\_columna)
- Solo se utiliza en la cláusula SELECT y solo acepta una expresión de columna como argumento

# Funciones GROUPING

- Ejemplo:

```
SELECT department_id, job_id, SUM(salary),  
       GROUPING(department_id) AS "Dept sub total",  
       GROUPING(job_id) AS "Job sub total"  
FROM employees  
WHERE department_id < 50  
GROUP BY CUBE (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)	Dept sub total	Job sub total
-	-	23400	1	1
-	MK_MAN	13000	1	0
-	MK_REP	6000	1	0
-	AD_ASST	4400	1	0
10	-	4400	0	1
10	AD_ASST	4400	0	0
20	-	19000	0	1
20	MK_MAN	13000	0	0
20	MK_REP	6000	0	0

ORACLE

Academy

DP 9-2

Uso de las Operaciones Rollup y Cube, y  
Grouping Sets

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

24

# Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - CUBE
  - FUNCIÓN GROUPING
  - GROUPING SETS
  - ROLLUP

# Resumen

- En esta lección, debe haber aprendido lo siguiente:
  - Utilizar ROLLUP para generar valores subtotales
  - Utilizar CUBE para generar valores de tabulación cruzada
  - Utilizar GROUPING SETS para generar un juego de resultados único
  - Utilizar la función GROUPING para identificar los valores de fila adicionales creados por una operación ROLLUP o CUBE



# **ORACLE**

## Academy