Database Systems
CSE 344

Lectures 5: Grouping & Aggregation

Wednesday June 28

Announcements

- HW2 is out
  - due next Wednesday 11pm
  - same format as HW1
  - uses joins, aggregation, grouping
- WQ2 due Sunday 11pm

Outline

- Last time:
  - outer joins
  - how to aggregate over all rows
- Today:
  - Grouping & Aggregations (6.4.3 – 6.4.6)

Aggregation

Purchase(product, price, quantity)

Find number of bagels sold for more than $1

```
SELECT    Sum(quantity) as Sold
FROM      Purchase
WHERE     price > 1 and product = 'bagel'
```

Grouping and Aggregation

Purchase(product, price, quantity)

Find number sold for more than $1 **for each product**

Group By

New Keyword

Grouping and Aggregation

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
~~Banana~~	~~0.5~~	~~50~~
Banana	2	10
Banana	4	10

Product	Sales
Bagel	40
Banana	20

```
SELECT   product, Sum(quantity) AS Sales
FROM     Purchase
WHERE    price > 1
GROUP BY product
```

Grouping and Aggregation

```
Purchase(product, price, quantity)
```

Find total quantities for all sales over $1, by product.

```
SELECT   product, Sum(quantity) AS Sales
FROM     Purchase
WHERE    price > 1
GROUP BY product
```

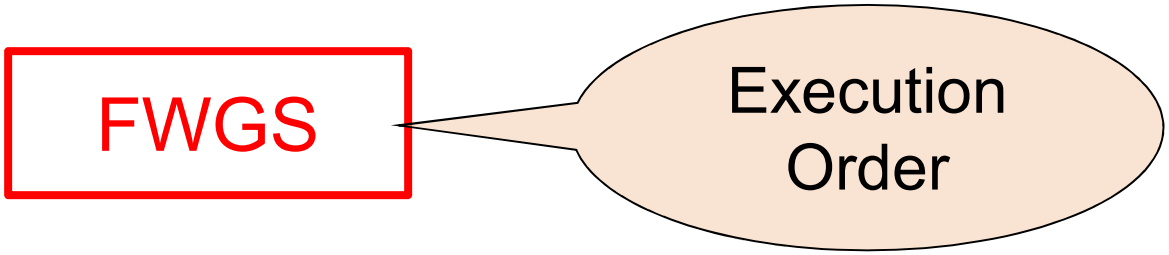
How is this query processed?

Grouping and Aggregation

1. Compute the **F**ROM and **W**HERE clauses.

2. Group by the attributes in the **G**ROUP BY

3. Compute the **S**ELECT clause:
   grouped attributes and aggregates.



1,2: From, Where

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
~~Banana~~	~~0.5~~	~~50~~
Banana	2	10
Banana	4	10

WHERE price > 1

```
SELECT    product, Sum(quantity) AS Sales
FROM      Purchase
WHERE     price > 1
GROUP BY  product
```

3,4. Grouping, Select

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	~~0.5~~	~~50~~
Banana	2	10
Banana	4	10

Product	Sales
Bagel	40
Banana	20

```
SELECT    product, Sum(quantity) AS Sales
FROM      Purchase
WHERE     price > 1
GROUP BY  product
```

Need to be Careful…

```
SELECT product,
       max(quantity)
FROM   Purchase
GROUP BY product
```

```
SELECT   product, quantity
FROM     Purchase
GROUP BY product
```

```
SELECT   quantity
FROM     Purchase
```

```
SELECT   product
FROM     Purchase
GROUP BY product
```

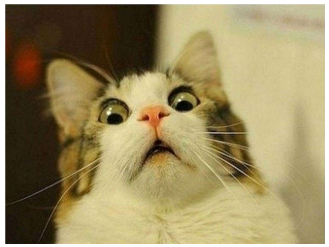
Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
	4	10

Product
Bagel
Banana

+

???

Quantity
20
20
50
10



Can't project a non-grouped / non-aggregated column!

Need to be Careful…

```
SELECT product,
       max(quantity)
FROM   Purchase
GROUP BY product
```

```
SELECT   product, quantity
FROM     Purchase
GROUP BY product, quanty
```

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

SQLite is WRONG on the second query.
Better DBMS (e.g. SQL Server) gives an error

Purchase(pid,product,price,quantity,month)

SQL Aside: 3-Valued Logic

↳ includes nulls

```
SELECT count(*) FROM Purchase
```

```
SELECT count(*)  FROM Purchase
WHERE price = 5.00 AND
  price < 5.00 AND price > 5.00
```

0

```
SELECT count(*) FROM Purchase
WHERE price = 5.00 OR
  price < 5.00 OR price > 5.00
```

everything but nulls

Is the result different between these? If so by how much?

```
SELECT count(*) FROM Purchase
WHERE price is NULL
```

Purchase(pid,product,price,quantity,month)

SQL Aside: 3-Valued Logic

- SQL has 3-valued logic

  FALSE = 0 (ex. price<25 is FALSE when price = 99)

  UNKNOWN = 0.5 (ex. price <25 is UNKNOWN when price=NULL)

  TRUE = 1 (ex. price<25 is TRUE when price = 19)

c1 AND c2	means min(c1,c2)
c1 OR c2	means max(c1,c2)
not c	means 1 - c

For SELECT ... FROM ... WHERE C do the following:

if C = TRUE then include the row in the output

if C = FALSE or C = unknown then do not include it

SQL Aside: 3-Valued Logic

Find employees who made more than 1200

Employ

Name	Salary	Bonus
Denise	1000	100
Larry	1000	500
Sharon	1800	NULL
Lawrence	1000	NULL

```
SELECT name , salary + bonus as take_home
FROM Employ WHERE take_home > 1200
```

Only return Larry (1800 + NULL > 1200 is UNKNOWN)

```
SELECT name , salary + coalesce(bonus,0) as take_home
From Employ WHERE take_home > 1200
```

SQL Aside: 3-Valued Logic

Employ

Name	Salary	Bonus
Denise	1000	100
Larry	1000	500
Sharon	1800	NULL
Lawrence	1000	NULL

Name	Salary	Bonus
Sharon	1800	NULL
Lawrence	1000	NULL
Larry	1000	500
Denise	1000	100

SELECT * from Employ Order By bonus asc, salary desc

The rule for ORDER By and GROUP BY is:

NULL values are the same (grouped together)

NULL values are less than anything else

Because SQL ¯_(ツ)_/¯

Purchase(pid,product,price,quantity,month)

Ordering Results

```
SELECT product, sum(price*quantity)
FROM   Purchase
GROUP BY product
ORDER BY sum(price*quantity) DESC
```

FWGOS

```
SELECT product, sum(price*quantity) as rev
FROM   Purchase
GROUP BY product
ORDER BY rev desc
```

Note: some SQL engines
want you to say ORDER BY sum(price*quantity)

Purchase(pid,product,price,quantity,month)

Filtering On Group By

Same query as earlier: (Products costing more than $1).
But now, only want ones with at least 30 sales.

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Purchase(pid,product,price,quantity,month)

HAVING Clause

Same query as earlier, except that we consider only products that had at least 30 sales.

```
SELECT      product, sum(price*quantity) as rev
FROM        Purchase
WHERE       price > 1
GROUP BY product
HAVING  rev > 30
ORDER BY rev
```

and rev > 30

Error FWGHOS

HAVING clause contains conditions on groups.

WHERE vs HAVING

- WHERE condition is applied to individual rows
  - The rows may or may not contribute to the aggregate
  - No aggregates allowed here
- HAVING condition is applied to the entire group
  - Entire group is returned, or not at all
  - May use aggregate functions in the group

Purchase(pid,product,price,quantity,month)

Exercise

Compute the total income per month displayed as "Rev"
Show only months with less than 10 items sold
Order by quantity sold and display as "Sold"

agg so no where

```
SELECT      month, sum(price*quantity) Rev,
            sum(quantity) as Sold
FROM        Purchase
GROUP BY    month
HAVING      Sold < 10
ORDER BY    sum(quantity)
```

FWGHOS

Purchase(pid,product,price,quantity,month)

Group By and Projection

```
SELECT      month, sum(quantity), max(price)
FROM        Purchase
GROUP BY    month
```

```
SELECT      month
FROM        Purchase
GROUP BY    month
```

```
SELECT DISTINCT month FROM Purchase
```

Lesson: DISTINCT is a special case of GROUP BY

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

JOINS and Multiple Atr GROUP BY

```
SELECT manufacturer, count(*)
FROM Product, Purchase
WHERE pname = product
GROUP BY  manufacturer
```

Let's figure out what these mean…

```
SELECT manufacturer, month, count(*)
FROM Product, Purchase
WHERE pname = product
GROUP BY  manufacturer, month
```

Nested Loop Semantics for SFW

```
SELECT  x1.a1, x2.a2, … xm.am
FROM    R1 as x1, R2 as x2, … Rm as xm
WHERE   Cond
```

```
for x1 in R1:
  for x2 in R2:
    ...
      for xm in Rm:
        if Cond(x1, x2…):
          output(x1.a1, x2.a2, … xm.am)
```

Nested loop semantics

Semantics for SFWGHO

SELECT	S
FROM	$R_1,\ldots,R_n$
WHERE	C1
GROUP BY	$g_1,\ldots,g_k$
HAVING	C2
ORDER BY	$a_1,\ldots,a_n$

Why ?

S = may contain attributes $g_1,\ldots,g_k$ and/or any aggregates but NO OTHER ATTRIBUTES

C1 = is any condition on the attributes in $R_1,\ldots,R_n$

C2 = is any condition on aggregate expressions and on attributes $g_1,\ldots,g_k$

Semantics for SFWGHO

SELECT	S
FROM	$R_1,\ldots,R_n$
WHERE	C1
GROUP BY	$g_1,\ldots,g_k$
HAVING	C2
ORDER BY	$a_1,\ldots,a_n$

Execution order:

FWGHOS

Evaluation steps:

1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes $a_1,\ldots,a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Aggregate + Join Example

What do these queries mean?

```
SELECT manufacturer, count(*)
FROM Product, Purchase
WHERE pname = product
GROUP BY  manufacturer
```

distinct products per manufacture

```
SELECT manufacturer, month, count(*)
FROM Product, Purchase
WHERE pname = product
GROUP BY  manufacturer, month
```

sum(quantity)

Empty Groups

- In the result of a group by query, there is one row per group in the result
- No group can be empty!
- In particular, count(*) is never 0

```
SELECT manufacturer, count(*)
FROM Product, Purchase
WHERE pname = product
GROUP BY manufacturer
```

What if there are no purchases for a manufacturer

Empty Group Solution: Outer Join

```
SELECT manufacturer, count(quantity)
FROM Product LEFT OUTER JOIN Purchase
ON pname = product
GROUP BY manufacturer
```

now include 0

Why count(quantity)? Why not count(*)?

Count(*) includes nulls

Purchase(~~pid~~,product,price,quantity,month)
Product(~~pid~~,pname,manufacturer)

Exercise 1

Find all manufacturers who have sold more than 10 items (of any product). Return manufacturer name and number of items sold (as sales).

Select manufacture, sum(quantity) as Sales,
From Purchase, Product
where product = pname
Group By manufactures
Having sales > 10;

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Exercise 1

Find all manufacturers who have sold more than 10 items (of any product). Return manufacturer name and number of items sold (as sales).

```
SELECT manufacturer, sum(quantity) as sales
FROM Product, Purchase
WHERE pname = product
GROUP BY manufacturer
HAVING sum(quantity) > 10
```

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Exercise 2

Find all manufacturers with more than 1 distinct product sold. Return the name of the manufacturer and number of distinct products sold

Select manufacture, count(distinct product) as products

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Exercise 2

Find all manufacturers with more than 1 distinct product sold. Return the name of the manufacturer and number of distinct products sold

```
SELECT manufacturer, count(distinct product)
FROM Product, Purchase
WHERE pname = product
GROUP BY manufacturer
HAVING count(distinct product) > 1
```

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Exercise 3

Find all products with more than 2 purchases.
Return the name of the product and max price it was sold

Purchase(pid,~~product~~,price,quantity,month)
Product(pid,pname,manufacturer)

if product not part of Purchase

Exercise 3

Find all products with more than 2 purchases.
Return the name of the product and max price it was sold

```
SELECT ~~product~~ pname, max(price)
FROM Purchase , Product
GROUP BY product
HAVING COUNT(*) > 2
```

where Pur.pid = Prod.pid

↳ not selection

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Exercise 4

Find all manufacturers with at least 5 purchases in one month
Return manufacturer name, month, and number of items sold

Purchase(pid,product,price,quantity,month)
Product(pid,pname,manufacturer)

Exercise 4

Find all manufacturers with at least 5 purchases in one month
Return manufacturer name, month, and number of items sold

```
SELECT manufacturer, month, sum(quantity) as sold
FROM Product, Purchase
GROUP BY manufacturer, month
HAVING count(*) > 5
```