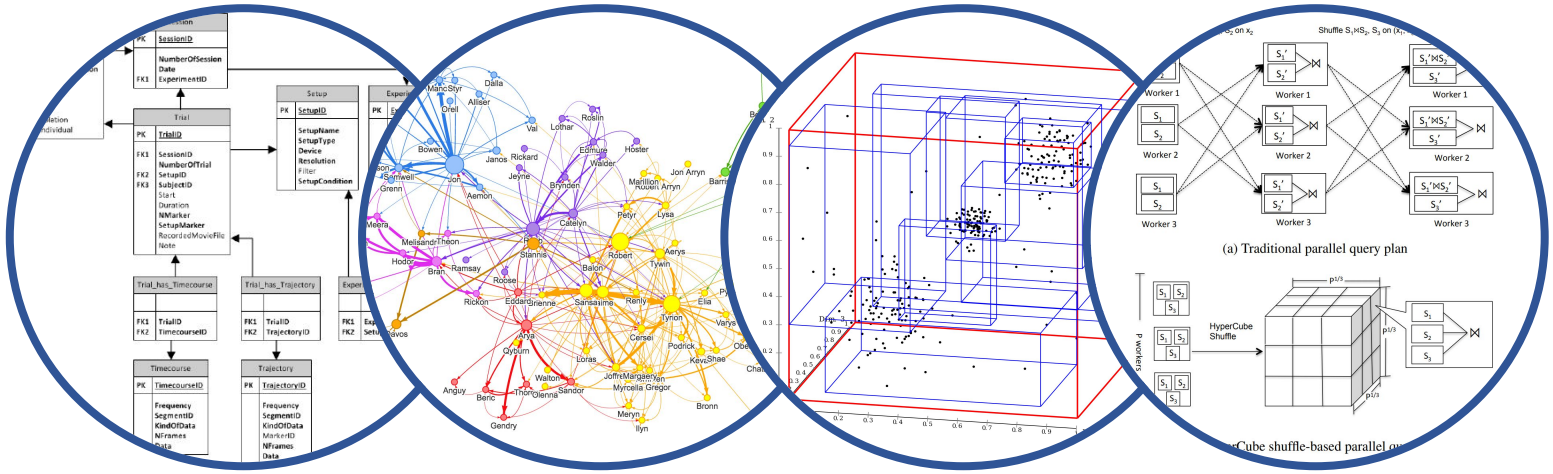

Introduction to Data Management

Grouping

Alyssa Pittman
Based on slides by Jonathan Leang, Dan Suciu, et al

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Announcements

- HW1 due tonight - push by 11 pm
- HW2 out today
  - Should be able to tackle the entire HW after this lecture

Recap - 3-valued logic

Comparisons with null result in *unknown*

false = 0

true = 1

unknown = .5

Formal definitions:

C1 AND C2 = min(C1,C2)

C1 OR C2 = max(C1,C2)

NOT C = 1 — C

Recap - Aggregate Functions

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - SELECT **COUNT**(*) FROM AnimeVideoViews …
  - SELECT **SUM**(cost) FROM CoffeeReceipts …
  - SELECT **AVG**(price) FROM CarDealers …
  - SELECT **MAX**(score) FROM StudentGrades …
  - SELECT **MIN**(price) FROM AveLunchPrices …

AGG(attr) □ computes **AGG** over non-NULL values
AGG(DISTINCT attr) is also possible

Recap - Aggregate Semantics

First evaluate the FROM clause

Next evaluate the WHERE clause

Last evaluate the SELECT clause

FWS

Goals for Today

- We have started to summarize results over our entire datasets
- Today we want to be able to summarize results for categories of data

Outline

- GROUP BY and HAVING clauses in SQL
- Augment our SQL query semantics
- Exercises

Table-wide aggregation

- We saw SQL aggregation over an entire table

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT SUM(quantity)
FROM Purchases
```

Table-wide aggregation

- We saw SQL aggregation over an entire table

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT SUM(quantity)
FROM Purchases
```

SUM(quantity)
110

Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a "whole-table" aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**

Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a "whole-table" aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a "whole-table" aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a "whole-table" aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Product	SUM(quantity)
Bagel	40
Banana	70

Different Groupings

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

```
SELECT Month,
       SUM(quantity)
FROM Purchases
GROUP BY Month
```

Different Groupings

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Product	SUM(quantity)
Bagel	40
Banana	70

```
SELECT Month,
       SUM(quantity)
FROM Purchases
GROUP BY Month
```

Month	SUM(quantity)
Jan	20
Feb	70
March	20

Grouping on Multiple Attributes

```
SELECT Product, Month, SUM(quantity)
FROM Purchases
GROUP BY Product, Month
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

Product	Month	SUM(quantity)
Bagel	Jan	20
Bagel	Feb	20
Banana	Feb	50
Banana	March	20

Selecting Multiple Aggregates

```
SELECT Product, SUM(quantity), MIN(price)
FROM Purchases
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

Product	SUM(quantity)	MIN(price)
Bagel	40	1.50
Banana	70	0.5

What does this mean?

```
SELECT Product, Price,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

What does this mean?

```
SELECT Product, Price,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

Product	Price	SUM(quantity)
Bagel	??	40
Banana	??	70

What does this mean?

```
SELECT Product, Price,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

DON'T DO THIS
sqlite will evaluate this and arbitrarily pick one of the prices
Other DBMSs will error

SELECT Clause Attributes

```
SELECT Product, Price,
       SUM(quantity)
FROM Purchases
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

Rule:
everything in SELECT clause needs to be either
- a group-by attribute or
- an aggregate

Filtering Rows with WHERE

- What if I only want to include some rows?

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT SUM(quantity)
FROM Purchases
WHERE Price > 1
GROUP BY Product
```

Filtering Rows with WHERE

- What if I only want to include some rows?

```
SELECT SUM(quantity)
FROM Purchases
WHERE Price > 1
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

Product	SUM(quantity)
Bagel	40
Banana	20

Filtering Rows with WHERE

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases

GROUP BY Product
```

```
SELECT Product,
       SUM(quantity)
FROM Purchases
WHERE Quantity = 20
GROUP BY Product
```

Filtering Rows with WHERE

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases

GROUP BY Product
```

Product	SUM(quantity)
Bagel	40
Banana	70

```
SELECT Product,
       SUM(quantity)
FROM Purchases
WHERE Quantity = 20
GROUP BY Product
```

Product	SUM(quantity)
Bagel	40

Filtering Rows with Where

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	2	10	March
Banana	4	10	March

```
SELECT Product,
       SUM(quantity)
FROM Purchases

GROUP BY Product
```

Product	SUM(quantity)
Bagel	40
Banana	70

```
SELECT Product,
                    ty)
FROM P
WHERE                20
GROUP BY Product
```

Empty groups are removed, so the results may have fewer groups

Product	SUM(quantity)
Bagel	40

GROUP BY Semantics

First evaluate the FROM clause

Next evaluate the WHERE clause

Group the attributes in the GROUPBY

Last evaluate the SELECT clause

FWGS

FWGS

```
SELECT Product, SUM(quantity)
FROM Purchases
WHERE Price < 4
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

FWGS

Purchases

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

```
FROM Purchases
```

FWGS

Purchases

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

WHERE `Price < 4`

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb

FROM `Purchases`

FWGS

Purchases

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

GROUP BY Product

Product	Price	Quantity	Month
Bagel	3	20	Jan
	1.50	20	Feb
Banana	0.5	50	Feb

WHERE Price < 4

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb

FROM Purchases

FWGS

Purchases

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

```
SELECT Product,
       SUM(quantity)
```

Product	SUM(quantity)
Bagel	40
Banana	50

```
GROUP BY Product
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
	1.50	20	Feb
Banana	0.5	50	Feb

```
WHERE Price < 4
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb

```
FROM Purchases
```

Filtering Groups with HAVING

- What if I only want to include some groups?

```
SELECT Product, SUM(quantity)
FROM Purchases
GROUP BY Product
HAVING SUM(quantity) > 20
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

Filtering Groups with HAVING

- What if I only want to include some groups?

```
SELECT Product, SUM(quantity)
FROM Purchases
GROUP BY Product
HAVING SUM(quantity) > 20
```

Product	Price	Quantity	Month
Bagel	3	20	Jan
Bagel	1.50	20	Feb
Banana	0.5	50	Feb
Banana	5	10	March
Apple	4	10	March

Product	SUM(quantity)
Bagel	40
Banana	60

HAVING Semantics

First evaluate the FROM clause

Next evaluate the WHERE clause

Group the attributes in the GROUPBY

Eliminate groups based on HAVING

Last evaluate the SELECT clause

FWGHS

Any More Semantics??

First evaluate the FROM clause

Next evaluate the WHERE clause

Group the attributes in the GROUPBY

Eliminate groups based on HAVING

Sort the results based on ORDER BY

Last evaluate the SELECT clause

FWGHOS™

General form of a GROUP BY query

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

S, O = any attributes $a_1, \ldots, a_k$ and/or any aggregates, but no other attributes

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

C2 = any condition on the aggregate expressions and attributes $a_1, \ldots, a_k$

Exercise

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

Product	Price	Quantity	Month

Exercise

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

Product	Price	Quantity	Month

```
FROM Product
```

Exercise

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

Product	Price	Quantity	Month

```
    FROM Product
GROUP BY Month
```

Exercise

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

Product	Price	Quantity	Month

```
   SELECT Month, SUM(Quantity*Price)

     FROM Product
GROUP BY Month
```

Exercise

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

Product	Price	Quantity	Month

```
   SELECT Month, SUM(Quantity*Price)

     FROM Product
 GROUP BY Month
   HAVING SUM(Quantity) < 10
```

Exercise

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

Product	Price	Quantity	Month

```
   SELECT Month, SUM(Quantity*Price),
          SUM(Quantity) as TotalSold
     FROM Product
 GROUP BY Month
   HAVING SUM(Quantity) < 10
 ORDER BY SUM(Quantity)
```

WHERE vs HAVING

WHERE condition applies to individual rows

- No aggregates allowed in the clause
- Occasionally, some groups become empty and are removed

HAVING condition applies to the entire group

- Entire group is returned or removed
- Aggregate functions allowed

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

UserID	Car	Year
123	Charger	2009
567	Civic	2016
567	Pinto	2000
789	Camry	2018

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Aggregate - COUNT and likely a GROUP

Attributes from two tables = JOIN

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

UserID	Car	Year
123	Charger	2009
567	Civic	2016
567	Pinto	2000
789	Camry	2018

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Step 1: think about the join

```
SELECT ...
  FROM Payroll p, Registry r
 WHERE p.UserID = r.UserID
```

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

UserID	Car	Year
123	Charger	2009
567	Civic	2016
567	Pinto	2000
789	Camry	2018

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Step 1: think about the join

```
SELECT ...
  FROM Payroll p, Registry r
 WHERE p.UserID = r.UserID
```

p.UserID	p.Name	p.Job	p.Salary	r.UserId	r.Car	r.Year
123	Jack	TA	50000	123	Charger	2009
567	Magda	Prof	90000	567	Civic	2016
567	Magda	Prof	90000	567	Pinto	2000
789	Dan	Prof	100000	789	Camry	2018

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Step 1: think about the join…and where

```
SELECT ...
  FROM Payroll p, Registry r
 WHERE p.UserID = r.UserID AND
       r.Year < 2017
```

p.UserID	p.Name	p.Job	p.Salary	r.UserId	r.Car	r.Year
123	Jack	TA	50000	123	Charger	2009
567	Magda	Prof	90000	567	Civic	2016
567	Magda	Prof	90000	567	Pinto	2000

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Step 1: think about the join…and where

Step 2: do the group-by on the join

```
   SELECT p.Name, COUNT(*)
     FROM Payroll p, Registry r
    WHERE p.UserID = r.UserID AND
          r.Year < 2017
 GROUP BY p.Name
```

p.UserID	p.Name	p.Job	p.Salary	r.UserId	r.Car	r.Year
123	Jack	TA	50000	123	Charger	2009
567	Magda	Prof	90000	567	Civic	2016
567	Magda	Prof	90000	567	Pinto	2000

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Step 1: think about the join…and where

Step 2: do the group-by on the join

```
   SELECT p.Name, COUNT(*) AS count
     FROM Payroll p, Registry r
    WHERE p.UserID = r.UserID AND
          r.Year < 2017
 GROUP BY p.Name
```

p.Name	count
Jack	1
Magda	2

Aggregates + Joins

Goal: how many cars made before 2017 does each person drive?

Step 1: think about the join…and where

Step 2: do the group-by on the join

```
SELECT p.Name, COUNT(*) AS count
  FROM Payroll p, Registry r
 WHERE p.UserID = r.UserID AND
       r.Year < 2017
GROUP BY p.UserID, p.Name
```

Probably want to group by UserID too, in case multiple people have the same name

p.Name	count
Jack	1
Magda	2

Including Empty Groups

Notice that empty groups were not included

- some people didn't have a car
- some people only had a new car

```
SELECT p.Name, COUNT(*) AS count
  FROM Payroll p, Registry r
 WHERE p.UserID = r.UserID AND
       r.Year < 2017
GROUP BY p.UserID, p.Name
```

p.Name	count
Jack	1
Magda	2

Including Empty Groups

Notice that empty groups were not included

- some people didn't have a car
- some people only ha ar

COUNT(*) will never be 0 for groups

```
SELECT p.Name, COUNT(*) AS count
  FROM Payroll p, Registry r
 WHERE p.UserID = r.UserID AND
       r.Year < 2017
GROUP BY p.UserID, p.Name
```

p.Name	count
Jack	1
Magda	2

Including Empty Groups

Remove our "older than 2017" constraint.

Now we want to include all people in the payroll, even if they don't have cars.

Any ideas for which type of join we could use?

```
   SELECT p.Name, COUNT(*) AS count
     FROM Payroll p, Registry r
    WHERE p.UserID = r.UserID
 GROUP BY p.UserID, p.Name
```

Including Empty Groups

```
SELECT p.Name, COUNT(r.UserID) AS count
  FROM Payroll p LEFT OUTER JOIN
       Registry r
    ON p.UserID = r.UserID
GROUP BY p.UserID, p.Name
```

p.UserID	p.Name	p.Job	p.Salary	r.UserId	r.Car	r.Year
123	Jack	TA	50000	123	Charger	2009
456	Allison	TA	60000	NULL	NULL	NULL
567	Magda	Prof	90000	567	Civic	2016
567	Magda	Prof	90000	567	Pinto	2000
789	Dan	Prof	100000	789	Camry	2018

Including Empty Groups

```
SELECT p.Name, COUNT(r.UserID) AS count
  FROM Payroll p LEFT OUTER JOIN
       Registry r
    ON p.UserID = r.UserID
GROUP BY p.UserID, p.Name
```

p.UserID	p.Name	p.Job	p.Salary	r.UserId	r.Car	r.Year
123	Jack	TA	50000	123	Charger	2009
456	Allison	TA	60000	NULL	NULL	NULL
567	Magda	Prof	90000	567	Civic	2016
567	Magda	Prof	90000	567	Pinto	2000
789	Dan	Prof	100000	789	Camry	2018

Including Empty Groups

```
SELECT p.Name, COUNT(r.UserID) AS count
  FROM Payroll p LEFT OUTER JOIN
       Registry r
    ON p.UserID = r.UserID
GROUP BY p.UserID, p.Name
```

p.Name	count
Jack	1
Allison	0
Magda	2
Dan	1

Including Empty Groups

COUNT(attr) excludes NULL, so can be 0

```
SELECT p.Name, COUNT(r.UserID) AS count
  FROM Payroll p LEFT OUTER JOIN
       Registry r
    ON p.UserID = r.UserID
GROUP BY p.UserID, p.Name
```

p.Name	count
Jack	1
Allison	0
Magda	2
Dan	1

Exercise

What do these compute?

```
   SELECT Month, SUM(Quantity), MAX(Price)
     FROM Product
GROUP BY Month
```

```
   SELECT Month, SUM(Quantity)
     FROM Product
GROUP BY Month
```

```
   SELECT Month
     FROM Product
GROUP BY Month
```

Exercise

What do these compute?

```
   SELECT Month, SUM(Quantity), MAX(Price)
     FROM Product
GROUP BY Month
```

```
   SELECT Month, SUM(Quantity)
     FROM Product
GROUP BY Month
```

```
   SELECT Month
     FROM Product
GROUP BY Month
```

DISTINCT is a special case of GROUP BY

Takeaways

- FWGHOS™
- Grouping allows us to answer questions about categories of data