

# Lecture 03: SQL

Monday, January 9, 2006

1

## Project

- <http://iisqlsrv.cs.washington.edu/444/Project/Default.aspx>
- 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

2

## 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)

3

## 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

4

## Aggregation: Count

COUNT applies to duplicates, unless otherwise stated:

```
SELECT Count(category)  same as Count(*)
FROM Product
WHERE year > 1995
```

We probably want:

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

5

## More Examples

Purchase(product, date, price, quantity)

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

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

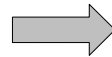
What do  
they mean ?

6

## Purchase Simple Aggregations

Product	Date	Price	Quantity
Bagel	10/21	1	20
Banana	10/3	0.5	10
Banana	10/10	1	10
Bagel	10/25	1.50	20

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



50 (= 20+30)

7

## Grouping and Aggregation

Purchase(product, date, price, quantity)

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...

8

# 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.

9

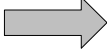
## 1&2. FROM-WHERE-GROUPBY

Product	Date	Price	Quantity
Bagel	10/21	1	20
Bagel	10/25	1.50	20
Banana	10/3	0.5	10
Banana	10/10	1	10

10

### 3. SELECT

Product	Date	Price	Quantity
Bagel	10/21	1	20
Bagel	10/25	1.50	20
Banana	10/3	0.5	10
Banana	10/10	1	10



Product	TotalSales
Bagel	50
Banana	15

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

11

### GROUP BY v.s. Nested Quereis

```
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

What does  
it mean ?

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

13

## 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.

14

## General form of Grouping and Aggregation

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

Why ?

S = may contain attributes  $a_1, \dots, a_k$  and/or any aggregates but NO OTHER ATTRIBUTES

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

C2 = is any condition on aggregate expressions

15

## General form of Grouping and Aggregation

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

Evaluation steps:

1. Evaluate FROM-WHERE, apply condition C1
2. Group by the attributes  $a_1, \dots, a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

16



# Advanced SQLizing

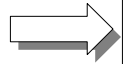
1. Getting around INTERSECT and EXCEPT
2. Quantifiers
3. Aggregation v.s. subqueries

17

INTERSECT and EXCEPT: not in SQL Server

## 1. INTERSECT and EXCEPT

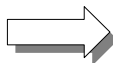
```
(SELECT R.A, R.B  
FROM R)  
INTERSECT  
(SELECT S.A, S.B  
FROM S)
```



```
SELECT R.A, R.B  
FROM R  
WHERE  
EXISTS(SELECT *  
FROM S  
WHERE R.A=S.A and R.B=S.B)
```

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

```
(SELECT R.A, R.B  
FROM R)  
EXCEPT  
(SELECT S.A, S.B  
FROM S)
```



```
SELECT R.A, R.B  
FROM R  
WHERE  
NOT EXISTS(SELECT *  
FROM S  
WHERE R.A=S.A and R.B=S.B)
```

18

## 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 ! 😊

19

## 2. Quantifiers

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

Find all companies that make only products with price < 100

same as:

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

Universal: hard ! 😞

20

## 2. Quantifiers

1. Find *the other* companies: i.e. s.t. some product  $\geq 100$

```
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  $< 100$

```
SELECT DISTINCT Company.cname
FROM Company
WHERE Company.cname NOT IN (SELECT Product.company
                            FROM Product
                            WHERE Produc.price >= 100)
```

21

## 3. Group-by v.s. Nested Query

Author(login,name)

Wrote(login,url)

- Find authors who wrote  $\geq 10$  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

22

### 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
```

This is  
SQL by  
an expert

No need for **DISTINCT**: automatically from **GROUP BY** 23

### 3. Group-by v.s. Nested Query

Author(login,name)

Wrote(login,url)

Mentions(url,word)

Find authors with vocabulary  $\geq 10000$  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
```

24