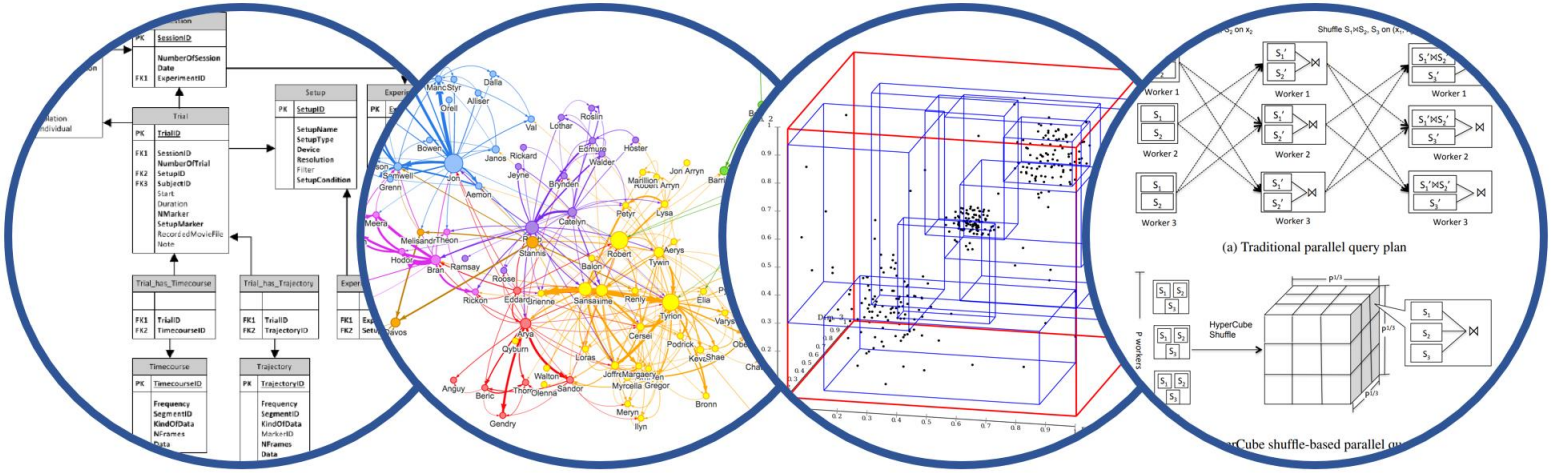


CSE 344: Intro to Data Management

SQL Subqueries

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

Announcements

- HW1 is graded: will be posted today.
- HW2 will probably be done early next week
- HW3 due on Wednesday, 10/23
- Midterm on Friday, 10/25 in class
  - Material up to date
  - Closed books, no cheat sheet (you won't need it)

Announcements (2/2)

No in-person lectures Monday&Wednesday next week!

- Lectures will be recorded: canvas→zoom
- Please watch the lectures

Predicates on Subqueries

- EXISTS (SELECT ….) checks if it is not empty
  NOT EXISTS (SELECT …) checks if it is empty

- X in (SELECT Y FROM …) checks output has X
  X not in (SELECT Y …) checks if it doesn't have X

- X > ALL(SELECT …)
  X > ANY(SELECT …)
  checks if X is > than one or all values in output

Recap: EXISTS

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist 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

Recap: EXISTS

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist 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

UserID	Name
123	Jack
567	Magda

Recap: EXISTS

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

Same as a semi-join

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

UserID	Name
123	Jack
567	Magda

Recap: NOT EXISTS

Find people who **do not** drive cars

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

Recap: NOT EXISTS

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist 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

Recap: NOT EXISTS

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID != R.UserID);
```

Does this work?

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

NO!
It returns everybody

Recap: NOT EXISTS

Find people who **do not** drive cars

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

Recap: NOT EXISTS

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist 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

Recap: NOT EXISTS

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist 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

UserID	Name
345	Allison
789	Dan

Unnesting EXISTS

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

```
SELECT DISTINCT P.UserID, P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID;
```

How do we unnest NOT EXISTS?

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

This doesn't work…

```
SELECT DISTINCT P.UserID, P.Name
FROM Payroll P, Regist R
WHERE P.UserID != R.UserID;
```

How do we unnest NOT EXISTS?

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

This query cannot be unnested without aggregates.

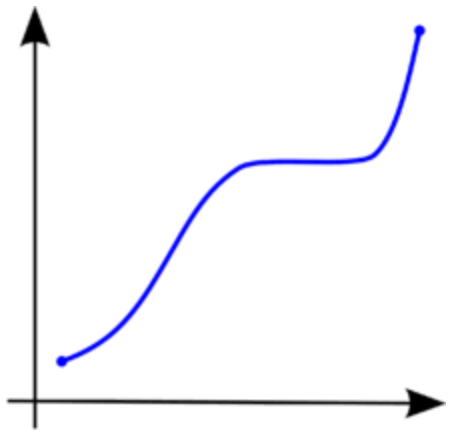
Proof next

Monotone Functions

Definition

A function $f: R \to R$ is monotone if $x \leq y$ implies $f(x) \leq f(y)$

Monotone:
$x^3 + x^2$,
e^x,
$\log(x)$,
…

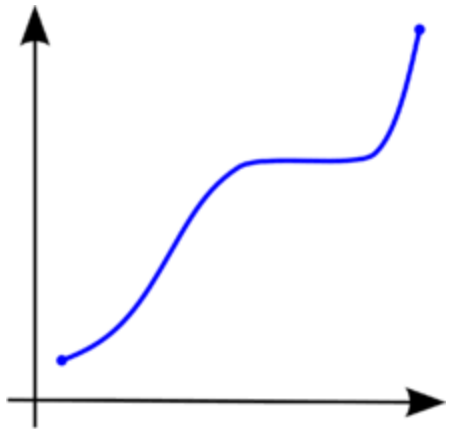


Non-Monotone:
$x^3 - x^2$,
e^{-x},
$\frac{1}{x}$,
…

Monotone Queries

Definition

A query Q is monotone if $I \subseteq J$ implies $q(I) \subseteq q(J)$

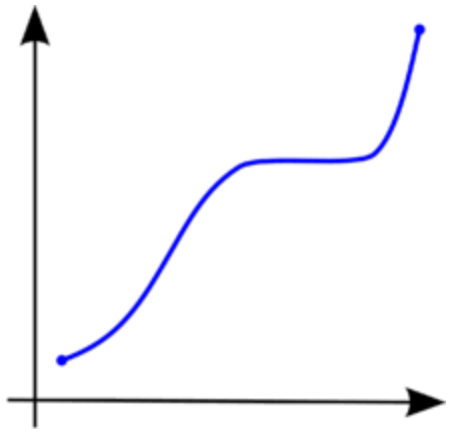


Monotone Queries

Definition

A query Q is monotone if $I \subseteq J$ implies $q(I) \subseteq q(J)$

Adding tuples to the input does
not remove tuples from the output



Monotone Queries

Find people who **do** drive cars

Is this query monotone?

Monotone Queries

Find people who **do** drive cars

Is this query monotone?

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

I

Monotone Queries

Find people who **do** drive cars

Is this query monotone?

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

I

UserID	Name
123	Jack
567	Magda

$q(I)$

Monotone Queries

Find people who **do** drive cars

Is this query monotone?

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

I

UserID	Name
123	Jack
567	Magda

$q(I)$

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
345	Tesla

J

Monotone Queries

Find people who **do** drive cars

Is this query monotone?

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

I

UserID	Name
123	Jack
567	Magda

$q(I)$

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
345	Tesla

J

UserID	Name
123	Jack
567	Magda
345	Allison

$q(J)$

Monotone Queries

Find people who **do** drive cars

Is this query monotone?

Yes, it is monotone

Monotone Queries

Find people who **do not** drive cars

Is this query monotone?

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

I

UserID	Name
345	Allison
789	Dan

$q(I)$

Monotone Queries

Find people who **do not** drive cars

Is this query monotone?

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

I

UserID	Name
345	Allison
789	Dan

$q(I)$

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
345	Tesla

J

Monotone Queries

Find people who **do not** drive cars

Is this query monotone?

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

I

UserID	Name
345	Allison
789	Dan

$q(I)$

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
345	Tesla

J

UserID	Name
~~345~~	~~Allison~~
789	Dan

$q(J)$

Monotone Queries

Find people who **do not** drive cars

Is this query monotone?

No, this query is not monotone

Monotone Queries

Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

Monotone Queries

Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

Proof. Consider a SQL query:

```
SELECT attrs
FROM T1, T2, ...
WHERE condition
```

Monotone Queries

Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

Proof. Consider a SQL query:

```
SELECT attrs
FROM T1, T2, ...
WHERE condition
```

Its nested loop semantics is:

```
for each r1 in T1:
 for each t2 in T2:
   for each t3 in T3:
      …
       if (condition):
         output (r1,r2,…)
```

If we insert a tuple into one of the input relations T_i, we will not remove any tuples from the output.

Monotone Queries

Consequence

The query “Find people who **do not** drive cars”
cannot be unnested without aggregates

Discussion

- The property whether the query is monotone or not does not depend on its SQL writeup

- Instead, it depends on the meaning of the query, regardless of how we write it in SQL

Monotone Queries

Count the number of employees.

```
SELECT count(*)
FROM Payroll
```

Monotone Queries

Count the number of employees.

```
SELECT count(*)
FROM Payroll
```

Monotone?

Payroll

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

I

count
4

$q(I)$

Monotone Queries

Count the number of employees.

```
SELECT count(*)
FROM Payroll
```

Monotone?

Payroll

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

I

count
4

$q(I)$

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000
555	Alice	TA	80000

J

count
5

$q(J)$

Monotone Queries

Count the number of employees.

```
SELECT count(*)
FROM Payroll
```

Monotone?

Payroll

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

I

count
4

$q(I)$

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000
555	Alice	TA	80000

J

count
5

$q(J)$

$\{4\} \not\subseteq \{5\}$

Monotone Queries

Count the number of employees.

```
SELECT count(*)
FROM Payroll
```

Monotone?

Payroll

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

I

count
4

$q(I)$

Not monotone

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000
555	Alice	TA	80000

J

count
5

$q(J)$

$\{4\} \not\subseteq \{5\}$

IN and NOT IN

Subqueries in WHERE/HAVING

X in (SELECT Y FROM …)

- Compute the subquery
- Check if $X \in Output$

Y
…
…

Subqueries in WHERE/HAVING

X in (SELECT Y FROM …)

- Compute the subquery
- Check if $X \in Output$

Y
…
…

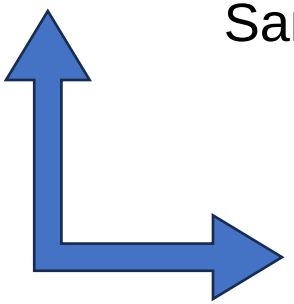
X not in (SELECT Y …) **not(X in (SELECT Y …))**

- Compute the subquery
- Check if $X \notin Output$

EXISTS v.s. IN

Find people who **do** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```



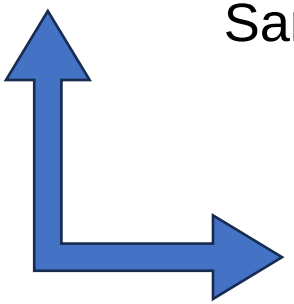
Same output

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID in
        (SELECT R.UserID
         FROM Regist R);
```

NOT EXISTS v.s. NOT IN

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```



Same output

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID not in
        (SELECT R.UserID
         FROM Regist R);
```

Computing NOT IN

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID not in
        (SELECT R.UserID
         FROM Regist R);
```

1. Compute subquery

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

Computing NOT IN

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID not in
        (SELECT R.UserID
         FROM Regist R);
```

1. Compute subquery

UserID
123
567
567

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

Computing NOT IN

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID not in
        (SELECT R.UserID
         FROM Regist R);
```

1. Compute subquery

UserID
123
567
567

2. For each Payroll, check if UserID ∉ subquery

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

UserID	Name
345	Allison
789	Dan

NOT EXISTS v.s. NOT IN

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

Which query is more efficient?

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID not in
        (SELECT R.UserID
         FROM Regist R);
```

NOT EXISTS v.s. NOT IN

Find people who **do not** drive cars

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE not exists
        (SELECT *
         FROM Regist R
         WHERE P.UserID = R.UserID);
```

Correlated subquery: computed repeatedly, once for each Payroll

Which query is more efficient?

Computed only once

```
SELECT P.UserID, P.Name
FROM Payroll P
WHERE P.UserID not in
        (SELECT R.UserID
         FROM Regist R);
```

ANY and ALL

ANY and ALL

< or <= or > or …

X < ANY (SELECT Y FROM …)

- Compute the subquery
- Check if there exists $Y \in Output$ s.t. X < Y

Y
…
…

ANY and ALL

< or <= or > or …

X < ANY (SELECT Y FROM …)

- Compute the subquery
- Check if there exists $Y \in Output$ s.t. X < Y

Y
…
…

X < ALL (SELECT Y FROM …)

- Compute the subquery
- Check if for all $Y \in Output$, X < Y

ANY and ALL

Find people who drive some car made after 2017

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

ANY and ALL

Find people who drive some car made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ANY(SELECT R.Year
     FROM Regist 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	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

ANY and ALL

Find people who drive some car made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ANY(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

Jack:

Year
2016
2018

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Jack	

ANY and ALL

Find people who drive some car made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ANY(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

Allison:

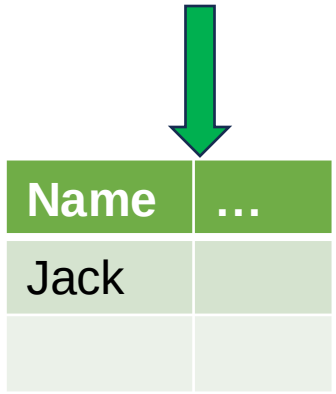
Year

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022



ANY and ALL

Find people who drive some car made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ANY(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

Magda:

Year
2020
2022

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Jack	
Magda	

ANY and ALL

Find people who drive some car made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ANY(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

Dan:

Year

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Jack	
Magda	

ANY and ALL

Find people who drive some car made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ANY(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

Same as a semi-join

```
SELECT DISTINCT P.*
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and R.Year > 2017
```

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Jack	
Magda	

ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist 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	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Jack:

Year
2016
2018

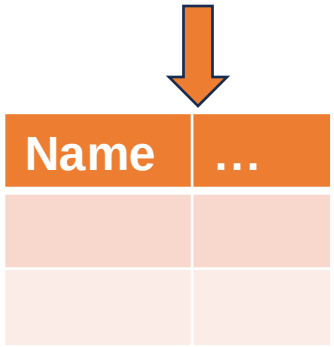
Do we output Jack?

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022



ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Jack:

Year
2016
2018

Do we output Jack?

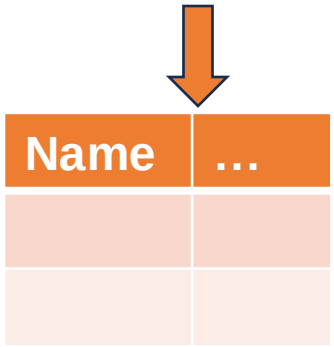
The test 2017 < ALL(…) fails for 2016

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022



ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Allison:

Year

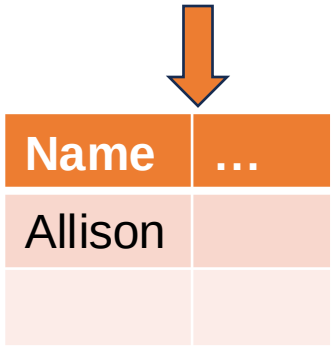
Do we output Allison?

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022



ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Allison:

Year

Do we output Allison?

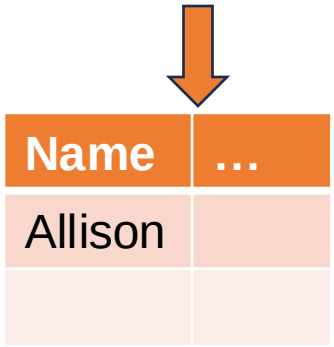
The test 2017 < ALL(…) does not fail anywhere. Hence it holds!!

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022



ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Magda:

Year
2020
2022

The test 2017 < ALL(…) does not fail anywhere.

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Allison	
Magda	

ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Dan:

Year

The test 2017 < ALL(…) does not fail anywhere. Hence it holds!!

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Allison	
Magda	
Dan	

ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Can we unnest this query?

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Allison	
Magda	
Dan	

ANY and ALL

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
   ALL(SELECT R.Year
       FROM Regist R
       WHERE P.UserID = R.UserID);
```

Can we unnest this query?

NO!
Non-monotone

Payroll

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

Regist

UserID	Car	Year
123	Charger	2016
123	Tesla	2018
567	Civic	2020
567	Pinto	2022

Name	…
Allison	
Magda	
Dan	

Recap: Predicates on Subqueries

- EXISTS / NOT EXISTS
- IN / NOT IN
- ANY / ALL

The are “equivalent” meaning that a query that you can write using one, you can also write using the others

They express QUANTIFIERS

Quantifiers

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ALL(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

Quantifiers

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ALL(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

```
SELECT P.*
FROM Payroll P
WHERE NOT EXISTS
  (SELECT *
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

Quantifiers

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ALL(SELECT R.Year
      FROM Regist R
      WHERE P.UserID = R.UserID);
```

```
SELECT P.*
FROM Payroll P
WHERE NOT EXISTS
  (SELECT *
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

```
SELECT P.*
FROM Payroll P
WHERE P.UserID NOT IN
     (SELECT R.UserID
      FROM Regist R
      WHERE R.Year <= 2017);
```

Quantifiers

Find people who drive only cars made after 2017

```
SELECT P.*
FROM Payroll P
WHERE 2017 <
 ALL(SELECT R.Year
     FROM Regist R
     WHERE P.UserID = R.UserID);
```

All these compute the same thing

```
SELECT P.*
FROM Payroll P
WHERE NOT EXISTS
  (SELECT *
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

```
SELECT P.*
FROM Payroll P
WHERE P.UserID NOT IN
     (SELECT R.UserID
      FROM Regist R
      WHERE R.Year <= 2017);
```

Discussion

- SQL can express naturally queries that represent existential quantifiers

- To write a query that uses a universal quantifier, use DeMorgan’s laws (next)

Quantifiers

Quantifiers

There are two types of quantifiers:

- **Exists** ($\exists x$, ...) there is at least 1 that satisfies predicate
- **Forall**: ($\forall x$, ...) all elements satisfy the predicate

Quantifiers

There are two types of quantifiers:

- **Exists** ($\exists x$, ...) there is at least 1 that satisfies predicate
- **Forall**: ($\forall x$, ...) all elements satisfy the predicate

SQL makes it easy to write **exists**

Quantifiers

There are two types of quantifiers:

- **Exists** ($\exists x$, ...) there is at least 1 that satisfies predicate
- **Forall**: ($\forall x$, ...) all elements satisfy the predicate

SQL makes it easy to write **exists**

To write **forall**, use double negation

predicate holds **forall** elements
if and only if
not (**exists** element where **not**(predicate) holds)

Using First Order Logic

Query: persons **P** that drive only cars made after 2017:

$$\forall \mathrm{R} \in \mathrm{Regist}, (\mathbf{P}.\mathrm{UserID} = \mathrm{R}.\mathrm{UserID}) \Rightarrow (\mathrm{R}.\mathrm{Year} > 2017)$$

Using First Order Logic

Query: persons **P** that drive only cars made after 2017:

$$\forall R \in \text{Regist}, (\mathbf{P}.\text{UserID} = \text{R}.\text{UserID}) \Rightarrow (\text{R}.\text{Year} > 2017)$$

Negation: persons **P** that drive some car made on/before 2017:

$$\exists R \in \text{Regist}, (\mathbf{P}.\text{UserID} = \text{R}.\text{UserID}) \text{ and } (\text{R}.\text{Year} \leq 2017)$$

Using First Order Logic

Query: persons **P** that drive only cars made after 2017:

$$\forall R \in \text{Regist}, (\mathbf{P}.\text{UserID} = \text{R}.\text{UserID}) \Rightarrow (\text{R}.\text{Year} > 2017)$$

Negation: persons **P** that drive some car made on/before 2017:

$$\exists R \in \text{Regist}, (\mathbf{P}.\text{UserID} = \text{R}.\text{UserID}) \text{ and } (\text{R}.\text{Year} \leq 2017)$$

Let's review this slowly

Brief Review of Logic

- Implication: A→B is same as: not(A) or B

Brief Review of Logic

- Implication: A→B is same as: not(A) or B

- DeMorgan's Laws:

$$\text{not}(\text{A and B}) = \text{not}(\text{A}) \text{ or } \text{not}(\text{B})$$
$$\text{not}(\text{A or B}) = \text{not}(\text{A}) \text{ and } \text{not}(\text{B})$$

Brief Review of Logic

- Implication: A→B is same as: not(A) or B

- DeMorgan's Laws:

$$\text{not}(\text{A and B}) = \text{not}(\text{A}) \text{ or } \text{not}(\text{B})$$
$$\text{not}(\text{A or B}) = \text{not}(\text{A}) \text{ and } \text{not}(\text{B})$$

$$\text{not}\big(\exists x, P(x)\big) = \forall x, \text{not}\big(P(x)\big)$$
$$\text{not}\big(\forall x, P(x)\big) = \exists x, \text{not}\big(P(x)\big)$$

Brief Review of Logic

- Implication: A→B is same as: not(A) or B

- DeMorgan's Laws:

$$\text{not}(\text{A and B}) = \text{not}(\text{A}) \text{ or } \text{not}(\text{B})$$
$$\text{not}(\text{A or B}) = \text{not}(\text{A}) \text{ and } \text{not}(\text{B})$$

$$\text{not}\big(\exists x, \text{P}(x)\big) = \forall x, \text{not}\big(\text{P}(x)\big)$$
$$\text{not}\big(\forall x, \text{P}(x)\big) = \exists x, \text{not}\big(\text{P}(x)\big)$$

- Consequences

$$\text{not}(\text{A} \rightarrow \text{B}) = (\text{A and not}(\text{B}))$$

Brief Review of Logic

- Implication: $A \rightarrow B$ is same as: not(A) or B

- DeMorgan's Laws:

$$\text{not}(A \text{ and } B) = \text{not}(A) \text{ or not}(B)$$
$$\text{not}(A \text{ or } B) = \text{not}(A) \text{ and not}(B)$$

$$\text{not}(\exists x, P(x)) = \forall x, \text{not}(P(x))$$
$$\text{not}(\forall x, P(x)) = \exists x, \text{not}(P(x))$$

- Consequences

$$\text{not}(A \rightarrow B) = (A \text{ and not}(B))$$

$$\text{not}(\forall x, (A(x) \rightarrow B(x))) = \exists x, (A(x) \wedge \text{not}(B(x)))$$

Using First Order Logic

Query: persons **P** that drive only cars made after 2017:

$$\forall \mathrm{R} \in \mathrm{Regist}, (\mathbf{P}.\mathrm{UserID} = \mathrm{R}.\mathrm{UserID}) \Rightarrow (\mathrm{R}.\mathrm{Year} > 2017)$$

Negation: persons **P** that drive some car made on/before 2017:

$$\exists \mathrm{R} \in \mathrm{Regist}, (\mathbf{P}.\mathrm{UserID} = \mathrm{R}.\mathrm{UserID}) \text{ and } (\mathrm{R}.\mathrm{Year} \leq 2017)$$

How to Write FORALL in SQL

Find person **P** drives **only** cars made after 2017

How to Write FORALL in SQL

Find person **P** drives **only** cars made after 2017

Negate: find the other persons

Find person **P** drives **some** car made on or before 2017

How to Write FORALL in SQL

Find person **P** drives **only** cars made after 2017

Negate: find the other persons

Find person **P** drives **some** car made on or before 2017

```
SELECT P.*
FROM Payroll P
WHERE EXISTS
  (SELECT R.Year
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

How to Write FORALL in SQL

Find person **P** drives **only** cars made after 2017

Negate: find the other persons
Find person **P** drives **some** car made on or before 2017

```
SELECT P.*
FROM Payroll P
WHERE EXISTS
  (SELECT R.Year
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

Negate again:
find the other other persons

```
SELECT P.*
FROM Payroll P
WHERE NOT EXISTS
  (SELECT R.Year
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

How to Write FORALL in SQL

Find person **P** drives **only** cars made after 2017

Negate: find the other persons

Find person **P** drives **some** car made on or before 2017

```
SELECT P.*
FROM Payroll P
WHERE EXISTS
  (SELECT R.Year
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

Negate again:
find the other other persons

```
SELECT P.*
FROM Payroll P
WHERE NOT EXISTS
  (SELECT R.Year
   FROM Regist R
   WHERE P.UserID = R.UserID
     and R.Year <= 2017);
```

Discussion

Writing universally quantified queries in SQL requires creativity

- Try using DeMorgan's laws: not exists, not in
- Try using ALL
- Try using aggregates, checking count=0