



# CSE 344: Intro to Data Management

## Outer Joins, NULL

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

# 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

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

NULL means  
"unknown" or  
"missing"

# 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

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)

ON, WHERE differ  
(next lecture)

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

# NULLS

# NULLs in SQL

A NULL value means missing, or unknown, or undefined, or inapplicable

payroll

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

# NULLs in SQL

A NULL value means missing, or unknown, or undefined, or inapplicable

```
.nullvalue NULL
INSERT INTO payroll
VALUES (123, 'Jack', 'TA', 50000),
       (345, 'Allison', NULL, 60000),
       (567, 'Magda', 'Prof', 90000),
       (789, 'Dan', 'Prof', NULL),
       (432, NULL, 'Prof', NULL);
```

payroll

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

# NULLs in SQL

A NULL value means missing, or unknown, or undefined, or inapplicable

Tells Sqlite how to print it

```
.nullvalue NULL
```

```
INSERT INTO payroll
VALUES (123, 'Jack', 'TA', 50000),
       (345, 'Allison', NULL, 60000),
       (567, 'Magda', 'Prof', 90000),
       (789, 'Dan', 'Prof', NULL),
       (432, NULL, 'Prof', NULL);
```

payroll

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

# NULLs in SQL

A NULL value means missing, or unknown, or undefined, or inapplicable

Complications:

- Expressions with NULLs?
- Conditions with NULLs?

payroll

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

# Expressions with NULLs

If any term is NULL, the entire expression is NULL

Give everyone a 10% raise

payroll

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



# Expressions with NULLs

If any term is NULL, the entire expression is NULL

Give everyone a 10% raise

```
SELECT name, salary*1.1 AS new_salary
FROM payroll;
```

payroll

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

# Expressions with NULLs

If any term is NULL, the entire expression is NULL

Give everyone a 10% raise

```
SELECT name, salary*1.1 AS new_sal
FROM payroll;
```

payroll

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



name	new_sal
Jack	55000
Allison	66000
Magda	99000
Dan	NULL
NULL	NULL

# Expressions with NULLs

If any term is NULL, the entire expression is NULL

```
SELECT name, salary*0 AS new_sal
FROM payroll;
```

payroll

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



name	new_sal
Jack	0
Allison	0
Magda	0
Dan	NULL
NULL	NULL

# Expressions with NULLs

If any term is NULL, the entire expression is NULL

```
SELECT name, salary*0 AS new_sal
FROM payroll;
```

NULL\*0 is not 0

payroll

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



name	new_sal
Jack	0
Allison	0
Magda	0
Dan	NULL
NULL	NULL

# Expressions with NULLs

If any term is NULL, the entire expression is NULL

```
SELECT name, 0 AS new_sal  
FROM payroll;
```

payroll

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



name	new_sal
Jack	0
Allison	0
Magda	0
Dan	0
NULL	0

now it works

# Conditions with NULLs

How should NULLs affect conditions in WHERE?

payroll

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

# Conditions with NULLs

How should NULLs affect conditions in WHERE?

```
SELECT name  
FROM payroll  
WHERE job = 'TA';
```

payroll

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

# Conditions with NULLs

How should NULLs affect conditions in WHERE?

```
SELECT name
FROM payroll
WHERE job = 'TA';
```

???



name	job
Jack	TA
Allison??	???

payroll

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



# Conditions with NULLs

How should NULLs affect conditions in WHERE?

```
SELECT name
FROM payroll
WHERE job = 'TA';
```

???



name	job
Jack	TA
Allison??	<del>???</del>

payroll

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

Not included:  
SQL uses  
3 valued logic.

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown =  $\min(1, 0.5) = \text{unknown}$

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

▪ true AND unknown =            **unknown**

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown =            **unknown**
- true OR unknown =

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown = **unknown**
- true OR unknown = **true**

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown = **unknown**
- true OR unknown = **true**
- unknown AND false =



# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown = **unknown**
- true OR unknown = **true**
- unknown AND false = **false**

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown = **unknown**
- true OR unknown = **true**
- unknown AND false = **false**

A = value

A < value

A > value

# Three-valued Logic

false = 0;    unknown = 0.5;    true = 1

$x \text{ AND } y = \min(x,y);$

$x \text{ OR } y = \max(x,y);$

$\text{not } x = 1-x$

What are these conditions?

- true AND unknown = **unknown**
- true OR unknown = **true**
- unknown AND false = **false**

A = value

A < value

A > value

When A is NULL  
then **unknown**

# Three-valued Logic

What does this query return?

```
SELECT *  
FROM payroll  
WHERE job != 'Prof'  
       OR salary > 80000;
```

payroll

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

True

# Three-valued Logic

What does this query return?

```
SELECT *  
FROM payroll  
WHERE job != 'Prof'  
       OR salary > 80000;
```

payroll

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

True

Unknown

# Three-valued Logic

What does this query return?

```
SELECT *  
FROM payroll  
WHERE job != 'Prof'  
       OR salary > 80000;
```

payroll

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

True

Unknown

True

# Three-valued Logic

What does this query return?

```
SELECT *  
FROM payroll  
WHERE job != 'Prof'  
       OR salary > 80000;
```

payroll

user_id	name	job	salary	
123	Jack	TA	50000	True
345	Allison	NULL	60000	Unknown
567	Magda	Prof	90000	True
789	Dan	Prof	NULL	Unknown
432	NULL	Prof	NULL	Unknown

# Three-valued Logic

What does this query return?

```
SELECT *  
FROM payroll  
WHERE job != 'Prof'  
      OR salary > 80000;
```

payroll

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

user_id	name	job	salary
123	Jack	TA	50000
567	Magda	Prof	90000

True

Unknown

True

Unknown

Unknown





# Three-valued Logic

NULLs are the nightmare of query optimizers

```
SELECT *  
FROM payroll  
WHERE job != 'Prof' OR job = 'Prof';
```

payroll

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



# Three-valued Logic

NULLs are the nightmare of query optimizers

```
SELECT *  
FROM payroll  
WHERE job != 'Prof' OR job = 'Prof';
```

payroll

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

Should return everyone, but...



# Three-valued Logic

NULLs are the nightmare of query optimizers

```
SELECT *  
FROM payroll  
WHERE job != 'Prof' OR job = 'Prof';
```

payroll

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

Should return everyone, but...

...we are missing Allison!

# Three-valued Logic

NULLs are the nightmare of query optimizers

```
SELECT *  
FROM payroll  
WHERE job != 'Prof' OR job = 'Prof' OR job IS NULL;
```

payroll

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



Now we get everyone!

# Discussion

- NULL: convenient way to represent missing values
- NULL as “unknown” vs NULL as “known to be absent”
- Need 3-valued logic
- However, leads to huge complications for the optimizer, and even counterintuitive query behavior
- Better avoid NULLs if possible
- One exception: LEFT OUTER Joins. Let’s revisit

# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

Steps 1,2,3 →

name	car
Jack	Charger
Allison	NULL
Magda	NULL

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

Steps 1,2,3 →

name	car
Jack	Charger
Allison	NULL
Magda	NULL

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
WHERE r.car = 'Charger';
```

What differs if we place r.car='Charger' in the WHERE clause?

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto



# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

Steps 1,2,3 →

name	car
Jack	Charger
Allison	NULL
Magda	NULL

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
WHERE r.car = 'Charger';
```

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

Steps 1,2,3 →

name	car
Jack	Charger
Allison	NULL
Magda	NULL

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
WHERE r.car = 'Charger';
```

Steps 1,2 →

name	car
Jack	Charger
Allison	NULL
Magda	Civic
Magda	Pinto

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

Steps 1,2,3 →

name	car
Jack	Charger
Allison	NULL
Magda	NULL

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
WHERE r.car = 'Charger';
```

Steps 1,2 →

name	car
Jack	Charger
Allison	NULL
Magda	Civic
Magda	Pinto

Step 3 →

name	car
Jack	Charger

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Outer Joins Revisited: ON v.s. WHERE

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

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
AND r.car = 'Charger';
```

Steps 1,2,3 →

name	car
Jack	Charger
Allison	NULL
Magda	NULL

```
SELECT p.name, r.car
FROM payroll AS P
LEFT OUTER JOIN regist AS R
ON p.user_id = r.user_id
WHERE r.car = 'Charger';
```

Steps 1,2 →

name	car
Jack	Charger
Allison	NULL
Magda	Civic
Magda	Pinto

Step 3 →

name	car
Jack	Charger

ON, WHERE differ

payroll

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

regist

user_id	car
123	Charger
567	Civic
567	Pinto