

# Resumen de SQL

Álvaro González Sotillo

30 de septiembre de 2020

## Índice

1. Orden de ejecución de una <i>query</i>	1
2. Funciones sobre valores	2
3. Funciones sobre grupos	2
4. Producto cartesiano	2
5. Vistas	5
6. <i>Query</i> en un <b>from</b>	5
7. <i>with</i>	5
8. <i>Query</i> traducida a valor	5
9. <i>Query</i> traducida a lista	6
10. <b>Having</b>	6
11. Referencias	6

## 1. Orden de ejecución de una *query*

1. from
2. where
3. group by
4. having
5. select
6. order by

---

```
select
  location, count(*) as ocupacion
from
  section
where
  capacity > 10
group by
  location
having
  count(*) < 10
order by
  ocupacion;
```

## 2. Funciones sobre valores

- upper, lower: Mayúsculas y minúsculas
- trunc: quita decimales a números, horas y minutos a las fechas
- substr: partes de un varchar
- mod, sqrt...: funciones matemáticas

## 3. Funciones sobre grupos

- avg: media
- max: mínimo
- min: máximo
- count: cuenta valores de columnas
- count(distinct ): cuenta valores distintos
- count(\*): cuenta todas las filas, incluidos los valores null

## 4. Producto cartesiano

- Es una operación de conjuntos
- Para calcular  $P = A \times B$ 
  - Por cada elemento  $a \in A$ 
    - Por cada elemento  $b \in B$ 
      - ◊  $(ab)$  es un elemento de  $P$
- Ejemplo
  - $A = \{\text{Juan, María}\}$
  - $B = \{\text{González, Pérez, García}\}$
  - $P = \{\text{Juan González, Juan Pérez, Juan García, María González, María Pérez, María García}\}$
- Se llama *producto* porque  $|P| = |A| \cdot |B|$

---

#### 4.1. Tabla original

- Solo un pedido al día
- No respeta 2FN (Precio depende de parte de la clave)

Cuadro 1: VENTAS				
Producto	Precio	Cantidad	Fecha pedido	Cliente
Pera	1	2	1-1	Pepe
Manzana	2	4	1-1	Pepe
Naranja	3	3	1-1	María
Manzana	2	6	1-2	María
Pera	1	5	1-2	Juan
Naranja	3	3	1-2	Juan

#### 4.2. Tablas normalizadas

Cuadro 2: PRODUCTOS	
Producto	Precio
Pera	1
Manzana	2
Naranja	3

Cuadro 3: PEDIDOS			
Producto	Cantidad	Fecha pedido	Cliente
Pera	2	1-1	Pepe
Manzana	4	1-1	Pepe
Naranja	3	1-1	María
Manzana	6	1-2	María
Pera	5	1-2	Juan
Naranja	3	1-2	Juan

#### 4.3. Cómo recuperar información original

- La tabla original VENTAS puede seguir siendo necesaria para un informe
- Se puede recuperar con los siguientes pasos:
  - Se calcula la tabla PRODUCTOS  $\times$  PEDIDOS
  - Quitamos las filas que no respeten la *foreign key*

#### 4.3.1. PRODUCTOS × PEDIDOS

PRODUCTO.producto	PRODUCTO.precio	PEDIDOS.producto	PEDIDOS.cantidad	PEDIDOS.Fecha pedido	PEDIDOS.producto
<u>Pera</u>	1	<u>Pera</u>	2	1-1	Pera
Pera	1	Manzana	4	1-1	Pera
Pera	1	Naranja	3	1-1	Manzana
Pera	1	Manzana	6	1-2	Manzana
<u>Pera</u>	1	<u>Pera</u>	5	1-2	Ju
Pera	1	Naranja	3	1-2	Ju
Manzana	2	Pera	2	1-1	Pera
<u>Manzana</u>	2	<u>Manzana</u>	4	1-1	Pera
Manzana	2	Naranja	3	1-1	Manzana
<u>Manzana</u>	2	<u>Manzana</u>	6	1-2	Manzana
Manzana	2	Pera	5	1-2	Ju
Manzana	2	Naranja	3	1-2	Ju
Naranja	3	Pera	2	1-1	Pera
Naranja	3	Manzana	4	1-1	Pera
<u>Naranja</u>	3	<u>Naranja</u>	3	1-1	Manzana
Naranja	3	Manzana	6	1-2	Manzana
Naranja	3	Pera	5	1-2	Ju
<u>Naranja</u>	3	<u>Naranja</u>	3	1-2	Ju

#### 4.3.2. PRODUCTOS × PEDIDOS, filtrado

- Nos quedamos solo con las filas where PRODUCTO.producto = PEDIDOS.producto

PRODUCTO.producto	PRODUCTO.precio	PEDIDOS.producto	PEDIDOS.cantidad	PEDIDOS.Fecha pedido	PEDIDOS.producto
<u>Pera</u>	1	<u>Pera</u>	2	1-1	Pera
<u>Pera</u>	1	<u>Pera</u>	5	1-2	Ju
Manzana	2	Manzana	4	1-1	Pera
Manzana	2	Manzana	6	1-2	Manzana
Naranja	3	Naranja	3	1-1	Manzana
Naranja	3	Naranja	3	1-2	Ju

#### 4.4. Sintaxis SQL

```
select
*
from
PRODUCTOS, PEDIDOS
where
PRODUCTOS.producto = PEDIDOS.producto;
```

```
select
*
from
PRODUCTOS join PEDIDOS on PRODUCTOS.producto = PEDIDOS.producto;
```

---

## 5. Vistas

- Una *query* puede guardarse como una vista
- Las vistas se comportan como tablas con la orden `select`, extrayendo información de las tablas originales
- En general, no se pueden modificar datos de una vista, hay que modificar las tablas de origen.

```
create view ALUMNOS as
select student_id as clave, first_name as nombre, last_name as apellidos from student;

select * from alumnos;
```

## 6. *Query* en un `from`

- En el `from` no es obligatorio poner tablas
- Se puede poner cualquier cosa con filas y columnas:
  - Tablas
  - Vistas
  - Otras *queries*

```
select * from (
  select student_id as clave, first_name as nombre, last_name as apellidos from student
);
```

## 7. *with*

- Si no se quiere definir una vista o usar una *query* dentro de `from`

```
select nombre, apellidos from (
  select student_id as clave, first_name as nombre, last_name as apellidos from student
)
where clave > 10;

with estudiantes as(
  select student_id as clave, first_name as nombre, last_name as apellidos from student
)
select nombre, apellidos
from estudiantes
where clave > 10;
```

## 8. *Query* traducida a valor

- Se puede poner una consulta que devuelva una fila y una columna en cualquier lugar donde se necesite un valor simple

```
select * from student
where upper(last_name) = (
  select max(upper(last_name)) from student
);
```

---

## 9. *Query* traducida a lista

- Se puede poner una consulta que devuelva una columna y muchas filas en una condición in

```
select distinct course_no from section where capacity = 25 order by course_no;
select * from course where course_no = 20 or course_no = 220 or course_no = 134;
select * from course where course_no in (20,220,134);

select * from course where course_no in (
  select distinct course_no from section where capacity = 25
);
```

## 10. Having

- having sirve para poner condiciones a los grupos de una consulta
- Se puede simular con una subconsulta en el from

```
select * from
(
  select
    location, count(*) as ocupacion
  from
    section
  group by
    location
)
where ocupacion < 10;

select
  location, count(*) as ocupacion
from
  section
group by
  location
having
  count(*) < 10;
```

## 11. Referencias

- Formatos:
  - [Transparencias](#)
  - [PDF](#)
  - [EPUB](#)
- Creado con:
  - [Emacs](#)
  - [org-re-reveal](#)
  - [Latex](#)
- Alojado en [Github](#)