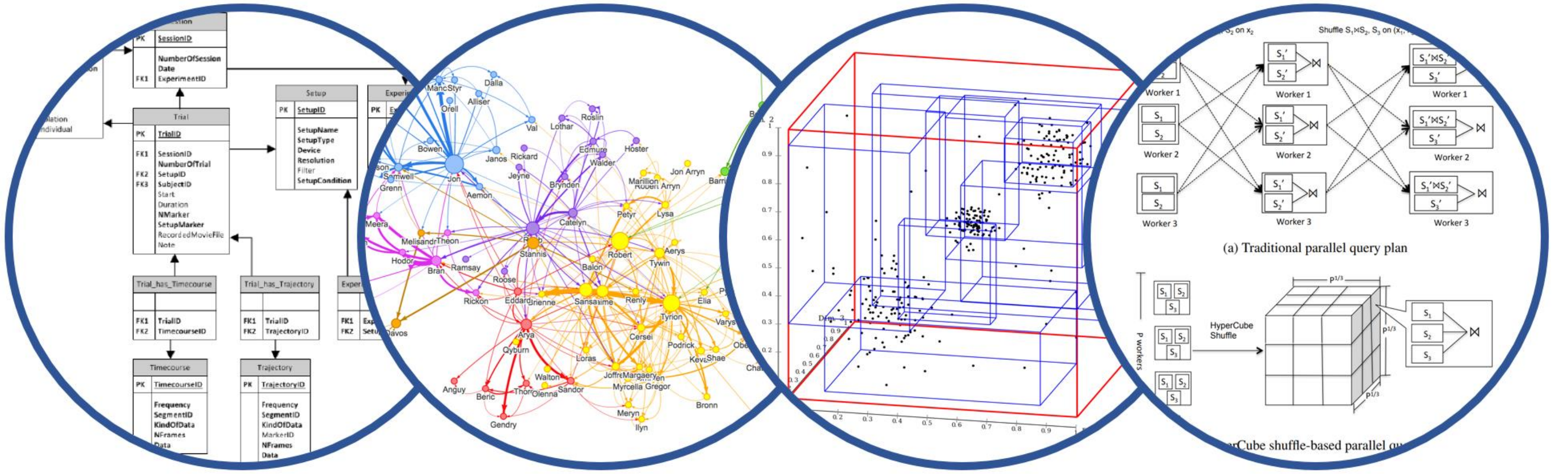


CSE 344: Intro to Data Management

Joining Tables

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

Recap: Keys and Foreign Keys

Foreign Key

A **Key** is one or more attributes that **uniquely** identify a row.
A **Foreign Key** is one or more attrs that uniquely identify a row in *another table*.

Recap: Keys and Foreign Keys

Foreign Key

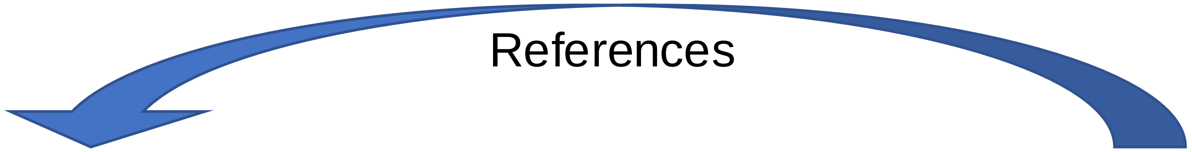
A **Key** is one or more attributes that **uniquely** identify a row.
A **Foreign Key** is one or more attrs that uniquely identify a row in *another table*.

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Recap: Keys and Foreign Keys

Foreign Key

A **Key** is one or more attributes that **uniquely** identify a row.
A **Foreign Key** is one or more attrs that uniquely identify a row in *another table*.



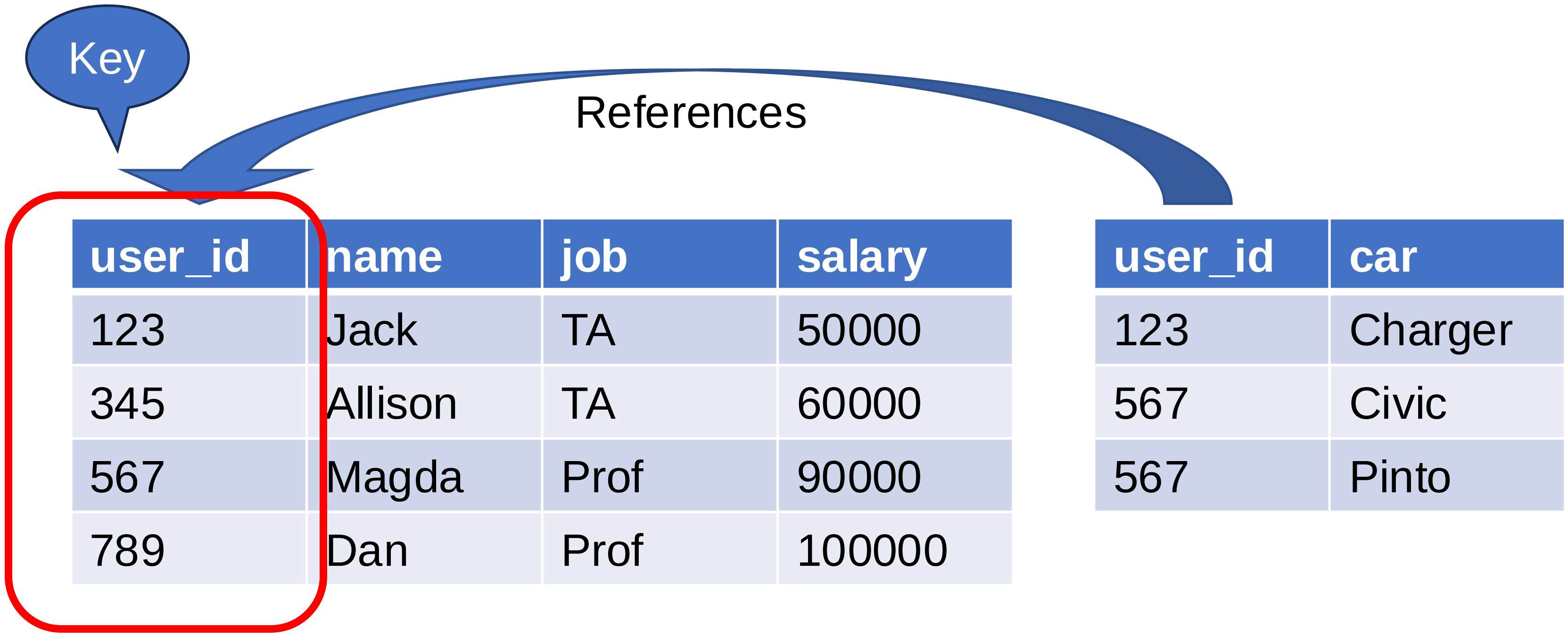
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

Recap: Keys and Foreign Keys

Foreign Key

A **Key** is one or more attributes that **uniquely** identify a row.
A **Foreign Key** is one or more attrs that uniquely identify a row in *another table*.



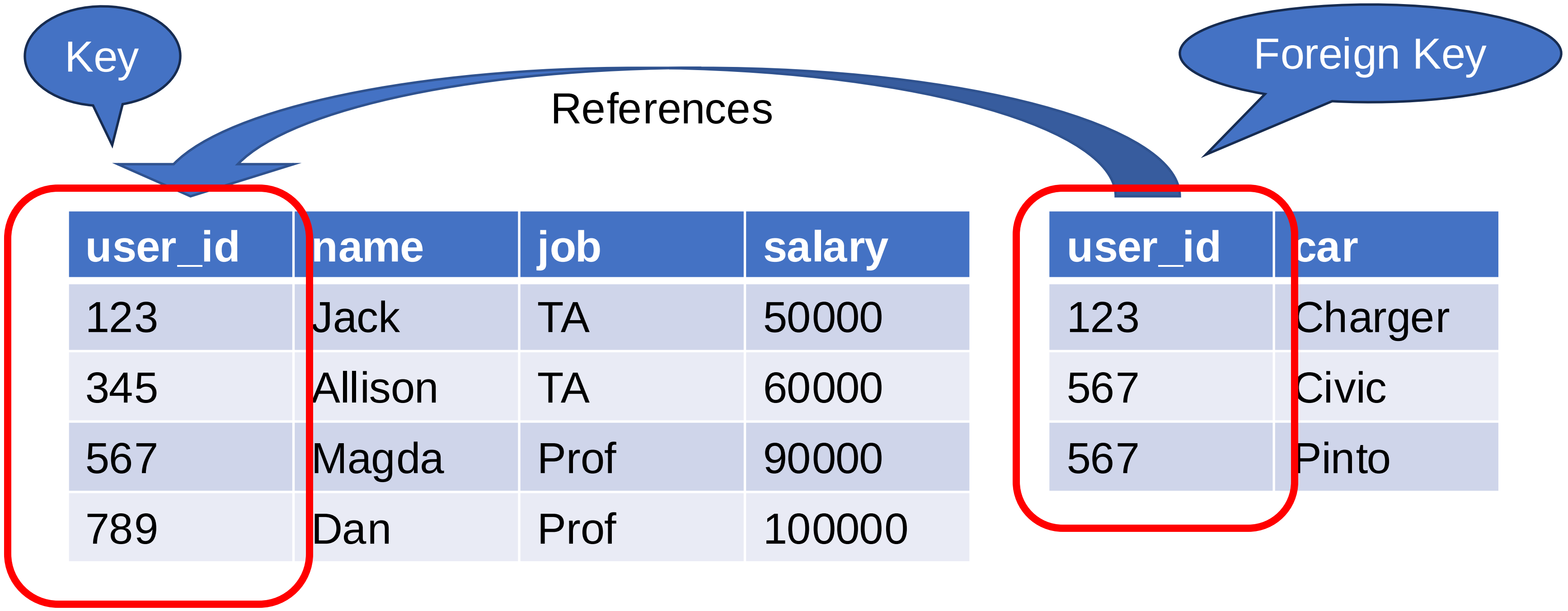
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

Recap: Keys and Foreign Keys

Foreign Key

A **Key** is one or more attributes that **uniquely** identify a row.
A **Foreign Key** is one or more attrs that uniquely identify a row in *another table*.



user_id	**name**	**job**	**salary**
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	**car**
123	Charger
567	Civic
567	Pinto

Joins

Joins

- Joins link records from different tables.
- May use the key / foreign-key relationship, but may also use any other relationships

Join

For each employee, find the cars that they drive

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each TA, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
  AND p.job = 'TA';
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each TA, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
  AND p.job = 'TA';
```

name	car
Jack	Charger

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each TA, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
  AND p.job = 'TA';
```

name	car
Jack	Charger

and is a Boolean expression

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Boolean Expression

In the WHERE clause: may use AND, OR, NOT

```
SELECT name
FROM payroll
WHERE job = 'TA' OR (salary > 55000 AND salary < 95000);
```

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Boolean Expression

In the WHERE clause: may use AND, OR, NOT

```
SELECT name
FROM payroll
WHERE job = 'TA' OR (salary > 55000 AND salary < 95000);
```

name
Jack
Allison
Magda

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Boolean Expression

In the WHERE clause: may use AND, OR, NOT

```
SELECT name
FROM payroll
WHERE job = 'TA' OR (salary > 55000 AND salary < 95000);
```

name
Jack
Allison
Magda

```
SELECT name
FROM payroll
WHERE job = 'TA' AND (salary > 55000 AND salary < 95000);
```

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Boolean Expression

In the WHERE clause: may use AND, OR, NOT

```
SELECT name
FROM payroll
WHERE job = 'TA' OR (salary > 55000 AND salary < 95000);
```

name
Jack
Allison
Magda

```
SELECT name
FROM payroll
WHERE job = 'TA' AND (salary > 55000 AND salary < 95000);
```

name
Allison

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Joins

- When we use joins we often have multiple conditions in the WHERE clause: and/or/not

- Next: two ways to write the join

Join: Two Ways to Write a Join

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join: Two Ways to Write a Join

```
SELECT p.name, r.car
FROM payroll AS p
JOIN regist AS r
  ON p.user_id = r.user_id;
```

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Means the same thing

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join: Two Ways to Write a Join

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
  AND p.job = ‘TA’;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join: Two Ways to Write a Join

```
SELECT p.name, r.car
FROM payroll AS p JOIN regist AS r
  ON p.user_id = r.user_id
WHERE p.job = ‘TA’;
```

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
  AND p.job = ‘TA’;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join: Two Ways to Write a Join

```
SELECT p.name, r.car
FROM payroll AS p JOIN regist AS r
  ON p.user_id = r.user_id
WHERE p.job = 'TA';
```

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
  AND p.job = 'TA';
```

```
SELECT p.name, r.car
FROM payroll AS p JOIN regist AS r
  ON p.user_id = r.user_id
 AND p.job = 'TA';
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join: Two Ways to Write a Join

ON same as WHERE
for now; but wait for it…

```
SELECT p.name, r.car
FROM payroll AS p JOIN regist AS r
   ON p.user_id = r.user_id
WHERE p.job = 'TA';
```

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id
   AND p.job = 'TA';
```

```
SELECT p.name, r.car
FROM payroll AS p JOIN regist AS r
   ON p.user_id = r.user_id
  and p.job = 'TA';
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Discussion

- A join is often between a key and a foreign key

- But not always! Let’s see some examples

Join

```
-- find the cars they are driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

```
-- find the cars they are driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

```
-- find the cars they are not driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id != r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

```
-- find the cars they are driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
-- find the cars they are not driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id != r.user_id;
```

name	car
Jack	Civic
Jack	Pinto
Allison	Charger
Allison	…
…	

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

```
-- find the cars they are driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
-- find the cars they are not driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id != r.user_id;
```

name	car
Jack	Civic
Jack	Pinto
Allison	Charger
Allison	…
…	

```
-- find WHAT??
SELECT p.name, r.car
FROM payroll AS p, regist AS r
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

```
-- find the cars they are driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
-- find the cars they are not driving
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id != r.user_id;
```

name	car
Jack	Civic
Jack	Pinto
Allison	Charger
Allison	…
…	

```
-- find WHAT??
SELECT p.name, r.car
FROM payroll AS p, regist AS r
```

. . . .

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Discussion

- FROM clause: several table names
- WHERE clause: some condition on these tables
- **Q**: What does it mean?
- **A**: For-Each semantics (Nested Loop Semantics)!

Nested Loop Semantics (again!)

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

How do we algorithmically get our results?

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

```
for each row1 in payroll:
   for each row2 in regist:
      if (row1.user_id = row2.user_id):
         output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger
Magda	Civic

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger
Magda	Civic

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
            output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

Final answer

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

```
for each row1 in payroll:
    for each row2 in regist:
        if (row1.user_id = row2.user_id):
           output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Key / Foreign-key join

```
for each row1 in payroll:
 for each row2 in regist:
  if (row1.user_id = row2.user_id):
    output (row1.name, row2.car)
```

Nested-Loop Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Key / Foreign-key join

```
for each row1 in payroll:
 for each row2 in regist:
  if (row1.user_id = row2.user_id):
    output (row1.name, row2.car)
```

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r;
```

Cross product

```
for each row1 in payroll:
 for each row2 in regist:
  output (row1.name,row2.car)
```

Summary: Nested-Loop Semantics

- FROM clause contains tables `T1, T2, T3, …`
- WHERE clause contains `condition`
- SELECT clause contains `attr1, attr2, …`

```
for each r1 in T1:
 for each t2 in T2:
   for each t3 in T3:
      …
      if (condition):
        output (attr1,attr2,…)
```

Set-builder semantics

Set-builder Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Key / Foreign-key join

$$\{(n,c) \mid (pu,n,j,s) \in \text{payroll} \wedge (ru,c) \in \text{regist} \wedge pu = ru \}$$

Set-builder Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Key / Foreign-key join

$$\{(n,c) \mid (pu,n,j,s) \in \text{payroll} \wedge (ru,c) \in \text{regist} \wedge pu = ru \}$$

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r;
```

Cross product

$$\{(n,c) \mid (pu,n,j,s) \in \text{payroll} \wedge (ru,c) \in \text{regist} \}$$

Set-builder Semantics

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT *
FROM payroll AS p, regist AS r;
```

Cross product

$$\{(pu, n, j, s, ru, c) \mid (pu, n, j, s) \in \text{payroll} \wedge (ru, c) \in \text{regist} \}$$

close to:

$$\text{payroll} \times \text{regist}$$

```
SELECT p.name, r.car
FROM payroll AS p, regist AS r
WHERE p.user_id = r.user_id;
```

Key / Foreign-key join

$$\{(n, c) \mid (pu, n, j, s) \in \text{payroll} \wedge (ru, c) \in \text{regist} \wedge pu = ru \}$$

Self-Joins

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

Intended answer

name	car1	car2
Magda	Civic	Pinto

Self Joins

Find all people who drive a ~~Civic and~~ Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

Let’s start with Pinto…

Self Joins

Find all people who drive a ~~Civic and~~ Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
       r.car = ‘Pinto’;
```

Let’s start with Pinto…

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
       r.car = ‘Civic’ AND
       r.car = ‘Pinto’;
```

Now both

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
       r.car = 'Civic' AND
       r.car = 'Pinto';
```

Now both

Will this work?

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
       r.car = 'Civic' AND
       r.car = 'Pinto';
```

Will this work?

Nope, returns the empty set.

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
        (r.car = 'Civic' OR
         r.car = 'Pinto');
```

Is this better?

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
       (r.car = 'Civic' OR
        r.car = 'Pinto');
```

Is this better?
Nope, it returns
both Jack and Magda.

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, r.car
  FROM payroll AS p, regist AS r
 WHERE p.user_id = r.user_id AND
       (r.car = 'Civic' OR
        r.car = 'Pinto');
```

Discuss with the people around you how you would solve this.

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, R1.car as car1, R2.car as car2
  FROM payroll AS p, regist AS r1, regist AS r2
 WHERE p.user_id = R1.user_id AND
       p.user_id = R2.user_id AND
       R1.car = 'Civic' AND
       R2.car = 'Pinto';
```

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

```
SELECT p.name, R1.car as car1, R2.car as car2
  FROM payroll AS p, regist AS r1, regist AS r2
 WHERE p.user_id = R1.user_id AND
       p.user_id = R2.user_id AND
       R1.car = 'Civic' AND
       R2.car = 'Pinto';
```

name	car1	car2
Magda	Civic	Pinto

Self Joins

Find all people who drive a Civic and Pinto

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

user_id	car
123	Charger
123	Pinto
123	Tesla
567	Civic
567	Pinto

The person we look for must drive TWO cars

```
SELECT p.name, R1.car as car1, R2.car as car2
  FROM payroll AS p, regist AS r1, regist AS r2
 WHERE p.user_id = R1.user_id AND
       p.user_id = R2.user_id AND
       R1.car = 'Civic' AND
       R2.car = 'Pinto';
```

name	car1	car2
Magda	Civic	Pinto

Self Joins

- When a relation occurs twice in the FROM clause we call it a “self-join”

- If we have a self-join, we must use table aliases; Otherwise, the attribute names are ambiguous

Outer Joins

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     JOIN regist AS r
     ON p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     JOIN regist AS r
     ON p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto

Allison, Dan are missing

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     LEFT OUTER JOIN regist AS r
     ON p.user_id = r.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     LEFT OUTER JOIN regist AS r
     ON p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto
Allison	NULL
Dan	NULL

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     LEFT OUTER JOIN regist AS r
     ON p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto
Allison	NULL
Dan	NULL

NULL means "unknown" or "missing"

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     LEFT OUTER JOIN regist AS r
     ON p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto
Allison	NULL
Dan	NULL

Left outer join:

1. Perform the join with the ON clause
2. Add all missing tuples from LEFT
3. Check the WHERE clause (if present)

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Join

For each employee, find the cars that they drive

```
SELECT p.name, r.car
FROM payroll AS p
     LEFT OUTER JOIN regist AS r
     ON p.user_id = r.user_id;
```

name	car
Jack	Charger
Magda	Civic
Magda	Pinto
Allison	NULL
Dan	NULL

ON, WHERE differ (next lecture)

Left outer join:

1. Perform the join with the ON clause
2. Add all missing tuples from LEFT
3. Check the WHERE clause (if present)

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

Outer Joins

- LEFT OUTER JOIN
  - Add missing tuples from the LEFT
- RIGHT OUTER JOIN
  - Add missing tuples from the RIGHT
- FULL OUTER JOIN
  - Add missing tuples from both

Outer Joins

- LEFT OUTER JOIN
  - Add missing tuples from the LEFT
- RIGHT OUTER JOIN
  - Add missing tuples from the RIGHT
- FULL OUTER JOIN
  - Add missing tuples from both

Useful, especially for aggregates (next lecture)

Rarely used