Lecture 03: SQL

Monday, January 9, 2006

Project

- Phase 0: form groups of two. 1/11
- Phase 1: design database. 1/25
- Phase 2: import data, provide logic. 2/8
- Phase 3: checkout logic. 2/22
- Phase 4: publish/consume XML data. 3/8
Outline

- Subqueries (6.3)
- Aggregations (6.4.3 – 6.4.6)

Read the entire chapter 6!

Suggestion:
“SQL for Nerds”: chapter 4, “More Complex queries”
(you will find it very useful for subqueries)

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:

```sql
SELECT Count(category) FROM Product WHERE year > 1995
```

same as Count(*)

We probably want:

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

More Examples

Purchase(product, date, price, quantity)

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

What do they mean?

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

<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>

\[
\text{SELECT } \text{Sum(price} \times \text{quantity)} \text{AS TotalSales}
\]

\[
\text{FROM } \text{Purchase}
\]

\[
\text{WHERE product = 'bagel'}
\]

\[= 50 \ (= 20+30)\]

Let’s see what this means…

Grouping and Aggregation

\[
\text{Purchase(product, date, price, quantity)}
\]

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

\[
\text{SELECT product, Sum(price*quantity) AS TotalSales}
\]

\[
\text{FROM Purchase}
\]

\[
\text{WHERE date > '10/1/2005'}
\]

\[
\text{GROUP BY product}
\]
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>

<table>
<thead>
<tr>
<th>Product</th>
<th>TotalSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>50</td>
</tr>
<tr>
<td>Banana</td>
<td>15</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 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 100 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 R_1, \ldots, R_n
WHERE C_1
GROUP BY a_1, \ldots, a_k
HAVING C_2

S = may contain attributes a_1, \ldots, a_k and/or any aggregates but NO OTHER ATTRIBUTES
C_1 = is any condition on the attributes in R_1, \ldots, R_n
C_2 = is any condition on aggregate expressions

General form of Grouping and Aggregation

Evaluation steps:
1. Evaluate FROM-WHERE, apply condition C_1
2. Group by the attributes a_1, \ldots, a_k
3. Apply condition C_2 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

1. INTERSECT and EXCEPT: not in SQL Server

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

\[
\begin{align*}
\text{(SELECT R.A, R.B FROM R) INTERSECT (SELECT S.A, S.B FROM S)} & \quad \text{SELECT R.A, R.B FROM R WHERE EXISTS(SELECT * FROM S WHERE R.A=S.A and R.B=S.B)} \\
\text{(SELECT R.A, R.B FROM R) EXCEPT (SELECT S.A, S.B FROM S)} & \quad \text{SELECT R.A, R.B FROM R WHERE NOT EXISTS(SELECT * FROM S WHERE R.A=S.A and R.B=S.B)}
\end{align*}
\]
2. Quantifiers

Product ( pname, price, company)
Company( cname, city)

Find all companies that make some products with price < 100

```
SELECT DISTINCT Company.cname
FROM Company, Product
WHERE Company.cname = Product.company and Product.price < 100
```

Existential: easy ! 😊

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

Universal: hard ! 😞
2. Quantifiers

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

   ```sql
   SELECT DISTINCT Company.cname
   FROM Company
   WHERE Company.cname IN (SELECT Product.company
                             FROM Product
                             WHERE Produc.price >= 100)
   ```

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

   ```sql
   SELECT DISTINCT Company.cname
   FROM Company
   WHERE Company.cname NOT IN (SELECT Product.company
                                  FROM Product
                                  WHERE Produc.price >= 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

   ```sql
   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
3. Group-by v.s. Nested Query

• Find all authors who wrote at least 10 documents:

• 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

This is SQL by an expert

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
```