Aggregation

Denis Miginsky

The question of the day

How many different material types are required for each product?

GROUP BY

SQL:

```
SELECT CLASS FROM CATEGORY
GROUP BY CLASS;
```

```
>> Fuel
   Mineral
   …
```

Works the same as

```
SELECT DISTINCT CLASS FROM CATEGORY
```

Looks not very useful yet…

GROUP BY, second try

SQL:

```
SELECT CLASS, COUNT(*)
FROM CATEGORY
GROUP BY CLASS;
```

```
>> Fuel         1
   Mineral      3

   …
```

Semantics of GROUP BY

CLASS	WARE
Food	Meat
Mineral	Water
Food	Grain
Mineral	Gold ore

Grouping →

Not a relation!

CLASS	WARE
Food	Meat
	Grain
Mineral	Water
	Gold ore

Projection & aggregation →

CLASS	COUNT()
Food	2
Mineral	2

Projection for GROUP BY

- Using aggregation functions (COUNT, etc.) **without** GROUP BY is turning the aggregation mode on and causes the query to produce the single group.
  This group will produce the single row after aggregation.
  The usage of any attributes without aggregation functions is prohibited in projection.

- Using **GROUP BY** is turning on the regular aggregation mode and multiple groups can be produced.
  In projection the attributes under GROUP BY can be used **without** the aggregation functions and all other attributes can only be used inside the aggregation functions.

- **DISTINCT** is useless in projection when GROUP BY is in use. However, it coluld be useful under aggregation functions

Q. of the day: answer?

```
SELECT p.WARE AS PRODUCT,
       COUNT(m.WARE) AS MATERIALS_NUM
FROM MATERIAL m
JOIN PRODUCT p
ON p.RECIPE_ID=m.RECIPE_ID
GROUP BY p.WARE;
```

Is it correct?

Questions on the question (and answer)

- Is the answer correct? How to fix errors?

- Is the query itself OK? Is the answer itself useful?

- What if we want the answer for **Food** category of products only? How will the selection work with the aggregation?

Statements execution order

1. All the JOINs with conditions (ON)
2. Selection (WHERE)
3. Grouping (GROUP BY)
4. Projection (SELECT)
5. Set operations (UNION, INTERSECT, etc.)
6. Ordering (ORDER BY)
7. Paging (OFFSET, LIMIT)

The final answer

```
SELECT p.WARE AS PRODUCT,
       GROUP_CONCAT(DISTINCT m.WARE) AS MATERIALS
FROM PRODUCT p
LEFT JOIN MATERIAL m
ON p.RECIPE_ID=m.RECIPE_ID
JOIN CATEGORY pc
ON pc.WARE=p.WARE
WHERE pc.CLASS='Food’
GROUP BY p.WARE;
```

Another question update

We are only interested in products with multiple materials (2 or more).

Can we use WHERE?

Yet another answer

```
SELECT p.WARE AS PRODUCT,
       GROUP_CONCAT(DISTINCT m.WARE) AS MATERIALS
FROM PRODUCT p
JOIN MATERIAL m
ON p.BILL_ID=m.BILL_ID
JOIN CATEGORY pc
ON pc.WARE=p.WARE
GROUP BY p.WARE
HAVING COUNT(DISTINCT m.WARE)>1;
```

Statements execution order (updated)

1. All the JOINs with conditions (ON)
2. Selection (WHERE)
3. Grouping (GROUP BY)
4. **After-grouping selection (HAVING)**
5. Projection (SELECT)
6. Set operations (UNION, INTERSECT, etc.)
7. Ordering (ORDER BY)
8. Paging (OFFSET, LIMIT)

Attributes usage in sections

- In all the sections **before GROUP BY** all the attributes are available by themselves i.e. without aggregation

- In the sections **after GROUP BY** the attributes used for grouping are available by themselves. Other attributes are available with aggregation only.

- **Rule:** if you are planning to use attribute in projection, HAVING section, etc., you must place it under GROUP BY, even in case it depends only on the grouping attributes.