

Consultas básicas y avanzadas

Gestión de Bases de Datos



Índice

Cláusula SELECT

Expresiones aritméticas

Alias

Operador de concatenación, cadenas de literales

Limitando filas seleccionadas

Operadores de comparación y lógicos

Cláusula ORDER BY

Funciones de agrupamiento

Cláusula GROUP BY

Cláusula HAVING

Subconsultas y cláusula UNION

Cláusula SELECT

- Recupera información de una base de datos:

```
SELECT atributo1, atributo2...  
FROM nombretabla1,  
[nombretabla2,...,nombretablaN]  
[WHERE condiciones-lógicas];
```

SELECT: (elegir) especifica todos los atributos que se quieren recuperar

FROM: (de) lista de tablas de donde se escogerán los atributos en la cláusula SELECT

WHERE: (donde) incluye las condiciones que deben cumplir los atributos (es opcional)

Nota: Si se quieren recuperar todos los atributos de una tabla se usa *



Expresiones aritméticas

- Se puede modificar la forma de mostrar los datos, realizando cálculos
 - usando expresiones aritméticas
- Una expresión aritmética puede contener
 - nombres de columnas
 - valores numéricos constantes
 - operadores aritméticos (+, -, *, /)



Expresiones aritméticas

- **Ejemplo:** Calcular el incremento del salario en 300.

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

Expresiones aritméticas

- **Precedencia de operadores**
- Multiplicación y división tienen prioridad sobre suma y resta
- Los operadores con igual prioridad se evalúan de izquierda a derecha.
- Los paréntesis son usados para forzar la evaluación de la prioridad y clarificar sentencias

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

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

Expresiones aritméticas

➤ Valor null

- Un valor nulo es un valor que no está disponible, no asignado, es desconocido o no aplicable en ninguno de los casos.
- NOT NULL o PRIMARY KEY, no pueden contener valores nulos
- Si se realiza alguna operación numérica sobre un valor NULL el resultado es NULL.

```
SELECT first_name, job_id, commission_pct  
FROM employees;
```

```
SELECT first_name, 12*salary+commission_pct  
FROM employees  
WHERE first_name='KING';
```



Alias

➤ Características:

- Renombra una columna o una tabla
- Se usa
 - en cálculos (columnas)
 - Para columnas denominadas igual en tablas diferentes (tablas)
- Se puede usar AS delante del alias (es opcional)
- Si contiene espacios o caracteres sensitivos (# o \$), se incluye entre " ".



Alias

- **Ejemplo:** Renombrar los campos first_name y salary de la tabla employees.

```
SELECT first_name AS nombre, salary salario  
FROM employees;
```

- **Ejemplo:** Renombrar los campos first_name y salary(incrementado por 12 meses, y que se llame incremento anual)

```
SELECT first_name "Nombre", salary*12  
"Incremento Anual"  
FROM employees;
```



Alias

- **Ejemplo:** Renombrar tablas

```
SELECT e.first_name, d.dept_name  
FROM employees e, dept d  
WHERE e.dept_id = d.dept_id;
```



Operadores de concatenación simples y complejos. Cadenas de caracteres

- Para concatenar columnas se puede usar
 - El operador ||
 - La función CONCAT
- Como resultado → se obtiene una columna de tipo cadena con la unión de las columnas indicadas




Operadores de concatenación simples y complejos. Cadenas de caracteres

- **Ejemplo:** Concatenar los valores de nombre y trabajo en una sola columna.

```
SELECT CONCAT(first_name, job_id) "Empleados"  
FROM employees;
```

- **Ejemplo:** Concatenar nombre y trabajo del empleado con espacios

```
SELECT first_name || ' ' || 'es un' || ' ' || job_id  
FROM employees;
```



Limitando las filas seleccionadas

- En una consulta, por defecto, se muestran todas las filas, incluyendo las que están repetidas

```
SELECT department_id, job_id  
FROM employees;
```

- Para eliminar las filas duplicadas, usamos la palabra reservada DISTINCT en la cláusula SELECT

```
SELECT DISTINCT department_id, job_id  
FROM employees;
```



Operadores de comparación. Operadores lógicos

► Comparación

=	Igual que
>	Mayor que
<	Menor que
>=, <=	Mayor o igual, menor o igual
<>	diferente



Operadores de comparación. Operadores lógicos

- **Ejemplo:** Mostrar el nombre, salario y comisión de la tabla empleados para todos los empleados cuyo salario mensual es menor o igual que 1.700

```
SELECT first_name, salary, commission_pct  
FROM employees  
WHERE salary<=1700;
```



Operadores de comparación. Operadores lógicos

► Otros operadores de comparación

BETWEEN limite_inf AND limite_sup	Comprobación entre dos valores incluidos
IN(lista valores)	Comprobación con cualquier valor de la lista de valores
LIKE	Comprobación con un patrón de caracteres, se usa para búsquedas mas avanzadas de cadenas de caracteres
IS NULL	Si coincide con un valor NULL



Operadores de comparación. Operadores lógicos

➤ Operador **BETWEEN**

- **Ejemplo:** Mostrar el nombre y el salario de los empleados que cobren entre 1000 y 1500 euros.

```
SELECT first_name, salary  
FROM employees  
WHERE salary BETWEEN 1000 AND 1500;
```



Operadores de comparación. Operadores lógicos

► Operador IN

- **Ejemplo:** Mostrar el número del empleado, el salario y el manager para todos los empleados cuyo manager sea el 104, 101 o 102.

```
SELECT employee_id, salary, manager_id  
FROM employees  
WHERE manager_id IN (104, 101, 102);
```



Operadores de comparación. Operadores lógicos

➤ **Operador LIKE**

➤ Se usa para mejorar las búsquedas de cadenas de caracteres:

➤ %: representa cero o mas caracteres

➤ _: representa un carácter

➤ **Ejemplo:** Muestra todos los empleados cuyo nombre comience por S

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



Operadores de comparación. Operadores lógicos

➤ Operador IS NULL

- **Ejemplo:** Mostrar el nombre, trabajo y la comisión de aquellos empleados cuya comisión sea nula.

```
SELECT first_name, job_id, commission_pct  
FROM employees  
WHERE commission_pct IS NULL;
```



Operadores de comparación. Operadores lógicos

► Operadores lógicos

AND	Devuelve verdadero si ambos componentes son verdaderos
OR	Devuelve verdadero si alguno de los dos es verdadero
NOT	Devuelve verdadero si la condición es falsa



Operadores de comparación. Operadores lógicos

➤ Operador AND

- **Ejemplo:** Mostrar el número de empleado, nombre, trabajo y salario para aquellos que ganen más de 1100 y que su trabajo sea 'ST_CLERK'

```
SELECT employee_id, first_name, job_id, salary  
FROM employees  
WHERE salary>1100  
AND job_id='ST_CLERK';
```



Operadores de comparación. Operadores lógicos

► Operador OR

- **Ejemplo:** Mostrar el número de empleado, nombre y salario de aquellos empleados que ganen más de 1100 o que su trabajo sea ST_Clerk.

```
SELECT employee_id, first_name, job_id, salary  
FROM employees  
WHERE salary>1100  
OR job_id='ST_CLERK';
```



Operadores de comparación. Operadores lógicos

► Operador NOT

- **Ejemplo:** Mostrar el nombre y el trabajo de aquellos empleados que no sean ni st_clerk, ni st_man.

```
SELECT first_name, job_id  
FROM employees  
WHERE job_id NOT IN  
( 'ST_CLERK', 'ST_MAN' );
```




Operadores de comparación. Operadores lógicos

► Prioridad

1º	Todos los operadores de comparación
2º	NOT
3º	AND
4º	OR

Operadores de comparación.

Operadores lógicos

- **Ejemplo:** Mostrar el nombre, trabajo y salario para aquellos empleados que sean presidentes y cobren más de 1500 o que sean Sa_man

```
SELECT first_name, job_id, salary
FROM employees
WHERE job_id='SA_MAN'
OR job_id='AD_PRES'
AND salary>1500;
```

- **Ejemplo:** Mostrar todos aquellos empleados que sean o sa_man o ad_pres pero que sea cual sea, cobren más de 1500

```
SELECT first_name, job_id, salary
FROM employees
WHERE (job_id='SA_MAN'
OR job_id='AD_PRES')
AND salary>1500;
```

Cláusula ORDER BY

- Sirve para ordenar las filas resultantes de una consulta de forma ascendente o descendente (por defecto es ascendente)
- Se incluye al final de la sentencia SELECT

```
SELECT expr  
FROM tabla  
[WHERE conditions]  
[ORDER BY columna, expression) [ASC|DESC];
```



Cláusula ORDER BY

- **Ejemplo:** Mostrar el nombre, trabajo y fecha de alta de forma descendente

```
SELECT first_name, job_id, hire_date  
FROM employees  
ORDER BY hire_date DESC;
```



Cláusula ORDER BY

- **Ejemplo:** Mostrar número de empleado, nombre y salario anual de forma ascendente y usando alias

```
SELECT employee_id, first_name, salary*12  
annual  
FROM employees  
ORDER BY annual ASC;
```

Cláusula ORDER BY

- **Ejemplo:** Mostrar el nombre, número de depto, y el salario ordenados por número de departamento y salario de forma descendente.

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



Trabajo

- Realizar las consultas del siguiente enlace:

<https://sql-playground.com/#>

- Base de datos “Tienda”. Los ejercicios del 1 al 19 (excepto 14 y 15).
- Base de datos “Empleado”. Los ejercicios del 1 al 7.

Funciones de agrupamiento

- Devuelven un único valor

Funciones de Agrupamiento

<i>Función</i>	<i>Cometido</i>	<i>Ejemplo</i>
COUNT(col)	Cuenta el número de filas agrupadas.	<code>select count(nombre),oficio from emp group by oficio;</code>
AVG(col)	Calcula el valor medio de todos los valores de la columna <i>col</i> .	<code>select avg(salario),oficio from emp group by oficio;</code>
MAX(col)	Calcula el valor máximo de todos los valores de la columna <i>col</i> .	<code>select max(salario),oficio from emp group by oficio;</code>
MIN(col)	Calcula el valor mínimo de todos los valores de la columna <i>col</i> .	<code>select min(salario),oficio from emp group by oficio;</code>
SUM(col)	Calcula la suma de los valores de la columna <i>col</i> .	<code>select sum(salario), oficio from emp group by oficio;</code>

Cláusula GROUP BY

- Divide la información de la tabla en grupos más pequeños
 - Se utiliza para agrupar resultados

```
SELECT columna, función_grupo(columna)
FROM tabla
[WHERE condición]
[GROUP BY columna]
[ORDER BY columna]
```

- No se puede usar alias de columnas en esta cláusula

Cláusula GROUP BY

- **Ejemplo:** Mostrar salario medio de cada uno de los departamentos

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

- **Ejemplo:** Mostrar salario medio de cada uno de los departamentos, ordenados por salario

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



Cláusula GROUP BY

- **Ejemplo:** Muestra el máximo salario medio de departamentos

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

Cláusula GROUP BY

- **Agrupando por más de una columna**
- **Ejemplo:** Muestra la suma de los salarios para cada trabajo, agrupados por departamento

```
SELECT department_id, job_id, SUM(salary)
FROM employees
GROUP BY department_id, job_id
```

- **Restricciones**
 - Cualquier columna o expresión en la lista SELECT que no sea una función de agregación debe estar en la cláusula GROUP BY
 - No se puede usar la cláusula WHERE para restringir grupos. Para eso, tenemos la cláusula HAVING

Cláusula HAVING

- La cláusula HAVING sirve para restringir grupos
 - Especifica una condición de búsqueda para grupos o funciones de agregación
- **Ejemplo:** Muestra la media de los salarios por departamento, siempre que esta sea mayor que 2000.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id
HAVING AVG(salary) > 2000;
```



Subconsultas



Formato

```
SELECT ...  
FROM ...  
WHERE columna operador_comparativo  
      (SELECT ....  
        FROM .....  
        WHERE .....);
```

Subconsultas

► Comparación en subconsultas

► >, <, <>, <=, >=, =

Ejemplo:

```
Select apellido  
From Empleado  
Where oficio = (Select oficio  
                from Empleado  
                Where apellido = 'Gil');
```



Subconsultas

- Pertenencia a un conjunto devuelto por una subconsulta (IN)

Ejemplo:

```
Select apellido  
From Empleado  
Where oficio IN (Select oficio  
                  From Empleado  
                  Where Dept_no = 20);
```




Subconsultas



Existencia (Exist, Not Exist)

Ejemplo: Listar los departamentos que tengan empleados

```
Select dnombre, dept_no  
From Departamento  
Where Exist (Select *  
              From Empleado  
              Where emple.dept_no =  
depart.dept_no);
```



Subconsultas

- Comparación cuantificada (ANY y ALL)

Se usan en conjunción con los operadores de comparación

- **ANY**

compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta. Si alguna comparación da TRUE, la consulta es TRUE, sino es FALSE.

- **ALL**

compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta. Si todas las comparaciones dan TRUE, la consulta es TRUE, sino es FALSE.



Subconsultas

➤ Ejemplo ANY

```
Select *  
From Empleado  
Where salario = ANY (Select salario  
                     From Empleado  
                     Where Dept_no = 30);
```

➤ Ejemplo ALL

```
Select *  
From Empleado  
Where salario < ALL (Select salario  
                    From Empleado  
                    Where Dept_no = 30);
```

Cláusula UNION

- Concatena los resultados de dos consultas en un único conjunto de resultados.
 - **UNION ALL**: incluye duplicados
 - **UNION**: se excluyen los duplicados

```
SELECT...  
FROM  
[WHERE ...]  
UNION  
SELECT...  
FROM  
[WHERE ...]
```

Cláusula UNION

➤ Ejemplo

```
select *  
from clientes  
where repclie = 101  
union  
select *  
from clientes  
where repclie = 102
```

Resultados de la consulta 1

NUMCLIE	NOMBRE	REPCLIE	LIMITECREDITO
2102	Alvaro Rodr��guez	101	65000
2105	Sntonio Canales	101	45000
2115	Vicente Mart��nez	101	20000

Resultados de la consulta 2

NUMCLIE	NOMBRE	REPCLIE	LIMITECREDITO
2106	Juan Su��rez	102	65000
2114	Cristina Bulini	102	20000
2120	Juan Malo	102	50000
2123	Jos�� Libros	102	40000

Resultados con el uni  n (1 y 2)

NUMCLIE	NOMBRE	REPCLIE	LIMITECREDITO
2102	Alvaro Rodr��guez	101	65000
2105	Sntonio Canales	101	45000
2106	Juan Su��rez	102	65000
2114	Cristina Bulini	102	20000
2115	Vicente Mart��nez	101	20000
2120	Juan Malo	102	50000
2123	Jos�� Libros	102	40000



Trabajo

- Realizar las consultas del siguiente enlace:

<https://sql-playground.com/#>

- Base de datos “Tienda”. Los ejercicios del 20 al 33. Y los ejercicios de multitabla (1 al 3) y subconsultas (1).
- Base de datos “Empleado”. Los ejercicios del 8 al 17.
- Base de datos “Ventas”. Los ejercicios del 1 al 11.