Introduction to Data Management CSE 344

Lectures 4 and 5: Aggregates in SQL

Magda Balazinska - CSE 344, Fall 2012

Announcements

- · Homework 1 is due tonight!
- Homework 2 is posted (due next week)
- · Quiz 1 due tonight!
- · Quiz 2 due next week

Magda Balazinska - CSE 344, Fall 2012

Outline

- Outer joins (6.3.8)
- Aggregations (6.4.3 6.4.6)
- Examples, examples, examples...

Magda Balazinska - CSE 344, Fall 2012

Outerjoins

Product(<u>name</u>, category) Purchase(prodName, store)

SELECT Product.name, Purchase.store

An "inner join": FROM Product, Purchase

WHERE Product.name = Purchase.prodName

Same as:

SELECT Product.name, Purchase.store FROM Product JOIN Purchase ON

Product.name = Purchase.prodName

But Products that never sold will be lost!

Magda Balazinska - CSE 344, Fall 2012

Outerjoins

Product(<u>name</u>, category) Purchase(prodName, store)

If we want the never-sold products, need an "outerjoin":

SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON
Product.name = Purchase.prodName

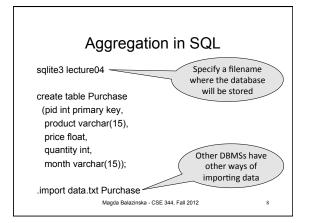
Magda Balazinska - CSE 344, Fall 2012

Product Purchase Category ProdName Store Gizmo Gizmo Wiz gadget Photo Camera Ritz OneClick Camera Wiz Store Gizmo Wiz Ritz Camera Wiz OneClick NII II I

Outer Joins

- · Left outer join:
 - Include the left tuple even if there's no match
- · Right outer join:
 - Include the right tuple even if there's no match
- · Full outer join:
 - Include both left and right tuples even if there's no match

Magda Balazinska - CSE 344, Fall 2012



Comment about SQLite

- One cannot load NULL values such that they are actually loaded as null values
- · So we need to use two steps:
 - Load null values using some type of special value
 - Update the special values to actual null values

update Purchase set price = null where price = 'null'

Magda Balazinska - CSE 344, Fall 2012

Simple Aggregations

Five basic aggregate operations in SQL

- select count(*) from Purchase
- · select count(quantity) from Purchase
- select sum(quantity) from Purchase
- · select avg(price) from Purchase
- select max(quantity) from Purchase
- select min(quantity) from Purchase

Except count, all aggregations apply to a single attribute

Magda Balazinska - CSE 344, Fall 2012

10

Aggregates and NULL Values

Null values are not used in aggregates

insert into Purchase values(12, 'gadget', NULL, NULL, 'april')

Let's try the following:

- select count(*) from Purchase
- select count(quantity) from Purchase
- select sum(quantity) from Purchase
- select sum(quantity) from Purchase where quantity is not null;

Magda Balazinska - CSE 344, Fall 2012

Counting Duplicates

COUNT applies to duplicates, unless otherwise stated:

SELECT Count(product)
FROM Purchase
WHERE price > 4.99

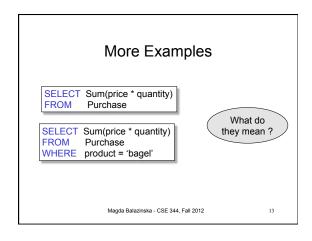
same as Count(*)

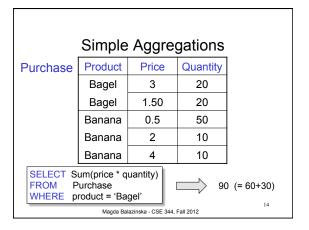
We probably want:

SELECT Count(DISTINCT product)
FROM Purchase
WHERE price> 4.99

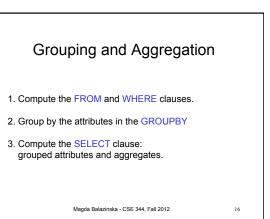
Magda Balazinska - CSE 344, Fall 2012

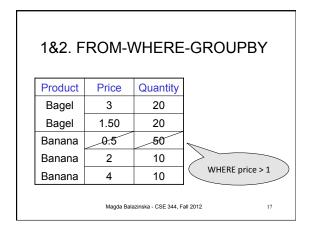
12

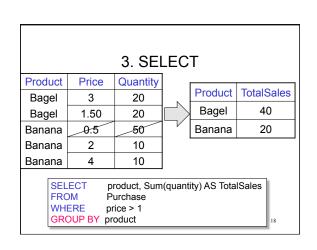


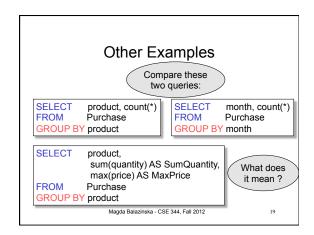


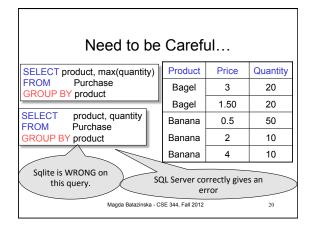












Ordering Results

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

Magda Balazinska - CSE 344, Fall 2012

23

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 aggregates.

Magda Balazinska - CSE 344, Fall 2012

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 al all
 - May use aggregate functions in the group

Magda Balazinska - CSE 344, Fall 2012

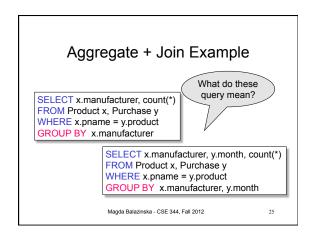
Aggregates and Joins

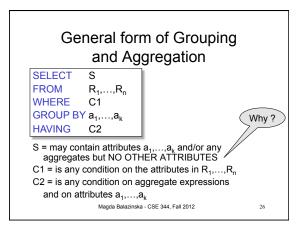
create table Product (pid int primary key, pname varchar(15), manufacturer varchar(15));

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');

Magda Balazinska - CSE 344, Fall 2012

4





Semantics of SQL With Group-By

 $\begin{array}{lll} \text{SELECT} & S \\ \text{FROM} & R_1, ..., R_n \\ \text{WHERE} & \text{C1} \\ \text{GROUP BY } a_1, ..., a_k \\ \text{HAVING} & \text{C2} \end{array}$

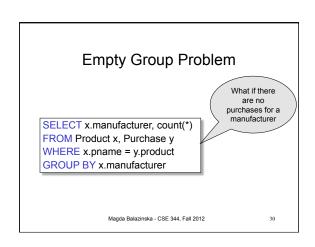
Evaluation steps:

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

Magda Balazinska - CSE 344, Fall 2012

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 x.manufacturer, count(*) FROM Product x, Purchase y WHERE x.pname = y.product

Empty Groups: Example SELECT product, count(*) FROM purchase GROUP BY product Select product, count(*) FROM purchase WHERE price > 2.0 GROUP BY product 3 groups in our example dataset Magda Balazinska - CSE 344, Fall 2012 29



Magda Balazinska - CSE 344, Fall 2012

GROUP BY x.manufacturer

Empty Group Solution: Outer Join

SELECT x.manufacturer, count(y.pid)
FROM Product x LEFT OUTER JOIN Purchase y
ON x.pname = y.product
GROUP BY x.manufacturer

Magda Balazinska - CSE 344, Fall 2012

21