Ingeniería de Datos

Bases de Datos Relacionales: JOIN, Subconsultas y Agrupamiento

Nadia Rodríguez

2013-1

Agenda

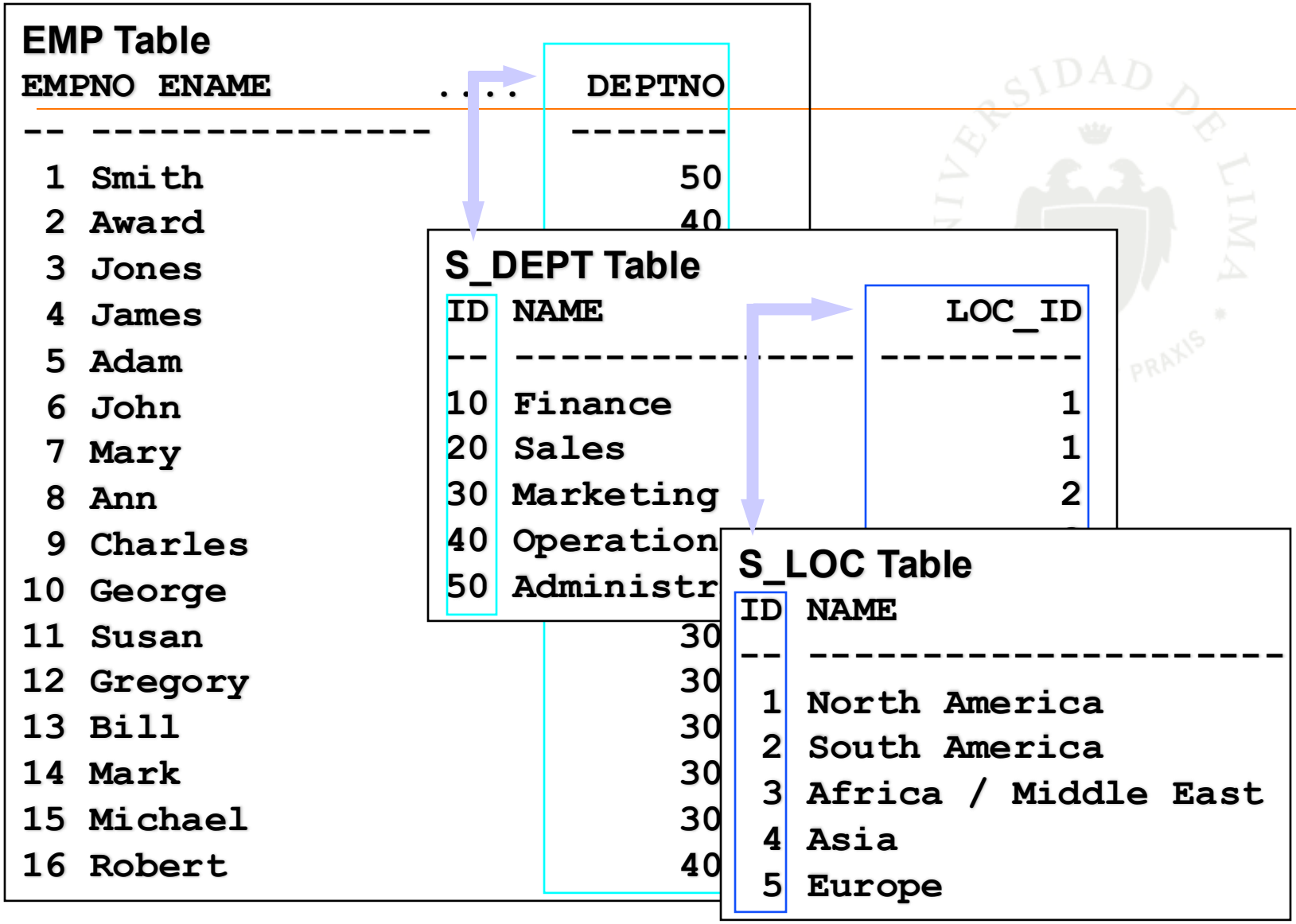
- JOIN:
  - Concepto de Join
  - Cross Join: Producto Cartesiano
  - Variables Tupla
  - Tipos de JOIN: Self Join, Equijoin, Non-Equijoin, Outer Join
- Subconsultas
  - Fila Simple
  - Uso de Funciones de Grupo
- Funciones de Grupo
  - MIN, MAX, AVG, SUM, COUNT
- SELECT
  - ORDER BY
  - GROUP BY
  - HAVING

JOIN

¿Qué es un Join?

- Un JOIN se emplea para consultar datos que se encuentran en más de una tabla.
- Las filas son enlazadas empleando valores comunes, típicamente entre el Primary y Foreign Key.
- Métodos de unión
  - Equijoin
  - Non-equijoin
  - Self join
  - Outer join

Relaciones entre Tablas



Cross Join (Producto Cartesiano)

- Se crea cuando:
  - Una condición join es omitida
  - Una condición join es invalida
  - Todas las filas de la primera tabla se juntan a todas las filas en la segunda tabla

EMP									DEPT		
PK			FK				FK		PK		
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO		DEPTNO	DNAME	LOC
7369	SMITH	CLERK	7902	17-DEC-80	800		20		10	ACCOUNTING	NEW YORK
7499	ALLEN	SALESMAN	7698	20-Feb-81	1600	300	30		20	RESEARCH	DALLAS
7521	WARD	SALESMAN	7698	22-Feb-81	1250	500	30		30	SALES	CHICAGO
7566	JONES	MANAGER	7839	02-APR-81	2975		20		40	OPERATIONS	BOSTON
7654	MARTIN	SALESMAN	7698	28-Sep-81	1250	1400	30				
7698	BLAKE	MANAGER	7839	01-May-81	2850		30				
7782	CLARK	MANAGER	7839	09-Jun-81	2450		10				
7788	SCOTT	ANALYST									
7839	KING	PRESIDENT									
7844	TURNER	SALESMAN									
7876	ADAMS	CLERK									
7900	JAMES	CLERK									
7902	FORD	ANALYST									
7934	MILLER	CLERK									

EMPXDEPT

7369	SMITH	CLERK	7902	17-DEC-80	800		20	10	ACCOUNTING	NEW YORK
7370	SMITH	CLERK	7903	17-DEC-81	800		20	20	RESEARCH	DALLAS
7371	SMITH	CLERK	7904	17-DEC-82	800		20	30	SALES	CHICAGO
7372	SMITH	CLERK	7905	17-DEC-83	800		20	40	OPERATIONS	BOSTON
7499	ALLEN	SALESMAN	7698	20-Feb-81	1600	300	30	10	ACCOUNTING	NEW YORK
7500	ALLEN	SALESMAN	7699	21-Feb-81	1600	300	30	20	RESEARCH	DALLAS
7501	ALLEN	SALESMAN	7700	22-Feb-81	1600	300	30	30	SALES	CHICAGO
7502	ALLEN	SALESMAN	7701	23-Feb-81	1600	300	30	40	OPERATIONS	BOSTON
7521	WARD	SALESMAN	7698	22-Feb-81	1250	500	30	10	ACCOUNTING	NEW YORK
7522	WARD	SALESMAN	7699	23-Feb-81	1250	500	30	20	RESEARCH	DALLAS
7523	WARD	SALESMAN	7700	24-Feb-81	1250	500	30	30	SALES	CHICAGO
7524	WARD	SALESMAN	7701	25-Feb-81	1250	500	30	40	OPERATIONS	BOSTON
7566	JONES	MANAGER	7839	02-APR-81	2975		20	10	ACCOUNTING	NEW YORK
7567	JONES	MANAGER	7840	02-APR-82	2975		20	20	RESEARCH	DALLAS
7568	JONES	MANAGER	7841	02-APR-83	2975		20	30	SALES	CHICAGO
7569	JONES	MANAGER	7842	02-APR-84	2975		20	40	OPERATIONS	BOSTON
7654	MARTIN	SALESMAN	7698	28-Sep-81	1250	1400	30	10	ACCOUNTING	NEW YORK
7655	MARTIN	SALESMAN	7699	29-Sep-81	1250	1400	30	20	RESEARCH	DALLAS
7656	MARTIN	SALESMAN	7700	30-Sep-81	1250	1400	30	30	SALES	CHICAGO
7657	MARTIN	SALESMAN	7701	01-Oct-81	1250	1400	30	40	OPERATIONS	BOSTON
7698	BLAKE	MANAGER	7839	01-May-81	2850		30	10	ACCOUNTING	NEW YORK
7699	BLAKE	MANAGER	7840	02-May-81	2850		30	20	RESEARCH	DALLAS
7700	BLAKE	MANAGER	7841	03-May-81	2850		30	30	SALES	CHICAGO
7701	BLAKE	MANAGER	7842	04-May-81	2850		30	40	OPERATIONS	BOSTON
7782	CLARK	MANAGER	7839	09-Jun-81	2450		10	10	ACCOUNTING	NEW YORK

Variables Tupla

- Simplifica las consultas mediante el uso de alias en los nombres de las tablas.

- SELECT

  C.nombre, ***OP***.numero_orden, ***OP***.numero_embarques
  FROM orden_pedido ***OP*** ***JOIN*** cliente ***C***
  ON ***OP***.id_cliente = ***C.***id_cliente;

Equijoin

```
SELECT    table1.column, table2.column
FROM      table1 JOIN table2
ON        table1.column1 = table2.column2;
```

- Cada nombre de columna debe ser precedido por el nombre de la tabla.
- En caso que las tablas tengan idéntico nombre usar alias.

Equijoin

- Resulta cuando el valor de una columna corresponde explícitamente al valor de alguna columna de otra tabla.

```
SELECT
C.nombre, OP.numero_orden, OP.numero_embarques
FROM orden_pedido OP JOIN cliente C
ON OP.id_cliente = C.id_cliente;
```

Non-Equijoins

- Resulta cuando el valor de una columna NO corresponde explícitamente al valor de alguna columna de otra tabla.
- El *join condition* contiene un operador diferente al igual (=).

- SELECT ***L***.nombre_linea
  FROM producto ***P*** JOIN linea_producto ***L***
  ON ***P***.precio BETWEEN ***L.***meta_venta_min AND
  L.meta_venta_max
  AND ***P***.idprod = ***L.***idprod;

Self Joins

- Utilizada para unir filas de una misma tabla con relación recursiva.
- Simula tener dos tablas en el FROM creando 2 alias diferentes para la misma tabla.

```
SELECT E.nombre || 'trabaja para'|| J.nombre
FROM empleado E JOIN empleado J
ON E.cod_jefe = J.cod_emp;
```

Outer Joins

- Se utiliza para ver filas que normalmente no puede presentarlas mediante una condición join.

- El operador del Outer Join puede ser **LEFT** o **RIGHT**, dependiendo en el lado del join que no tenga un valor para el join.

Outer Joins

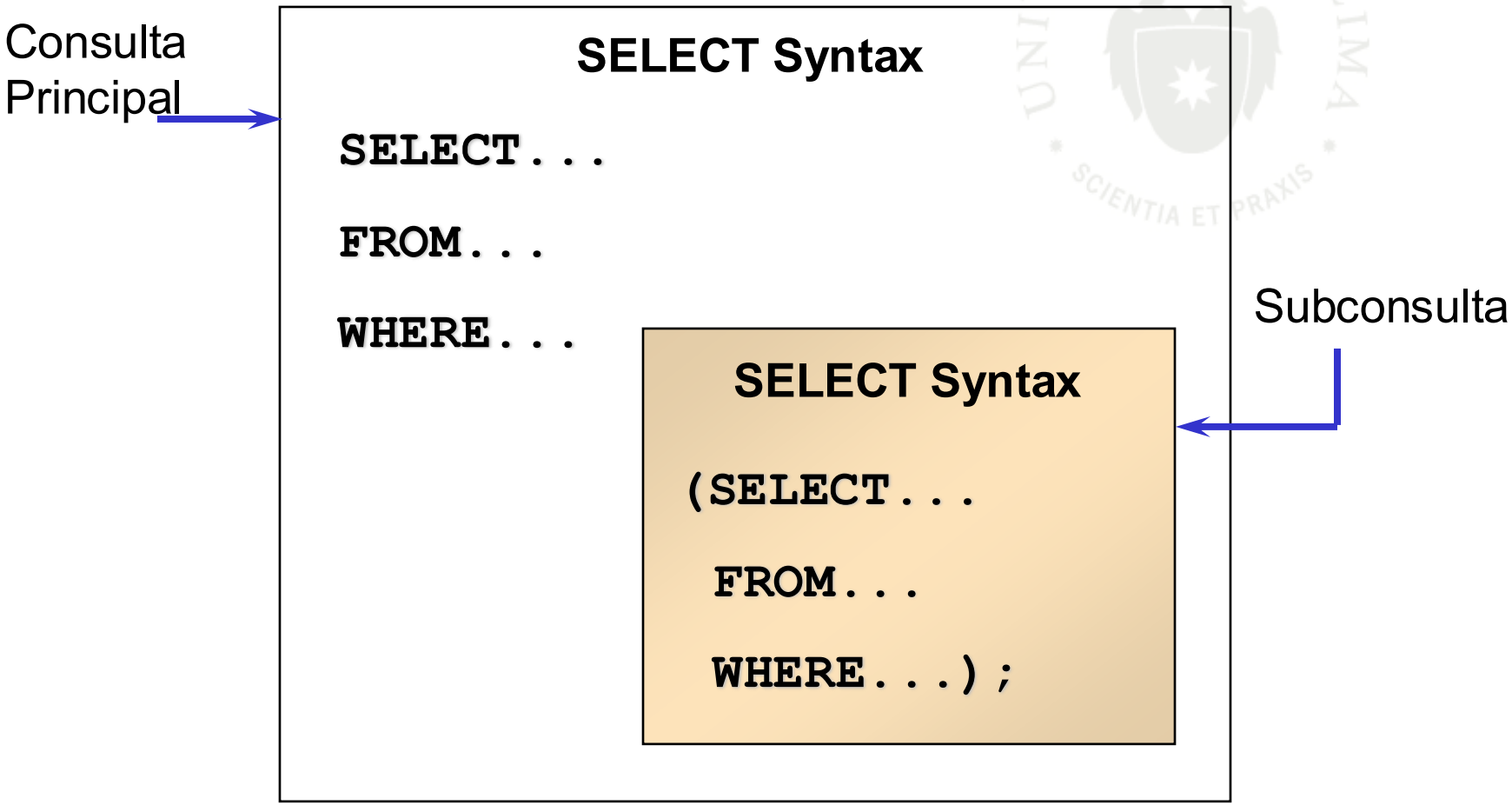
```
SELECT    table1.column, table2.column
FROM      table1 [LEFT |RIGHT] OUTER JOIN table2
WHERE     table1.column1 = table2.column2;
```

- SELECT nombre, numero_orden, numero_embarques FROM cliente ***LEFT OUTER JOIN*** orden_pedido ***ON*** cliente.id_cliente = orden_pedido.id_cliente;

- SELECT nombre, numero_orden, numero_embarques FROM orden_pedido ***RIGTH OUTER JOIN*** cliente ***ON*** cliente.id_cliente = orden_pedido.id_cliente;

SUBCONSULTAS

¿Qué es una Subconsulta?



Subconsultas

```
SELECT      select_list
FROM        table
WHERE       expr operator
            (SELECT      select_list
             FROM        table);
```

- Las subconsultas se ejecutan antes que la consulta principal.
- El resultado de la subconsulta es usada por la consulta principal o externa.

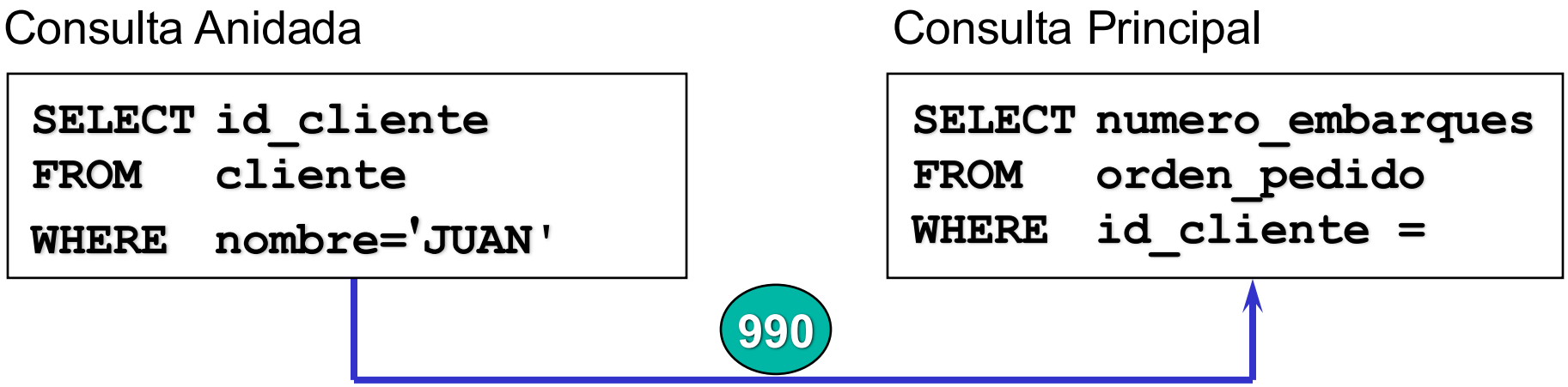
Reglas para Subconsultas

- Son usadas cuando una consulta está basada en valores desconocidos
- Deben estar encerradas entre paréntesis
- Dos clases de operadores de comparación pueden usarse en las subconsultas: de simple y de múltiple fila
- El operador debe aparecer al lado izquierdo de la subconsulta
- Pueden usar varios comandos del SQL
- No pueden contener una cláusula ORDER BY
- Pueden contener más de una sentencia SELECT
- Las subconsultas son procesadas primero, luego la consulta principal es ejecutada y los resultados servirán de base para las cláusulas WHERE o HAVING de la consulta principal.

¿Cómo son procesadas las Subconsultas Anidadas?

```
SELECT numero_embarques
FROM   orden_pedido
WHERE  id_cliente = (SELECT id_cliente
                     FROM   cliente
                     WHERE  nombre='JUAN')
```

1. La sentencia SELECT anidada es ejecutada
2. El resultado es pasado a la consulta principal.



Subconsultas de fila simple: Operadores

- La subconsulta retorna como resultado una sola fila
- Sólo utiliza operador de fila simple

=	>	>=	<	<=	< >

Subconsultas de Fila Simple

```
SELECT    numero_embarques
FROM      orden_pedido
WHERE     id_cliente =
                    (SELECT   id_cliente
                     FROM     cliente
                     WHERE    nombre = 'JUAN')
```

Solo debe haber un cliente llamado 'JUAN' en la tabla cliente

Subconsultas de Fila Simple

```
SELECT    id_cliente
FROM      orden_pedido
WHERE     numero_embarques =

                  (SELECT MAX(numero_embarques)
                   FROM   orden_pedido)
```

Subconsultas de Fila Múltiple

- La subconsulta de fila múltiple retorna varias filas de resultado.

IN	Igual a cualquier miembro de la lista
NOT IN	No es igual a ningún miembro de la lista

Subconsultas de Fila Múltiple

- Utilizar el operador de fila múltiple en la cláusula WHERE (Ej. IN).

```
SELECT          numero_embarques
  FROM          orden_pedido
 WHERE          id_cliente IN (SELECT id_cliente
                               FROM   cliente
                               WHERE tipo = 'NAC' OR
                                     tipo = 'INT');
```

AGRUPAMIENTO

Funciones de Agrupamiento

- Son aquellas que nos permiten obtener resultados a partir de los agrupamientos de filas.
- Tipos
  - MIN (n)
  - MAX (n)
  - AVG (n)
  - SUM (n)
  - COUNT (x)

Funciones de Agrupamiento (cont.)

- SELECT MIN (numero_embarques),
  MAX (numero_embarques),
  AVG(numero_embarques),
  SUM(numero_embarques)
  FROM orden_pedido
  WHERE fecha_llenado LIKE '%07';

- SELECT COUNT(*)
  FROM cliente;

La clásula GROUP BY

- Para dividir en subconjuntos de filas de una tabla se utiliza la cláusula GROUP BY.

- Se debe incluir una lista de columnas en la cláusula GROUP BY siempre y cuando éstas columnas hayan sido considerada en el SELECT.

- Para modificar el orden de presentación utilizar la cláusula ORDER BY. Por default el ordenamiento es ascendente.

La cláusula GROUP BY

```
SELECT      columna, grupo_funcion
FROM        tabla
[WHERE      condicion]
[GROUP BY   grupo_por_expresion]
[ORDER BY   columna];
```

SELECT *tipo*, COUNT(*)
FROM cliente
WHERE *apellido* LIKE 'A%'
GROUP BY *tipo*
ORDER BY *tipo* DESC;

Muestra los tipos de cliente y la cantidad por cada tipo de cliente, solo de aquellos clientes cuyos apellidos empiezan con A, en orden descendente.

Las cláusulas GROUP BY y HAVING en la sentencia SELECT

- HAVING es utilizada para especificar que grupos serán mostrados en el resultado de la consulta

- Toda cláusula de HAVING requiere de una cláusula de GROUP BY precedente

Las cláusulas GROUP BY y HAVING en la sentencia SELECT

```
SELECT      columna, grupo_funcion
FROM        tabla
[WHERE      condicion]
[GROUP BY   grupo_por_expresion]
[HAVING     grupo_condicion]
[ORDER BY   columna];
```

```
SELECT id_cliente, COUNT(*)
FROM orden_pedido
GROUP BY id_cliente
HAVING COUNT(*) > 5;
```

Muestra los id's de los clientes y la cantidad de ordenes de pedido por cada cliente, solo de aquellos clientes que hayan generado más de 5 órdenes de pedido

Cláusula HAVING con Subconsultas

- Se puede usar subconsultas en la cláusula HAVING.
- Se retorna los resultados a la consulta principal de la cláusula HAVING.

```
SELECT    id_cliente, COUNT(*)
FROM      orden_pedido
GROUP BY id_cliente
HAVING    COUNT(*) > (SELECT AVG(COUNT(*)) FROM orden_pedido
                      GROUP BY id_cliente)
```