

like, between, in, not null, order by, desc

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

ORACLE

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.
...

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

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

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

ORACLE

Página 19 de 23 | - | +

DEPARTMENT_ID
-
90
20
110
80
50
10
60

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');
```

ORACLE

Academy

CITY	STATE_PROVINCE	COUNTRY_ID
Toronto	Ontario	CA
Oxford	Oxford	UK

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;
```

LAST_NAME	SALARY
Zlotkey	10500
Abel	11000
Hunold	9000

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

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

- 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

IS NULL, IS NOT NULL

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

LAST_NAME
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

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 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

Reglas de Prioridad o Qué Ocurre

- 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

Operador NOT

- El operador NOT devolverá las filas que no cumplen 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

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

DISTINCT (para que las respuestas

no se repitan) Ej: **SELECT DISTINCT department_id FROM EMPLOYEES**

Ordenamiento 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

[Mostrar más](#)

A partir del documento importado

AURA M: 671 kB

AURA M: 1 MB

AURA M: 1.009 kB

AURA MO: 1 MB

ERGI PO: 921 kB

ERGI PO: 993 kB

AURA M: 940 kB

ORACLE

Ordenamiento con Varias Columnas

- A continuación se muestra un ejemplo de ordenamiento 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

ORACLE

-----CHARS-----

alfabéticos en minúscula

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

ORACLE Academy

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

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

12

Funciones de Manipulación de Mayúsculas/Minúsculas

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

```
SELECT last_name  
FROM employees  
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';
```

Examples:	Result
SELECT CONCAT('Hello', 'World') FROM DUAL;	HelloWorld

SUBSTR (extrae un string en una posición determinada)

Examples:	Result
SELECT SUBSTR('HelloWorld', 1, 5) FROM DUAL;	Hello
SELECT SUBSTR('HelloWorld', 6) FROM DUAL;	World
SELECT SUBSTR(last_name, 1, 3) FROM employees;	Abe Dav

LENGTH (muestra la cantidad de caracteres de un string)

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

INSTR (muestra la posición numérica de un carácter)

Examples:	Result
SELECT INSTR('HelloWorld', 'W') FROM DUAL;	6
SELECT last_name, INSTR(last_name, 'a') FROM employees;	Abel 0 Davies 2 ...

LPAD (rellena con símbolos a la izquierda de la variable)

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

Examples:	Result
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

Examples:	Result
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

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

Enter a value for the bind variable	
Bind Variable	Value
:ENTER_DEPT_ID	<input type="text"/>

Los bloques de datos se utilizan para insertar datos en una consulta.

ROUND(Hacia arriba si esta por delante del 0.5), TRUNC(HACIA ABAJO)

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

Funciones de...

- MOD
- ROUND
- TRUNC

---FECHAS---DATOS---

SYSDATE

- SYSDATE es una función de fecha que devuelve la fecha y hora actuales del servidor de base de datos

```
SELECT SYSDATE
FROM dual;
```

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

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

DP 4-3

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

- **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

- **MONTHS_BETWEEN:** toma 2 argumentos DATE y
devuelve el número de meses de calendario entre l
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 ...

- **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;	04-Jul-2015

- **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

- **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

---TO CHAR---TO DATE---TO NUMBER--- CONVERSION

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
---	----------------

ORACLE

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

- 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;
```

ORACLE

CONVERT
10-May-1989

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

-----NVL-----

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

ORACLE

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 las expresiones 2 sean datos de caracteres, en cuyo caso, el tipo devuelto es VARCHAR2

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

ORACLE
Academy

DP-S-2

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

17

-----Condicionales, CASE, DECODE-----

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.

ORACLE
Academy

DP 5-3

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

10

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

ORACLE
Academy

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.

-----JOINS-----

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

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

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
...

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

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
...	

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
...				

ORACLE

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
...		

ORACLE

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	-	-



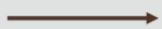
Uniones Externas Izquierdas y Derechas



- 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



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



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);
```

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
...

-----MIN-----MAX---COUNT---AVG--SUM--

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

COUNT: devuelve el número de filas

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

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

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

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

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

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

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

---GROUP BY----- HAVING-----COUNT

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

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

ORACLE

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

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
```

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

ORACLE

-----SUB CONSULTAS----

Ejemplo de Subconsulta

- ¿Qué hacer si desea averiguar los nombres de los empleados contratados después Peter Vargas?
- Lo primero que debe saber es la respuesta a la pregunta "¿Cuándo se contrató a Peter Vargas?"
- Una vez que sepa su fecha de contratación, puede seleccionar aquellos empleados cuyas fechas de contratación sean posteriores a la suya

FIRST_NAME	LAST_NAME	HIRE_DATE
Eleni	Zlotkey	29-Jan-2000
Kimberely	Grant	24-May-1999
Kevin	Mourgos	16-Nov-1999
Diana	Lorentz	07-Feb-1999

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

ORACLE

Academy

DP 10-1
Conceptos Fundamentales de las Subconsultas

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

11

Subconsultas de Diferentes Tablas

- Más de una subconsulta puede devolver la información a la consulta externa

```
SELECT last_name, job_id, salary, department_id
  FROM employees
 WHERE job_id =
    (SELECT job_id
       FROM employees
      WHERE employee_id = 141)
 AND department_id =
    (SELECT department_id
       FROM departments
      WHERE location_id = 1500);
```

Resultado de la 1^a
subconsulta

JOB_ID
ST_CLERK

Resultado de la 2^a
subconsulta

DEPARTMENT_ID
50

ORACLE
Academy

LAST_NAME	JOB_ID	SALARY	DEPARTMENT_ID
Rajs	ST_CLERK	3500	50
Davies	ST_CLERK	3100	50
Matos	ST_CLERK	2600	50
Vargas	ST_CLERK	2500	50

DP 10-2
Subconsultas de Una Sola Fila

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

8

SUBCONSULTA CON HAVING

Ejemplo de Subconsulta

- ¿Qué departamentos tienen un salario más bajo que sea mayor que el salario más bajo del departamento 50?
- En este ejemplo, la subconsulta selecciona y devuelve el menor salario del departamento 50

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) >
       (SELECT MIN(salary)
        FROM employees
        WHERE department_id = 50);
```

DEPARTMENT_ID	MIN(SALARY)
-	7000
90	17000
20	6000
110	8300
80	8600
10	4400
60	4200

ORACLE

Academy

DP 10-2

MIN(SALARY)
2500

Resultado de la subconsulta

Copyright © 2020, Oracle y/o sus filiales. Todas las derechos reservados.

12

SUBCONSULTAS DE VARIAS FILAS--ALL-- ANY -- IN --

IN

- El operador IN se utiliza en la cláusula WHERE de la consulta externa para seleccionar solo las filas que están EN la lista de valores devueltos de la consulta interna
- Por ejemplo, estamos interesados en todos los empleados contratados el mismo año que un empleado del departamento 90

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) IN
      (SELECT EXTRACT(YEAR FROM hire_date)
       FROM employees
       WHERE department_id=90);
```

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

ORACLE

ANY

- El operador ANY se utiliza cuando deseamos que la cláusula WHERE de la consulta externa seleccione las filas que coinciden con los criterios (<, >, =, etc.) de al menos un valor en el juego de resultados de la subconsulta
- En el ejemplo mostrado se devolverá cualquier empleado cuyo año de contratación sea menor que al menos un año de contratación que los empleados del departamento 90

```
SELECT last_name, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ANY
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```

ORACLE

Academy

DP 10.2



LAST_NAME	HIRE_DATE
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987
Hunold	03-Jan-1990
Ernst	21-May-1991

ALL

- El operador ALL se utiliza cuando deseamos que la cláusula WHERE de la consulta externa seleccione las filas que coinciden con los criterios (<, >, =, etc.) de todos los valores en el juego de resultados de la subconsulta
- El operador ALL compara un valor con todos los valores devueltos por la consulta interna
- Puesto que no se contrató a ningún empleado antes de 1987, no se devuelve ninguna fila

```
SELECT last_name, hire_date FROM employees
WHERE EXTRACT(YEAR FROM hire_date) < ALL
  (SELECT EXTRACT(YEAR FROM hire_date)
   FROM employees
   WHERE department_id=90);
```



Valores NULL

- Suponga que uno de los valores devueltos por una subconsulta de varias filas es nulo, pero otros valores no lo son
- Si se utiliza IN o ANY, la consulta externa devolverá filas que coinciden con los valores no nulos

```
SELECT last_name,
employee_id
FROM employees
WHERE employee_id IN
(SELECT manager_id
FROM employees);
```

MANAGER_ID
-
100
100
101
101
205
100
...

LAST_NAME	EMPLOYEE_ID
King	100
Kochhar	101
De Haan	102
Higgins	205
...	

GROUP BY y HAVING

- Esta es la sentencia SQL:

```
SELECT department_id, MIN(salary)
FROM employees
GROUP BY department_id
HAVING MIN(salary) < ANY
(SELECT salary
FROM employees
WHERE department_id IN (10,20))
ORDER BY department_id;
```

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
50	2500
60	4200
80	8600
110	8300
-	7000

LAST_NAME	DEPT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000

Resultado de la subconsulta

--INSERT FILA-- COPY --

Sintaxis para Crear una Copia de una Tabla

- Cree la sintaxis de tabla:

```
CREATE TABLE copy_tablename
AS (SELECT * FROM tablename);
```

- Por ejemplo:

```
CREATE TABLE copy_employees
AS (SELECT * FROM employees);
```

```
CREATE TABLE copy_departments
AS (SELECT * FROM departments);
```

```
INSERT INTO copy_departments
  (department_id, department_name, manager_id, location_id)
VALUES (200, 'Human Resources', 205, 1500);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
200	Human Resources	205	1500

ORACLE
Academy

- La sentencia INSERT en este ejemplo se ha escrito sin nombrar explícitamente las columnas
- Sin embargo, para mayor claridad, es mejor utilizar los nombres de columna en una cláusula INSERT

```
INSERT INTO copy_departments
VALUES
(210, 'Estate Management', 102, 1700);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
210	Estate Management	102	1700

Inserción de Filas con Valores Nulos

- Una inserción implícita insertará automáticamente un valor nulo en las columnas que permiten valores nulos
- Para agregar explícitamente un valor nulo a una columna que permite valores nulos, utilice la palabra clave NULL en la lista VALUES

--INSERTAR VALORES FECHA----

Inserción de Valores de Fecha Específicos

- El modelo de formato por defecto para tipos de dato de fecha es DD-Mes-AAAA
- Con este formato de fecha, la hora por defecto de medianoche (00:00:00) también se incluye
- En la sección anterior, hemos aprendido cómo utilizar la función TO_CHAR para convertir una fecha en una cadena de caracteres cuando queremos recuperar y mostrar un valor de fecha con un formato que no es el formato por defecto

ORACLE
Academy

DP 12.1
Guía Oracle Database

Copyright © 2000, Oracle y/o sus filiales. Todos los derechos reservados.

26

Inserción de Valores de Fecha Específicos

- Este es un recordatorio de TO_CHAR:

```
SELECT first_name, TO_CHAR(hire_date,'Month, fmdd, yyyy')
FROM employees
WHERE employee_id = 101;
```

FIRST_NAME	TO_CHAR(HIRE_DATE,'MONTH,FMDD,YYYY')
Neena	September, 21, 1989

Inserción de Valores de Fecha Específicos

- Del mismo modo, si deseamos INSERTAR una fila con un formato que no sea el formato por defecto para una columna de fecha, debemos utilizar la función TO_DATE para convertir el valor de fecha (cadena de caracteres) en una fecha

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number,
  hire_date, job_id, salary)
VALUES
(301, 'Katie', 'Hernandez', 'khernandez', '8586667641',
 TO_DATE('July 8, 2017', 'Month fmdd, yyyy'),
 'MK_REP', 4200);
```

--INSERTAR CON SUBCONSULTA--

Uso de una Subconsulta para Copiar Filas

- Cada sentencia INSERT que hemos visto hasta el momento solo agrega una fila a la tabla
- Sin embargo, suponga que deseamos copiar 100 filas de una tabla a otra
- No queremos tener que escribir y ejecutar 100 sentencias INSERT independientes, una tras otra
- Eso llevaría mucho tiempo
- Afortunadamente, SQL nos permite utilizar una subconsulta dentro de una sentencia INSERT

- En el ejemplo que se muestra, una nueva tabla denominada SALES_REPS se está llenando con copias de algunas de las filas y columnas de la tabla EMPLEADOS
- La cláusula WHERE selecciona aquellos empleados que tengan identificadores de trabajo como "%REP%"

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

---UPDATE--- DELETE--

```
UPDATE copy_employees
SET phone_number = '654321', last_name = 'Jones'
WHERE employee_id >= 303;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER
303	Angelina	Jones	654321
304	Test	Jones	654321

DELETE se utiliza para eliminar filas existentes de una tabla. Necesita:

- el nombre de la tabla
- la condición que identifica las filas que se van a suprimir

```
DELETE from copy_employees
WHERE employee_id = 303;
```

Subconsulta DELETE

- Las subconsultas también se pueden utilizar en sentencias DELETE
- En el ejemplo, se suprimen las filas de la tabla de empleados para todos los empleados que trabajan en el departamento de envíos
- Es posible que este departamento haya cambiado de nombre o se haya cerrado

```
DELETE FROM copy_employees
WHERE department_id =
(SELECT department_id
FROM departments
WHERE department_name = 'Shipping'); 5 filas eliminadas.
```

---ERRORES INTEGRIDAD---

- ¿Cuál de las siguientes sentencias devolverá un error?

1. UPDATE employees SET department_id = 15
WHERE employee_id = 100;
2. DELETE FROM departments WHERE department_id = 10;
3. UPDATE employees SET department_id = 10
WHERE department_id = 20;

Errores de Restricción de Integridad

- Al modificar las copias de tablas (por ejemplo, copy_employees), es posible que vea errores de restricción de valor no nulo, pero no verá ningún error de restricción de clave primaria-clave ajena
- El motivo es que la sentencia CREATE TABLE ... AS (SELECT ...) que se utiliza para crear la copia de la tabla, copia las filas y las restricciones de valores no nulos, pero no copia las restricciones de clave primaria-clave ajena
- Por lo tanto, no existe ninguna restricción de clave primaria-clave ajena en ninguna de las tablas copiadas
- La adición de restricciones se abordará en otra lección

-----CREATE TABLE-----PK--FK--PRIMARY KEY--FOREIGN KEY---CHECK

Nivel Columna

```
CREATE TABLE clients
(client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
last_name      VARCHAR2(13) CONSTRAINT clients_last_name_nn NOT NULL,
email          VARCHAR2(80) CONSTRAINT clients_email_pk UNIQUE);
```

Nivel Tabla

```
CREATE TABLE clients (
    client_number NUMBER(6) NOT NULL,
    first_name      VARCHAR2(20),
    last_name       VARCHAR2(20),
    phone           VARCHAR2(20),
    email           VARCHAR2(10) NOT NULL,
    CONSTRAINT clients_phone_email_uk UNIQUE (email,phone))
```

Reglas Básicas para Restricciones

- Las restricciones que hacen referencia a más de una columna se deben especificar en el nivel de tabla
- La restricción NOT NULL se puede especificar solo en el nivel de columna, no en el nivel de tabla
- Las restricciones UNIQUE, PRIMARY KEY, FOREIGN KEY y CHECK se pueden definir en el nivel de columna o de tabla
- Si se utiliza la palabra CONSTRAINT en la sentencia CREATE TABLE, debe asignar un nombre a la restricción

VIOLACIÓN DE CLAVE ÚNICA COMPUSTA
Las claves compuestas se deben definir en nivel de tabla

```
CREATE TABLE clients(
    client_number  NUMBER(6),
    first_name     VARCHAR2(20),
    last_name      VARCHAR2(20),
    phone          VARCHAR2(20) CONSTRAINT phone_email_uk
                    UNIQUE(email,phone),
    email          VARCHAR2(10) CONSTRAINT NOT NULL,
    CONSTRAINT emailclients_email NOT NULL,
    CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));
```

VIOLACIÓN DE NOT NULL
Las restricciones NOT NULL solo se pueden definir en el nivel de columna

VIOLACIÓN DE NOMBRE
Cuando se utiliza el término CONSTRAINT, debe ir seguido de un nombre de restricción

ORACLE

Ejemplo de CHECK

```
CREATE TABLE PRUEBA_CHECK
(ID NUMBER(4) CONSTRAINT PRUEBA_ID PRIMARY KEY,
NAME VARCHAR2(6),
GENRE VARCHAR2(1) CHECK(GENRE='M' OR GENRE='W'),
AGE NUMBER(3) CHECK(AGE<120) );
```

-- DDL --- ALTER TABLE ---- DROP ---- RENAME ---- TRUNCATE--

- Para agregar una nueva columna, utilice la sintaxis SQL que se muestra a continuación:

```
ALTER TABLE tablename  
ADD (column name data type [DEFAULT expression],  
column name data type [DEFAULT expression], ...)
```

- Por ejemplo:

```
ALTER TABLE my_cd_collection  
ADD (release_date DATE DEFAULT SYSDATE);
```

```
ALTER TABLE my_friends  
ADD (favorite_game VARCHAR2(30));
```

ALTER TABLE: Borrado de una Columna

- Sintaxis SQL:

```
ALTER TABLE tablename  
DROP COLUMN column name;
```

- Por ejemplo:

```
ALTER TABLE my_cd_collection  
DROP COLUMN release_date;
```

```
ALTER TABLE my_friends  
DROP COLUMN favorite_game;
```

- Ejemplo: se ha creado una tabla con dos columnas:

```
CREATE TABLE mod_emp  
  (last_name VARCHAR2(20),  
   salary NUMBER(8,2));
```

- ¿Cuál de estas modificaciones estaría permitida y cuál no?
- Considere las respuestas con y sin filas de datos en la tabla

```
ALTER TABLE mod_emp  
  MODIFY (last_name VARCHAR2(30));  
ALTER TABLE mod_emp  
  MODIFY (last_name VARCHAR2(10));  
ALTER TABLE mod_emp  
  MODIFY (salary NUMBER(10,2));  
ALTER TABLE mod_emp  
  MODIFY (salary NUMBER(8,2) DEFAULT 50);
```

RENAME

- Para cambiar el nombre de una tabla, utilice la sentencia RENAME

- Esto solo lo puede hacer el propietario de la tabla (DBA)

- Sintaxis:

```
RENAME old_name to new_name;
```

- Ejemplo:

```
RENAME my_cd_collection TO my_music;
```

DR_IS_S_ESP.PPT

SET UNUSED para Columnas

- Borrar una columna de una tabla grande puede llevar mucho tiempo
- Una alternativa más rápida es marcar la columna como no utilizable
- Los valores de columna permanecen en la base de datos, pero no se puede acceder a ellos de ninguna forma, por lo que el efecto es el mismo que borrar la columna
- De hecho, puede agregar una nueva columna a la base de datos con el mismo nombre que la columna no utilizada
- Las columnas no utilizadas existen, pero son invisibles
- Sintaxis: `ALTER TABLE tablename SET UNUSED (column name);`

DROP TABLE

- Sintaxis:

```
DROP TABLE tablename;
```

- Ejemplo:

```
DROP TABLE copy_employees;
```

En caso de haber eliminado la tabla use flashback

La sintaxis es la siguiente:

```
FLASHBACK TABLE tablename TO BEFORE DROP;
```

TRUNCATE

- Sintaxis:

```
TRUNCATE TABLE tablename;
```

Presented by 16:12 11 ene

Pero como no se genera información de rollback, TRUNCATE es irreversible.

A partir del documento importado

FROM 1.1.1.1

SECUENCIAS, SEQUENCE

```
CREATE SEQUENCE sequence
  [INCREMENT BY n]
  [START WITH n]
  [{MAXVALUE n | NOMAXVALUE}]
  [{MINVALUE n | NOMINVALUE}]
  [{CYCLE | NOCYCLE}]
  [{CACHE n | NOCACHE}];
```

```
CREATE SEQUENCE runner_id_seq
  INCREMENT BY 1
  START WITH 1
  MAXVALUE 50000
  NOCACHE
  NOCYCLE;
```

secuencia	es el nombre del generador de secuencias (objeto).
INCREMENT BY n	especifica el intervalo entre los números de secuencia, donde n es un número entero (si esta cláusula se omite, la secuencia se va incrementando en 1).
START WITH n	especifica el primer número de secuencia que se va a generar (si se omite esta cláusula, la secuencia empieza con 1).

MAXVALUE n	especifique el valor máximo que puede generar la secuencia.
NOMAXVALUE	especifica un valor máximo de 10^27 para una secuencia ascendente y –1 para una secuencia descendente (valor por defecto).
MINVALUE n	especifica el valor mínimo de secuencia.

NOMINVALUE	especifica un valor máximo de 1 para una secuencia ascendente y de – (10^26) para una secuencia descendente (por defecto).
CYCLE NOCYCLE	especifica si la secuencia sigue generando valores después de alcanzar su valor máximo o mínimo (NOCYCLE es la opción por defecto).

CACHE n NOCACHE	especifica cuántos valores asigna previamente y mantiene Oracle Server en la memoria. (Por defecto, Oracle Server almacena en caché 20 valores). Si el sistema falla, los valores se pierden.
-------------------	---

Confirmación de Secuencias

```
SELECT sequence_name, min_value, max_value, increment_by,
last_number
FROM user_sequences;
```

NEXTVAL, CURRVAL

```
INSERT INTO departments
  (department_id, department_name, location_id)
VALUES  (departments_seq.NEXTVAL, 'Support', 2500);
```

```
INSERT INTO employees
  (employee_id, department_id, ...)
VALUES (employees_seq.NEXTVAL, dept_deptid_seq .CURRVAL,
...);
```

Eliminar y alterar secuencias, ALTER SEQUENCE

- Por ejemplo, no se puede ejecutar un nuevo valor MAXVALUE menor que el número de secuencia actual

```
ALTER SEQUENCE runner_id_seq
    INCREMENT BY 1
    MAXVALUE 90
    NOCACHE
    NOCYCLE;
```

```
DROP SEQUENCE runner_id_seq;
```

Uso de una Secuencia

- No puede utilizar NEXTVAL y CURRVAL en los siguientes contextos:
 - La lista SELECT de una vista
 - Una sentencia SELECT con la palabra clave DISTINCT
 - Una sentencia SELECT con las cláusulas GROUP BY, HAVING o ORDER BY
 - Una subconsulta en una sentencia SELECT, DELETE o UPDATE
 - La expresión DEFAULT en una sentencia CREATE TABLE o ALTER TABLE

VISTAS VIEWS

Creación de Vistas

- Ejemplo:

```
CREATE OR REPLACE VIEW view_euro_countries
AS SELECT country_id, region_id, country_name, capitol
      FROM wf_countries
     WHERE location LIKE '%Europe';
```

```
CREATE VIEW ART_DEPORTES AS
SELECT NOMBREARTICULO, SECCIÓN, PRECIO FROM PRODUCTOS
WHERE SECCION='DEPORTES'|
```

Modificación de Vistas

- Para modificar una vista existente sin tener que borrarla y, a continuación, volver a crearla, utilice la opción OR REPLACE en la sentencia CREATE VIEW
- La vista antigua se sustituye por la nueva versión
- Por ejemplo:

```
CREATE OR REPLACE VIEW view_euro_countries
AS SELECT country_id, region_id, country_name, capitol
      FROM wf_countries
     WHERE location LIKE '%Europe';
```

Vista Compleja

- Las vistas complejas son vistas que pueden contener funciones de grupo y uniones
- El siguiente ejemplo crea una vista que deriva datos de dos tablas

```
CREATE OR REPLACE VIEW view_euro_countries
  ("ID", "Country", "Capitol City", "Region")
AS SELECT c.country_id, c.country_name, c.capitol,
r.region_name
  FROM wf_countries c JOIN wf_world_regions r
  USING (region_id)
 WHERE location LIKE '%Europe';

SELECT *
  FROM view_euro_countries;
```

Eliminar vista

```
DROP VIEW viewname;
```

VISTAS RESTRICCIONES DML

Sentencias DML y Vistas

- Las operaciones DML INSERT, UPDATE y DELETE se pueden realizar en vistas simples
- Estas operaciones se pueden utilizar para cambiar los datos en las tablas base subyacentes

Vistas con la Opción CHECK

- WITH CHECK OPTION garantiza que las operaciones DML realizadas en la vista se mantengan en el dominio de la vista
- Cualquier intento de cambiar el número de departamento de una fila de la vista fallará porque viola la restricción WITH CHECK OPTION
- Observe en el ejemplo siguiente que a la restricción WITH CHECK OPTION se le asignó el nombre view_dept50_check

```
CREATE OR REPLACE VIEW view_dept50
AS SELECT department_id, employee_id, first_name, last_name, salary
      FROM employees
     WHERE department_id = 50
WITH CHECK OPTION CONSTRAINT view_dept50_check;
```

Vistas con READ ONLY

- La opción WITH READ ONLY garantiza que no se produce ninguna operación DML en la vista
- Cualquier intento de ejecutar una sentencia INSERT, UPDATE o DELETE producirá un error de Oracle Server

```
CREATE OR REPLACE VIEW view_dept50
AS SELECT department_id, employee_id, first_name, last_name,
salary
      FROM employees
     WHERE department_id = 50
WITH READ ONLY;
```

-----USUARIOS--USERS-----

```
CREATE USER user
IDENTIFIED BY password;
```

```
GRANT privilege [, privilege...]
TO user [, user| role, PUBLIC...];
```

cambiar su contraseña

- Ejemplo:

```
ALTER USER scott
IDENTIFIED BY imscott35;
```

```
GRANT create session, create table, create sequence, create
view
TO scott;
```

- Puede otorgar los privilegios UPDATE, REFERENCES e INSERT en columnas individuales de una tabla

- Por ejemplo:

```
GRANT UPDATE (salary)
      ON employees TO steven_king
```

Palabra Clave PUBLIC

- Un propietario de una tabla puede otorgar acceso a todos los usuarios mediante la palabra clave PUBLIC
- En el segundo ejemplo que se muestra a continuación, se permite a todos los usuarios del sistema consultar datos de la tabla DEPARTMENTS de Alice

```
GRANT select  
  ON alices.departments  
  TO PUBLIC;
```

Revocación de Privilegios de Objeto

- Utilice la siguiente sintaxis para revocar privilegios de objeto:

```
REVOKE {privilege [, privilege...]}|ALL  
ON object  
FROM {user[, user...]|role|PUBLIC}  
[CASCADE CONSTRAINTS];
```

- En el ejemplo siguiente, se revocan los privilegios SELECT e INSERT proporcionados al usuario Scott en la tabla de clientes

```
REVOKE SELECT, INSERT  
ON clients  
FROM scott_king;
```

Si el propietario revoca un privilegio de un usuario que ha otorgado privilegios a otros usuarios, la sentencia de revocación tiene un efecto en cascada en todos los privilegios otorgados

eliminar usuario = DROP USER NOMBREUSUARIO;

Funciones

- Para crear y asignar un rol, en primer lugar el DBA debe crear el rol
- A continuación, el DBA puede asignar privilegios al rol, y dicho rol a los usuarios

```
CREATE ROLE manager;
```

Role created.

```
GRANT create table, create view TO manager;
```

Grant succeeded.

```
GRANT manager TO jennifer_cho;
```

Grant succeeded.

Ejemplo de crear usuarios fuente youtube

--crear usuario

```
CREATE USER USER1 IDENTIFIED BY  
"1111";
```

--asignar rol

```
GRANT "CONNECT" TO YOUTUBE;
```

--privilegios

```
GRANT ALTER TABLESPACE TO USER1;  
GRANT EXECUTE ANY PROCEDURE TO  
USER1;
```

-----PRIVILEGIOS-----

CREATE USER Los usuarios con privilegios pueden crear otros usuarios de Oracle (privilegio necesario para el rol DBA).

DROP USER El usuario con privilegios puede borrar otro usuario.

DROP ANY TABLE El usuario con privilegios puede borrar una tabla en cualquier esquema.

BACKUP ANY TABLE El usuario con privilegios puede realizar una copia de seguridad de tablas en cualquier esquema con la utilidad de exportación.

SELECT ANY TABLE El usuario con privilegios puede consultar las tablas, vistas o instantáneas en cualquier esquema.

CREATE ANY TABLE El usuario con privilegios puede crear una tabla en cualquier esquema.

CREATE SESSION Conectar a la base de datos.

CREATE TABLE Crear tablas en el esquema del usuario.

CREATE SEQUENCE Crear una secuencia en el esquema del usuario.

CREATE VIEW Crear una vista en el esquema del usuario.

CREATE PROCEDURE Crear un paquete, función o procedimiento en el esquema del usuario.

----privilegios de visualizacion----

ROLE_SYS_PRIVS Privilegios del sistema otorgados a roles

ROLE_TAB_PRIVS Privilegios de tabla otorgados a roles

USER_ROLE_PRIVS Roles a los que puede acceder el usuario

USER_TAB_PRIVS_MADE Privilegios de objeto otorgados a objetos del usuario

USER_TAB_PRIVS_REC'D Privilegios de objeto otorgados al usuario

USER_COL_PRIVS_MADE Privilegios de objeto otorgados a columnas de objetos del usuario

USER_COL_PRIVS_REC'D Privilegios de objeto otorgados al usuario en columnas específicas

USER_SYS_PRIVS Muestra privilegios del sistema otorgados al usuario

-----EXPRESIONES REGULARES-----SÍMBOLOS-----