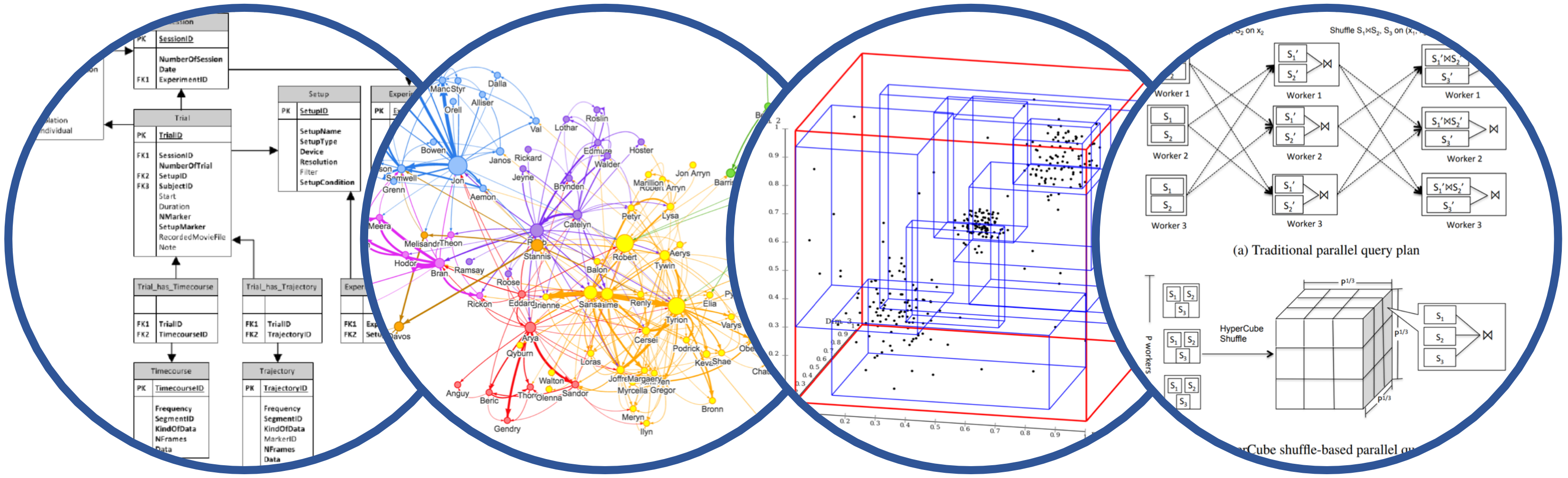


Introduction to Data Management

Grouping and Aggregates

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

Announcements

- HW 1 due today: tag by 11pm
- HW 2 releases tomorrow

  Complex joins and grouping

  Start early!

- No class on Friday

Recap

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

What am I aggregating over in a SELECT-FROM-WHERE query?

Answer:
The resulting tuples AFTER the join

There is an implicit "order of operations"

Order of operations

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

FROM → WHERE → ORDER BY → SELECT

"FWOS"

Order of operations

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

"FWOS"

$$\begin{tikzcd}
\text{SELECT} \\
\text{ORDER BY} \arrow[u] \\
\text{WHERE} \arrow[u] \\
\text{FROM} \arrow[u]
\end{tikzcd}$$

Aggregation Semantics

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

AVG(P.Salary)
76666

Aggregate (SELECT)

P.UserID	P.Name	P.Job	P.Salary	R.UserID	R.Car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto

Join (FROM → WHERE)

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

UserID	Car
123	Charger
567	Civic
567	Pinto

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**

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

```
…
GROUP BY Job
…
```

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**

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

```
…
GROUP BY Job
…
```

Grouping Example

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
```

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

Grouping Example

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
```

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

Job	MAX(Salary)
TA	60000
Prof	100000

Grouping on Multiple Attributes

```
SELECT Name, MAX(Salary)
  FROM Payroll
 GROUP BY Job, Name
```

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

Name	Salary
Jack	50000
Allison	60000
Magda	90000
Dan	100000

Filtering Groups with HAVING

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

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

Filtering Groups with HAVING

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

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

Job	MAX(Salary)
Prof	100000

Aggregation Order

How is aggregation processed internally?

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Aggregation Order

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Job	maxSal
Prof	100000

$$SELECT_{\ Job, maxSal}$$

Job	maxSal	minSal
Prof	100000	90000

$$HAVING\ minSal > 80000$$

Job	maxSal	minSal
TA	60000	50000
Prof	100000	90000

$$FROM\ \rightarrow GROUP\ BY\ Job$$

UserID	Name	Job	Salary
…	…	…	…

Order of operations

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

FROM → WHERE → ORDER BY → SELECT

"FWOS"

New keywords:

FWGHOS™

FROM → WHERE → GROUP BY → HAVING → ORDER BY → SELECT

The Witnessing Problem

- Also known as argmax/argmin
- Ex: Return the person with the highest salary for each job type

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

The Witnessing Problem

- Also known as argmax/argmin
- Ex: Return the person with the highest salary for each job type

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

Easy right?

```
  SELECT Name, MAX(Salary)
    FROM Payroll
GROUP BY Job
```

The Witnessing Problem

- Also known as argmax/argmin
- Ex: Return the person with the highest salary for each job type

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

Easy right?

```
SELECT Name, MAX(Salary)
  FROM Payroll
```

The Witnessing Problem

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

Name	MAX(Salary)
???	60000
???	100000

```
SELECT Name, MAX(Salary)
  FROM Payroll
 GROUP BY Job
```

The Witnessing Problem

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

Name	MAX(Salary)
???	60000
???	100000

```
SELECT Name, MAX(Salary)
  FROM Payroll
 GROUP BY Job
```

The Witnessing Problem

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

In SQLite:
… This works somehow. SQLite selects an arbitrary field

```
SELECT Name, MAX(Salary)
  FROM Payroll
 GROUP BY Job
```

The Witnessing Problem

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

In PostgreSQL or SQL Server:
"Failed to execute query. Error: Column 'Payroll.name' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause."

```
SELECT Name, MAX(Salary)
  FROM Payroll
GROUP BY Job
```

The Witnessing Problem

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

SELECT, HAVING, ORDER BY
Must use aggregate functions or attributes in GROUP BY

[illegible]me	MAX(Salary)
[illegible]	60000
???	100000

```
SELECT Name, MAX(Salary)
  FROM Payroll
GROUP BY Job
```

The Witnessing Problem

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

Return the person with the highest salary for each job type

How do we witness the maxima for a group?
Discuss!
Conceptual ideas are great

The Witnessing Problem

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

Return the person with the highest salary for each job type

Main idea: we need to join the respective maxima to each row

The Witnessing Problem

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

Return the person with the highest salary for each job type

Main idea: we need to join the respective maxima to each row

The Witnessing Problem

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

Return the person with the highest salary for each job type

Main idea: we need to join the respective maxima to each row

The Witnessing Problem

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

Return the person with the highest salary for each job type

```
SELECT P1.Name, MAX(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
 GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

The Witnessing Problem

```
SELECT P1.Name, MAX(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
 GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

Join on "original" grouping attributes

P1 P2

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

The Witnessing Problem

```
SELECT P1.Name, MAX(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
 GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

Group on additional attributes that you are argmax-ing for

P1 | P2

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

The Witnessing Problem

```
SELECT P1.Name, MAX(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

Group on additional attributes that you are argmax-ing for

P1 P2

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

The Witnessing Problem

```
SELECT P1.Name, MAX(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
 GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

P1 P2

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

The Witnessing Problem

```
SELECT P1.Name, MAX(P2.Salary)
  FROM Payroll AS P1, Payroll AS P2
 WHERE P1.Job = P2.Job
 GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

Name	MAX(Salary)
Allison	60000
Dan	100000

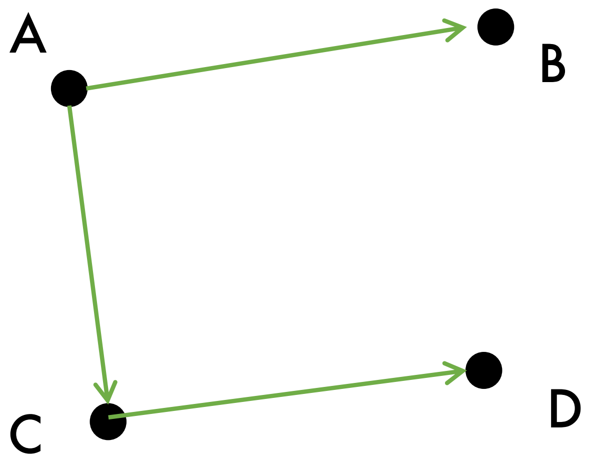
Self Join Example

Self Joins can be used
to find paths in a graph!

Very useful for HW 2

Self Join Example

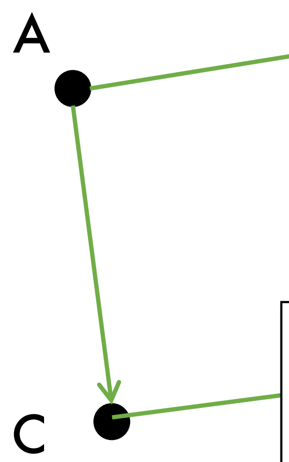
`Edge(start, end)`



start	end
A	B
A	C
C	D

Self Join Example

```
Edge(start, end)
```



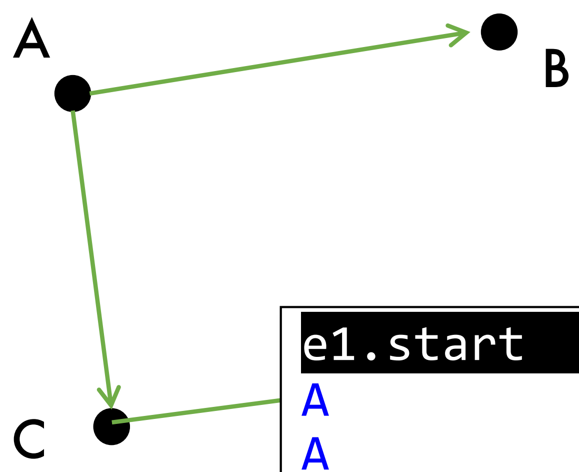
```
SELECT *
FROM   Edge e1, Edge e2
```

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C
C	D	C	D

start
A
A
C

Self Join Example

`Edge(start, end)`



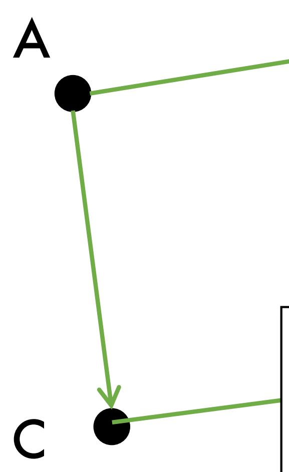
```
SELECT *
FROM   Edge e1, Edge e2
WHERE e1.end = e2.start
```

start
A
A
C

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C
C	D	C	D

Self Join Example

```
Edge(start, end)
```



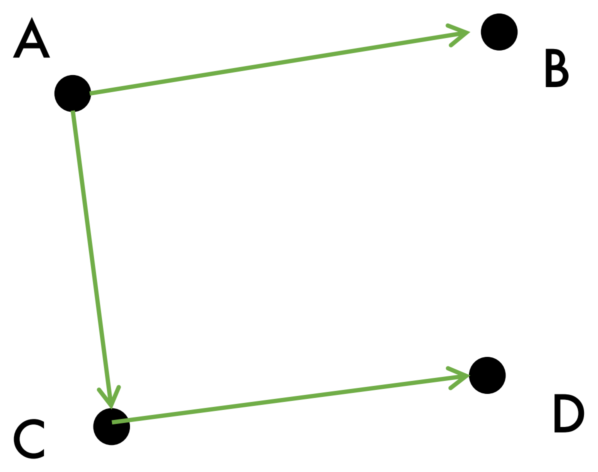
start
A
A
C

```
SELECT *
FROM   Edge e1, Edge e2
WHERE e1.end = e2.start
```

e1.start	e1.end	e2.start	e2.end
A	B	A	B
A	B	A	C
A	B	C	D
A	C	A	B
A	C	A	C
A	C	C	D
C	D	A	B
C	D	A	C
C	D	C	D

Self Join Example

```
Edge(start, end)
```



start	end
A	B
A	C
C	D

```
SELECT e1.start, e2.end
FROM   Edge e1, Edge e2
WHERE e1.end = e2.start
```

```
e1.start      e2.end
A             D
```