

## C.F.G.S.: DESARROLLO DE APLICACIONES WEB

### Módulo: BASES DE DATOS

#### TEMA 05: SQL Realización de Consultas

##### U5. 01 Introducción a SQL

- SQL (Structured Query Language), lenguaje de consulta estructurado, es un lenguaje surgido de un proyecto de investigación de IBM para el **acceso a bases de datos relacionales**. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores.
- Lenguaje de 4ª generación que nos permite definir y manipular los datos almacenados en la Base de Datos
- Llamado anteriormente SEQUEL proviene de un proyecto de investigación de IBM a mediados de los años setenta sobre la base de datos relacional llamada SYSTEM R
- 1979 Oracle Corporation introduce la primera implementación comercializada de SQL
- IBM desarrolló productos herederos del prototipo SYSTEM R como DB2 y SQL/DS
- ANSI (American National Standards Institute) adoptó SQL como lenguaje estándar para sistemas de BD relacionales en 1986.
- 1987 lo adopta ISO (International Standardization Organization).
- Actualmente lo comercializan diversos SGBD (Sistemas gestores de bases de datos)
  - SQL/DS (SQL/Data System) se ejecuta bajo SO DOS y VMS de IBM
  - Database 2 (DB2) se ejecuta bajo MVS
  - ORACLE
  - DBASE IV
  - INFORMIX
- Visual Basic, por ejemplo, incorpora sentencias SQL para acceder a bases de datos relacionales.
- Características principales:
  - Lenguaje para todo tipo de usuarios: administradores, desarrolladores y usuarios normales.
  - Se especifica **qué** se quiere, no **dónde** ni **cómo** buscarlo
  - Lenguaje para consultas, actualizaciones, definición de datos y control.

### Tipos de Sentencias en SQL

El lenguaje SQL proporciona un gran repertorio de sentencias que se utilizan en variadas tareas, como consultar datos de la base de datos, crear, actualizar y eliminar objetos, controlar el acceso a la base de datos y a los objetos. Dependiendo de las tareas, podemos clasificar las sentencias SQL en varios tipos:

- **DML : (Data Manipulation Language)**

Manipulación de datos

SELECT	Recupera filas de la base de datos
INSERT	Añade nuevas filas en la la base de datos
DELETE	Suprime filas en la base de datos
UPDATE	Modifica datos de las filas de la base de datos

- **DDL : (Data Definition Language)**

Definición de datos

CREATE	Table, view, index, synonym...	Crear objetos
DROP		Borrar objetos
ALTER		Modificar la definición de un objeto

- **DCL : (Data Control Language)**

Control de accesos, restricciones

GRANT	Concede privilegios de acceso a usuarios
REVOKE	Suprime privilegios de acceso a usuarios

### Componentes sintácticos de una sentencia:

Casi todas las sentencias SQL tienen una forma básica. Comienzan por un verbo, que es una palabra clave que describe que hace la sentencia (por ejemplo SELECT, INSERT, CREATE...), a continuación se especifica los datos con los que opera la sentencia y acaban con cláusulas opcionales u obligatorias que especifican los datos con los que se trabaja.

- La información en una base de datos relacional se almacena en **tablas**.
- Una tabla se compone de una serie de **campos** (también llamados **atributos o columnas**), cada uno de ellos de un **tipo de datos determinado**.

## TEMA 5: SQL Realización de Consultas

- Una vez definida y creada la tabla se podrán cargar datos en ella, insertar **filas** que contendrán la información que se quiere almacenar en la base de datos.

Ejemplo:

```
DROP TABLE DEPART;

CREATE TABLE DEPART (
  DEPT_NO NUMBER(2) NOT NULL,
  DNOMBRE VARCHAR2(14),
  LOC VARCHAR2(14) );

INSERT INTO DEPART VALUES (10,'CONTABILIDAD','SEVILLA');
INSERT INTO DEPART VALUES (20,'INVESTIGACION','MADRID');
INSERT INTO DEPART VALUES (30,'VENTAS','BARCELONA');
INSERT INTO DEPART VALUES (40,'PRODUCCION','BILBAO');
```

En este ejemplo hemos creado una tabla y la hemos cargado con datos, cuatro filas o registros.

Cada vez que nos conectemos a la base de datos podremos consultar esta información:

```
SQL> SELECT * FROM DEPART;
```

DEPT_NO	DNOMBRE	LOC
10	CONTABILIDAD	SEVILLA
20	INVESTIGACION	MADRID
30	VENTAS	BARCELONA
40	PRODUCCION	BILBAO

## U5. 02 CONSULTAS BÁSICAS

Para recuperar información de la base de datos, es decir, para realizar consultas a la base de datos utilizaremos la sentencia SELECT.

### Sentencia SELECT

**SELECT** [ALL | DISTINCT ]

<nombre\_campo> [{,<nombre\_campo>}]

**FROM** <nombre\_tabla> | <nombre\_vista>

[{,<nombre\_tabla> | <nombre\_vista>}]

[**WHERE** <condicion> [{ **AND** | **OR** <condicion>}]]

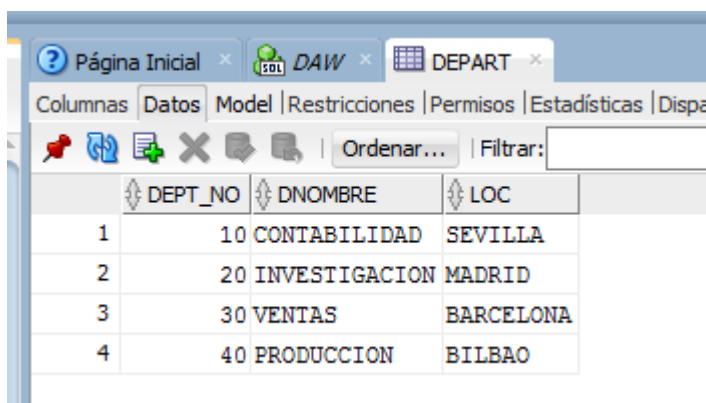
[**GROUP BY** <nombre\_campo> [{,<nombre\_campo> }]]

[**HAVING** <condicion> [{ **AND** | **OR** <condicion>}]]

[**ORDER BY** <nombre\_campo> [**ASC** | **DESC**]

[{,<nombre\_campo> [**ASC** | **DESC** }]]]

La única cláusula obligatoria es la cláusula **FROM** que especifica la tabla o tablas de las que se recuperarán los datos.



	DEPT_NO	DNOMBRE	LOC
1	10	CONTABILIDAD	SEVILLA
2	20	INVESTIGACION	MADRID
3	30	VENTAS	BARCELONA
4	40	PRODUCCION	BILBAO

## TEMA 5: SQL Realización de Consultas

Start Page x usuasir x EMPLE x								
Columns   Data   Model   Constraints   Grants   Statistics   Triggers   Flashback   Dependencies   Details   Partitions   Indexes   SQL								
Sort..   Filter:								
	EMP_NO	APELLIDO	OFICIO	RESPONSABLE	FECHA_ALTA	SALARIO	COMISION	DEPT_NO
1	7369	SANCHEZ	ADMINISTRATIVO	7902	17/12/2005	1040	(null)	20
2	7499	ARROYO	VENDEDOR	7698	20/02/2007	1500	390	30
3	7521	SALA	VENDEDOR	7698	22/02/2008	1625	650	30
4	7566	JIMENEZ	DIRECTOR	7839	02/04/2008	2900	(null)	20
5	7654	MARTIN	VENDEDOR	7698	29/09/2012	1600	1020	30
6	7698	ROJO	DIRECTOR	7839	10/05/2012	3005	(null)	30
7	7782	CEREZO	DIRECTOR	7839	09/06/2005	2885	(null)	10
8	7788	GIL	ANALISTA	7566	09/11/2007	3000	(null)	20
9	7839	REY	PRESIDENTE	(null)	17/11/2007	4100	(null)	10
10	7844	TOVAR	VENDEDOR	7698	08/09/2007	1350	0	30
11	7876	ALONSO	ADMINISTRATIVO	7788	23/09/2008	1335	(null)	20
12	7900	JIMENO	ADMINISTRATIVO	7698	03/12/2008	1335	(null)	30
13	7902	FERNANDEZ	ANALISTA	7566	03/12/2007	3000	(null)	20
14	7934	MUÑOZ	ADMINISTRATIVO	7782	23/01/2009	1690	(null)	10

**SELECT \***

**FROM TABLA;**

Muestra todas las columnas de la tabla

**SELECT campo1, campo2,... campon**

**FROM TABLA;**

Muestra los campos especificados de la tabla.

**SQL> select \* from depart;**

DEPT_NO	DNOMBRE	LOC
-----		
10	CONTABILIDAD	SEVILLA
20	INVESTIGACION	MADRID
30	VENTAS	BARCELONA
40	PRODUCCION	BILBAO

**SQL> select dept\_no, loc from depart;**

DEPT_NO	LOC
-----	
10	SEVILLA
20	MADRID
30	BARCELONA
40	BILBAO

## ALL / DISTINCT

**ALL:** Recupera todas las filas aunque estén repetidas, es la opción por omisión.

**DISTINCT:** Sólo recupera las filas que son distintas, no muestra resultados repetidos

```
SELECT ALL DEPT_NO FROM EMPLE;
```

```
SELECT DISTINCT DEPT_NO FROM EMPLE;
```

```
SQL> SELECT DISTINCT DEPT_NO FROM EMPLE;
```

```
DEPT_NO
-----
      10
      20
      30
```

```
SQL> SELECT ALL DEPT_NO FROM EMPLE;
```

```
DEPT_NO
-----
      20
      30
      30
      20
      30
      30
      10
      20
      10
      30
      20
```

```
DEPT_NO
-----
      30
      20
      10
```

14 filas seleccionadas.

---

## Cláusula WHERE

Se utiliza para expresar una o varias condiciones que han de cumplir las filas que se muestran

El formato de la condición es

*expresión operador expresión*

donde *expresión* puede ser una columna, una expresión aritmética, una constante, un valor nulo o el resultado de aplicar una función sobre cualquiera de los anteriores.

Operadores:

**Operadores de comparación:**

=, <, >, <=, >=, !=, <>

IN, NOT IN, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE

**Operadores lógicos:**

OR, AND, NOT

AND: Devuelve TRUE cuando las dos condiciones son verdaderas

OR: Devuelve TRUE cuando al menos una de las dos condiciones es verdadera

NOT: Devuelve TRUE cuando la condición es falsa

Ejemplos:

```
SELECT * FROM EMPLE
```

```
WHERE OFICIO = 'VENDEDOR';
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2500;
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2000 AND SALARIO < 3000;
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2000 OR DEPT_NO <> 30;
```

```
SELECT * FROM EMPLE WHERE SALARIO > 2000 AND DEPT_NO <> 30;
```

```
SELECT * FROM EMPLE
```

```
WHERE OFICIO = 'EMPLEADO' OR DEPT_NO = 30 AND SALARIO < 1600;
```

```
SELECT * FROM EMPLE
```

```
WHERE (OFICIO = 'EMPLEADO' OR DEPT_NO = 30 ) AND SALARIO < 1600;
```

Está permitido utilizar paréntesis para forzar el orden de evaluación.

---

**Comprobación en un conjunto de valores:**

- **Operador IN / NOT IN**

Permite comprobar si una columna o expresión pertenece o no a un conjunto de valores

<expresión|columna> IN (conjunto de valores separados por comas)

<expresión|columna> NOT IN (conjunto de valores separados por comas)

```
SELECT * FROM EMPLE
```

```
WHERE DEPT_NO IN (10,30);
```

```
SELECT * FROM EMPLE
```

```
WHERE DEPT_NO NOT IN (10,30);
```

- **Operador BETWEEN / NOT BETWEEN**

Permite comprobar si una columna o expresión está o no dentro de un rango de valores

<expresión|columna> BETWEEN valor\_inicial AND valor\_final

<expresión|columna> NOT BETWEEN valor\_inicial AND valor\_final

```
SELECT * FROM EMPLE
```

```
WHERE SALARIO BETWEEN 1800 AND 2500;
```

```
SELECT * FROM EMPLE
```

```
WHERE DEPT_NO NOT BETWEEN 1800 AND 2500;
```

---

## NULL / NOT NULL

Una columna es nula (NULL) cuando está completamente vacía.

Para preguntar si un campo es nulo preguntamos en la condición

**CAMPO IS NULL**

O para preguntar si no es nulo

**CAMPO IS NOT NULL**

```
SELECT * FROM EMPLE WHERE COMISION IS NULL;
```

```
SELECT * FROM EMPLE WHERE COMISION IS NOT NULL;
```

---

## Operadores de comparación de cadenas de caracteres: LIKE, NOT LIKE

LIKE nos permite comparar cadenas de caracteres utilizando los siguientes caracteres especiales:

% : Comodín, representa cualquier cadena de 0 o más caracteres

\_ : Marcador de posición, representa un carácter cualquiera

```
SELECT * FROM EMPLE WHERE APELLIDO LIKE 'G%';
```



Muestra los empleados cuyo apellido comienza por G

```
SELECT * FROM EMPLE WHERE APELLIDO LIKE '%O%O%';
```

Muestra los empleados cuyo apellido contiene al menos dos O

```
SELECT * EMPLE WHERE APELLIDO LIKE '_A%';
```

Muestra los empleados cuyo apellido tiene una A en la segunda posición

```
SELECT * EMPLE WHERE APELLIDO NOT LIKE '%A';
```

Muestra los empleados cuyo apellido no termina en A

---

### Crear y utilizar alias de columnas

Cuando se consulta la base de datos, los nombres de las columnas se usan como cabeceras de presentación. Si el nombre resulta demasiado largo, corto o críptico, existe la posibilidad de cambiarlo con la misma sentencia SQL de consulta creando un ALIAS. El ALIAS se pone entre comillas dobles a la derecha de la columna.

```
SELECT DNOMBRE "Nombre departamento",  
       DEPT_NO "Número de departamento"  
FROM EMPLE
```

---

### Operadores aritméticos: +, -, \*, /

#### Suma (+) , resta (-) , multiplicación (\*) y división (/)

```
SELECT SALARIO, SALARIO + 100 "SALARIO MAS 100" FROM EMPLE
```

```
SELECT SALARIO, SALARIO + 100 SALARIOMAS100 FROM EMPLE;
```

```
SELECT SALARIO + SALARIO*10/100 FROM EMPLE;
```

---

### Cláusula ORDER BY

Sirve para ordenar las filas resultado de la consulta.

```
ORDER BY expre_columna [DESC|ASC], ..., expre_columna [DESC|ASC]
```

## TEMA 5: SQL Realización de Consultas

Si no se especifica nada el orden es ascendente (ASC)

Esta cláusula es siempre la última dentro de la sentencia SELECT

```
SELECT DEPT_NO, APELLIDO, SALARIO FROM EMPL  
ORDER BY DEPT_NO, APELLIDO;
```

```
SELECT DEPT_NO, APELLIDO , SALARIO FROM EMPL  
ORDER BY SALARIO DESC;
```

```
SELECT DEPT_NO, APELLIDO , SALARIO FROM EMPL  
ORDER BY SALARIO ASC;
```

```
SELECT DEPT_NO, APELLIDO, SALARIO FROM EMPL  
ORDER BY DEPT_NO DESC, APELLIDO ASC;
```

### U5. 03 COMBINACIÓN DE TABLAS (JOIN)

A menudo necesitaremos realizar consultas que nos muestren datos de más de una tabla a la vez (Por ejemplo si quisiéramos ver el apellido, el salario y la localidad en la que trabajan los empleados cuyo oficio es director)

En este caso:

- Tendremos que poner en la cláusula FROM todas las tablas de las que queremos obtener información.
- Podremos poner en la cláusula SELECT columnas de las tablas especificadas en el FROM
- Cuando el nombre de una columna se repita en dos o más tablas especificadas en el FROM, habrá que indicarle a cuál de las tablas pertenece la columna poniendo nombre\_tabla.nombre\_columna
- Generalmente, para que la consulta tenga sentido, se especificará el criterio para combinar las tablas en la cláusula WHERE, indicando el campo o campos por los que se relacionan las tablas. Si esto se omite el resultado será el producto cartesiano de las tablas y se emparejará cada una de las filas de una tabla con todas las filas de la otra.

Por ejemplo:

Partimos de la tabla DEPART y EMPLE. Ambas tienen un campo que se llama dept\_no y por el que se relacionan. En la tabla EMPLE se guarda, para cada empleado, el código del departamento al que pertenece (dept\_no). Si con este código acudimos a la tabla DEPART podemos encontrar la información de la localidad del departamento en el que el empleado trabaja:

```
SELECT APELLIDO, EMPL.DEPT_NO, DEPART.DEPT_NO, LOC
FROM EMPL, DEPART
WHERE EMPL.DEPT_NO = DEPART.DEPT_NO
ORDER BY EMPL.DEPT_NO;
```

	APELLIDO	DEPT_NO	DEPT_NO_1	LOC
1	CEREZO	10	10	SEVILLA
2	REY	10	10	SEVILLA
3	MUÑOZ	10	10	SEVILLA
4	JIMENEZ	20	20	MADRID
5	FERNANDEZ	20	20	MADRID
6	ALONSO	20	20	MADRID
7	SANCHEZ	20	20	MADRID
8	GIL	20	20	MADRID
9	SALA	30	30	BARCELONA
10	TOVAR	30	30	BARCELONA
11	ARROYO	30	30	BARCELONA
12	JIMENO	30	30	BARCELONA
13	NEGRO	30	30	BARCELONA
14	MARTIN	30	30	BARCELONA

TEMA 5: SQL Realización de Consultas

Si no unimos las tablas en el WHERE por el campo por el que se relacionan, lo que se produce es un producto cartesiano en el que las 14 filas de EMPLE se combinarán con las 4 de DEPART

```
SELECT APELLIDO, EMPL.DEPT_NO, DEPART.DEPT_NO, LOC
FROM EMPL, DEPART
ORDER BY EMPL.DEPT_NO;
```

	APELLIDO	DEPT_NO	DEPT_NO_1	LOC
1	CEREZO	10	20	MADRID
2	CEREZO	10	10	SEVILLA
3	REY	10	20	MADRID
4	MUÑOZ	10	20	MADRID
5	MUÑOZ	10	30	BARCELONA
6	CEREZO	10	30	BARCELONA
7	CEREZO	10	40	BILBAO
8	REY	10	40	BILBAO
9	REY	10	10	SEVILLA
10	REY	10	30	BARCELONA
11	MUÑOZ	10	40	BILBAO
12	MUÑOZ	10	10	SEVILLA
13	JIMENEZ	20	20	MADRID
14	SANCHEZ	20	20	MADRID
15	FERNANDEZ	20	10	SEVILLA
16	ALONSO	20	10	SEVILLA
17	GIL	20	10	SEVILLA
18	JIMENEZ	20	10	SEVILLA
19	SANCHEZ	20	10	SEVILLA
20	FERNANDEZ	20	40	BILBAO
21	ALONSO	20	40	BILBAO
22	GIL	20	40	BILBAO
23	JIMENEZ	20	40	BILBAO
24	SANCHEZ	20	40	BILBAO
25	FERNANDEZ	20	30	BARCELONA
26	ALONSO	20	30	BARCELONA

Podemos utilizar alias para renombrar las tablas y utilizar estos alias para calificar las columnas:

```
SELECT APELLIDO, E.DEPT_NO, LOC
FROM EMPL E, DEPART D
WHERE E.DEPT_NO = D.DEPT_NO;
```

Relacionar las tablas en una consulta para que los datos sean coherentes se hace también de la siguiente manera:

```
SELECT APELLIDO, DEPART.DEPT_NO, LOC
FROM EMPL JOIN DEPART ON DEPART.DEPT_NO = EMPL.DEPT_NO;
```

#### U5. 04 SUBCONSULTAS

A veces, para realizar alguna operación de consulta, necesitamos los datos devueltos por otra consulta; así, si queremos obtener los datos de los empleados que tengan el mismo oficio que 'PEPE', hemos de averiguar el oficio de 'PEPE' (primera consulta). Una vez conocido este dato, podemos averiguar qué empleados tienen el mismo oficio que 'PEPE' (segunda consulta). Este problema se puede resolver usando una subconsulta, que no es más que una sentencia SELECT dentro de otra SELECT. Las subconsultas son aquellas sentencias SELECT que forman parte de una cláusula WHERE de una sentencia SELECT anterior.

Una subconsulta consistirá en incluir una declaración SELECT como parte de una cláusula WHERE. El formato de una subconsulta es similar a éste:

```
SELECT ...
FROM ...
WHERE columna operador-comparativo (SELECT ...
                                FROM ...
                                WHERE ...);
```

La subconsulta (el comando SELECT entre paréntesis) se ejecutará primero y, posteriormente, el valor extraído es “introducido” en la consulta principal.

Las condiciones de búsqueda que nos podemos encontrar en una subconsulta son:

- Test de comparación en subconsultas (=, >, <, >=, <=, <>) compara el valor de una expresión con un valor único producido por una subconsulta.

Empleados con el mismo oficio que el de ARROYO

```
select apellido, oficio, salario, dept_no from emple
where oficio = (select oficio from emple
               where apellido = 'ARROYO');
```

- Test de pertenencia a un conjunto de valores devuelto por una subconsulta (IN, NOT IN) comprueba si el valor de una expresión coincide con uno del conjunto de valores producido por una subconsulta)

Empleados con uno de los oficios del departamento 20

```
select apellido, oficio from emple
where oficio IN (select oficio from emple
               where dept_no = 20);
```

Ejemplos:

Empleados que ganan más que MARTIN

```
select apellido, oficio , salario from emple
where salario > (select salario from emple
                 where apellido = 'MARTIN');
```

Empleados que trabajan en el departamento de CONTABILIDAD

```
select apellido, oficio , salario, dept_no from emple
where dept_no = (select dept_no from depart
                 where dnombre = 'CONTABILIDAD');
```

Empleados que trabajan en el departamento de CONTABILIDAD

```
select apellido, oficio , salario, dept_no from emple
where dept_no IN (select dept_no from depart
                 where dnombre = 'CONTABILIDAD');
```

Empleados que no trabajan en el departamento de CONTABILIDAD

```
select apellido, oficio , salario, dept_no from emple
where dept_no NOT IN (select dept_no from depart
                     where dnombre = 'CONTABILIDAD');
```

Empleados con el mismo oficio que ARROYO y que ganan más que JIMENO

```
select apellido, oficio , salario, dept_no from emple
where oficio = (select oficio from emple
               where apellido = 'ARROYO')
and salario > (select salario from emple
               where apellido = 'JIMENO');
```

## U5. 05 FUNCIONES

Las funciones actúan sobre los valores de columnas, variables o constantes y devuelven un resultado.

Se pueden clasificar en cinco tipos:

- Aritméticas
- De cadenas de caracteres
- De manejo de fechas
- De conversión
- Otras

### Funciones aritméticas

- ✓ Trabajan con datos de tipo numérico (number).
- ✓ Se dividen en tres grupos:
  - De valores simples
  - De grupos de valores
  - De listas de valores

### Funciones de valores simples

ABS (n)	Devuelve el valor absoluto de “n”.
CEIL (n)	Obtiene el valor entero inmediatamente superior o igual a “n”
FLOOR(n)	Devuelve el valor entero inmediatamente inferior o igual a “n”
MOD(m, n)	Devuelve el resto resultante de dividir “m” entre “n”.
NVL (valor, expresión)	Esta función se utiliza para sustituir un valor nulo por otro valor.
POWER(m, exponente)	Calcula la potencia de un número.
ROUND(número [,m])	Redondea números con el número de dígitos de precisión indicados.
SIGN(valor)	Esta función indica el signo del “valor”.
SQRT(n)	Devuelve la raíz cuadrada de “n”.
TRUNC (número, [m])	Trunca números para que tengan una cierta cantidad de dígitos de precisión.
VARIANCE (valor)	Devuelve varianza de un conjunto de valores.

## TEMA 5: SQL Realización de Consultas

Ejemplos:

```
SELECT ABS(10), ABS(-10), ABS(0) FROM DUAL;
```

ABS(10)	ABS(-10)	ABS(0)
10	10	0

```
SELECT CEIL(10.6), CEIL(10.3), CEIL(-10.6) FROM DUAL;
```

CEIL(10.6)	CEIL(10.3)	CEIL(-10.6)
11	11	-10

```
SELECT FLOOR(10.6), FLOOR(10.3), FLOOR(-10.6) FROM DUAL;
```

FLOOR(10.6)	FLOOR(10.3)	FLOOR(-10.6)
10	10	-11

```
SELECT MOD(9,3), MOD(9,4), MOD(-9,4), MOD(9,-4), MOD(9.2,3), MOD(9.2,3.1) FROM DUAL;
```

MOD(9,3)	MOD(9,4)	MOD(-9,4)	MOD(9,-4)	MOD(9.2,3)	MOD(9.2,3.1)
0	1	-1	1	,2	3

```
SELECT COMISION, NVL(COMISION,0), SALARIO+NVL(COMISION,0) FROM EMPLE
WHERE DEPT_NO = 30;
```

COMISION	NVL(COMISION,0)	SALARIO+NVL(COMISION,0)
390	390	1890
650	650	2275
1020	1020	2620
	0	3005
0	0	1350
	0	1335



## TEMA 5: SQL Realización de Consultas

```
SELECT POWER (2,3), POWER(2,-3), POWER(-2,3), POWER (-2,0) FROM DUAL;
```

POWER(2,3)	POWER(2,-3)	POWER(-2,3)	POWER(-2,0)
8	,125	-8	1

```
SELECT ROUND (123.4567,0), ROUND (123.4567,1),ROUND (123.4567,2), ROUND (123.4567,-1) FROM DUAL;
```

ROUND(123.4567,0)	ROUND(123.4567,1)	ROUND(123.4567,2)	ROUND(123.4567,-1)
123	123,5	123,46	120

```
SELECT TRUNC (123.4567,0), TRUNC (123.4567,1),TRUNC (123.4567,2), TRUNC (123.4567,-1) FROM DUAL;
```

TRUNC(123.4567,0)	TRUNC(123.4567,1)	TRUNC(123.4567,2)	TRUNC(123.4567,-1)
123	123,4	123,45	120

```
SELECT SIGN(33), SIGN(-33), SIGN (0) FROM DUAL;
```

SIGN(33)	SIGN(-33)	SIGN(0)
1	-1	0

```
SELECT SQRT(25) FROM DUAL;
```

SQRT(25)
5

## TEMA 5: SQL Realización de Consultas

## Funciones de grupos de valores

AVG(n)	Calcula el valor medio de n ignorando los valores nulos.
COUNT (*) (cuenta filas) COUNT (expresión) COUNT (distinct expresión)	Cuenta el número de veces que la expresión evalúa algún dato con valor no nulo. La opción "*" cuenta todas las filas seleccionadas.
MAX (expresión)	Calcula el máximo valor de la expresión.
MIN(expresión)	Calcula el mínimo valor de la expresión.
SUM(expresión)	Obtiene la suma de valores de la expresión.

Ejemplos:

```
select count(*), count(oficio), count(comision), count(distinct oficio)
from emple;
```

```

COUNT(*) COUNT(OFICIO) COUNT(COMISION) COUNT(DISTINCTOFICIO)
-----
14          14          4          5

```

Mostrar la media de salarios, la suma y el mejor y peor salario del departamento 20

```
SELECT AVG(SALARIO), SUM(SALARIO), MAX(SALARIO), MIN(SALARIO)
FROM EMPL
WHERE DEPT_NO = 20;
```

Mostrar el apellido del empleado que mayor salario tiene.

```
SELECT APELLIDO
FROM EMPL
WHERE SALARIO = (SELECT MAX(SALARIO)
FROM EMPL);
```

Mostrar el apellido del último empleado que ha entrado en la empresa.

```
SELECT APELLIDO
FROM EMPLE
WHERE FECHA_ALT = (SELECT MAX(FECHA_ALT)
FROM EMPLE);
```

Mostrar el salario más bajo del departamento de contabilidad.

```
SELECT MIN (SALARIO)
FROM EMPLE
WHERE DEPT_NO = (SELECT DEPT_NO
FROM DEPART
WHERE DNOMBRE='CONTABILIDAD');
```

Mostrar el apellido de la persona que menos gana en el departamento de contabilidad.

```
SELECT APELLIDO
FROM EMPLE E, DEPART D
WHERE E.DEPT_NO = D.DEPT_NO
AND DNOMBRE = 'CONTABILIDAD'
AND SALARIO = (SELECT MIN (SALARIO)
FROM EMPLE E, DEPART D
WHERE E.DEPT_NO = D.DEPT_NO
AND DNOMBRE = 'CONTABILIDAD') ;
```

### Funciones de cadenas de caracteres

- ✓ Trabajan con datos de tipo carácter (CHAR o VARCHAR2) Estos datos incluyen cualquier carácter alfanumérico: letras, números y caracteres especiales.
- ✓ Se dividen en tres grupos:
  - Funciones que devuelven valores carácter
  - Funciones que devuelven valores numéricos

### Funciones que devuelven valores carácter

CHR(n)	Devuelve el carácter cuyo valor en binario (ASCII o EBCDIC ) es equivalente a "n".
CONCAT (cad1, cad2)	Devuelve "cad1" concatenada con "cad2".
	El operador    también concatena y es mucho más cómodo de utilizar

**TEMA 5: SQL Realización de Consultas**

LOWER (cad)	Devuelve la cadena "cad" con todas sus letras convertidas a minúsculas.
UPPER (cad)	Devuelve la cadena "cad" con todas sus letras convertidas a mayúsculas.
INITCAP (cad)	Convierte la cadena "cad" a tipo título.
LPAD (cad1, n [, cad2 ])	Esta función añade caracteres a la izquierda de la cadena, hasta que tiene una cierta longitud.
RPAD (cad1, n [, cad2 ])	Añade caracteres a la derecha de la cadena hasta conseguir una cierta longitud.
LTRIM(cad [, set])	Suprime un conjunto de caracteres a la izquierda de la cadena.
RTRIM(cad [, set])	Suprime un conjunto de caracteres a la derecha de la cadena.
REPLACE (cad, cadena búsqueda[,cadena_sustitución])	Sustituye un carácter o caracteres de una cadena con O o más caracteres
SUBSTR(cad, m [,n])	Obtiene parte de una cadena, a partir de la posición m, n caracteres.
TRANSLATE (cad1, cad2, cad3)	Convierte caracteres de una cadena en caracteres diferentes, según un plan de sustitución marcado por el usuario.

Ejemplos:

```
SELECT CHR (65), CHR(97) FROM DUAL;
```

```

C C
- -
A a

```

## TEMA 5: SQL Realización de Consultas

```
SELECT CONCAT ('HOLA','ADIOS') FROM DUAL;
```

```
CONCAT('H
-----
HOLAADIOS
```

```
SELECT APELLIDO||' '||OFICIO FROM EMPLE;
```

```
APELLIDO||' '||OFICIO
-----
SANCHEZ EMPLEADO
ARROYO VENDEDOR
SALA VENDEDOR
JIMENEZ DIRECTOR
MARTIN VENDEDOR
NEGRO DIRECTOR
CEREZO DIRECTOR
GIL ANALISTA
```

```
SELECT UPPER(APELLIDO),LOWER(APELLIDO), INITCAP(APELLIDO)
FROM EMPLE;
```

```
UPPER(APEL LOWER(APEL INITCAP(AP
-----
SANCHEZ      sánchez      Sanchez
ARROYO      arroyo      Arroyo
SALA        sala        Sala
JIMENEZ     jimenez     Jimenez
MARTIN      martin      Martin
NEGRO       negro      Negro
CEREZO      cerezo      Cerezo
```

```
SELECT LPAD(APELLIDO,10,'*.*'), RPAD(APELLIDO,10,'*.*')
FROM EMPLE;
```

```
LPAD(APELL RPAD(APELL
-----
*.*SANCHEZ SANCHEZ*.*
*.*ARROYO ARROYO*.*
*.*SALA SALA*.*
*.*JIMENEZ JIMENEZ*.*
*.*MARTIN MARTIN*.*
*.*NEGRO NEGRO*.*
```

```
SELECT RTRIM ('**..**HOLA...','.'), LTRIM ('**..**ADIOS...','.')
FROM DUAL;
```

```
RTRIM('**.. LTRIM('**
-----
**..**HOLA ADIOS....
```

```
SELECT SUBSTR('HOLA CARACOLA',6,3), TRANSLATE ('HOLA CARACOLA','AIOOU','EEEE'),
```

```
SUB TRANSLATE('HO REPLACE('HOLACA
-----
CAR HELE CERECELE HO*.* CARACO*.*
1 fila seleccionada.
```

### Funciones que devuelven valores numéricos

ASCII (cad)	Devuelve el valor ASCII de la primera letra de la cadena cad <sup>t</sup>
INSTR(cad1, cad2 [,comienzo [ ,m ]])	Busca un conjunto de caracteres en una cadena y devuelve la posición en la que lo encuentra.
LENGTH (cad)	Devuelve el número de caracteres de cad.

```
SELECT APELLIDO, LENGTH (APELLIDO), SUBSTR(APELLIDO,1,1), SUBSTR(APELLIDO,LENGTH(APELLIDO),1)
FROM EMPLE;
```

```
APELLIDO  LENGTH(APELLIDO) S S
-----
SANCHEZ      7 S Z
ARROYO       6 A O
SALA         4 S A
JIMENEZ      7 J Z
MARTIN       6 M N
NEGRO        5 N O
CEREZO       6 C O
GIL          3 G L
REY          3 R Y
```

```
SELECT APELLIDO, INSTR(APELLIDO, 'A') FROM EMPLE;
```

APELLIDO	INSTR(APELLIDO, 'A')
SANCHEZ	2
ARROYO	1
SALA	2
JIMENEZ	0
MARTIN	2
NEGRO	0

```
SELECT APENOM, SUBSTR(APENOM, (INSTR(APENOM, ',')+2))
FROM ALUMNOS;
```

APENOM	SUBSTR(APENOM, (INSTR(APENOM, ',')+2))
Alcalde García, Elena	Elena
Cerrato Vela, Luis	Luis
Díaz Fernández, María	María
Sanz Martín, Roberto	Roberto

### Funciones para el manejo de fechas

Oracle dispone del tipo **DATE** para almacenar fechas.

Los datos de tipo fecha contienen una fecha en la que se guarda:

Siglo, Año, Mes, Día, Hora, Minutos, Segundos

Tiene un formato por omisión que es DD/MM/YY pero este se puede modificar para la sesión en curso:

```
select sysdate from dual;
```

SYSDATE
01/11/13

ALTER SESSION SET NLS-DATE-FORMAT = 'formato que queremos mostrar'

## TEMA 5: SQL Realización de Consultas

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY HH24:MI:SS';
select sysdate from dual;
```

Sesión modificada.

```
SYSDATE
-----
01-11-2013 18:27:26
```

Operaciones con fechas:

- La resta entre dos fechas tiene por resultado el número de días entre ambas

```
SELECT SYSDATE - FECHA_ALT
FROM EMPLE
WHERE APELLIDO = 'SALA';
```

```
SYSDATE-FECHA_ALT
-----
2079,77569
1 fila seleccionada.
```

Podemos transformarlo en años

```
SELECT TRUNC((SYSDATE - FECHA_ALT)/365)
FROM EMPLE
WHERE APELLIDO = 'SALA';
```

```
TRUNC((SYSDATE-FECHA_ALT)/365)
-----
5
1 fila seleccionada.
```

- A una fecha podemos sumarle (o restarle) una cantidad de días

```
-- QUE DIA SERÁ MAÑANA, QUE DÍA FUE AYER
SELECT SYSDATE + 1, SYSDATE - 1 FROM DUAL;
```

```
SYSDATE+1 SYSDATE-1
-----
02/11/2013 31/10/2013
1 fila seleccionada.
```



### FUNCIONES CON FECHAS

<b>SYSDATE</b>	Devuelve la fecha del sistema.
<b>ADD_MONTHS (fecha, n)</b>	Devuelve la fecha 'fecha' incrementada en "n" meses.
<b>LAST_DAY (fecha)</b>	Devuelve la fecha del último día del mes que contiene 'fecha'.
<b>MONTHS_BETWEEN(fecha1, fecha2)</b>	Devuelve la diferencia en meses entre las fechas "fecha1" y "fecha2".
<b>NEXT_DAY(fecha, cad)</b>	Devuelve la fecha del primer día de la semana indicado por cad

```
SELECT SYSDATE, ADD_MONTHS (SYSDATE,3), LAST_DAY(SYSDATE), NEXT_DAY (SYSDATE,'LUN') FROM DUAL;
```

```

SYSDATE      ADD_MONTHS LAST_DAY(S NEXT_DAY(S
-----
01/11/2013 01/02/2014 30/11/2013 04/11/2013

1 fila seleccionada.
```

```
SELECT FECHA_ALT, MONTHS_BETWEEN (SYSDATE, FECHA_ALT)
FROM EMPLE
WHERE DEPT_NO = 30;
```

```

FECHA_ALT  MONTHS_BETWEEN(SYSDATE,FECHA_ALT)
-----
20/02/2007                80,4124179
22/02/2008                68,3479017
29/09/2008                61,1220953
01/05/2008                  66
08/09/2007                73,7995146
03/12/2008                58,960805
```

### Funciones de conversión

<b>TO_CHAR (fecha, formato)</b>	Transforma un tipo DATE en una cadena de caracteres.
<b>TO_CHAR (numero, formato)</b>	Transforma un tipo DATE o NUMBER en una cadena de caracteres.

**TEMA 5: SQL Realización de Consultas**

TO_DATE (cadena, formato)	Transforma una cadena en DATE.
TO_NUMBER (cadena, formato)	Transforma una cadena de caracteres en NUMBER.

**TABLA DE CONTROL DE FORMATO DE FECHAS**

cc o scc	Valor del siglo.
y, yy o sy,yyy	Año con coma, con o sin signo.
yyyy	Año sin signo.
yyy	Ultimos tres dígitos del año
yy	Ultimos dos dígitos del año
y	Ultimo dígito del año
q	Número del trimestre
ww	Número de la semana del año.
w	Número de semana del mes.
mm	Número de mes.
ddd	Número de día del año.
dd	Número de día del mes.
d	Número de día de la semana.
hh o hh12	Hora (1-12).
hh24	Hora (1-24).
mi	Minutos.
ss	Segundos.
sssss	Segundos transcurridos desde medianoche.
j	Juliano
syear o year	Año en inglés (por ejemplo: mneteen-eighty two)
month	Nombre del mes (ENERO).
mon	Abreviatura de tres letras del nombre del mes (ENE).
day	Nombre del día de la semana (LUNES).
dy	Abreviatura de tres letras del nombre del día (LUN).
a .m. o p .m.	Muestra a.m. o p.m., dependiendo del momento del día.
b . c .o a. d.	Indicador para el año (antes de Cristo o después de Cristo).

```

SELECT TO_CHAR(SYSDATE, 'DD-MM-YYYY HH24:MI:SS') FROM DUAL;

TO_CHAR(SYSDATE, 'DD
-----
01-11-2013 19:03:51
1 fila seleccionada.

```

```

SELECT APELLIDO, TO_CHAR(FECHA_ALT, 'DAY MONTH')
FROM EMPLE
WHERE DEPT_NO = 30;
|

```

```

*****
ARROYO      MARTES    FEBRERO
SALA        VIERNES   FEBRERO
MARTIN      LUNES     SEPTIEMBRE
NEGRO       JUEVES    MAYO
TOVAR       SÁBADO    SEPTIEMBRE
JIMENO      MIÉRCOLES  DICIEMBRE

```

6 filas seleccionadas.

Empleados del departamento 30 que empezaron a trabajar un lunes

```

SELECT APELLIDO, TO_CHAR(FECHA_ALT, 'DAY MONTH')
FROM EMPLE
WHERE DEPT_NO = 30
AND TO_CHAR(FECHA_ALT, 'DAY') = 'LUNES';
|

```

ninguna fila seleccionada

Internamente guarda LUNES seguido de varios espacios en blanco por lo que tenemos que modificar la condición del WHERE

```

SELECT APELLIDO, TO_CHAR(FECHA_ALT, 'DAY MONTH')
FROM EMPLE
WHERE DEPT_NO = 30
AND TO_CHAR(FECHA_ALT, 'DAY') LIKE 'LUNES%';
|

```

```

*****
APELLIDO    TO_CHAR(FECHA_ALT, 'D
-----
MARTIN      LUNES     SEPTIEMBRE

```

1 fila seleccionada.

O

```

SELECT APELLIDO, TO_CHAR(FECHA_ALT, 'DAY MONTH')
FROM EMPLE
WHERE DEPT_NO = 30
AND TO_CHAR(FECHA_ALT, 'D') = 1;
|

```

```

*****
APELLIDO    TO_CHAR(FECHA_ALT, 'D
-----
MARTIN      LUNES     SEPTIEMBRE

```

1 fila seleccionada.

## TEMA 5: SQL Realización de Consultas

```
SELECT TO_DATE ('19062010','DDMMYYYY') FROM DUAL;
```

```
TO_DATE('1
-----
19/06/2010
1 fila seleccionada.
```

Nos resultará útil para pasar cadenas a formato fecha para poder trabajar con ellas como si lo fueran.

Por ejemplo:

¿Cuánto días han pasado desde el día 12 de junio de este año a hoy?

```
SELECT SYSDATE - TO_DATE('12062013','DDMMYYYY') FROM DUAL;
```

ó mejor

```
SELECT SYSDATE - TO_DATE('1206' || TO_CHAR(SYSDATE,'YYYY'),'DDMMYYYY') FROM DUAL;
```

## OTRAS FUNCIONES

<b>USER</b>	Devuelve el usuario conectado
<b>DECODE</b> (var, val1, cod1, val2, cod2... valor_por_defecto)	Esta función sustituye un valor por otro, Si var vale val1 lo sustituye por cod1, si vale val2 por cod2,... y si no es ninguno de los valores especificados lo sustituye por valor_por_defecto

```
SELECT USER, UID FROM DUAL;
```

```
USER          UID
-----
USUDAW        72
```

```

SELECT DNI, NOTA, DECODE (NOTA,5,'SUFICIENTE',6,'BIEN',7,'NOTABLE',8,'NOTABLE',
                             9,'SOBRESALIENTE',10,'SOBRESALIENTE','SUSPENSO')
FROM NOTAS |;
    
```

DNI	NOTA	DECODE (NOTA, 5
12344345	6	BIEN
12344345	5	SUFICIENTE
12344345	6	BIEN
12344345	6	BIEN
4448242	6	BIEN
4448242	8	NOTABLE
4448242	4	SUSPENSO
4448242	5	SUFICIENTE
56882942	8	NOTABLE
56882942	7	NOTABLE
56882942	8	NOTABLE
56882942	9	SOBRESALIENTE
2112212	3	SUSPENSO
2112212	3	SUSPENSO
2112212	2	SUSPENSO
2112212	6	BIEN

#### U5. 05 Cláusulas avanzadas de selección. Agrupación de elementos: GROUP BY y HAVING

Una select devuelve una serie de filas. Podemos agrupar esas filas para obtener información general del grupo.

Podremos agrupar por uno o varios campos y la información que podremos mostrar es aquello por lo que agrupamos y el resultado de aplicar funciones de grupo sobre los campos. Estas funciones de grupo son las que vimos en funciones:

- SUM (para la suma)
- AVG (para la media)
- MAX (para el máximo)
- MIN (para el mínimo)
- COUNT (para contar cuantos)

**Para realizar los grupos utilizaremos la cláusula GROUP BY**

Por ejemplo:

Agrupar los empleados por departamento y mostrar el código, el máximo y el mínimo salario de cada departamento

```
SELECT DEPT_NO, MAX(SALARIO), MIN(SALARIO)
FROM EMPLE
GROUP BY DEPT_NO;
```

```
DEPT_NO MAX(SALARIO) MIN(SALARIO)
-----
10      4100          1690
20      3000          1040
30      3005          1335
```

3 filas seleccionadas.

**Podemos seguir poniendo condiciones a la hora de seleccionar las filas que queremos después agrupar**

Por ejemplo:

Agrupar los empleados por departamento sin tener en cuenta a los directores ni al presidente y mostrar el código, el máximo y el mínimo salario de cada departamento

```
SELECT DEPT_NO, MAX(SALARIO), MIN(SALARIO)
FROM EMPLE
WHERE OFICIO <> 'DIRECTOR' AND OFICIO <> 'PRESIDENTE'
GROUP BY DEPT_NO;
```

```
DEPT_NO MAX(SALARIO) MIN(SALARIO)
-----
10      1690          1690
20      3000          1040
30      1625          1335
```

3 filas seleccionadas.

**Podemos imponer condiciones sobre los datos una vez agrupados mediante HAVING**

Por ejemplo:

Agrupar los empleados por departamento y mostrar el código, el máximo y el mínimo salario de cada departamento pero solo de departamentos con más de 4 empleados

```
SELECT DEPT_NO, MAX(SALARIO), MIN(SALARIO), COUNT(*)
FROM EMPLE
GROUP BY DEPT_NO
HAVING COUNT(*) > 4;
```

DEPT_NO	MAX(SALARIO)	MIN(SALARIO)	COUNT(*)
20	3000	1040	5
30	3005	1335	6

2 filas seleccionadas.

**Podemos agrupar por más de un campo**

Por ejemplo:

Agrupar a los empleados por departamento y oficio y mostrar cuantos hay en cada grupo y la suma de sus salarios

```
SELECT DEPT_NO, OFICIO, COUNT(*), SUM(SALARIO)
FROM EMPLE
GROUP BY DEPT_NO, OFICIO;
```

DEPT_NO	OFICIO	COUNT(*)	SUM(SALARIO)
10	DIRECTOR	1	2885
10	EMPLEADO	1	1690
10	PRESIDENTE	1	4100
20	ANALISTA	2	6000
20	DIRECTOR	1	2900
20	EMPLEADO	2	2470
30	DIRECTOR	1	3005
30	EMPLEADO	1	1335
30	VENDEDOR	4	6075

9 filas seleccionadas.

**Podemos agrupar por el resultado de una función sobre un campo**

Por ejemplo:

Agrupar a los empleados por año de entrada en la empresa y mostrar cuantos entraron en cada año

```
SELECT TO_CHAR(FECHA_ALT, 'YYYY')||, COUNT(*)
FROM EMPLE
GROUP BY TO_CHAR(FECHA_ALT, 'YYYY');
```

```
TO_C    COUNT(*)
-----
2005          2
2007          5
2008          6
2009          1

4 filas seleccionadas.
```

Podemos seguir combinando tablas

Por ejemplo:

Agrupar a los empleados por localidad y mostrar cuantos trabajan en cada una

```
SELECT LOC, COUNT(*)
FROM EMPLE E, DEPART D
WHERE E.DEPT_NO = D.DEPT_NO
GROUP BY LOC;
```

```
LOC          COUNT(*)
-----
BARCELONA          6
MADRID              5
SEVILLA             3

3 filas seleccionadas.
```

Y podemos realizar consultas más elaboradas utilizando subconsultas:

Mostrar el departamento que más empleados tiene:



```

SELECT DEPT_NO, COUNT(*)
FROM EMPLE E
GROUP BY DEPT_NO;

SELECT DEPT_NO, COUNT(*)
FROM EMPLE E
GROUP BY DEPT_NO
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
FROM EMPLE
GROUP BY DEPT_NO);

```

```

DEPT_NO  COUNT(*)
-----
10         3
20         5
30         6

3 filas seleccionadas.

DEPT_NO  COUNT(*)
-----
30         6

1 fila seleccionada.

```

Mostrar el año en el que han entrado menos empleados

```

SELECT TO_CHAR(FECHA_ALT, 'YYYY'), COUNT(*)
FROM EMPLE E
GROUP BY TO_CHAR(FECHA_ALT, 'YYYY');

SELECT TO_CHAR(FECHA_ALT, 'YYYY'), COUNT(*)
FROM EMPLE E
GROUP BY TO_CHAR(FECHA_ALT, 'YYYY')
HAVING COUNT(*) = (SELECT MIN(COUNT(*))
FROM EMPLE
GROUP BY TO_CHAR(FECHA_ALT, 'YYYY'));

```

```

TO_C      COUNT(*)
-----
2005         2
2007         5
2008         6
2009         1

4 filas seleccionadas.

TO_C      COUNT(*)
-----
2009         1

```

Mostrar el dni del alumno que más suspensos tiene

```
SELECT DNI, COUNT(*)
FROM NOTAS
WHERE NOTA < 5
GROUP BY DNI;

SELECT DNI, COUNT(*)
FROM NOTAS
WHERE NOTA < 5
GROUP BY DNI
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
FROM NOTAS
WHERE NOTA < 5
GROUP BY DNI);
```

```
DNI          COUNT(*)
-----
2112212      3
4448242      1
```

2 filas seleccionadas.

```
DNI          COUNT(*)
-----
2112212      3
|
```

1 fila seleccionada.

Mostrar el nombre del alumno que más suspensos tiene

```
SELECT APENOM, COUNT(*)
FROM ALUMNOS AL, NOTAS N
WHERE AL.DNI = N.DNI
AND NOTA < 5
GROUP BY APENOM;

SELECT APENOM, COUNT(*)
FROM ALUMNOS AL, NOTAS N
WHERE AL.DNI = N.DNI
AND NOTA < 5
GROUP BY APENOM
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
FROM NOTAS
WHERE NOTA < 5
GROUP BY DNI);
```

```
APENOM          COUNT(*)
-----
Cerrato Vela, Luis      1
Sanz Martín, Roberto    3
```

2 filas seleccionadas.

```
APENOM          COUNT(*)
-----
Sanz Martín, Roberto    3
```

1 fila seleccionada.

### U5. 06 Combinación externa (Outer Joins)

Cuando combinamos tablas en una select por su campo(s) común(es), los resultados que se obtienen son las filas de una combinadas con las correspondientes de la otra. Para una fila de una tabla que no tiene correspondientes filas en la otra tabla, no se obtendrán resultados.

Por ejemplo, tenemos los departamentos

```
select * from depart;
```

DEPT_NO	DNOMBRE	LOC
10	CONTABILIDAD	SEVILLA
20	INVESTIGACION	MADRID
30	VENTAS	BARCELONA
40	PRODUCCION	BILBAO

El departamento 40 no tiene empleados

```
select * from emple
where dept_no = 40;
```

ninguna fila seleccionada

Al combinar las tablas EMPLE y DEPART, el departamento 40 no aparece

```
select d.dept_no, loc, apellido
from emple e, depart d
where e.dept_no = d.dept_no;
```

DEPT_NO	LOC	APELLIDO
10	SEVILLA	CEREZO
10	SEVILLA	REY
10	SEVILLA	MUÑOZ
20	MADRID	SANCHEZ
20	MADRID	ALONSO
20	MADRID	FERNANDEZ
20	MADRID	GIL
20	MADRID	JIMENEZ
30	BARCELONA	ARROYO
30	BARCELONA	NEGRO
30	BARCELONA	MARTIN
30	BARCELONA	JIMENO
30	BARCELONA	TOVAR
30	BARCELONA	SALA

14 filas seleccionadas.

En esta combinación el departamento 40 no aparece porque no hay ningún empleado en la tabla EMPLE que tenga dicho departamento.

## TEMA 5: SQL Realización de Consultas

Existe una variedad de combinación de tablas que se llama OUTER JOIN y que permite seleccionar las filas de una tabla que no tienen ninguna correspondencia con las filas de la otra con la que se combina

Consiste en añadir (+) al campo para el que puede no haber valores al hacer la combinación. En este caso, como puede no haber valores en EMPLE para algún departamento, cuando combinemos haremos

where e.dept\_no (+) = d.dept\_no

```
select d.dept_no, loc, apellido
from emple e, depart d
where e.dept_no (+) = d.dept_no;
```

DEPT_NO	LOC	APELLIDO
10	SEVILLA	CEREZO
10	SEVILLA	REY
10	SEVILLA	MUÑOZ
20	MADRID	SANCHEZ
20	MADRID	ALONSO
20	MADRID	FERNANDEZ
20	MADRID	GIL
20	MADRID	JIMENEZ
30	BARCELONA	ARROYO
30	BARCELONA	NEGRO
30	BARCELONA	MARTIN
30	BARCELONA	JIMENO
30	BARCELONA	TOVAR
30	BARCELONA	SALA
40	BILBAO	

15 filas seleccionadas.

También podemos expresarlo de la siguiente manera:

```
SELECT DEPART.DEPT_NO, LOC, APELLIDO
FROM EMPL RIGHT OUTER JOIN DEPART ON depart.dept_no = emple.dept_no;
```

Además existe el LEFT OUTER JOIN

Quitamos la obligatoriedad del campo dept\_no en EMPL para probar que ocurre si tenemos un empleado con dept\_no a NULL y queremos mostrar todos los empleados con sus departamentos incluso si no los tiene.

```
INSERT INTO EMPL VALUES (9000,'BARRIO','JEFAZA',NULL,SYSDATE,10000,NULL,NULL);
```

```
select apellido, depart.dept_no
from depart, emple
where depart.dept_no (+) = emple.dept_no;
```

```
SELECT APELLIDO, DEPART.DEPT_NO
FROM EMPL LEFT OUTER JOIN DEPART ON depart.dept_no = empl.dept_no;
```

### U5. 07 Union, intersect y minus

Union, intersect y minus son operadores de conjunto que nos permiten operar entre el resultado de dos selects.

SELECT ... FROM ... WHERE...

operador\_de\_conjunto

SELECT ... FROM ... WHERE ...

Vamos a probar con las siguientes tablas

Archivo Editar Hoja de trabajo Ayuda																										
<pre>select * from ALUM; select * from NUEVOS; select * from ANTIGUOS;</pre>																										
<table> <thead> <tr> <th>NOMBRE</th><th>EDAD</th><th>LOCALIDAD</th></tr> </thead> <tbody> <tr><td>JUAN</td><td>18</td><td>COSLADA</td></tr> <tr><td>PEDRO</td><td>19</td><td>COSLADA</td></tr> <tr><td>ANA</td><td>17</td><td>ALCALA</td></tr> <tr><td>LUISA</td><td>18</td><td>TORREJÓN</td></tr> <tr><td>MARÍA</td><td>20</td><td>MADRID</td></tr> <tr><td>ERNESTO</td><td>21</td><td>MADRID</td></tr> <tr><td>RAQUEL</td><td>19</td><td>TOLEDO</td></tr> </tbody> </table> <p>7 filas seleccionadas.</p>			NOMBRE	EDAD	LOCALIDAD	JUAN	18	COSLADA	PEDRO	19	COSLADA	ANA	17	ALCALA	LUISA	18	TORREJÓN	MARÍA	20	MADRID	ERNESTO	21	MADRID	RAQUEL	19	TOLEDO
NOMBRE	EDAD	LOCALIDAD																								
JUAN	18	COSLADA																								
PEDRO	19	COSLADA																								
ANA	17	ALCALA																								
LUISA	18	TORREJÓN																								
MARÍA	20	MADRID																								
ERNESTO	21	MADRID																								
RAQUEL	19	TOLEDO																								
<table> <thead> <tr> <th>NOMBRE</th><th>EDAD</th><th>LOCALIDAD</th></tr> </thead> <tbody> <tr><td>JUAN</td><td>18</td><td>COSLADA</td></tr> <tr><td>MAITE</td><td>15</td><td>ALCALA</td></tr> <tr><td>SOFÍA</td><td>14</td><td>ALCALA</td></tr> <tr><td>ANA</td><td>17</td><td>ALCALA</td></tr> <tr><td>ERNESTO</td><td>21</td><td>MADRID</td></tr> </tbody> </table> <p>5 filas seleccionadas.</p>			NOMBRE	EDAD	LOCALIDAD	JUAN	18	COSLADA	MAITE	15	ALCALA	SOFÍA	14	ALCALA	ANA	17	ALCALA	ERNESTO	21	MADRID						
NOMBRE	EDAD	LOCALIDAD																								
JUAN	18	COSLADA																								
MAITE	15	ALCALA																								
SOFÍA	14	ALCALA																								
ANA	17	ALCALA																								
ERNESTO	21	MADRID																								
<table> <thead> <tr> <th>NOMBRE</th><th>EDAD</th><th>LOCALIDAD</th></tr> </thead> <tbody> <tr><td>MARÍA</td><td>20</td><td>MADRID</td></tr> <tr><td>ERNESTO</td><td>21</td><td>MADRID</td></tr> <tr><td>ANDRÉS</td><td>26</td><td>LAS ROZAS</td></tr> <tr><td>IRENE</td><td>24</td><td>LAS ROZAS</td></tr> </tbody> </table> <p>4 filas seleccionadas.</p>			NOMBRE	EDAD	LOCALIDAD	MARÍA	20	MADRID	ERNESTO	21	MADRID	ANDRÉS	26	LAS ROZAS	IRENE	24	LAS ROZAS									
NOMBRE	EDAD	LOCALIDAD																								
MARÍA	20	MADRID																								
ERNESTO	21	MADRID																								
ANDRÉS	26	LAS ROZAS																								
IRENE	24	LAS ROZAS																								

TEMA 5: SQL Realización de Consultas

- **UNION:** devuelve las filas de una consulta seguidas de las de la otra consulta. Las filas duplicadas solo aparecen una vez.
- **UNION ALL:** como la union pero además si una fila es aparece en las dos selects, la saca dos veces
- **INTERSECT:** devuelve las filas que son iguales en ambas consultas
- **MINUS:** devuelve aquellas filas que están en la primera select y no están en la segunda.

```
select * from NUEVOS
UNION
select * from ANTIGUOS;
```

NOMBRE	EDAD	LOCALIDAD
ANA	17	ALCALA
ANDRÉS	26	LAS ROZAS
ERNESTO	21	MADRID
IRENE	24	LAS ROZAS
JUAN	18	COSLADA
MAITE	15	ALCALA
MARÍA	20	MADRID
SOFÍA	14	ALCALA

8 filas seleccionadas.

```
select * from NUEVOS
UNION ALL
select * from ANTIGUOS;
```

NOMBRE	EDAD	LOCALIDAD
JUAN	18	COSLADA
MAITE	15	ALCALA
SOFÍA	14	ALCALA
ANA	17	ALCALA
ERNESTO	21	MADRID
MARÍA	20	MADRID
ERNESTO	21	MADRID
ANDRÉS	26	LAS ROZAS
IRENE	24	LAS ROZAS

9 filas seleccionadas.

```
select * from NUEVOS
INTERSECT
select * from ANTIGUOS;
```

```

NOMBRE                EDAD LOCALIDAD
-----
ERNESTO                21 MADRID

1 fila seleccionada.
```

```
select * from ALUM
MINUS
select * from ANTIGUOS;
```

9 filas seleccionadas.

```

NOMBRE                EDAD LOCALIDAD
-----
ANA                    17 ALCALA
JUAN                   18 COSLADA
LUISA                  18 TORREJÓN
PEDRO                  19 COSLADA
RAQUEL                 19 TOLEDO

5 filas seleccionadas.
```

#### Reglas para la utilización de operadores de conjuntos:

- Las columnas de las dos consultas se relacionan en orden, de izquierda a derecha.
- Los nombres de columna de la primera sentencia select no tienen por qué ser los mismos que los nombres de la segunda
- Las select han de tener el mismo número de columnas
- Los tipos de datos deben coincidir, aunque la longitud no tiene que ser la misma

#### U5. 08 Recuperación jerárquica

La cláusula **CONNECT BY** se emplea para recuperar filas siguiendo una estructura de árbol. Solo se puede aplicar cuando los datos de origen tienen tal estructura.

Vamos a verlo con un ejemplo en la tabla EMPLE, dónde el campo DIR contiene un número de empleado (EMP\_NO) que corresponde a la persona que es directora de cada empleado.

TEMA 5: SQL Realización de Consultas

```

SELECT APELLIDO, DEPT_NO, EMP_NO, DIR, LEVEL
FROM EMPLE
CONNECT BY PRIOR EMP_NO = DIR
START WITH APELLIDO = 'REY';

```

APELLIDO	DEPT_NO	EMP_NO	DIR	LEVEL
REY	10	7839		1
JIMENEZ	20	7566	7839	2
GIL	20	7788	7566	3
ALONSO	20	7876	7788	4
FERNANDEZ	20	7902	7566	3
SANCHEZ	20	7369	7902	4
NEGRO	30	7698	7839	2
ARROYO	30	7499	7698	3
SALA	30	7521	7698	3
MARTIN	30	7654	7698	3
TOVAR	30	7844	7698	3
JIMENO	30	7900	7698	3
CEREZO	10	7782	7839	2
MUÑOZ	10	7934	7782	3

14 filas seleccionadas.

```

SELECT EMP_NO, DIR, SUBSTR(LPAD(APELLIDO, (LEVEL-1)*5+LENGTH(APELLIDO) ),1,30)
FROM EMPLE
CONNECT BY PRIOR EMP_NO = DIR
START WITH APELLIDO = 'REY';

```

EMP_NO	DIR	SUBSTR(LPAD(APELLIDO, (LEVEL-1)*5+LENGTH(APELLIDO) ),1,30)
7839	REY	
7566	7839	JIMENEZ
7788	7566	GIL
7876	7788	ALONSO
7902	7566	FERNANDEZ
7369	7902	SANCHEZ
7698	7839	NEGRO
7499	7698	ARROYO
7521	7698	SALA
7654	7698	MARTIN
7844	7698	TOVAR
7900	7698	JIMENO
7782	7839	CEREZO
7934	7782	MUÑOZ

14 filas seleccionadas.