Producto cartesiano y JOIN

Álvaro González Sotillo

24 de marzo de 2020

Índice

1. Funciones sobre valores

- `upper`, `lower`: Mayúsculas y minúsculas
- `trunc`: quita decimales a números, horas y minutos a las fechas

2. Funciones sobre grupos

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

3. 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 $|P| = |A| \cdot |B|$

3.1. 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

3.2. 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

3.3. 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 × PEDIDOS
  - Quitamos las filas que no respeten la *foreign key*

3.3.1. PRODUCTOS × PEDIDOS

PRODUCTO.producto	PRODUCTO.precio	PEDIDOS.producto	PEDIDOS.cantidad	PEDIDOS.Fecha pedido	PE
Pera	1	**Pera**	2	1-1	Pe
Pera	1	Manzana	4	1-1	Pe
Pera	1	Naranja	3	1-1	M
Pera	1	Manzana	6	1-2	M
Pera	1	**Pera**	5	1-2	Ju
Pera	1	Naranja	3	1-2	Ju
Manzana	2	Pera	2	1-1	Pe
Manzana	2	**Manzana**	4	1-1	Pe
Manzana	2	Naranja	3	1-1	M
Manzana	2	**Manzana**	6	1-2	M
Manzana	2	Pera	5	1-2	Ju
Manzana	2	Naranja	3	1-2	Ju
Naranja	3	Pera	2	1-1	Pe
Naranja	3	Manzana	4	1-1	Pe
Naranja	3	**Naranja**	3	1-1	M
Naranja	3	Manzana	6	1-2	M
Naranja	3	Pera	5	1-2	Ju
Naranja	3	**Naranja**	3	1-2	Ju

3.3.2. 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	PE
Pera	1	**Pera**	2	1-1	Pe
Pera	1	**Pera**	5	1-2	Ju
Manzana	2	**Manzana**	4	1-1	Pe
Manzana	2	**Manzana**	6	1-2	M
Naranja	3	**Naranja**	3	1-1	M
Naranja	3	**Naranja**	3	1-2	Ju

3.4. Sintaxis SQL

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

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

4. 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;
```

5. *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
);
```

6. *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
);
```

7. *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
);
```

8. `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;
```

9. 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;
```