

SQL

Select

José R. Paramá



Guion

Introducción

Sqlplus

Tablas usadas en los ejemplos

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Introducción

Parte del SEQUEL (Structured English QUery Language), desarrollado en laboratorios de IBM para el SYSTEM R.

Se basa en el álgebra relacional y el cálculo relacional.

Este lenguaje evolucionó al SQL (Structured Query Language).

ANSI decidió estandarizarlo, se le llamó SQL-86 o SQL1. ISO la aceptó en el 87.

Aparecieron luego otros estándares: SQL-92 (SQL2), SQL:1999 (SQL3), SQL:2003, SQL:2006, SQL:2008, SQL:2011

Introducción

La mayor parte es seguida por los fabricantes, pero hay pequeñas divergencias, por eso el SQL que sigue cada producto se denomina *dialecto*.

A pesar de su nombre, no es sólo un lenguaje de consulta y cubre:

- DDL o LDD (lenguaje de definición de datos).

- DML o LMD (lenguaje de manipulación de datos).

Con SQL se puede realizar cualquier tarea dentro del SGBD(crear usuarios, dar permisos, control concurrencia, creación de estructuras de almacenamiento y acceso a los datos, etc.)

Introducción

Otras características:

Es posible incrustar SQL dentro de programas escritos con lenguajes de propósito general.

Es un lenguaje no procedimental (se especifica lo *qué* queremos, y no especificamos el *cómo*).

No manipula conjuntos de filas (como el modelo relacional teórico), maneja *colecciones* de filas (no hay orden, pero puede haber filas repetidas).

Cambia también la terminología: *tablas*, *filas* y *columnas* en lugar de *relaciones*, *tuplas* y *atributos*.

SQL*Plus de Oracle

Programa para ejecutar sentencias de SQL en un entorno Oracle.

Tiene una serie de comandos propios.

Entrar:

```
$ sqlplus
```

Nombre de usuario: nombre de usuario de la UDC entre comillas dobles, ej: "pepe.gotera"

Contraseña: DNI más letra en mayúscula ej: 41222444R

Si entras anteponiendo *rlwrap*:

```
$ rlwrap sqlplus
```

Con las teclas ↑↓ puedes acceder a las últimas consultas

Cambiar contraseña:

```
password
```

SQL*Plus

Los comandos SQL*plus se escriben en una única línea y no necesitan ;

```
DESCRIBE emp
```

Las sentencias SQL se pueden escribir en varias filas y acaban en ;

```
SQL> select *  
      2  from emp;
```

SQL*Plus

Comandos:

DESC[RIBE] <tabla> muestra la estructura (columnas y tipos de datos) de la tabla.

```
SQL> DESCRIBE emp
```

Nombre	00 Nulo?	Tipo
-----	-----	-----
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

EXIT salir

SQL*Plus

SQL*Plus usa un buffer para almacenar la última sentencia SQL ejecutada (no afecta a los comandos SQL*Plus). Incorpora un editor de líneas elemental para realizar cambios mínimos en esa sentencia SQL:

Comandos para usar el buffer:

`L[IST]`: Visualiza el contenido del buffer

`ED[IT]`: Abre el contenido del buffer en un editor (modo texto).

`R[UN]`: Ejecuta el contenido del buffer.

`SAV[E] fichero[.sql]`: Salva el contenido del buffer en un fichero con nombre `fichero[.sql]`.

`GET fichero[.sql]`: Carga el contenido del fichero `fichero[.sql]` en el buffer.

Para configurar el editor de texto y otros aspectos del entorno, descarga el fichero `login.sql` del Moodle, y ponlo en un directorio donde *siempre debes arrancar* el SQL*plus.

Si no tienes cargado el `login.sql`, puedes escribir `DEFINE_EDITOR=pico` (o `DEFINE_EDITOR=vi` si lo sabes manejar) para configurar el editor.

Tablas usadas en los ejemplos

EMP		
Campo	Tipo	Descripción
<u>EMPNO</u> :	NUMBER(4) NOT NULL	Número o código del empleado. Es la clave primaria de la tabla.
ENAME	VARCHAR2(10)	Nombre del empleado
JOB	VARCHAR2(9)	Trabajo del empleado
MGR	NUMBER(4)	Código del jefe del empleado. Clave foránea que referencia (cíclicamente) la tabla EMP
HIREDATE	DATE	Fecha de contratación.
SAL	NUMBER(7, 2)	Salario mensual del empleado
COMM	NUMBER(7, 2)	Comisión
DEPTNO	NUMBER(2)	Código del departamento al que el empleado está adscrito. Clave foránea que referencia la tabla DEPT

Un nulo en COMM significa que el empleado no trabaja a comisión (el valor no procede).

Un nulo en MGR significa que no tiene jefe (también "no procede")

DEPT		
Campo	Tipo	Descripción
<u>DEPTNO</u>	NUMBER(2) NOT NULL	Número o código del departamento. Es la clave primaria de la tabla.
DNAME	VARCHAR2(14)	Nombre del departamento.
LOC	VARCHAR2(13)	Localidad (o ciudad) donde el departamento está ubicado.

PRO		
Campo	Tipo	Descripción
<u>PRONO</u>	NUMBER(4) NOT NULL	Número o código del Proyecto. Es la clave primaria de la tabla.
PNAME	VARCHAR2(10)	Nombre del proyecto.
LOC	VARCHAR2(13)	Ciudad donde se realiza el proyecto.
DEPTNO	NUMBER(2)	Número del departamento controlador del proyecto. Clave foránea que referencia la tabla DEPT

EMPPRO		
Campo	Tipo	Descripción
<u>EMPNO</u>	NUMBER(4) NOT NULL	Número o código del empleado. Clave foránea que referencia la tabla EMP
<u>PRONO</u>	NUMBER(4) NOT NULL	Número o código del proyecto. Clave foránea que referencia la tabla PRO
HOURS	NUMBER(2)	Horas que ha trabajado un empleado en un proyecto.

Es la clave primaria de la tabla

Tablas usadas en los ejemplos

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30
7566	JONES	MANAGER	7839	02/04/81	2,975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20
7839	KING	PRESIDENT		17/11/81	5,000		10
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1,100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3,000		20
7934	MILLER	CLERK	7782	23/01/82	1,300		10

14 filas seleccionadas.

```
SQL> select * from dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 filas seleccionadas.

```
SQL> select * from emppro;
```

EMPNO	PRONO	HOURS
7499	1004	15
7499	1005	12
7521	1004	10
7521	1008	8
7654	1001	16
7654	1006	15
7654	1008	5
7844	1005	6
7934	1001	4

9 filas seleccionadas.

```
SQL> select * from pro;
```

PRONO	PNAME	LOC	DEPTNO
1001	P1	BOSTON	20
1004	P4	CHICAGO	30
1005	P5	CHICAGO	30
1006	P6	LOS ANGELES	30
1008	P8	NEW YORK	30

5 filas seleccionadas.

Guion

Introducción

Conceptos previos

Nulos

Expresiones

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Conceptos previos

Nulos:

El valor nulo **NULL** representa la *ausencia de información*, o bien por desconocimiento del dato, o bien porque no procede.

Debe diferenciarse de cualquier otro valor, entre ellos del valor 0 si se trata de un dato numérico, y de la cadena de caracteres vacía, si es un dato de tipo carácter.

Conceptos previos

Expresiones¹:

Una expresión es la formulación de una secuencia de operaciones, o sea, una combinación de operadores, operandos y paréntesis, que, cuando se ejecuta, *devuelve un único valor escalar como resultado*.

Los operandos pueden ser constantes, nombres de columna, funciones, otras expresiones y otros elementos.

El tipo de dato de cada operando de una expresión debe ser el mismo. Si un operando es nulo, el resultado también es nulo

Operadores numéricos: + - * /

Operador alfanumérico: || (concatenación de cadenas de texto)

Ejemplos:

- 3
- 'Casa'
- 3+2
- 'A' || 'BC'
- ENAME
- SAL*1.5
- 0.5 * COMM
- SAL + COMM
- (Select COMM FROM EMP WHERE EMPNO = 7499)

No son expresiones:

- SAL < 1500
- (SAL+COMM) >=10

¹Restringimos la definición de expresión a la versión “Core SQL” del estándar.

Conceptos previos

En Oracle, el texto (literal de texto) va entre comillas simples, y es sensible a las mayúsculas/minúsculas.

`'casa'`

`'Casa'`

`'Casa bonita'`

Los identificadores (de usuario, nombres de columnas,...) van entre comillas dobles. Se pueden omitir las comillas cuando el identificador no tiene espacios o símbolos de puntuación.

`psanchez`

`"Pedro Sanchez"`

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsulta

Composición de consultas

Sentencia Select

La sentencia SELECT permite seleccionar u obtener datos de una o de varias tablas. Parte de una o de varias tablas de la BD y el resultado es otra tabla, denominada a veces tabla resultado, pero que no formará parte de la BD.

```
SELECT [DISTINCT|ALL] {* | <expr1>[, <expr2>] ...}  
  FROM <tabla1>[[INNER|LEFT|RIGHT|FULL|CROSS] JOIN <tabla2> ...]  
  [WHERE <condicion_where>]  
  [GROUP BY <columna1>[,<columna2>, ...]  
  [HAVING <condicion_having>]  
  [ORDER BY <expr_orderby1>[, ...]]
```

Sentencia Select

```
SELECT [DISTINCT|ALL] { * | <expr1>[, <expr2>] ...}  
FROM <tabla1> [[INNER|LEFT|RIGHT|FULL|CROSS] JOIN <tabla2> ..  
[WHERE <condicion_where>]  
[GROUP BY <columna1>[, <columna2>, ...]  
[HAVING <condicion_having>]  
[ORDER BY <expr_orderby1>[, ...]]
```

El orden de ejecución de las cláusulas y la función de cada una es:

1. **FROM**(obligatoria)

Partiendo de una o más tablas *obtiene una única tabla* que será procesada por el resto de cláusulas

2. **WHERE** (optativa)

De las filas que le pasa el FROM, *elimina las filas que NO HACEN CIERTA la condición* especificada

3. **GROUP BY** (optativa)

4. **HAVING** (optativa)

5. **SELECT** (obligatoria)

Cada fila que le llega, es usada para obtener una fila del resultado.

Se procesan las filas de una en una, cuando se procesa una fila se evalúa sobre las expresiones, *cada expresión da lugar a una columna de la tabla resultado*.

Alternativamente un * indica que en el resultado se añadan todas las columnas.

Si hubiese *filas repetidas*, de forma predeterminada aparecen, pero no lo hacen si se incluye **DISTINCT**.

6. **ORDER BY** (optativa)

Permite determinar el criterio de ordenación de las filas de la tabla resultado. Sin ella obtendremos las mismas filas, pero no hay garantía de en qué orden, que será el que dicte la estrategia seguida por el SGBD para extraer los datos.

Sentencia Select

```
Select 'Nombre: ' || ename, sal*0.20  
from emp  
where deptno=10;
```

El resultado de este paso es

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30
7566	JONES	MANAGER	7839	02/04/81	2,975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20
7839	KING	PRESIDENT		17/11/81	5,000		10
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1,100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3,000		20
7934	MILLER	CLERK	7782	23/01/82	1,300		10

La sentencia Select

```
Select 'Nombre: ' || ename, sal*0.20  
from emp  
where deptno=10;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	
7369	SMITH	CLERK	7902	17/12/80	800		20	Falso
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30	Falso
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30	Falso
7566	JONES	MANAGER	7839	02/04/81	2,975		20	Falso
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30	Falso
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30	Falso
7782	CLARK	MANAGER	7839	09/06/81	2,450		10	Cierto: continúa
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20	Falso
7839	KING	PRESIDENT		17/11/81	5,000		10	Cierto: continúa
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30	Falso
7876	ADAMS	CLERK	7788	23/05/87	1,100		20	Falso
7900	JAMES	CLERK	7698	03/12/81	950		30	Falso
7902	FORD	ANALYST	7566	03/12/81	3,000		20	Falso
7934	MILLER	CLERK	7782	23/01/82	1,300		10	Cierto: continúa

El resultado de este paso es

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7839	KING	PRESIDENT		17/11/81	5,000		10
7934	MILLER	CLERK	7782	23/01/82	1,300		10

La sentencia Select

```
Select 'Nombre: ' || ename, sal*0.20  
from emp  
where deptno=10;
```

Cada coma separa dos expresiones,
Y cada expresión da lugar a una columna
en el resultado

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7839	KING	PRESIDENT		17/11/81	5,000		10
7934	MILLER	CLERK	7782	23/01/82	1,300		10

'Nombre: ' ename	sal*0.20
'Nombre: ' CLARK	2450*0.20
'Nombre: ' KING	5000*0.20
'Nombre: ' MILLER	1300*0.20

El resultado de este paso es

'NOMBRE: ' ENAME	SAL*0.20
Nombre: CLARK	490
Nombre: KING	1000
Nombre: MILLER	260

La sentencia Select

Se puede cambiar el nombre de una columna.

SELECT <expr1> [AS] nuevo_nombre, ...

```
select ename as nombre, sal salario, sal+comm as "ingresos mensuales", hiredate "fecha contratación"
from emp;
```

nombre character varying(10)	salario numeric(7,2)	ingresos mensuales numeric	fecha contratación date
KING	5000.00		1981-11-17
JONES	2975.00		1981-04-02
FORD	3000.00		1981-12-03
SMITH	800.00		1980-12-17
BLAKE	2850.00		1981-05-01
ALLEN	1600.00	1900.00	1981-02-20
WARD	1250.00	1750.00	1981-02-22
MARTIN	1250.00	2650.00	1981-09-28
CLARK	2450.00		1981-06-09
SCOTT	3000.00		1982-12-09
TURNER	1500.00	1500.00	1981-09-08
ADAMS	1100.00		1983-01-12
JAMES	950.00		1981-12-03
MILLER	1300.00		1982-01-23

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Distinct

Si se incluye ***DISTINCT*** antes de las expresiones en la cláusula select, se eliminarán ***FILAS REPETIDAS***.

```
SQL> select job, deptno  
2   from emp  
3   order by job, deptno;
```

JOB	DEPTNO
-----	-----
ANALYST	20
ANALYST	20
CLERK	10
CLERK	20
CLERK	20
CLERK	30
MANAGER	10
MANAGER	20
MANAGER	30
PRESIDENT	10
SALESMAN	30
SALESMAN	30
SALESMAN	30
SALESMAN	30

14 filas seleccionadas.

```
SQL> select distinct job, deptno  
2   from emp  
3   order by job, deptno;
```

JOB	DEPTNO
-----	-----
ANALYST	20
CLERK	10
CLERK	20
CLERK	30
MANAGER	10
MANAGER	20
MANAGER	30
PRESIDENT	10
SALESMAN	30

9 filas seleccionadas.

Distinct

Los nulos, *para el distinct, son iguales*

```
SQL> select deptno, comm  
2  from emp  
3  order by deptno, comm;
```

DEPTNO	COMM
--------	------

10	
10	
10	
20	
20	
20	
20	
20	
30	0
30	300
30	500
30	1,400
30	
30	

14 filas seleccionadas.

```
SQL> select distinct deptno, comm  
2  from emp  
3  order by deptno, comm;
```

DEPTNO	COMM
--------	------

10	
20	
30	0
30	300
30	500
30	1,400
30	

7 filas seleccionadas.

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Order by

ORDER BY <expr_orderby1> [ASC|DESC], <expr_orderby2> [ASC|DESC],...

Ordena las **FILAS** obtenidas.

Las expresiones de order by pueden ser *expresiones que no sean un literal* (una constante).

Tanto la expresión como la columna no tienen que aparecer necesariamente en la cláusula select.

Correctas

```
select ename, job  
from emp  
order by hiredate;
```

```
select ename, job  
from emp  
order by sal+comm;
```

Order by

Si no se indica nada el ordenamiento por defecto es ascendente (ASC).

```
select ename, sal  
from emp  
order by sal;
```

ename character varying(10)	sal numeric(7,2)
SMITH	800.00
JAMES	950.00
ADAMS	1100.00
WARD	1250.00
MARTIN	1250.00
MILLER	1300.00
TURNER	1500.00
ALLEN	1600.00
CLARK	2450.00
BLAKE	2850.00
JONES	2975.00
SCOTT	3000.00
FORD	3000.00
KING	5000.00

```
select ename, sal  
from emp  
order by sal DESC;
```

ename character varying(10)	sal numeric(7,2)
KING	5000.00
SCOTT	3000.00
FORD	3000.00
JONES	2975.00
BLAKE	2850.00
CLARK	2450.00
ALLEN	1600.00
TURNER	1500.00
MILLER	1300.00
MARTIN	1250.00
WARD	1250.00
ADAMS	1100.00
JAMES	950.00
SMITH	800.00

Order by

Se puede usar también el nombre de columna (en lugar de usar la expresión que la define).

```
select ename as nombre, sal salario, sal+comm as "ingresos mensuales",  
hiredate "fecha contratación"  
from emp  
order by "ingresos mensuales"
```

Order by

Si hay varias expresiones de ordenamiento, se ordenan las *filas* primero por la primera expresión de ordenamiento, para aquellas filas con el mismo valor en la primera expresión de ordenamiento, se desempata por la segunda expresión de ordenamiento, y así sucesivamente.

```
select ename, deptno, sal
from emp
order by deptno, sal;
```

ename character varying(10)	deptno numeric(2,0)	sal numeric(7,2)
MILLER	10	1300.00
CLARK	10	2450.00
KING	10	5000.00
SMITH	20	800.00
ADAMS	20	1100.00
JONES	20	2975.00
FORD	20	3000.00
SCOTT	20	3000.00
JAMES	30	950.00
MARTIN	30	1250.00
WARD	30	1250.00
TURNER	30	1500.00
ALLEN	30	1600.00
BLAKE	30	2850.00

Se ordena
por la
segunda

Igual valor en la
primera

Order by

Se puede ordenar ascendentemente en unas expresiones, y descendente en otras

```
select ename, deptno, sal  
from emp  
order by deptno, sal desc;
```

ename character varying(10)	deptno numeric(2,0)	sal numeric(7,2)
KING	10	5000.00
CLARK	10	2450.00
MILLER	10	1300.00
FORD	20	3000.00
SCOTT	20	3000.00
JONES	20	2975.00
ADAMS	20	1100.00
SMITH	20	800.00
BLAKE	30	2850.00
ALLEN	30	1600.00
TURNER	30	1500.00
WARD	30	1250.00
MARTIN	30	1250.00
JAMES	30	950.00

Descendente

Ascendente

Order by

Para el order by, por convención, los nulos se consideran *mayores* que cualquier valor.

```
select ename, sal, comm  
from emp  
order by comm;
```

ename character varying(10)	sal numeric(7,2)	comm numeric(7,2)
TURNER	1500.00	0.00
ALLEN	1600.00	300.00
WARD	1250.00	500.00
MARTIN	1250.00	1400.00
BLAKE	2850.00	
CLARK	2450.00	
SCOTT	3000.00	
ADAMS	1100.00	
JAMES	950.00	
KING	5000.00	
MILLER	1300.00	
JONES	2975.00	
FORD	3000.00	
SMITH	800.00	

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Predicados elementales

Los predicados permiten especificar una condición.

Se pueden usar en las partes *where* y *having*.

<expres1> <op_condición> <expres2>

<op_condición> puede ser: < <= = != <> >= >

El resultado de un predicado dentro de una cláusula *where*, como hemos visto, se aplica a ***una única fila*** y su resultado puede ser: ***cierto (true)***, ***falso (false)*** o ***desconocido (null)***.

El motivo del tercer resultado posible es la presencia de nulos.

Cuando <expres1> o <expres2> es un nulo, el resultado es ***desconocido***.

Predicados elementales

```
Select 'Nombre: ' || ename, sal*0.20
from emp
where comm>1000;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30
7566	JONES	MANAGER	7839	02/04/81	2,975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20
7839	KING	PRESIDENT		17/11/81	5,000		10
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1,100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3,000		20
7934	MILLER	CLERK	7782	23/01/82	1,300		10

- Desconocido
- Falso
- Falso
- Desconocido
- Cierto: *continúa*
- Desconocido
- Desconocido
- Desconocido
- Desconocido
- Falso
- Desconocido
- Desconocido
- Desconocido
- Desconocido

Resultado final

?column? text	?column? numeric
nombre: MARTIN	250.0000

Predicados elementales

`<expres1> <op_condición> <expres2>`

Observa que a los dos lados de la condición puede haber *expresiones*

```
select ename, job, sal, comm, sal+comm
from emp
where sal+comm > 2500;
```

```
select ename, job, sal, comm, sal+comm
from emp
where 1000 = 1000;          ?????
```

```
select ename, job, sal, comm, sal+comm
from emp
where null = null;          ?????
```

```
select ename, job, sal, comm, sal+comm
from emp
where comm= null;           ?????
```

Predicados de nulos

Los predicados de comparación no sirven para determinar los valores nulos.

Como hemos visto, no es válido **COMM = NULL** porque sería discernir si un valor (que también puede ser desconocido) es igual a desconocido.

Se requiere un predicado especial, con formato: **<expr> IS [NOT] NULL**

```
select ename, sal, comm, sal+comm total
from emp
where sal+comm is null;
```

ename character varying(10)	sal numeric(7,2)	comm numeric(7,2)	total numeric
KING	5000.00		
JONES	2975.00		
FORD	3000.00		
SMITH	800.00		
BLAKE	2850.00		
CLARK	2450.00		
SCOTT	3000.00		
ADAMS	1100.00		
JAMES	950.00		
MILLER	1300.00		

```
select ename, sal, comm, sal+comm total
from emp
where sal+comm is not null;
```

ename character varying(10)	sal numeric(7,2)	comm numeric(7,2)	total numeric
ALLEN	1600.00	300.00	1900.00
WARD	1250.00	500.00	1750.00
MARTIN	1250.00	1400.00	2650.00
TURNER	1500.00	0.00	1500.00

Predicado Between

Predicado de rango o predicado **BETWEEN**

Compara si los valores de una expresión están o no entre los valores de otras dos (incluyendo los extremos).

Formato: **<expr1> [NOT] BETWEEN <expr2> AND <expr3>**

```
SELECT *  
FROM emp  
WHERE sal BETWEEN 1500 AND 3000;
```

Predicados de pertenencia a conjunto

Predicado de pertenencia a conjunto (IN)

Comprueba si el valor de una expresión coincide con alguno de los valores incluidos en una lista de expresiones.

Formato: **<expr1> [NOT] IN (<expr2>[, <expr3>, ...])**

```
SELECT *  
FROM emp  
WHERE deptno IN (10, 30, 40) ;
```

```
SELECT ename  
FROM emp  
WHERE job IN ( 'CLERK' , ' SALESMAN' ) ;
```

<expr1> IN (<expr2>,<expr3>) es lo mismo que **<expr1>=<expr2> OR <expr1>=<expr3>**
<expr1> NOT IN (<expr2>,<expr3>) es lo mismo que **<expr1>!=<expr2> AND <expr1>!=<expr3>**

Predicado LIKE

Predicado de correspondencia con un patrón o modelo

Comprueba si el valor de una expresión alfanumérica se corresponde con un modelo. El modelo puede incluir dos caracteres que actúan como comodines:

- Indica un único carácter, incluido el blanco.
- % Indica una cadena de caracteres de cualquier longitud, incluida la cadena vacía.

Formato: **<expr1> [NOT] LIKE <modelo>**

```
SELECT *  
FROM emp  
WHERE ename LIKE '%NE%'
```

```
SELECT *  
FROM emp  
WHERE ename LIKE '_____'
```


Predicados compuestos

Son la unión de dos o más predicados mediante los operadores lógicos **AND**, **OR** y **NOT**.

Al existir una lógica de tres valores, debemos considerar el efecto del valor **NULL**.

X	Y	X AND Y	X OR Y	NOT X
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	DESC.	DESC.	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	DESC.	FALSE	DESC.	TRUE
DESC.	TRUE	DESC.	TRUE	DESC.
DESC.	FALSE	FALSE	DESC.	DESC.
DESC.	DESC.	DESC.	DESC.	DESC.

Predicados compuestos

```
Select *
from emp
where sal+comm>2500;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	
7369	SMITH	CLERK	7902	17/12/80	800		20	Desconocido
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30	Falso
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30	Falso
7566	JONES	MANAGER	7839	02/04/81	2,975		20	Desconocido
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30	Cierto
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30	Desconocido
7782	CLARK	MANAGER	7839	09/06/81	2,450		10	Desconocido
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20	Desconocido
7839	KING	PRESIDENT		17/11/81	5,000		10	Desconocido
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30	Falso
7876	ADAMS	CLERK	7788	23/05/87	1,100		20	Desconocido
7900	JAMES	CLERK	7698	03/12/81	950		30	Desconocido
7902	FORD	ANALYST	7566	03/12/81	3,000		20	Desconocido
7934	MILLER	CLERK	7782	23/01/82	1,300		10	Desconocido

```
Select *
from emp
where sal+comm>2500
      or sal > 2500;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	
7369	SMITH	CLERK	7902	17/12/80	800		20	Desc.(Desc. OR Falso)
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30	Falso (Falso OR Falso)
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30	Falso
7566	JONES	MANAGER	7839	02/04/81	2,975		20	Cierto: (Desc. OR Cierto)
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30	Cierto: (Cierto OR Falso)
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30	Cierto: (Desc. OR Cierto)
7782	CLARK	MANAGER	7839	09/06/81	2,450		10	Desc (Desc. OR Falso):
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20	Cierto:
7839	KING	PRESIDENT		17/11/81	5,000		10	Cierto:
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30	Falso (Falso OR Falso)
7876	ADAMS	CLERK	7788	23/05/87	1,100		20	Desc.(Desc. OR Falso)
7900	JAMES	CLERK	7698	03/12/81	950		30	Desc
7902	FORD	ANALYST	7566	03/12/81	3,000		20	Cierto: (Desc. OR Cierto)
7934	MILLER	CLERK	7782	23/01/82	1,300		10	Desc.

Ejercicios

1. Muestra los puestos de trabajo que hay en cada departamento (código de dept y nombre del puesto de trabajo). No deben aparecer repetidos.
2. Muestra los códigos de empleados que son jefes. En el resultado no debe aparecer filas con nulos.
3. Muestra las ciudades donde se ejecutan proyectos controlados por el departamento 30. No debe aparecer repetidos en el resultado.
4. Muestra empleados que no tienen jefe.
5. Muestra empleados que tengan jefe y que ganen (incluyendo salario y comisión) más de 2500.
6. Muestra los empleados cuyo nombre empieza por 'S'.
7. Muestra los empleados que ganan (incluyendo salario y comisión) entre 1500 y 2500 euros.
8. Muestra los empleados que son 'CLERK', 'SALESMAN' o 'ANALYST' y gana (incluyendo salario y comisión) más de 1250

Ejercicios

1.

```
Select distinct deptno, job
from emp
```
2.

```
Select mgr
from emp
where mgr is not null
```
3.

```
Select distinct loc
from pro
where deptno=30
```
4.

```
Select empno, ename
from emp
where mgr is null
```
5.

```
Select empno, ename
from emp
where mgr is not null
and (sal>2500 or
sal+comm>2500)
```
6.

```
Select empno, ename
from emp
where ename like 'S%'
```
7.

```
Select empno, ename
from emp
where (sal between 1500 and 2500
and comm is null)
or
(sal+comm between 1500 and 2500)
```
8.

```
Select empno, ename, sal, comm, job
from emp
where job in ('ANALYST','CLERK','SALESMAN')
and
(sal+comm > 1250
or sal >1250)
```

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Escalares

Colectivas

Group by

Having

Join

Subconsultas

Composición de consultas

Funciones Escalares

Son funciones que se aplican dentro de expresiones *normales*, y por tanto se puede utilizar en *cualquier sitio* donde se espere una expresión.

Es decir, sobre expresiones que se aplican **SOBRE UNA FILA** y devuelven **un valor para esa fila**.

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10

```
select sqrt(sal)
from emp;
```



sqrt numeric
70.710678118654752
54.543560573178572
54.772255750516611
28.284271247461901
53.385391260156556
40.000000000000000
35.355339059327376
35.355339059327376
49.497474683058327
54.772255750516611
38.729833462074169
33.166247903553998
30.822070014844882
36.055512754639893

Funciones Escalares

Hay muchas, algunos ejemplos:

Numéricas o aritméticas:

SQRT (<exp_numerica>) Raíz cuadrada. Ej.: SQRT(81)

ABS (<exp_numerica>) Valor absoluto. Ej.: ABS(-11)

POWER (<exp1>, <exp2>) Potencia. Ej.: POWER(9,2) = 81

Alfanuméricas o de cadenas de caracteres:

SUBSTR (<exp1>, <exp2> [, <exp3>]) Subcadena de <exp1> empezando en la posición <exp2> y de longitud <exp3>.

Ej.: SUBSTR('Materia',2,4) = 'ater' SUBSTR('Materia',5) = 'ria'

UPPER (<exp_caracter>) Pasa a mayúsculas. Ej.: UPPER('Materia') = 'MATERIA'

LOWER (<exp_caracter>) Pasa a minúsculas. Ej.: LOWER('Materia') = 'materia'

LENGTH (<exp_carácter>) cuenta nº caracteres

De fecha y tiempo:

CURRENT_DATE Fecha actual del sistema. Ej.: SELECT CURRENT_DATE FROM DUAL

Funciones Escalares

Función útil cuando hay valores nulos en expresiones.

Formato: **COALESCE**(<expr1>, <expr2>, ...)

Funcionamiento:

Evalúa la <expr1>. Si su valor es distinto de NULL, devuelve dicho valor. En caso contrario, evalúa la <expr2> y devuelve el resultado, y así sucesivamente.

```
SELECT COALESCE(sal + comm, sal)
FROM emp;
```

En este ejemplo se evalúa la suma del salario y la comisión de cada empleado. Si el resultado es distinto de NULL, se devuelve dicho resultado. Si el resultado es NULL (debido a que la comisión del empleado es NULL), se evalúa el salario de cada empleado y se devuelve ese valor.

COALESCE(<expr1>, <expr2>, ...) es una expresión, y por tanto se puede usar en *cualquier sitio* donde se espere una expresión, por ej:

```
SELECT COALESCE(sal + comm, sal)
FROM emp
WHERE COALESCE(comm, 0) + sal > 2500
```


Funciones Colectivas

Las funciones colectivas (o de agrupamiento, o de conjuntos), son funciones que se aplican sobre una **COLECCIÓN DE FILAS** y devuelve **un valor para esa colección de filas**.

Una expresión que contiene una función colectiva sigue siendo una expresión, pero ahora ya no se puede usar en cualquier sitio, existen restricciones.

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10

```
select sum(sal)
from emp;
```

sum numeric
29025.00

En este ejemplo al no haber where llega toda la tabla a la cláusula select

Funciones Colectivas

```
select sum(sal)
from emp
where job='CLERK';
```

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10

sum numeric
4150.00

Como aquí hay where
lo que llega a la cláusula
select es el resultado del
where

Primero se
ejecuta el
where

Y a
continuación
la cláusula
select

Funciones Colectivas

Formato: **func (<expr>)**

Muchas permiten ALL y DISTINCT: **<func> ([ALL | DISTINCT] <expr>)**

Si aparece DISTINCT se eliminan los valores repetidos del argumento, antes de calcular la función.

Las más frecuentes:

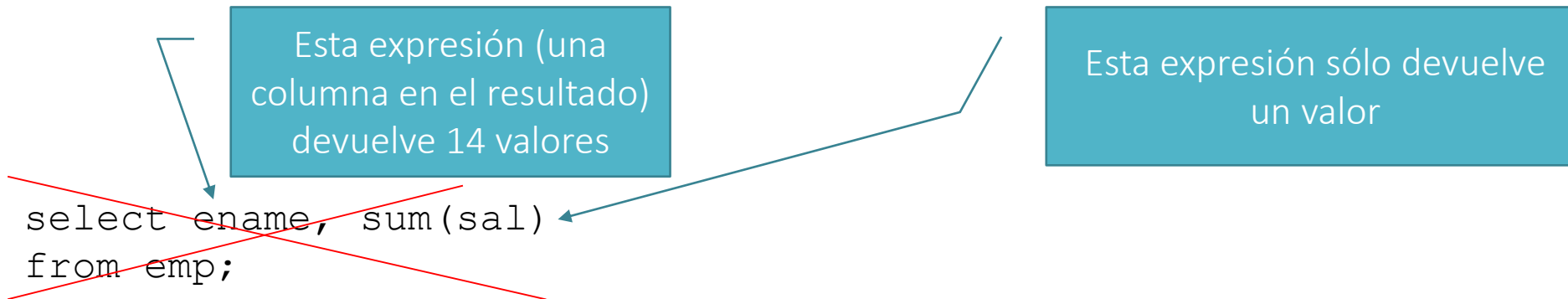
AVG	Media	COUNT	Contar
MAX	Máximo	MIN	Mínimo
SUM	Suma	VAR	Varianza

El estándar indica que la expresión no puede ser una subconsulta ni una expresión con una función colectiva (no se puede anidar funciones colectivas).

Aunque algunos SGBD sí permiten un nivel de anidamiento.

Funciones Colectivas

Si se incluye una función de agrupamiento en la cláusula SELECT todas las expresiones en dicha cláusula deben tener un valor único para el conjunto de las filas.



No se puede construir una tabla resultado con una columna de 14 valores y otra columna con sólo un valor. La tabla tiene una *bolsa* de filas, y cada fila tiene que tener un valor (o un nulo) para cada una de las columnas (2 en este ejemplo).

El propio SGBD da un error:

```
select ename, sum(sal)
      2  from emp;
select ename, sum(sal)
      *
```

ERROR en línea 1:

ORA-00937: la función de grupo no es de grupo único

Funciones Colectivas

Las funciones colectivas eliminan (casi siempre) los nulos antes de realizar su operación.

```
select comm, mgr  
from emp;
```

comm numeric(7,2)	mgr numeric(4,0)
	7839
	7566
	7902
	7839
300.00	7698
500.00	7698
1400.00	7698
	7839
	7566
0.00	7698
	7788
	7698
	7782

10
nulos

1 nulo y 13 distintos
de nulo

```
select sum(comm), count(mgr)  
from emp;
```

sum numeric	count bigint
2200.00	13

Suma de los
distintos de
nulo

Cuenta los
distintos de
nulo

Funciones Colectivas

Se puede incluir un *distinct* dentro de cada función colectiva (es el único modo de que pueda aparecer más de un distinct en una cláusula select).

<func> ([ALL | DISTINCT] <expr>)

```
select deptno
from emp;
```

deptno numeric(2,0)
10
20
20
20
30
30
30
30
10
20
30
20
30
10

14 valores y 3
distintos (10,20,30)

```
select sum(distinct deptno), count(distinct deptno)
from emp;
```

sum numeric	count bqint
60	3

Cuenta (10,20,30)

Suma (10,20,30)

Funciones Colectivas

La función `count` tiene pequeñas diferencias:

`count ([ALL] <expresión>)` cuenta cuántos valores distintos de nulo hay en la columna correspondiente a `<expresión>` en la tabla resultante.

`count (DISTINCT <expresión>)` cuenta cuántos valores distintos de nulo y distintos entre sí hay en la columna correspondiente a `<expresión>` en la tabla resultante.

`count (*)` cuenta cuántas filas hay en la tabla resultante (aún cuando las filas sean todo nulos).

```
select *  
from tabla;
```

```
select count(col1), count(distinct col1), count(*)  
from tabla;
```

col1 integer	col2 integer
1	1
1	1
2	2

Dos filas todo nulos

5 filas, 3 distintos de nulo y 2 distintos (1,2)

count bigint	count bigint	count bigint
3	2	5

Ejercicios

1. Muestra cuántos empleados hay y a cuánto ascienden sus ingresos (sumando los de todos e incluyendo salario y comisión) que sean SALESMAN o CLERK.
2. Cuántos empleados tienen comisión, cuántos no tienen comisión, a cuánto asciende el salario medio, y cuánto asciende la comisión media.
3. Empleados con un nombre de más de 5 letras.
4. Cuántos empleados trabajan para los departamentos 20 y 30, y cuántos trabajos distintos se desempeñan en esos departamentos.
5. Cuántos empleados tienen jefe, cuántos son jefes y cuántos no son jefes.
6. Cuántos son los ingresos (salario más comisión) medios de los empleados contratados después del 01-08-1981.

Ejercicios

1.

```
Select count(*),sum(sal+coalesce(comm,0))  
from emp  
where job in ('SALESMAN','CLERK')
```
2.

```
Select count(comm), count(*)-count(comm),avg(sal), avg(comm)  
from emp
```
3.

```
Select ename  
from emp  
where length(ename)>5
```
4.

```
Select count(*), count(distinct job)  
from emp  
where deptno in (20,30)
```
5.

```
Select count(mgr), count(distinct mgr), count(empno)-count(distinct mgr)  
from emp
```
6.

```
select avg(coalesce(sal+comm,sal))  
from emp  
where hiredate >'01-08-1981'
```

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Group by

Hasta ahora, las funciones de agrupamiento actuaban sobre el conjunto de las filas que le llegan a la cláusula select.

En lugar de aplicar las funciones colectivas sobre todas las filas, éstas se pueden agrupar, formando más de un grupo de filas, y entonces aplicar las funciones sobre cada uno de esos grupos.

Esos grupos se crean indicando una o más *columnas de agrupamiento*, así los grupos de filas están formados por todas las filas que tienen el mismo valor en las columnas de agrupamiento.

En el resultado, habrá una fila por cada uno de estos grupos.

```
SELECT ...  
  FROM ...  
  [WHERE ...]  
  [GROUP BY <columna1>[,<columna2>, ...]
```

Algunos SGBDs (como Oracle) permiten expresiones en el group by.

Group by

```
SELECT count(*), sum(sal)
FROM emp
GROUP BY job
```

empno numeric(4,0)	ename character(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

count bigint	sum numeric
2	6000.00
4	4150.00
3	8275.00
1	5000.00
4	5600.00

Por cada grupo de filas con el mismo *job*, se genera **UNA FILA** en el resultado

Group by

```
SELECT job, count(*), sum(sal)
FROM emp
GROUP BY job
```

Es posible poner la columna/s de agrupamiento en la cláusula select

Pero cualquiera otra columna **NO puede aparecer** en la cláusula select

empno numeric(4,0)	ename character varying(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

job character varying(9)	count bigint	sum numeric
ANALYST	2	6000.00
CLERK	4	4150.00
MANAGER	3	8275.00
PRESIDENT	1	5000.00
SALESMAN	4	5600.00

Todas las filas del grupo tienen el mismo *job*

Por tanto hay UN único *job* grupo, que puede aparecer en el resultado

Group by

```
SELECT job, deptno, count(*), sum(sal)
FROM emp
GROUP BY job
```

Es posible poner la columna/s de agrupamiento en la cláusula select

Pero cualquiera otra columna **NO puede aparecer** en la cláusula select

empno numeric(4)	ename character varying(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

No hay un único deptno para este grupo

job character varying(9)	count bigint	sum numeric
ANALYST	2	6000.00
CLERK	4	4150.00
MANAGER	3	8275.00
PRESIDENT	1	5000.00
SALESMAN	4	5600.00

En una fila sólo puede haber para cada atributo un único valor. Pero tenemos varios!!!!

```
SQL>
1  SELECT job, deptno, count(*), sum(sal)
2    FROM emp
3    GROUP BY job
4*
SELECT job, deptno, count(*), sum(sal)
      *
ERROR en línea 1:
ORA-00979: no es una expresión GROUP BY

SQL>
```

Group by

```
SELECT job, deptno, count(*), sum(sal)
FROM emp
GROUP BY job, deptno
```

Ahora los grupos son formados por filas con igual valor en job y deptno

Por lo tanto al incluir deptno en el group by cambian los grupos

empno numeric(4,0)	ename character varying(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

job character varying(9)	deptno numeric(2,0)	count bigint	sum numeric
ANALYST	20	2	6000.00
CLERK	10	1	1300.00
CLERK	20	2	1900.00
CLERK	30	1	950.00
MANAGER	10	1	2450.00
MANAGER	20	1	2975.00
MANAGER	30	1	2850.00
PRESIDENT	10	1	5000.00
SALESMAN	30	4	5600.00

Group by

```
SELECT comm, count(*)  
FROM emp  
GROUP BY comm
```

Los nulos son “iguales” para el group by

empno numeric(4,0)	ename character(10)	sal numeric(7,2)	deptno numeric(2,0)	comm numeric(7,2)
7844	TURNER	1500.00	30	0.00
7499	ALLEN	1600.00	30	300.00
7521	WARD	1250.00	30	500.00
7654	MARTIN	1250.00	30	1400.00
7698	BLAKE	2850.00	30	
7782	CLARK	2450.00	10	
7788	SCOTT	3000.00	20	
7876	ADAMS	1100.00	20	
7900	JAMES	950.00	30	
7839	KING	5000.00	10	
7934	MILLER	1300.00	10	
7566	JONES	2975.00	20	
7902	FORD	3000.00	20	
7369	SMITH	800.00	20	

comm numeric(7,2)	count bigint
0.00	1
300.00	1
500.00	1
1400.00	1
	10

Ejercicios

1. Cuántos empleados hay en cada departamento, cuántos tienen comisión, cuántos no tienen comisión y cuales son los ingresos medios (incluyendo salario y comisión).
2. Muestra los departamentos que tienen empleados con comisión. No puede haber valores repetidos.
3. Para cada departamento muestra la comisión media, si no tiene empleados con comisión, se debe indicar con un 0.
4. Para cada departamento muestra cuántos puestos de trabajo distintos desempeñan sus trabajadores.
5. Para cada departamento muestra cuántos empleados hay de cada puesto de trabajo.
6. Muestra cuántos empleados tienen unos ingresos superiores a 2500 € en cada departamento.

Ejercicios

1.

```
Select deptno, count(*), count(comm), count(*)-count(comm), avg(coalesce(sal+comm, sal))  
from emp  
group by deptno
```
2.

```
Select distinct deptno  
from emp  
where comm is not null
```
3.

```
Select deptno, coalesce(avg(comm), 0)  
from emp  
group by deptno
```
4.

```
Select deptno, count(distinct job)  
from emp  
group by deptno
```
5.

```
Select deptno, job, count(*)  
from emp  
group by deptno, job
```
6.

```
select deptno, count(*)  
from emp  
where coalesce(sal+comm, sal)>2500  
group by deptno
```

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Having

De igual forma que la cláusula *where* permite filtrar *filas*, la cláusula *having* permite filtrar grupos.

La cláusula *having* permite establecer una condición (predicado) que se evalúa sobre cada grupo de filas, y aquellos grupos que hacen cierta la condición, pasan a la cláusula *select*.

```
SELECT job, count(*), sum(sal)
FROM emp
GROUP BY job
HAVING min(sal) > 1500
```

empno numeric(4,0)	ename character varying(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

→ Cierto

→ Falso

→ Cierto

→ Cierto

→ Falso

job character varying(9)	count bigint	sum numeric
ANALYST	2	6000.00
MANAGER	3	8275.00
PRESIDENT	1	5000.00

Having

En el predicado de un having se puede utilizar todas herramientas usadas en los predicados where: todos los predicados, subconsultas, etc.

Pero hay que tener cuidado con las expresiones que se incluyan: éstas sólo pueden contener constantes, funciones (incluyendo las colectivas) aplicadas sobre cualquier columna, pero, fuera de funciones colectivas, sólo pueden aparecer las **columnas de agrupamiento**.

```
SELECT job, count(*), sum(sal)
FROM emp
GROUP BY job
HAVING sal > 1500
```

empno numeric(4)	ename character varying(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

→ ¿sal?>1500

El predicado se aplica UNO A UNO a cada **UNO de los grupos de filas**.

Por tanto cuando se aplique sobre este grupo, ¿por qué valor sustituimos el nombre de columna **SAL**

Having

Pero hay que tener cuidado con las expresiones que se incluyan: éstas sólo pueden contener contantes, funciones (incluyendo las colectivas) aplicadas sobre cualquier columna, pero, fuera de funciones colectivas sólo pueden aparecer las *columnas de agrupamiento*.

```
SELECT job, count(*), sum(sal)
FROM emp
GROUP BY job
HAVING min(sal) > 1500
```

empno numeric(4,0)	ename character varying(10)	sal numeric(7,2)	job character varying(9)	deptno numeric(2,0)
7788	SCOTT	3000.00	ANALYST	20
7902	FORD	3000.00	ANALYST	20
7934	MILLER	1300.00	CLERK	10
7369	SMITH	800.00	CLERK	20
7876	ADAMS	1100.00	CLERK	20
7900	JAMES	950.00	CLERK	30
7782	CLARK	2450.00	MANAGER	10
7566	JONES	2975.00	MANAGER	20
7698	BLAKE	2850.00	MANAGER	30
7839	KING	5000.00	PRESIDENT	10
7844	TURNER	1500.00	SALESMAN	30
7654	MARTIN	1250.00	SALESMAN	30
7521	WARD	1250.00	SALESMAN	30
7499	ALLEN	1600.00	SALESMAN	30

→ ¿3000>1500?

→ ¿800>1500?

→ ¿2450>1500?

→ ¿5000>1500?

→ ¿1250>1500?

El predicado se aplica UNO A UNO a cada uno de los grupos de filas.

Having

Orden de ejecución en la sentencia select:

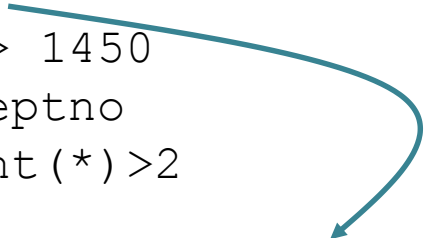
1. FROM
2. WHERE
3. GROUP By
4. HAVING
5. SELECT
6. ORDER BY

Having

La condición *where* se aplica a *filas*

La condición *having* se aplica a *grupos de filas*

```
SELECT deptno, count(*), sum(sal)
FROM emp
WHERE sal > 1450
GROUP BY deptno
HAVING count(*) > 2
```



Primer paso

empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7839	KING	PRESIDENT		1981-11-17	5000.00		10

Having

La condición *where* se aplica a *filas*

La condición *having* se aplica a *grupos de filas*

```
SELECT deptno, count(*), sum(sal)
FROM emp
WHERE sal > 1450
GROUP BY deptno
HAVING count(*) > 2
```

Segundo paso

empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7839	KING	PRESIDENT		1981-11-17	5000.00		10

Falso

Cierto

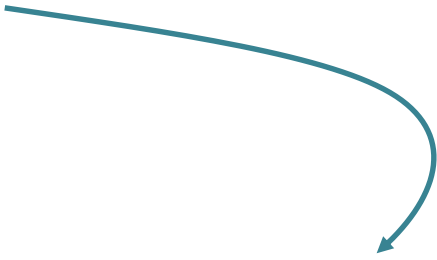
empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30

Having

La condición *where* se aplica a *filas*

La condición *having* se aplica a *grupos de filas*

```
SELECT deptno, count(*), sum(sal)
FROM emp
WHERE sal > 1450
GROUP BY deptno
HAVING count(*) > 2
```



Tercer paso

empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30

Having

La condición *where* se aplica a *filas*

La condición *having* se aplica a *grupos de filas*

```
SELECT deptno, count(*), sum(sal)
FROM emp
WHERE sal > 1450
GROUP BY deptno
HAVING count(*) > 2
```

Cuarto paso

empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30

Falso

Cierto

Cierto

empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30

Having

La condición *where* se aplica a *filas*

La condición *having* se aplica a *grupos de filas*

```
SELECT deptno, count(*), sum(sal)
FROM emp
WHERE sal > 1450
GROUP BY deptno
HAVING count(*) > 2
```

Quinto paso

empno numeric(4)	ename character	job character var	mgr numeric	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30

deptno numeric(2,0)	count bigint	sum numeric
20	3	8975.00
30	3	5950.00

Ejercicios

1. Para cada departamento muestra cuántos empleados tienen unos ingresos (sal+comm) superiores a 2500 €.
2. Muestra los departamentos con unos ingresos medios superiores a los 2500 €. Muestra para cada uno, cuántos empleados tienen.
3. Departamentos con al menos dos 'MANAGER'
4. Departamentos con al menos dos empleados con comisión. Para cada departamento muestra cuántos empleados tiene (en total) y cuántos con comisión.
5. Departamentos con al menos dos empleados con el mismo puesto de trabajo. No puede aparecer repetidos.

Ejercicios

```
1. Select deptno, count(*)  
   from emp  
   where coalesce(sal+comm,sal)>2500  
   group by deptno
```

```
2. Select deptno, count(*)  
   from emp  
   group by deptno  
   having avg(coalesce(sal+comm,sal))>2500
```

```
3. Select deptno  
   from emp  
   where job = 'MANAGER'  
   group by deptno  
   having count(*)>=2
```

```
4. Select deptno, count(*), count(comm)  
   from emp  
   group by deptno  
   having count(comm)>=2
```

```
5. Select distinct deptno  
   from emp  
   group by deptno, job  
   having count(*)>=2
```

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Más de una tabla en el FROM

Cómo puedo obtener una tabla que para cada *empleado* me muestre su *nombre* y el *nombre del departamento* para el que trabaja.

En la cláusula `select` los nombres de columna que aparezcan deben estar en alguna tabla en el `from`.

Pero en la tabla `emp` sólo tenemos el número de departamento.

```
SQL> select * from emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30
7566	JONES	MANAGER	7839	02/04/81	2,975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20
7839	KING	PRESIDENT		17/11/81	5,000		10
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1,100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3,000		20
7934	MILLER	CLERK	7782	23/01/82	1,300		10

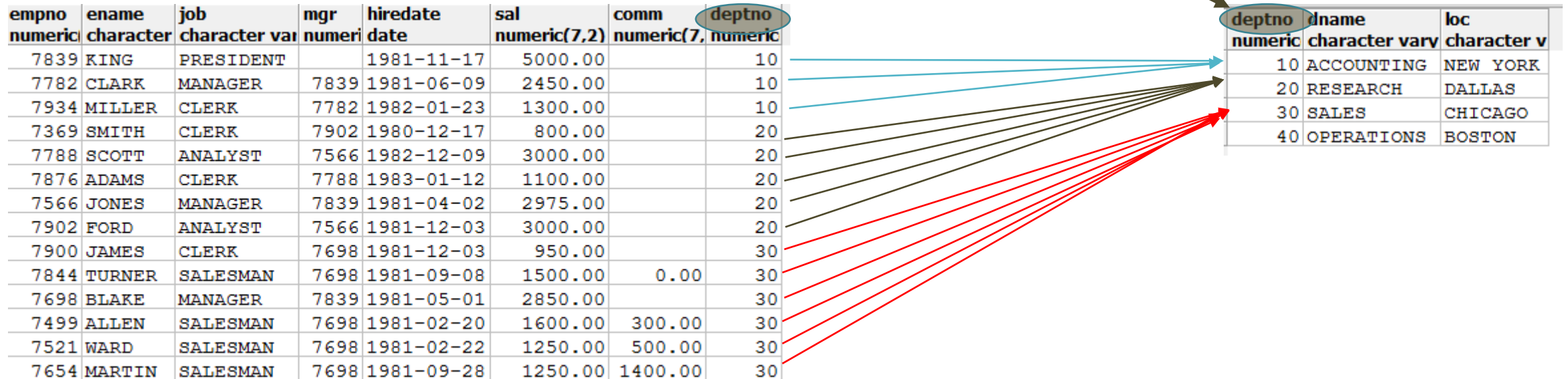
```
SQL> select * from dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Necesitamos acceder a la tabla `emp` y a la tabla `dept`

Más de una tabla en el FROM

Cuando partimos la información en tablas, dejamos claves foráneas que mantienen los vínculos entre la información partida.



Mediante esas claves foráneas, podemos enlazar la información partida.

El mecanismo que se usa en SQL se llama JOIN (*reunir* en español).

JOIN

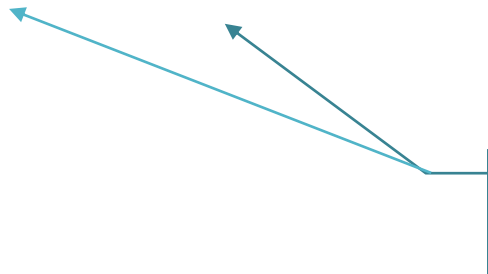
```
SELECT ...  
FROM <tabla1> [INNER|LEFT|RIGHT|FULL] JOIN <tabla2>  
ON <condición de join>
```

El más normal es este: que obtiene, del producto cartesiano, las filas que hacen cierta la condición de join

```
SELECT ...  
FROM <tabla1> [INNER] JOIN <tabla2>  
ON <condición de join>
```

Y lo más normal es que la condición de join sea una igualdad entre una clave foránea y una clave primaria. Por ej.

```
SELECT *  
FROM emp JOIN dept  
ON emp.deptno=dept.deptno
```



Hay 2 columnas deptno, así que
hay que desambiguar

JOIN

Conceptualmente (no lo hace en realidad), primero crea el producto cartesiano

Todas columnas
de las dos tablas

Todas las filas de
emp pegadas a
todas las de
dept

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)	deptno numeric(2,0)	dname character varying(14)	loc character varying(13)
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-09-08	2975.00		20	10	ACCOUNTING	NEW YORK
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	10	ACCOUNTING	NEW YORK
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	10	ACCOUNTING	NEW YORK
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	10	ACCOUNTING	NEW YORK
7499	ALLEN	SALESMAN	7698	1981-09-19	1600.00	300.00	30	10	ACCOUNTING	NEW YORK
7521	WARD	SALESMAN	7698	1981-02-12	1250.00	500.00	30	10	ACCOUNTING	NEW YORK
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	10	ACCOUNTING	NEW YORK
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	10	ACCOUNTING	NEW YORK
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	10	ACCOUNTING	NEW YORK
7876	ADAMS	CLERK	7788	1983-01-23	1100.00		20	10	ACCOUNTING	NEW YORK
7900	JAMES	CLERK	7698	1981-12-03	950.00		30	10	ACCOUNTING	NEW YORK
7934	MILLER	CLERK	7782	1982-07-06	1300.00		10	10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT		1981-11-17	5000.00		10	20	RESEARCH	DALLAS
7566	JONES	MANAGER	7839	1981-09-08	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	1981-09-19	1600.00	300.00	30	20	RESEARCH	DALLAS
7521	WARD	SALESMAN	7698	1981-02-12	1250.00	500.00	30	20	RESEARCH	DALLAS
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	20	RESEARCH	DALLAS
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	20	RESEARCH	DALLAS
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	20	RESEARCH	DALLAS

... continúa con más filas...

JOIN

```
SELECT *  
FROM emp JOIN dept  
ON emp.deptno=dept.deptno
```

EMP							DEPT			
empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)	deptno numeric(2,0)	dname character varying(14)	loc character varying(13)
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-09-01	2975.00		20	10	ACCOUNTING	NEW YORK
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	10	ACCOUNTING	NEW YORK
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	10	ACCOUNTING	NEW YORK
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	10	ACCOUNTING	NEW YORK
7499	ALLEN	SALESMAN	7698	1981-09-08	1600.00	300.00	30	10	ACCOUNTING	NEW YORK
7521	WARD	SALESMAN	7698	1981-09-08	1250.00	500.00	30	10	ACCOUNTING	NEW YORK
7654	MARTIN	SALESMAN	7698	1981-09-08	1250.00	1400.00	30	10	ACCOUNTING	NEW YORK
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	10	ACCOUNTING	NEW YORK

Sobre el producto cartesiano se seleccionan las filas que hacen cierta la condición de join

Este tipo de filas son las que hacen cierta la condición de join

Estas filas no salen porque no hacen cierta la condición de join

JOIN

```
SELECT *  
FROM emp JOIN dept  
ON emp.deptno=dept.deptno
```

EMP							DEPT			
empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	dname	loc
numeric	character	character var	numeric(4,0)	date	numeric(7,2)	numeric(7,2)	numeric(2)	numeric(2)	character var	character var
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	30	SALES	CHICAGO
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20	20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	1981-12-03	950.00		30	30	SALES	CHICAGO
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	10	ACCOUNTING	NEW YORK

Se puede desambiguar usando alias.

```
SELECT ename, E.deptno, dname  
FROM emp E JOIN dept D  
ON E.deptno=D.deptno
```

Sólo es necesario desambiguar en los nombres de columna que aparecen en las dos tablas.

JOIN

Recordar los pasos

1. FROM(obligatoria)

Partiendo de una o más tablas *obtiene una única tabla* que será procesada por el resto de cláusulas

2. WHERE (optativa)

3. GROUP BY (optativa)

4. HAVING (optativa)

3. SELECT (obligatoria)


4. ORDER BY (optativa)

Así, que todo lo visto hasta ahora funciona igual, porque el FROM es lo primero que se ejecuta, y devuelve una única tabla.

JOIN

```
SELECT ename, E.deptno, dname  
FROM emp E JOIN dept D  
ON E.deptno=D.deptno  
WHERE coalesce(comm,0)+sal>2500
```

Primer paso



empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	dname	loc
numeric	character	character var	numeric(4,0)	date	numeric(7,2)	numeric(7,2)	numeric(2,0)	numeric(2,0)	character var	character var
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	30	SALES	CHICAGO
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20	20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	1981-12-03	950.00		30	30	SALES	CHICAGO
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	10	ACCOUNTING	NEW YORK

JOIN

```
SELECT ename, E.deptno, dname
FROM emp E JOIN dept D
ON E.deptno=D.deptno
WHERE coalesce(comm,0)+sal>2500
```

empno numeric	ename character	job character var	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,	deptno numeric(deptno numeric(dname character vary	loc character v:
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	30	SALES	CHICAGO
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20	20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	1981-12-03	950.00		30	30	SALES	CHICAGO
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	10	ACCOUNTING	NEW YORK

Segundo paso

empno numeric	ename character	job character var	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,	deptno numeric(deptno numeric(dname character vary	loc character v:
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS

Tercer paso

ename character varying(10)	deptno numeric(2,0)	dname character varying(14)
KING	10	ACCOUNTING
JONES	20	RESEARCH
FORD	20	RESEARCH
BLAKE	30	SALES
MARTIN	30	SALES
SCOTT	20	RESEARCH

JOIN

Se puede usar más de una copia de una misma tabla.

```
SELECT s.ename subordinado, s.mgr, j.empno, j.ename jefe  
FROM emp s JOIN emp j ON s.mgr=j.empno
```

subordinado character varying(10)	mgr numeric(4,0)	empno numeric(4,0)	jefe character varying(10)
JONES	7839	7839	KING
FORD	7566	7566	JONES
SMITH	7902	7902	FORD
BLAKE	7839	7839	KING
ALLEN	7698	7698	BLAKE
WARD	7698	7698	BLAKE
MARTIN	7698	7698	BLAKE
CLARK	7839	7839	KING
SCOTT	7566	7566	JONES
TURNER	7698	7698	BLAKE
ADAMS	7788	7788	SCOTT
JAMES	7698	7698	BLAKE
MILLER	7782	7782	CLARK

JOIN

La condición de join puede ser cualquier predicado.

```
SELECT a.ename, a.sal, b.ename, b.sal
FROM emp a JOIN emp b ON a.sal>b.sal
```

ename character varying(10)	sal numeric(7,2)	ename character varying(10)	sal numeric(7,2)
KING	5000.00	JONES	2975.00
KING	5000.00	FORD	3000.00
KING	5000.00	SMITH	800.00
KING	5000.00	BLAKE	2850.00
KING	5000.00	ALLEN	1600.00
KING	5000.00	WARD	1250.00
KING	5000.00	MARTIN	1250.00
KING	5000.00	CLARK	2450.00
KING	5000.00	SCOTT	3000.00
KING	5000.00	TURNER	1500.00
KING	5000.00	ADAMS	1100.00
KING	5000.00	JAMES	950.00
KING	5000.00	MILLER	1300.00
JONES	2975.00	SMITH	800.00
JONES	2975.00	BLAKE	2850.00
JONES	2975.00	ALLEN	1600.00
JONES	2975.00	WARD	1250.00
JONES	2975.00	MARTIN	1250.00
JONES	2975.00	CLARK	2450.00
JONES	2975.00	TURNER	1500.00
JONES	2975.00	ADAMS	1100.00
JONES	2975.00	JAMES	950.00
JONES	2975.00	MILLER	1300.00
FORD	3000.00	JONES	2975.00
FORD	3000.00	SMITH	800.00
FORD	3000.00	BLAKE	2850.00

...

```
SELECT s.ename sub, s.sal, j.ename jefe, j.sal
FROM emp s JOIN emp j
ON s.mgr=j.empno AND s.sal>j.sal
```

sub character varying(10)	sal numeric(7,2)	jefe character varying(10)	sal numeric(7,2)
FORD	3000.00	JONES	2975.00
SCOTT	3000.00	JONES	2975.00

JOIN

Recalcamos que en el resultado, están las filas (del producto cartesiano) que cumplen la condición de join.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1,600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1,250	500	30
7566	JONES	MANAGER	7839	02/04/81	2,975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1,250	1,400	30
7698	BLAKE	MANAGER	7839	01/05/81	2,850		30
7782	CLARK	MANAGER	7839	09/06/81	2,450		10
7788	SCOTT	ANALYST	7566	19/04/87	3,000		20
7839	KING	PRESIDENT		17/11/81	5,000		10
7844	TURNER	SALESMAN	7698	08/09/81	1,500	0	30
7876	ADAMS	CLERK	7788	23/05/87	1,100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81	3,000		20
7934	MILLER	CLERK	7782	23/01/82	1,300		10

```
SELECT *
FROM emp JOIN dept
ON emp.deptno=dept.deptno
```

empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	dname	loc
numeric	character	character var	numeric(4,0)	date	numeric(7,2)	numeric(7,2)	numeric(2,0)	numeric(2,0)	character var	character var
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	30	SALES	CHICAGO
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20	20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	1981-12-03	950.00		30	30	SALES	CHICAGO
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	10	ACCOUNTING	NEW YORK

En el resultado no hay rastro del departamento 40, porque no hay ningún empleado que trabaje para ese departamento y por tanto nunca se hace cierta la condición de join

JOIN exterior

```
SELECT ...  
FROM <tabla1> [LEFT|RIGHT|FULL] JOIN <tabla2>  
ON <condición de join>
```

Podemos forzar a que las filas de una (o de las dos) tabla de entrada que en el INNER join no aparecen por no hacer cierta la condición de join, salgan rellenando las columnas del otro lado con nulos.

```
SELECT *
```

```
FROM emp RIGHT JOIN dept  
ON emp.deptno=dept.deptno
```

Vamos forzar que salgan las filas del lado derecho que no salen en el INNER

empno numeric(4,0)	ename character	job character vary	mgr numeri	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric	deptno numeric	dname character vary	loc character vary
7839	KING	PRESIDENT		1981-11-17	5000.00		10	10	ACCOUNTING	NEW YORK
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20	20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20	20	RESEARCH	DALLAS
7369	SMITH	CLERK	7902	1980-12-17	800.00		20	20	RESEARCH	DALLAS
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30	30	SALES	CHICAGO
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10	10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20	20	RESEARCH	DALLAS
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20	20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	1981-12-03	950.00		30	30	SALES	CHICAGO
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	10	ACCOUNTING	NEW YORK
								40	OPERATIONS	BOSTON

Las filas del lado derecho que “emparejan”, salen igual que en el INNER join

Las filas del lado derecho que “no emparejan”, salen rellenado las columnas del lado izquierdo con nulos

Nulos

JOIN exterior

```
SELECT s.ename subordinado, s.mgr, j.empno, j.ename jefe
FROM emp s LEFT JOIN emp j ON s.mgr=j.empno
```

subordinado character varying(10)	mgr numeric(4,0)	empno numeric(4,0)	jefe character varying(10)
KING			
JONES	7839	7839	KING
FORD	7566	7566	JONES
SMITH	7902	7902	FORD
BLAKE	7839	7839	KING
ALLEN	7698	7698	BLAKE
WARD	7698	7698	BLAKE
MARTIN	7698	7698	BLAKE
CLARK	7839	7839	KING
SCOTT	7566	7566	JONES
TURNER	7698	7698	BLAKE
ADAMS	7788	7788	SCOTT
JAMES	7698	7698	BLAKE
MILLER	7782	7782	CLARK

} Forzada por el left join

} Encaje "normal"

JOIN exterior

```
SELECT s.ename subordinado, s.mgr, j.empno, j.ename jefe
FROM emp s FULL JOIN emp j ON s.mgr=j.empno
```

subordinado character varying(10)	mgr numeric(4,0)	empno numeric(4,0)	jefe character varying(10)
KING			
JONES	7839	7839	KING
FORD	7566	7566	JONES
SMITH	7902	7902	FORD
BLAKE	7839	7839	KING
ALLEN	7698	7698	BLAKE
WARD	7698	7698	BLAKE
MARTIN	7698	7698	BLAKE
CLARK	7839	7839	KING
SCOTT	7566	7566	JONES
TURNER	7698	7698	BLAKE
ADAMS	7788	7788	SCOTT
JAMES	7698	7698	BLAKE
MILLER	7782	7782	CLARK
		7844	TURNER
		7369	SMITH
		7876	ADAMS
		7934	MILLER
		7521	WARD
		7900	JAMES
		7499	ALLEN
		7654	MARTIN

} Forzada por el left join

Encaje "normal"

} Forzada por el right join

Join de más de dos tablas

```
SELECT ...  
FROM <tabla1> [INNER|LEFT|RIGHT|FULL] JOIN <tabla2> ON <condición de join12>  
          [INNER|LEFT|RIGHT|FULL] JOIN <tabla3> ON <condición de join123>  
          [INNER|LEFT|RIGHT|FULL] JOIN <tabla4> ON <condición de join1234>  
          ...
```

En primer lugar se unen <tabla1> y <tabla2>, con su condición de join.

Esto da lugar a una tabla.

Esa tabla resultante se une a la <tabla3>, con su condición de join, lo que resulta en otra tabla.

Y así sucesivamente.

Join de más de dos tablas

```
SELECT ...  
FROM <tabla1> JOIN <tabla2> JOIN <tabla3> JOIN <tabla4>  
ON <condición de join12>  
  and <condición de join123>  
  and <condición de join1234>  
  ...
```

Esto **NO ES CORRECTO!!**

Join de más de dos tablas

```
SELECT ename, pname, hours
FROM emp e JOIN emppro ep ON e.empno=ep.empno
      JOIN pro p ON ep.prono=p.prono
```

Primer paso

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7369	SMITH	CLERK	7902	1980-12-17	800.00		20
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00		20
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1983-01-12	1100.00		20
7900	JAMES	CLERK	7698	1981-12-03	950.00		30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7499	1004	15
7499	1005	12
7521	1004	10
7521	1008	8
7654	1001	16
7654	1006	15
7654	1008	5
7844	1005	6
7934	1001	4

emp

emppro

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)	empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1004	15
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1005	12
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1004	10
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1008	8
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1001	16
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1006	15
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1008	5
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	7844	1005	6
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	7934	1001	4

Join de más de dos tablas

```
SELECT ename, pname, hours
FROM emp e JOIN emppro ep ON e.empno=ep.empno
      JOIN pro p ON ep.prono=p.prono
```

Segundo paso

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)	empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1004	15
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1005	12
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1004	10
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1008	8
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1001	16
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1006	15
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1008	5
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	7844	1005	6
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	7934	1001	4

prono numeric	pname character	loc character varying	deptno numeric
1001	P1	BOSTON	20
1004	P4	CHICAGO	30
1005	P5	CHICAGO	30
1006	P6	LOS ANGELES	30
1008	P8	NEW YORK	30

emp

emppro

pro

empno numeric(4,0)	ename character varying(10)	job character varying(9)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)	empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)	prono numeric(4,0)	pname character varying(10)	loc character varying(13)	deptno numeric(2,0)
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1004	15	1004	P4	CHICAGO	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1005	12	1005	P5	CHICAGO	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1004	10	1004	P4	CHICAGO	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1008	8	1008	P8	NEW YORK	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1001	16	1001	P1	BOSTON	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1006	15	1006	P6	LOS ANGELES	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1008	5	1008	P8	NEW YORK	30
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	7844	1005	6	1005	P5	CHICAGO	30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	7934	1001	4	1001	P1	BOSTON	20

Join de más de dos tablas

```
SELECT ename, pname, hours
FROM emp e JOIN emppro ep ON e.empno=ep.empno
      JOIN pro p ON ep.prono=p.prono
```

empno numeric(4,0)	ename character varying(10)	job character varying(10)	mgr numeric(4,0)	hiredate date	sal numeric(7,2)	comm numeric(7,2)	deptno numeric(2,0)	empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)	prono numeric(4,0)	pname character varying(10)	loc character varying(13)	deptno numeric(2,0)
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1004	15	1004	P4	CHICAGO	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30	7499	1005	12	1005	P5	CHICAGO	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1004	10	1004	P4	CHICAGO	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30	7521	1008	8	1008	P8	NEW YORK	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1001	16	1001	P1	BOSTON	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1006	15	1006	P6	LOS ANGELES	30
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30	7654	1008	5	1008	P8	NEW YORK	30
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30	7844	1005	6	1005	P5	CHICAGO	30
7934	MILLER	CLERK	7782	1982-01-23	1300.00		10	7934	1001	4	1001	P1	BOSTON	20

ename character varying(10)	pname character varying(10)	hours numeric(2,0)
ALLEN	P4	15
ALLEN	P5	12
WARD	P4	10
WARD	P8	8
MARTIN	P1	16
MARTIN	P6	15
MARTIN	P8	5
TURNER	P5	6
MILLER	P1	4

Ejercicios

1. Para cada proyecto muestra su nombre y el nombre del departamento que los controla.
2. Para cada empleado muestra su nombre y los códigos de proyectos para los que trabaja.
3. Para cada empleado muestra su nombre y los códigos de proyectos para los que trabaja. Si hay empleados que no trabajan en proyectos, éstos deben aparecer con el código de proyecto a nulo.
4. Para cada empleado muestra el nombre de su jefe, si no tiene jefe, muestra un nulo en el nombre del jefe.
5. Para cada empleado muestra su nombre, el nombre de su jefe, y el departamento para el que trabaja su jefe.
6. Devuelve los empleados que tienen un salario más alto que su jefe.

Ejercicios

1. `Select pname, dname
from pro p join dept d on p.deptno=d.deptno`
2. `Select ename, proname
from emp e join emppro ep on e.empno=ep.empno`
3. `Select ename, proname
from emp e left join emppro ep on e.empno=ep.empno`
4. `Select e.ename, j.ename
from emp e left join emp j on e.mgr=j.empno`
5. `Select e.ename, j.ename, d.dname
from emp e join emp j on e.mgr=j.empno join dept d
on j.deptno=d.deptno`
6. `Select e.ename, e.sal, j.ename, j.sal
from emp e join emp j on e.mgr=j.empno
where e.sal>j.sal`

Ejercicios

1. Para empleado muestra su nombre y cuántas horas trabajó en proyectos.
2. Para cada departamento, muestra su nombre y cuántos empleados tiene.
3. Para cada jefe, muestra su nombre y cuántos subordinados tiene.
4. Muestra el nombre de proyectos donde se ha trabajado (en total, todos los empleados) más de 15 horas
5. Muestra los departamentos (nombre) que controlan más de dos proyectos.
6. Muestra los departamentos (nombre) donde hay por lo menos dos empleados con el mismo puesto de trabajo. No debe aparecer repetidos.
7. Para cada departamento mostrar su nombre y cuántos empleados tiene, si no tiene ninguno, indicarlo con un 0.
8. Para cada empleado mostrar las horas que trabajó en proyectos, si no trabajó en ninguno, indicarlo con un 0.
9. Para cada jefe, cuántos subordinados ganan más que él, si no gana ninguno indicarlo con un cero.

Ejercicios

1.

```
Select ename, sum(hours)
from emp e join emppro ep on e.empno=ep.empno
group by e.empno, ename
```
2.

```
Select dname, count(ename)
from emp e join dept d on e.deptno=d.deptno
group by d.deptno, dname
```
3.

```
Select j.ename, count(e.empno)
from emp e join emp j on e.mgr=j.empno
group by j.empno, j.ename
```
4.

```
Select pname, sum(hours)
from emppro ep join pro p on ep.prono=p.prono
group by p.prono, pname
having sum(hours)>15
```
5.

```
Select dname, count(prono)
from dept d join pro p on d.deptno=p.deptno
group by d.deptno, dname
having count(*)>2
```
6.

```
Select distinct dname
from emp e join dept d on e.deptno=d.deptno
group by d.deptno, dname, job
having count(*)>=2
```
7.

```
Select dname, count(empno) /*no count(*)*/
from emp e right join dept d on e.deptno=d.deptno
group by d.deptno, dname
```
8.

```
Select ename, coalesce(sum(hours),0)
from emp e left join emppro ep on e.empno=ep.empno
group by e.empno, ename
```
9.

```
Select j.ename, count(e.ename)
from emp e right join emp j on e.mgr=j.empno
and e.sal>j.sal
group by j.empno, j.ename
```

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Subconsultas

Hasta ahora, todos los nombres de columnas que aparecen en todas las expresiones deben ser de las tablas que aparecen en el `FROM` de la sentencia `select`.

Si quisiésemos saber los empleados que trabajan en 'DALLAS' (sin join) tendríamos un problema, porque necesitamos la tabla EMP para obtener los datos de los empleados.

```
SELECT *  
FROM emp  
WHERE deptno=20
```

Pero necesitamos saber que el departamento localizado en DALLAS es el que tiene número 20.

Para esto necesitamos ejecutar una consulta previa

```
SELECT deptno  
FROM dept  
WHERE LOC='DALLAS'
```

Necesitamos un mecanismo que me permita consultar el número de departamento que está en DALLAS en la misma consulta, sin que el usuario tenga que hacer una consulta previa.

Subconsultas

SQL nos permite incluir consultas dentro de otras consultas.

Así en lugar de escribir el 20, podemos incluir una consulta que obtenga dicho valor.

```
SELECT *  
FROM emp  
WHERE deptno=(SELECT deptno  
               FROM dept  
               WHERE LOC= 'DALLAS' )
```

Esto se denominan *subconsultas*.

Una consulta puede contener, a su vez, subconsultas.

La consulta inicial se conoce como consulta principal.

El resultado de la consulta lo utiliza la consulta de nivel superior en un predicado, pero su resultado no se puede trasladar al resultado de la consulta.

Es decir, en nuestro ejemplo, no podemos poner ninguna expresión en la cláusula SELECT principal conteniendo nombres de columnas de la tabla DEPT.


Subconsultas

La subconsulta se ejecuta (conceptualmente) antes de comenzar la ejecución de la principal, y su resultado se usa para evaluar el predicado donde está.

El primer caso que hemos visto es una **subconsulta escalar**, que es un tipo especial de expresión, ya que devuelve un único valor escalar.

```
SELECT *  
FROM emp  
WHERE deptno=
```

```
(SELECT deptno  
FROM dept  
WHERE LOC= 'DALLAS' )
```



deptno
numeric(2,0)
20

Dicho de otro modo, la subconsulta devuelve una tabla con una fila y una columna. Por lo tanto es equivalente a escribir una expresión literal (una constante).

Este es el único tipo de subconsulta válida con predicados elementales.

```
SELECT *  
FROM emp  
WHERE deptno = (SELECT deptno  
FROM dept  
WHERE loc IN ('DALLAS', 'NEW YORK'))
```

ERROR en línea 3:
ORA-01427: la subconsulta
de una sola fila devuelve
más de una fila

Subconsultas

La subconsulta puede usar todas las herramientas usadas hasta el momento, por ej:

```
SELECT *  
FROM emp  
WHERE sal > (SELECT AVG(sal)  
             FROM emp  
             WHERE deptno=10)
```

Si usamos un predicado IN, la subconsulta puede devolver más de una fila. En tal caso, la subconsulta, ya no se considera una expresión.

La subconsulta debe devolver una tabla de sólo una columna.

```
SELECT *  
FROM emp  
WHERE deptno IN (SELECT deptno  
                FROM dept  
                WHERE loc IN ('DALLAS', 'NEW YORK'))
```

Subconsultas

Es posible que la subconsulta devuelva más de una fila usando predicados elementales si los operadores están *cuantificados*.

Los operadores **ANY** o **SOME** y **ALL** modifican el operador para que se pueda comparar un valor escalar con una lista de valores (una tabla de una sola columna)

```
... <expre> operador_comparacion { ALL | SOME | ANY } (subconsulta)
```

Operadores de comparacion: < <= = != <> >= >

ANY, SOME: La expresión se compara con cada uno de los valores de la subconsulta y si para alguno es verdadera, el resultado es verdadero.

ALL: La expresión se compara con cada uno de los valores de la subconsulta y si para todos es verdadera, el resultado es verdadero.

```
SELECT *  
FROM emp  
WHERE sal > ANY(SELECT sal  
                FROM emp  
                WHERE deptno=10)
```

```
SELECT *  
FROM emp  
WHERE sal > ALL(SELECT sal  
                FROM emp  
                WHERE deptno=10)
```

Ejercicios

1. Empleados que tienen un salario mayor al salario medio de la empresa
2. Para cada departamento mostrar cuántos empleados tiene que ganen más del salario medio de la empresa. Muestra el nombre del departamento.
3. Empleados que son jefe. Muestra su nombre.
4. Empleados que no son jefe. Muestra su nombre.
5. Muestra el empleado/s (nombre) con el salario más alto.
6. Muestra el departamento (nombre) con la suma de salarios más alta.
7. Para los departamentos que tienen empleados con comisión, muestra cuántos empleados tienen comisión, y cuántos no. Muestra nombre del departamento.

Ejercicios

1.

```
Select empno, ename, sal
from emp
where sal > (Select avg(sal) from emp)
```
2.

```
Select dname, count(*)
from emp e join dept d on e.deptno=d.deptno
where sal > (Select avg(sal) from emp)
group by d.deptno, dname
```
3.

```
Select ename
from emp
where empno in (Select mgr from emp)
```
4.

```
Select ename
from emp
where empno not in (Select mgr from emp
                    where mgr is not null)
```
5.

```
Select ename
from emp
where sal= (Select max(sal) from emp)
```
6.

```
Select dname, sum(sal)
from emp e join dept d on e.deptno=d.deptno
group by d.deptno, dname
having sum(sal) >= ALL (select sum(sal)
                       from emp
                       group by deptno)
```
7.

```
Select dname, count(*)-count(comm) "Sin comisión", count(comm) "Con comisión"
from emp e join dept d on e.deptno=d.deptno
where d.deptno in (Select deptno
                  from emp
                  where comm is not null)
group by d.deptno, dname
```

Subconsultas correlacionadas

Las subconsultas vistas hasta ahora, se ejecutaban una única vez antes de ejecutar la principal, y su resultado lo utilizaba la consulta principal como un literal o lista de literales.

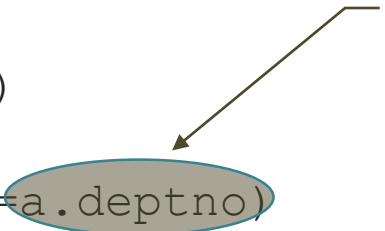
Es decir, la subconsulta era totalmente independiente de la principal.

Las subconsultas correlacionadas varían con respecto a las normales en:

Tienen al menos una referencia a una de las columnas de las tablas en el FROM de la consulta principal.

La subconsulta se ejecuta una vez por cada fila de la principal

```
SELECT *  
FROM emp a  
WHERE sal = (SELECT MAX(sal)  
             FROM emp  
             WHERE deptno=a.deptno)
```



Referencia a
columna de la
consulta principal

Subconsultas correlacionadas

```
SELECT *
```

```
FROM emppro a
```

```
WHERE hours = (SELECT MAX(hours)  
               FROM emppro  
               WHERE prono=a.prono)
```

Para cada proyecto, ¿cuál es el empleado que más horas trabaja

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7934	1001	4
7654	1001	16
7499	1004	15
7521	1004	10
7844	1005	6
7499	1005	12
7654	1006	15
7654	1008	5
7521	1008	8

```
SELECT MAX(hours)  
FROM emppro  
WHERE prono=1001
```

Para cada fila proveniente del FROM de la consulta principal, se ejecuta la subconsulta, sustituyendo las referencias a columnas de la consulta principal por el valor de la fila procesada

Subconsultas correlacionadas

```
SELECT *  
FROM emppro a  
WHERE hours = (SELECT MAX(hours)  
               FROM emppro  
               WHERE prono=a.prono)
```

Comprobación de la
condición where de la
consulta principal

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7934	1001	4
7654	1001	16
7499	1004	15
7521	1004	10
7844	1005	6
7499	1005	12
7654	1006	15
7654	1008	5
7521	1008	8

WHERE 4 = 16

```
SELECT MAX(hours)  
FROM emppro  
WHERE prono=1001
```

Ejecución de la
subconsulta
para la fila en
color rojo

7934 no es el que más horas trabaja en el proyecto 1001

Subconsultas correlacionadas

```
SELECT *  
FROM emppro a  
WHERE hours = (SELECT MAX(hours)  
                FROM emppro  
                WHERE prono=a.prono)
```

Comprobación de la
condición where de la
consulta principal

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7934	1001	4
7654	1001	16
7499	1004	15
7521	1004	10
7844	1005	6
7499	1005	12
7654	1006	15
7654	1008	5
7521	1008	8

WHERE 16 = 16

```
SELECT MAX(hours)  
FROM emppro  
WHERE prono=1001
```

Ejecución de la
subconsulta
para la fila en
color rojo

7654 es el que más horas trabaja en el proyecto 1001

Subconsultas correlacionadas

```
SELECT *  
FROM emppro a  
WHERE hours = (SELECT MAX(hours)  
               FROM emppro  
               WHERE prono=a.prono)
```

Comprobación de la
condición where de la
consulta principal

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7934	1001	4
7654	1001	16
7499	1004	15
7521	1004	10
7844	1005	6
7499	1005	12
7654	1006	15
7654	1008	5
7521	1008	8

WHERE 15 = 15

```
SELECT MAX(hours)  
FROM emppro  
WHERE prono=1004
```

Ejecución de la
subconsulta
para la fila en
color rojo

7499 es el que más horas trabaja en el proyecto 1004

Subconsultas correlacionadas

```
SELECT *  
FROM emppro a  
WHERE hours = (SELECT MAX(hours)  
               FROM emppro  
               WHERE prono=a.prono)
```

Comprobación de la
condición where de la
consulta principal

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7934	1001	4
7654	1001	16
7499	1004	15
7521	1004	10
7844	1005	6
7499	1005	12
7654	1006	15
7654	1008	5
7521	1008	8

WHERE 10 = 15

```
SELECT MAX(hours)  
FROM emppro  
WHERE prono=1004
```

Ejecución de la
subconsulta
para la fila en
color rojo

7521 no es el que más horas trabaja en el proyecto 1004

Subconsultas correlacionadas

```
SELECT *  
FROM emppro a  
WHERE hours = (SELECT MAX(hours)  
                FROM emppro  
                WHERE prono=a.prono)
```

Comprobación de la
condición where de la
consulta principal

empno numeric(4,0)	prono numeric(4,0)	hours numeric(2,0)
7934	1001	4
7654	1001	16
7499	1004	15
7521	1004	10
7844	1005	6
7499	1005	12
7654	1006	15
7654	1008	5
7521	1008	8

WHERE 6 = 12

```
SELECT MAX(hours)  
FROM emppro  
WHERE prono=1005
```

Ejecución de la
subconsulta
para la fila en
color rojo

7844 no es el que más horas trabaja en el proyecto 1005

Y así hasta acabar con todas las filas de la consulta principal

Subconsultas correlacionadas con predicado exists

Predicado de existencia (**EXISTS**)

Comprueba si la subconsulta devuelve o no filas. Devuelve CIERTO si la subconsulta devuelve filas y FALSO si no tiene filas.

Formato: **[NOT] EXISTS (subconsulta)**

La subconsulta puede devolver una tabla con cualquier número de columnas y filas.

```
SELECT dname
FROM dept d
WHERE EXISTS (select *
              from emp
              where deptno=d.deptno)
```

Ejercicios

1. Muestra el empleado/s con el salario más alto de cada departamento.
2. Muestra el código del empleado/s que más horas trabajan en cada proyecto.
3. Muestra el nombre de empleado/s que más horas trabajan en cada proyecto
4. Muestra el nombre de empleado/s que más horas trabajan en cada proyecto.
Muestra también el nombre del proyecto.
5. Para cada departamento muestra su nombre y cuántos empleados de ese departamento tienen un salario mayor al salario medio de su departamento.
6. Para cada departamento muestra su nombre y cuántos empleados ganan más que su jefe.

Ejercicios

1.

```
Select ename, deptno, sal
from emp e
where sal= (Select max(sal) from emp
           where deptno=e.deptno)
```
2.

```
Select pronon, empno, hours
from emppro ep
where hours= (Select max(hours) from emppro
             where pronon=ep.pronon)
```
3.

```
Select pronon, ename, hours
from emppro ep join emp e on e.empno=ep.empno
where hours= (Select max(hours) from emppro
             where pronon=ep.pronon)
```
4.

```
Select pname, ename, hours
from emppro ep join emp e on e.empno=ep.empno
              join emp p on ep.pronon=p.pronon
where hours= (Select max(hours) from emppro
             where pronon=ep.pronon)
```
5.

```
Select dname, count(*)
from emp e join dept d on e.deptno=d.deptno
where sal > (Select avg(sal) from emp
            where deptno=d.deptno)
group by d.deptno, dname
```
6.

```
Select dname, count(*)
from emp e join dept d on e.deptno=d.deptno
where sal > (Select sal from emp
            where empno=e.mgr)
group by d.deptno, dname
```

Guion

Introducción

Conceptos previos

Sentencia Select

Distinct

Order by

Predicados

Funciones

Group by

Having

Join

Subconsultas

Composición de consultas

Composición de consultas

SQL dispone de tres operadores de conjuntos: unión (UNION), intersección (INTERSECT) y diferencia (EXCEPT).

Permiten realizar esas operaciones con las filas resultantes de dos sentencias select.

Formato: **consulta1**

{UNION|INTERSECT|EXCEPT} [ALL|DISTINCT]

consulta2

[order by <expre1>,...]

Las dos consultas deben ser “unión compatibles”:

- Deben tener igual número de columnas.

- Correspondencia de tipos entre las columnas ubicadas en la misma posición (contando desde la izquierda).

ALL permite filas duplicadas, DISTINCT elimina filas duplicadas. El predeterminado es DISTINCT.

Sólo puede haber un order by, que se aplicaría sobre el resultado de la operación conjuntista

Composición de consultas

```
SELECT ename, sal+comm AS "Ingresos totales", 'Incluye comisión' AS "Comisión?"  
FROM emp  
WHERE comm is not null  
  
UNION
```

```
SELECT ename, sal, 'No tiene comisión'  
FROM emp  
WHERE comm is null  
ORDER BY "Ingresos totales"
```

ename character varying	Ingresos totales numeric	Comisión? text
SMITH	800.00	No tiene comisión
JAMES	950.00	No tiene comisión
ADAMS	1100.00	No tiene comisión
MILLER	1300.00	No tiene comisión
TURNER	1500.00	Incluye comisión
WARD	1750.00	Incluye comisión
ALLEN	1900.00	Incluye comisión
CLARK	2450.00	No tiene comisión
MARTIN	2650.00	Incluye comisión
BLAKE	2850.00	No tiene comisión
JONES	2975.00	No tiene comisión
FORD	3000.00	No tiene comisión
SCOTT	3000.00	No tiene comisión
KING	5000.00	No tiene comisión