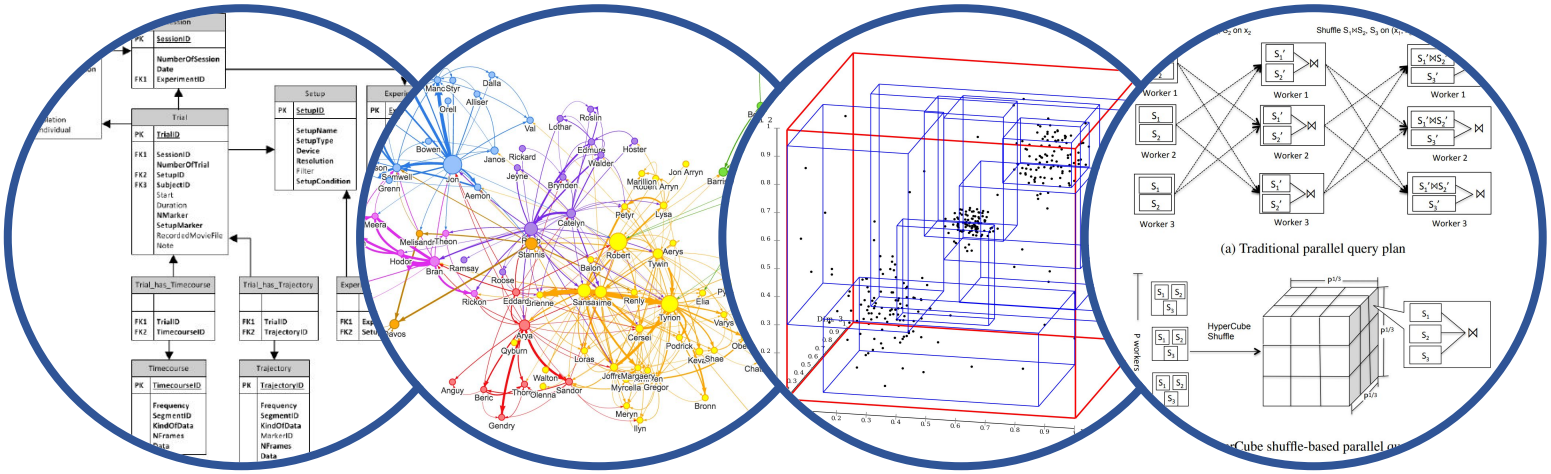


Introduction to Data Management

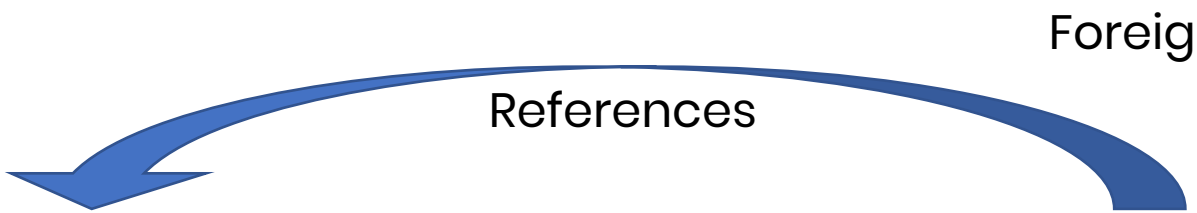
Aggregates

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

Recap – Keys and Foreign Keys

- Modeling multiple tables in the same database
  - Keys and foreign keys

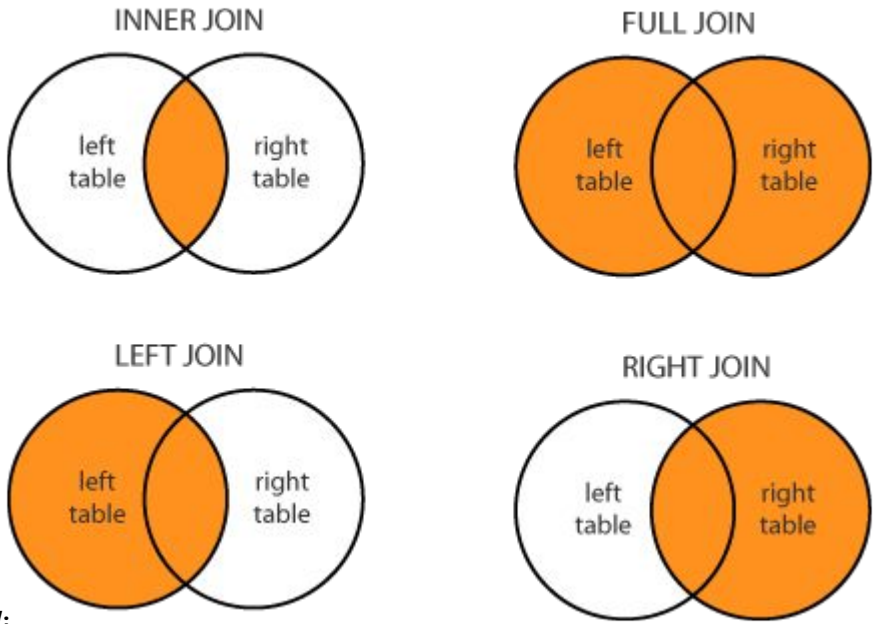


UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

UserID	Car
123	Charger
567	Civic
567	Pinto

Recap – Joins

- Join to combine data from different tables
  - Nested-loop semantics
  - Inner join (the most common)
  - Outer joins can preserve information
  - Self join pattern



https://www.dofactory.com/sql/join

Goals for Today

- We have started to build our SQL toolbox
  - Not just reading and filtering data anymore
  - Starting to answer complex questions
- Today we want to effectively summarize results

Outline

- Discussion of null values
- Aggregation functions
- More demo

Null review

- Real-world data often has missing values
- DBMSs often model missing data with null
- Null is a placeholder for
  - missing,
  - unknown,
  - non-applicable

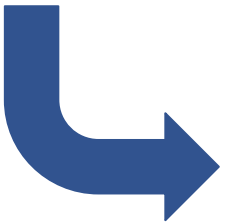
Null comparisons

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

```
SELECT *
  FROM Payroll
 WHERE Salary IS NULL
```

Null comparisons

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



UserID	Name	Job	Salary
123	Jack	TA	null
567	Magda	Prof	null

```
SELECT *
  FROM Payroll
 WHERE Salary IS NULL
```

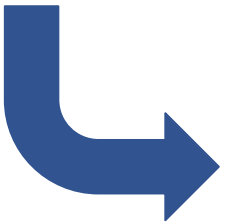
Null comparisons

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

```
SELECT *
  FROM Payroll
 WHERE Salary IS NOT NULL
```

Null comparisons

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

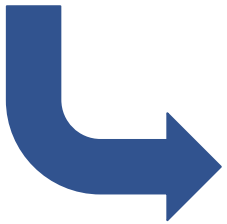


UserID	Name	Job	Salary
345	Allison	TA	60000
789	Dan	Prof	100000

```
SELECT *
  FROM Payroll
 WHERE Salary IS NOT NULL
```

Null comparisons

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



?

```
SELECT *
  FROM Payroll
 WHERE Salary > 70000
```

3-valued logic

- Null isn't true or false
- A comparison with null isn't true or false
  - It's a third value: unknown

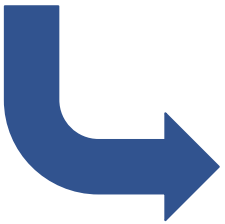
salary 60000 > 1000 ? → true

salary 900 > 1000? → false

salary null > 1000? → unknown

3-valued logic

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

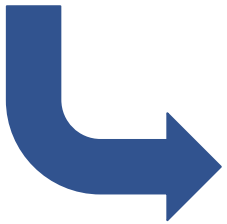


UserID	Name	Job	Salary
789	Dan	Prof	100000

```
SELECT *
  FROM Payroll
 WHERE Salary > 70000
```

3-valued logic

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



?

```
SELECT *
  FROM Payroll
 WHERE Salary > 70000 OR Salary <= 70000
```

3-valued logic

false = 0

true = 1

unknown = .5

Formal definitions:

C1 AND C2 = min(C1,C2)

C1 OR C2 = max(C1,C2)

NOT C = 1 — C

3-valued logic

false = 0

true = 1

unknown = .5

work out the truth table for AND....

x AND y	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

Formal definitions:

C1 AND C2 = min(C1,C2)

C1 OR C2 = max(C1,C2)

NOT C = 1 — C

3-valued logic

How do we use this in our queries?

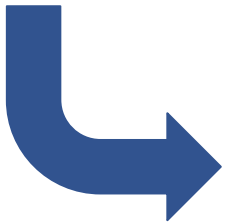
The rule for SELECT … FROM … WHERE … is the following:

if C = TRUE, then include the row in the output

if C = FALSE or UNKNOWN, then do not include it

3-valued logic

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



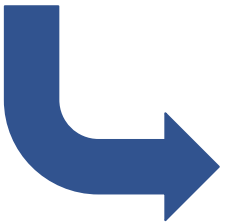
UserID	Name	Job	Salary
345	Allison	TA	60000
789	Dan	Prof	100000

```
SELECT *
  FROM Payroll
 WHERE Salary > 70000 OR Salary <= 70000
```

Always true? Nope!

3-valued logic

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



UserID	Name	Job	Salary
345	Allison	TA	60000
789	Dan	Prof	100000

```
SELECT  *
  FROM  Payroll
 WHERE  Salary = Salary
```

Also a weird one

Aggregates

A new form of SQL queries:

Aggregates

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - "How popular is this anime?"

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - "How popular is this anime?" □ COUNT

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - "How popular is this anime?" □ COUNT
  - "Do I spend too much on coffee?"

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this dealer?”

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this dealer?” □ AVG

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this dealer?” □ AVG
  - “Who got the highest grade in the class?”

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this dealer?” □ AVG
  - “Who got the highest grade in the class?” □ MAX

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this dealer?” □ AVG
  - “Who got the highest grade in the class?” □ MAX
  - “What’s the cheapest food on the Ave?”

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this anime?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this dealer?” □ AVG
  - “Who got the highest grade in the class?” □ MAX
  - “What’s the cheapest food on the Ave?” □ MIN

Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - COUNT
  - SUM
  - AVG
  - MAX
  - MIN

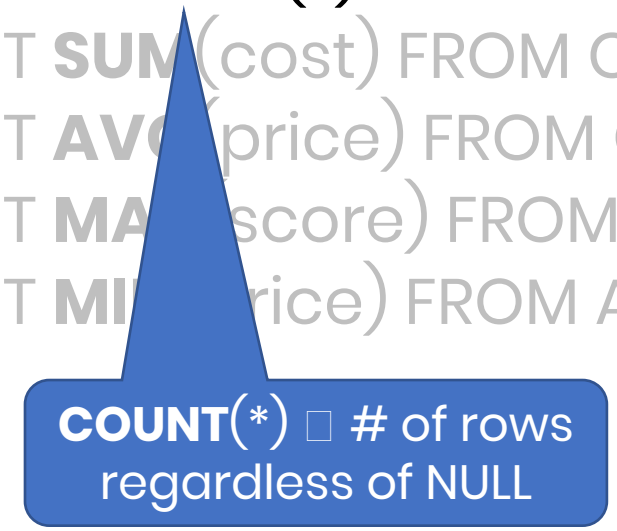
Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - COUNT
  - SUM
  - AVG
  - MAX
  - MIN

Very common functions found in DBMSs

Actionable Results

- 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 …



Actionable Results

- 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) □ computes AGG over distinct non-NULL values

Aggregation Semantics

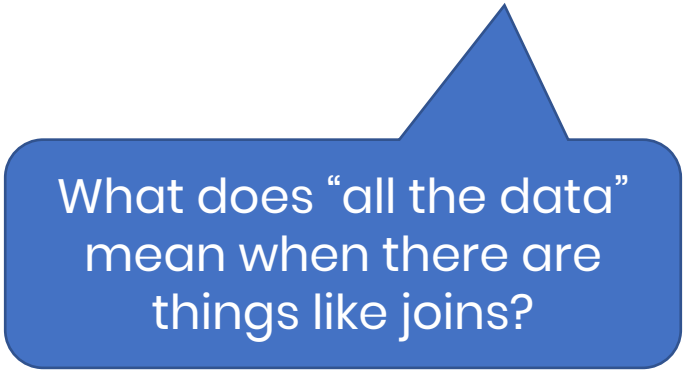
What am I aggregating over in a SELECT-FROM-WHERE query?

Intuitively: “all the data”

Aggregation Semantics

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

Intuitively: “all the data”



Aggregation Semantics

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

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

Payroll

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

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Aggregation Semantics

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

```
  FROM Payroll AS P, Regist AS 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
123	Charger
567	Civic
567	Pinto

Aggregation Semantics

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

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

```
FROM Payroll AS P, Regist AS 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
123	Charger
567	Civic
567	Pinto

Aggregation Semantics

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

```
SELECT AVG(P.Salary)
```

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

```
FROM Payroll AS P, Regist AS 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
123	Charger
567	Civic
567	Pinto

Aggregation Semantics

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

```
SELECT AVG(P.Salary)
```

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

```
FROM Payroll AS P, Regist AS 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
123	Charger
567	Civic
567	Pinto

Aggregation Semantics

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

AVG(P.Salary)
76666

```
SELECT AVG(P.Salary)
```

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

```
  FROM Payroll AS P, Regist AS 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
123	Charger
567	Civic
567	Pinto

Aggregation semantics

First evaluate the FROM clause

Next evaluate the WHERE clause

Last evaluate the SELECT clause

FWS

Aggregates and nulls

Null values aren't used in aggregates

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

```
SELECT COUNT(*) FROM Payroll;

SELECT COUNT(salary) FROM Payroll;

SELECT SUM(salary) FROM Payroll;

SELECT COUNT(*) FROM Payroll
  WHERE salary IS NOT NULL;
```

Aggregates and nulls

Null values aren't used in aggregates

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

```
SELECT COUNT(*) FROM Payroll; → 4

SELECT COUNT(salary) FROM Payroll;

SELECT SUM(salary) FROM Payroll;

SELECT COUNT(*) FROM Payroll
  WHERE salary IS NOT NULL;
```

Aggregates and nulls

Null values aren't used in aggregates

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

```
SELECT COUNT(*) FROM Payroll; → 4

SELECT COUNT(salary) FROM Payroll; → 2

SELECT SUM(salary) FROM Payroll;

SELECT COUNT(*) FROM Payroll
  WHERE salary IS NOT NULL;
```

Aggregates and nulls

Null values aren't used in aggregates

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

```
SELECT COUNT(*) FROM Payroll; → 4

SELECT COUNT(salary) FROM Payroll; → 2

SELECT SUM(salary) FROM Payroll; → 16000

SELECT COUNT(*) FROM Payroll
  WHERE salary IS NOT NULL;
```

Aggregates and nulls

Null values aren't used in aggregates

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

```
SELECT COUNT(*) FROM Payroll; → 4

SELECT COUNT(salary) FROM Payroll; → 2

SELECT SUM(salary) FROM Payroll; → 16000

SELECT COUNT(*) FROM Payroll
  WHERE salary IS NOT NULL; → 2
```

Aggregates and duplicates

Aggregates apply to duplicates…

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

```
SELECT COUNT(Job) FROM Payroll; → 4
```

….unless we tell them otherwise

```
SELECT COUNT(DISTINCT Job) FROM Payroll;
      → 2
```

Takeaways

- SQL nulls sometimes behave unintuitively
- Aggregation lets us summarize data
- FWS