Introduction to Database Systems
CSE 444

Lecture 03: SQL
April 4, 2008

Outline

• Subqueries (6.3)
• Aggregations (6.4.3 – 6.4.6)
• Examples, examples, examples…

Read the entire chapter 6!

Aggregation

```
SELECT avg(price)
FROM Product
WHERE maker="Toyota"
```

```
SELECT count(*)
FROM Product
WHERE year > 1995
```

SQL supports several aggregation operations:

sum, count, min, max, avg

Except count, all aggregations apply to a single attribute

Aggregation: Count

COUNT applies to duplicates, unless otherwise stated:

We probably want:

```
SELECT Count(DISTINCT category)
FROM Product
WHERE year > 1995
```
More Examples

Purchase(product, date, price, quantity)

```
SELECT Sum(price * quantity)
FROM Purchase
```

What do they mean?

```
SELECT Sum(price * quantity)
FROM Purchase
WHERE product = 'bagel'
```

Simple Aggregations

```
SELECT Sum(price * quantity)
FROM Purchase
WHERE product = 'bagel'
```

50 (= 20+30)

![Table of purchases](image)

<table>
<thead>
<tr>
<th>Product</th>
<th>Date</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>10/21</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>10/3</td>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>10/10</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Bagel</td>
<td>10/25</td>
<td>1.50</td>
<td>20</td>
</tr>
</tbody>
</table>

Grouping and Aggregation

Find total sales after 10/1/2005 per product.

```
SELECT product, Sum(price*quantity) AS TotalSales
FROM Purchase
WHERE date > '10/1/2005'
GROUP BY product
```

Let’s see what this means…

Grouping and Aggregation

1. Compute the FROM and WHERE clauses.
2. Group by the attributes in the GROUPBY
3. Compute the SELECT clause: grouped attributes and aggregates.
1&2. FROM-WHERE-GROUPBY

<table>
<thead>
<tr>
<th>Product</th>
<th>Date</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>10/21</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>10/25</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>10/3</td>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>10/10</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

3. SELECT

<table>
<thead>
<tr>
<th>Product</th>
<th>Date</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>10/21</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Bagel</td>
<td>10/25</td>
<td>1.50</td>
<td>20</td>
</tr>
<tr>
<td>Banana</td>
<td>10/3</td>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>Banana</td>
<td>10/10</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

```
SELECT product, Sum(price*quantity) AS TotalSales
FROM Purchase
WHERE date > '10/1/2005'
GROUP BY product
```

GROUP BY v.s. Nested Queries

```
SELECT product, Sum(price*quantity) AS TotalSales
FROM Purchase
WHERE date > '10/1/2005'
GROUP BY product
```

```
SELECT DISTINCT x.product, (SELECT Sum(y.price*y.quantity)
FROM Purchase y
WHERE x.product = y.product
AND y.date > '10/1/2005')
AS TotalSales
FROM Purchase x
WHERE x.date > '10/1/2005'
```

Another Example

```
SELECT product, sum(price * quantity) AS SumSales
max(quantity) AS MaxQuantity
FROM Purchase
GROUP BY product
```

What does it mean?
HAVING Clause

Same query, except that we consider only products that had at least 30 buyers.

```
SELECT product, Sum(price * quantity)
FROM Purchase
WHERE date > '10/1/2005'
GROUP BY product
HAVING Sum(quantity) > 30
```

HAVING clause contains conditions on aggregates.

General form of Grouping and Aggregation

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

Why?

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

Evaluation steps:
1. Evaluate FROM-WHERE, apply condition C1
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

Advanced SQLizing

1. Getting around INTERSECT and EXCEPT
2. Quantifiers
3. Aggregation v.s. subqueries
4. Two examples (study at home)
1. INTERSECT and EXCEPT:

If \( R, S \) have no duplicates, then can write without subqueries:

\[
\text{INTERSECT (SELECT } R.A, R.B \text{ FROM } R) \text{ INTERSECT (SELECT } S.A, S.B \text{ FROM } S) \]

\[
\text{EXCEPT (SELECT } R.A, R.B \text{ FROM } R) \text{ EXCEPT (SELECT } S.A, S.B \text{ FROM } S) \]

2. Quantifiers

Product ( \( \text{pname, price, company} \) )

Company ( \( \text{cname, city} \) )

Find all companies that make some products with price < 100

\[
\text{SELECT DISTINCT Company.cname FROM Company, Product WHERE Company.cname} = \text{Product.company and Product.price < 100}
\]

Existential: easy 😊

2. Quantifiers

1. Find the other companies: i.e. s.t. some product ≥ 100

\[
\text{SELECT DISTINCT Company.cname FROM Company WHERE Company.cname} \in (\text{SELECT Product.company FROM Product WHERE Product.price} \geq 100)
\]

Universal: hard 😊

2. Find all companies s.t. all their products have price < 100

\[
\text{SELECT DISTINCT Company.cname FROM Company WHERE Company.cname} \notin (\text{SELECT Product.company FROM Product WHERE Product.price} \geq 100)
\]
3. Group-by v.s. Nested Query

**Author**(login, name)

**Wrote**(login, url)

- Find authors who wrote ≥ 10 documents:
  - Attempt 1: with nested queries

```
SELECT DISTINCT Author.name
FROM Author
WHERE count(SELECT Wrote.url
             FROM Wrote
             WHERE Author.login=Wrote.login)
       > 10
```

This is SQL by a novice

- Attempt 2: SQL style (with GROUP BY)

```
SELECT Author.name
FROM Author, Wrote
WHERE Author.login=Wrote.login
GROUP BY Author.name
HAVING count(wrote.url) > 10
```

No need for DISTINCT: automatically from GROUP BY

3. Group-by v.s. Nested Query

**Author**(login, name)

**Wrote**(login, url)

**Mentions**(url, word)

Find authors with vocabulary ≥ 10000 words:

```
SELECT Author.name
FROM Author, Wrote, Mentions
WHERE Author.login=Wrote.login AND Wrote.url=Mentions.url
GROUP BY Author.name
HAVING count(distinct Mentions.word) > 10000
```

4. Two Examples

**Store**(sid, sname)

**Product**(pid, pname, price, sid)

Find all stores that sell *only* products with price > 100

same as:

Find all stores s.t. all their products have price > 100)
**Two Examples**

Store(sid, sname)  
Product(pid, pname, price, sid)

For each store, find its most expensive product

---

**Two Examples**

This is easy but doesn’t do what we want:

```
SELECT Store.sname, max(Product.price)  
FROM Store, Product  
WHERE Store.sid = Product.sid  
GROUP BY Store.sid, Store.sname
```

Better:

```
SELECT Store.sname, x.pname  
FROM Store, Product x  
WHERE Store.sid = x.sid and  
x.price >= ALL (SELECT y.price  
FROM Product y  
WHERE Store.sid = y.sid)  
GROUP BY Store.sname
```

But may return multiple product names per store

---

**Two Examples**

Finally, choose some pid arbitrarily, if there are many with highest price:

```
SELECT Store.sname, max(x.pname)  
FROM Store, Product x  
WHERE Store.sid = x.sid and  
x.price >= ALL (SELECT y.price  
FROM Product y  
WHERE Store.sid = y.sid)  
GROUP BY Store.sname
```