Database Systems
CSE 414

Lectures 5: Grouping & Aggregation

Announcements

• HW1 is due next Monday, 11pm

Outline

• Last time:
  – outer joins
  – how to aggregate over all rows
• Grouping & aggregations (6.4.3 – 6.4.6)

Aggregation

Purchase(product, price, quantity)

Find number of bagels sold for more than $1

\[
\text{SELECT } \text{Sum(quantity) as TotalSold} \\
\text{FROM } \text{Purchase} \\
\text{WHERE } \text{price > 1 and product = 'bagel'}
\]

Grouping and Aggregation

Find number sold for more than $1 for each product

\[
\text{SELECT } \text{product, Sum(quantity)} \\
\text{FROM } \text{Purchase} \\
\text{WHERE } \text{price > 1} \\
\text{GROUP BY } \text{product}
\]

Let's see what this means…
1&2. FROM-WHERE-GROUPBY

```
<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>
```

WHERE price > 1

3. SELECT

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

<table>
<thead>
<tr>
<th>Product</th>
<th>Sum(quantity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>40</td>
</tr>
<tr>
<td>Banana</td>
<td>20</td>
</tr>
</tbody>
</table>

Other Examples

```
SELECT product, count(*) AS Count
FROM Purchase
GROUP BY product
```

Ordering Results

```
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) as rev
HAVING Clause

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

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

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

Lesson: DISTINCT is a special case of GROUP BY

Aggregates and Joins

```
create table Product(
  pid int primary key,
  pname varchar(15),
  manufacturer varchar(15));
insert into Product values(1,"bagel","Sunshine Co.");
insert into Product values(2,"banana","BusyHands");
insert into Product values(3,"gizmo","GizmoWorks");
insert into Product values(4,"gadget","BusyHands");
insert into Product values(5,"powerGizmo","PowerWorks");
```

Exercise

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

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

Mystery Query

What do they compute?

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

Aggregate + Join Example

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

Let's figure out what these mean...
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)

Semantics for SFWGH

```
SELECT S
FROM R1, ..., Rn
WHERE C1
GROUP BY a1, ..., ak
HAVING C2
```

Evaluation steps:
1. Evaluate FROM-WHERE using Nested Loop Semantics
2. Group by the attributes a1, ..., ak
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Aggregate + Join Example

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

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

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

Execution order:
```
FWGHOS
```

S = may contain attributes a1, ..., ak and/or any aggregates, but NO OTHER ATTRIBUTES
C1 = is any condition on the attributes in R1, ..., Rn
C2 = is any condition on aggregate expressions and on attributes a1, ..., ak
Empty Group Solution: Outer Join

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

Why not `count(*)`?

Exercise:

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

Find all manufacturers with more than 10 items sold.
Return manufacturer name and number of items sold.

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

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

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

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

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

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