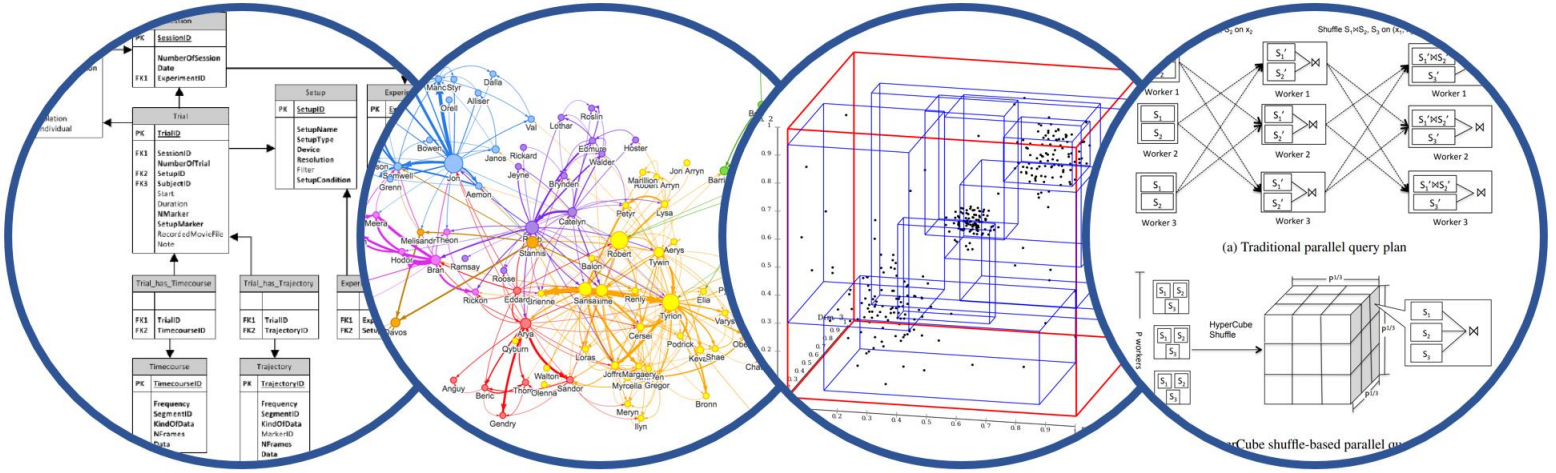


Introduction to Data Management

Relational Algebra

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

Announcements

- 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)
  - Some practice midterms on the course website

Relational Algebra

Motivation

- SQL is a declarative language:
  we say what, we don't say how
- The query optimizer needs to convert the query into some intermediate language that can be both optimized, and executed
- That language is Relational Algebra

The Five Basic Relational Operators

1. Selection $\sigma_{\text{condition}}(S)$
2. Projection $\Pi_{\text{attrs}}(S)$
3. Join $R \bowtie_\theta S = \sigma_\theta(R \times S)$
4. Union $\cup$
5. Set difference $-$

Rename ρ

Let's discuss them one by one

1. Selection

$$\sigma_{\text{condition}}(T)$$

Returns those tuples in T that satisfy the condition:

```
SELECT *
FROM T
WHERE condition;
```

1. Selection

$$\sigma_{\text{condition}}(T)$$

Returns those tuples in T that satisfy the condition:

```
SELECT *
FROM T
WHERE condition;
```

$$\sigma_{\text{salary} \geq 55000}(\text{Payroll}) =$$

Payroll

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

1. Selection

$$\sigma_{\text{condition}}(T)$$

Returns those tuples in T that satisfy the condition:

```
SELECT *
FROM T
WHERE condition;
```

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

$$\sigma_{\text{salary}\geq 55000}(\text{Payroll}) =$$

Payroll

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

1. Selection

$$\sigma_{\text{condition}}(T)$$

Returns those tuples in T that satisfy the condition:

```
SELECT *
FROM T
WHERE condition;
```

$$\sigma_{\text{salary} \geq 55000 \text{ and Job}='\text{TA}'}(\text{Payroll}) =$$

Payroll

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

1. Selection

$\sigma_{condition}(T)$

UserID	Name	Job	Salary
345	Allison	TA	60000

Returns those tuples in T that satisfy the condition:

```
SELECT *
FROM T
WHERE condition;
```

$\sigma_{salary \geq 55000 \text{ and } Job='TA'}(Payroll) =$ 

Payroll

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

2. Projection

$$\Pi_{attrs}(T)$$

Returns all tuples in T keeping
only the attributes in the subscript:

```
SELECT attrs
FROM T;
```

2. Projection

$$\Pi_{attrs}(T)$$

Returns all tuples in T keeping only the attributes in the subscript:

$$\Pi_{Name,Salary}(Payroll) =$$

```
SELECT attrs
FROM T;
```

Payroll

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

2. Projection

$$\Pi_{attrs}(T)$$

Name	Salary
Jack	50000
Allison	60000
Magda	90000
Dan	100000

Returns all tuples in T keeping only the attributes in the subscript:

$$\Pi_{Name,Salary}(Payroll) =$$

```
SELECT attrs
FROM T;
```

Payroll

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

2. Projection

$$\Pi_{attrs}(T)$$

Returns all tuples in T keeping only the attributes in the subscript:

$$\Pi_{Job}(Payroll) =$$

```
SELECT attrs
FROM T;
```

Payroll

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

2. Projection

$$\Pi_{attrs}(T)$$

Job
TA
TA
Prof
Prof

Returns all tuples in T keeping only the attributes in the subscript:

$$\Pi_{Job}(Payroll) =$$

```
SELECT attrs
FROM T;
```

Payroll

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

2. Projection

$$\Pi_{attrs}(T)$$

Job
TA
TA
Prof
Prof

RA can be defined using bag semantics or set semantics.
We always need to say which one we mean.

Job
TA
Prof

Returns all tuples in T keeping only the attributes in the subscript:

$$\Pi_{Job}(Payroll) =$$

```
SELECT attrs
FROM T;
```

Payroll

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

3. Join

$$S \bowtie_{\theta} T$$

Join S and T using condition θ

```
SELECT *
FROM S,T
WHERE θ;
```

3. Join

$$S \bowtie_\theta T$$

Join S and T using condition θ

```
SELECT *
FROM S,T
WHERE θ;
```

$$\text{Payroll} \bowtie_{\text{UserID}=\text{UserID}} \text{Regist} =$$

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

3. Join

$$S \bowtie_{\theta} T$$

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

Join S and T using condition θ

$$\text{Payroll} \bowtie_{\text{UserID=UserID}} \text{Regist} =$$

```
SELECT *
FROM S,T
WHERE θ;
```

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

Many Variants of Join

- Eq-join: $\text{Payroll} \bowtie_{\text{UserID}=\text{UserID}} \text{Regist}$
- Theta-join: $\text{Payroll} \bowtie_{\text{UserID}<\text{UserID}} \text{Regist}$
- Cartesian product: $\text{Payroll} \times \text{Regist}$
- Natural Join: $\text{Payroll} \bowtie \text{Regist}$

Many Variants of Join

- Eq-join: Payroll $\bowtie_{\text{UserID}=\text{UserID}}$ Regist

Only =

- Theta-join: Payroll $\bowtie_{\text{UserID}<\text{UserID}}$ Regist

Any condition

- Cartesian product: Payroll $\times$ Regist
- Natural Join: Payroll $\bowtie$ Regist

Many Variants of Join

- Eq-join: $\text{Payroll} \bowtie_{\text{UserID}=\text{UserID}} \text{Regist}$

Only =

- Theta-join: $\text{Payroll} \bowtie_{\text{UserID}<\text{UserID}} \text{Regist}$

Any condition

- Cartesian product: $\text{Payroll} \times \text{Regist}$
- Natural Join: $\text{Payroll} \bowtie \text{Regist}$

Next

Cartesian Product / Cross Product

$$S \times T$$

Cross product of S and T

```
SELECT *
FROM S,T
```

Cartesian Product / Cross Product

$$S \times T$$

Cross product of S and T

```
SELECT *
FROM S,T
```

$$\text{Payroll} \times \text{Regist} =$$

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

Cartesian Product / Cross Product

$S \times T$

12 tuples

UserID	Name	Job	Salary	UserID	Car
123	Jack	TA	50000	123	Charger
123	Jack	TA	50000	567	Civic
			. . .		
789	Dan	Prof	100000	567	Pinto

Cross product of S and T

$Payroll \times Regist =$

```
SELECT *
FROM S,T
```

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

Cartesian Product / Cross Product

$S \times T$

Cross product of S and T

```
SELECT *
FROM S,T
```

Join = cartesian product + selection

$$R \bowtie_\theta S = \sigma_\theta(R \times S)$$

Natural Join

$S \bowtie T$

Join S, T on common attributes, retain only one copy of those attributes

Natural Join

$S \bowtie T$

Join S, T on common attributes, retain only one copy of those attributes

$Payroll \bowtie Regist =$

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

Natural Join

$S \bowtie T$

Join S, T on common attributes, retain only one copy of those attributes

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

Only one UserID attr

$\text{Payroll} \bowtie \text{Regist} =$

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

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$
- $R(A, B) \bowtie S(C, D)$
- $R(A, B) \bowtie S(A, B)$

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$

- $R(A, B) \bowtie S(C, D)$

- $R(A, B) \bowtie S(A, B)$

R

A	B
1	10
2	10
2	20

S

B	C
10	8
10	9
20	8
50	7

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$
  eqjoin on attribute B (5 tuples)

R

A	B
1	10
2	10
2	20

S

B	C
10	8
10	9
20	8
50	7

- $R(A, B) \bowtie S(C, D)$

- $R(A, B) \bowtie S(A, B)$

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$
  eqjoin on attribute B (5 tuples)

R

A	B
1	10
2	10
2	20

S

B	C
10	8
10	9
20	8
50	7

- $R(A, B) \bowtie S(C, D)$

R

A	B
1	10
2	10
2	20

S

C	D
8	u
9	v
8	v
7	w

- $R(A, B) \bowtie S(A, B)$

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$
  eqjoin on attribute B (5 tuples)

R

A	B
1	10
2	10
2	20

S

B	C
10	8
10	9
20	8
50	7

- $R(A, B) \bowtie S(C, D)$
  cross product (12 tuples)

R

A	B
1	10
2	10
2	20

S

C	D
8	u
9	v
8	v
7	w

- $R(A, B) \bowtie S(A, B)$

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$
  eqjoin on attribute B (5 tuples)

R

A	B
1	10
2	10
2	20

S

B	C
10	8
10	9
20	8
50	7

- $R(A, B) \bowtie S(C, D)$
  cross product (12 tuples)

R

A	B
1	10
2	10
2	20

S

C	D
8	u
9	v
8	v
7	w

- $R(A, B) \bowtie S(A, B)$

R

A	B
1	10
2	10
2	20

S

A	B
1	10
2	20

Natural Join

What do these natural joins output?

- $R(A, B) \bowtie S(B, C)$
  eqjoin on attribute B (5 tuples)

R

A	B
1	10
2	10
2	20

S

B	C
10	8
10	9
20	8
50	7

- $R(A, B) \bowtie S(C, D)$
  cross product (12 tuples)

R

A	B
1	10
2	10
2	20

S

C	D
8	u
9	v
8	v
7	w

- $R(A, B) \bowtie S(A, B)$
  intersection (2 tuples)

R

A	B
1	10
2	10
2	20

S

A	B
1	10
2	20

Even More Joins

- Inner join $\bowtie$
  - Eq-join, theta-join, cross product, natural join
- Outer join
  - Left outer join $⟕$
  - Right outer join $⟖$
  - Full outer join $⟗$
- Semi join $\ltimes$

4. Union

$S \cup T$

The union of S and T

```
S UNION T;
```

SQL

4. Union

$S \cup T$

The union of S and T

$$\text{Regist} \cup \text{Bicycle} =$$

```
S UNION T;
```

Regist

UserID	Model
123	Charger
567	Civic
567	Pinto

Bicycle

UserID	Model
345	Schwinn
567	Sirrus

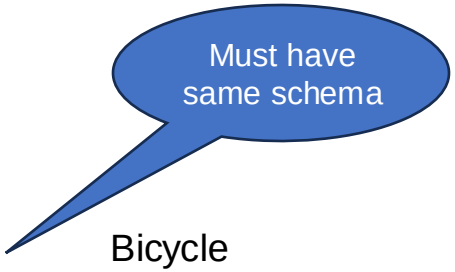
4. Union

$S \cup T$

The union of S and T

```
S UNION T;
```

$\text{Regist} \cup \text{Bicycle} =$



Regist

UserID	Model
123	Charger
567	Civic
567	Pinto

Bicycle

UserID	Model
345	Schwinn
567	Sirrus

4. Union

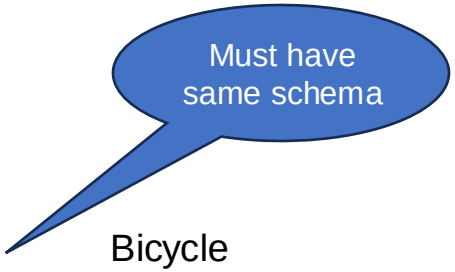
$S \cup T$

The union of S and T

```
S UNION T;
```

UserID	Model
123	Charger
567	Civic
567	Pinto
345	Schwinn
567	Sirrus

$\text{Regist} \cup \text{Bicycle} =$



Regist

UserID	Model
123	Charger
567	Civic
567	Pinto

Bicycle

UserID	Model
345	Schwinn
567	Sirrus

5. Difference

$$S - T$$

The set difference of S and T

```
S EXCEPT T;
```

SQL

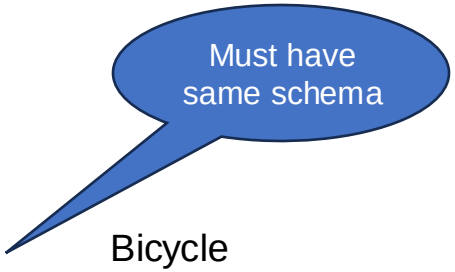
5. Difference

$$S - T$$

The set difference of S and T

$$\text{Regist} - \text{Bicycle} =$$

```
S EXCEPT T;
```



Regist

UserID	Model
123	Charger
567	Civic
567	Pinto

Bicycle

UserID	Model
345	Schwinn
567	Civic

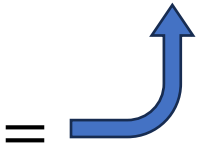
5. Difference

$$S - T$$

UserID	Model
123	Charger
567	Pinto

The set difference of S and T

$$Regist - Bicycle =$$



Must have same schema

```
S EXCEPT T;
```

Regist

UserID	Model
123	Charger
567	Civic
567	Pinto

Bicycle

UserID	Model
345	Schwinn
567	Civic

Renaming

$$\rho_{attrs'}(T)$$

Rename attributes

```
SELECT a1 as a1',
       a2 as a2',
       ...
FROM T;
```

Renaming

$$\rho_{attrs'}(T)$$

Rename attributes

$$\rho_{\text{UserID,Model}}(\text{Regist}) =$$

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

```
SELECT a1 as a1',
       a2 as a2',
       ...
FROM T;
```

Renaming

$\rho_{attrs'}(T)$

Rename attributes

```
SELECT a1 as a1',
       a2 as a2',
       ...
FROM T;
```

UserID	Model
123	Charger
567	Civic
567	Pinto

$\rho_{\text{UserID,Model}}(\text{Regist}) =$ 

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

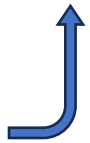
Renaming

$\rho_{attrs'}(T)$

Rename attributes

```
SELECT a1 as a1',
       a2 as a2',
       ...
FROM T;
```

UserID	Model
123	Charger
567	Civic
567	Pinto

$\rho_{\text{UserID,Model}}(\text{Regist}) =$ 

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Corrected union:

$\rho_{\text{UserID,Model}}(\text{Regist}) \cup \text{Bicycle}$

The Five Basic Relational Operators

1. Selection $\sigma_{\text{condition}}(S)$
2. Projection $\Pi_{\text{attrs}}(S)$
3. Join $R \bowtie_\theta S = \sigma_\theta(R \times S)$
4. Union $\cup$
5. Set difference $-$

Rename ρ

Which operators are monotone?

The Five Basic Relational Operators

1. Selection $\sigma_{\text{condition}}(S)$
2. Projection $\Pi_{\text{attrs}}(S)$
3. Join $R \bowtie_\theta S = \sigma_\theta(R \times S)$
4. Union $\cup$
5. Set difference $-$

Monotone

Non-monotone

Rename ρ — Monotone, but doesn't do anything

Which operators are monotone?

Query Plans

Relational Algebra Plan, or Query Plan

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

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

Relational Algebra Plan, or Query Plan

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

$$\Pi_{\text{Name}}(\sigma_{\text{Job}='\text{TA}'}(\text{Payroll} \bowtie \text{Regist}))$$

Payroll

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

Regist

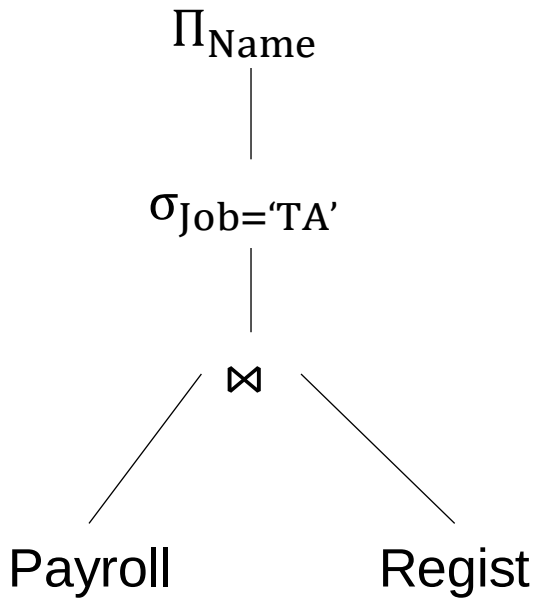
UserID	Car
123	Charger
567	Civic
567	Pinto

Relational Algebra Plan, or Query Plan

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

We write it as

a query plan



$$\Pi_{Name}(\sigma_{Job='TA'}(Payroll \bowtie Regist))$$

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

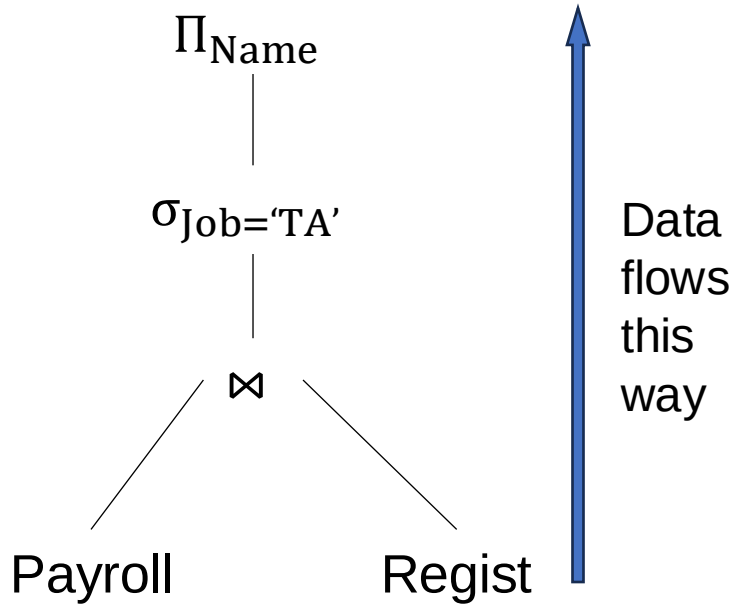
Relational Algebra Plan, or Query Plan

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

$\Pi_{\text{Name}}(\sigma_{\text{Job}='\text{TA}'}(\text{Payroll} \bowtie \text{Regist}))$

We write it as

a query plan



Payroll

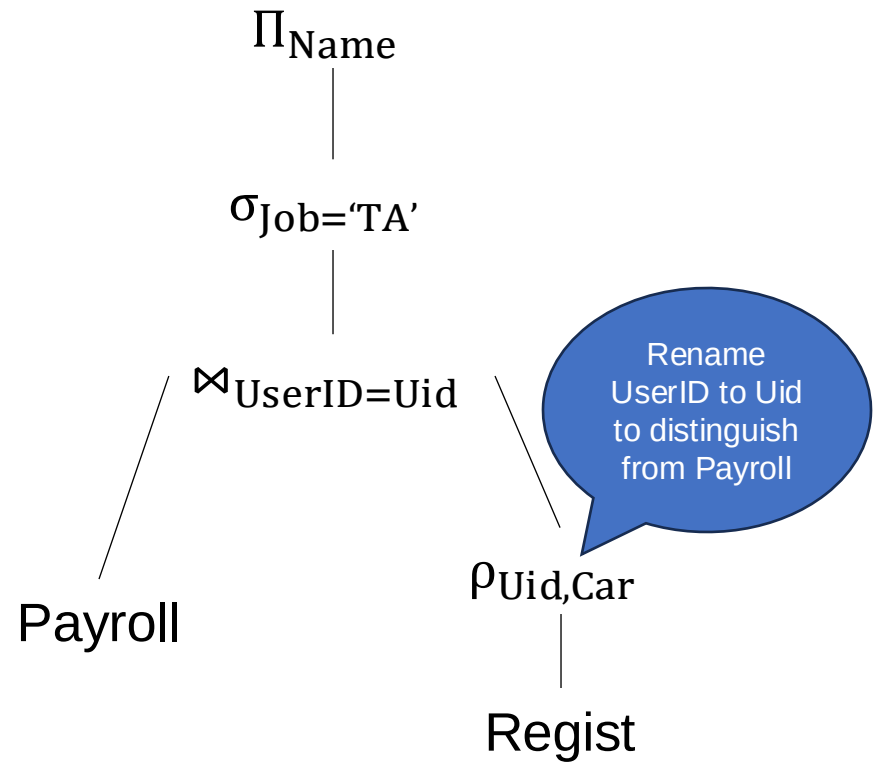
UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

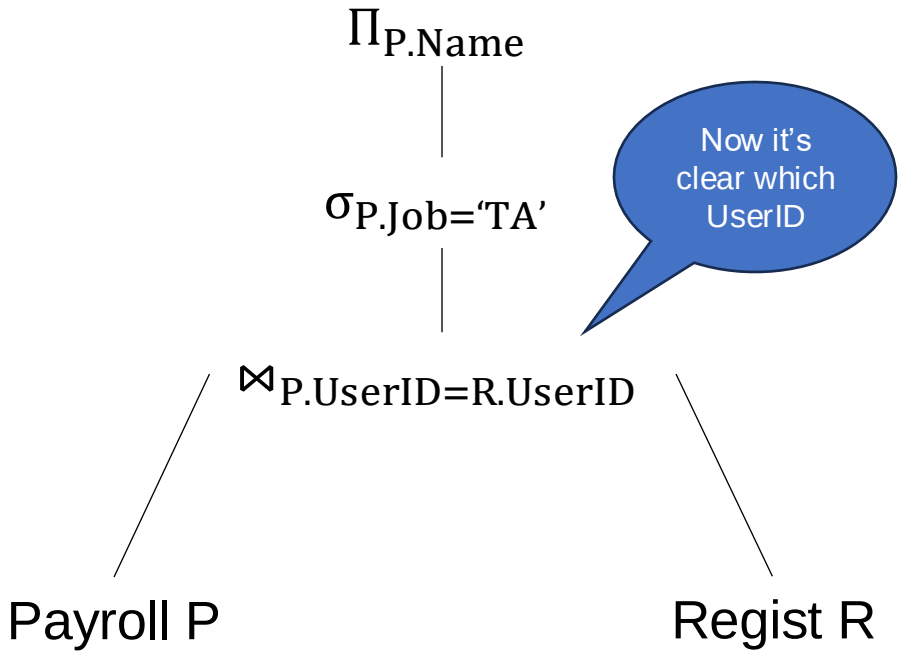
UserID	Car
123	Charger
567	Civic
567	Pinto

Query Plan: Attribute Names

Managing attribute names correctly is tedious



Better: use aliases, much like in SQL



Query Plan: Execution Order

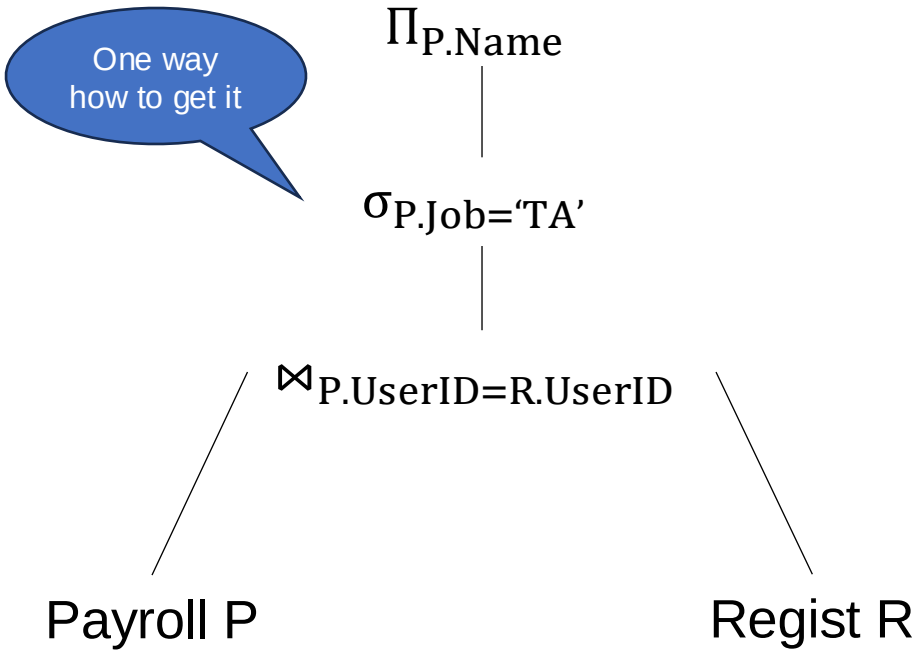
```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

We say what we want,
not how to get it

Query Plan: Execution Order

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

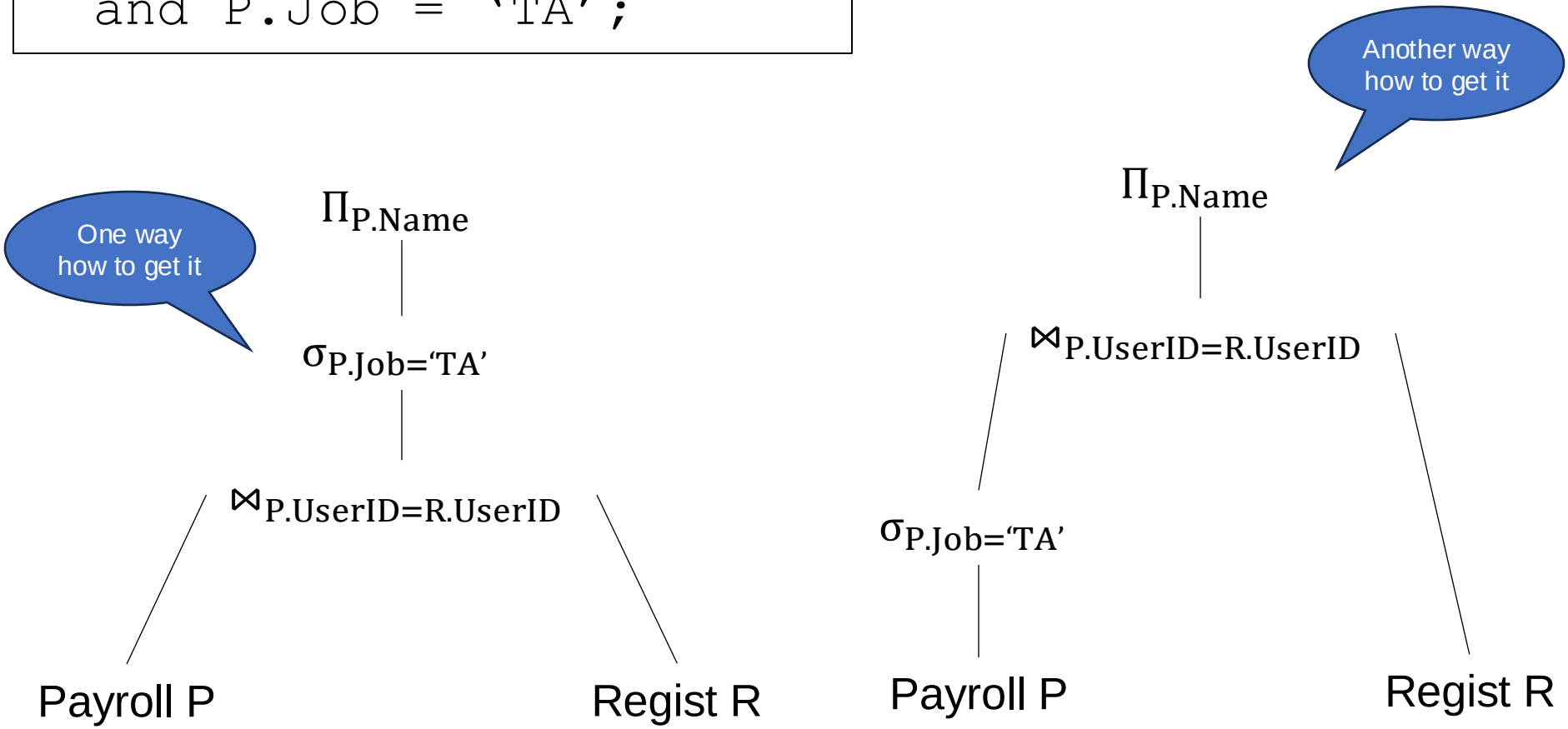
We say what we want,
not how to get it



Query Plan: Execution Order

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

We say what we want,
not how to get it



Query Plan: Execution Order

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

We say what we want,
not how to get it

One way
how to get it

$\Pi_{P.Name}$

$\sigma_{P.Job='TA'}$

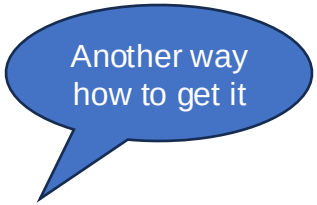
$\bowtie_{P.UserID=R.UserID}$

Payroll P

Regist R

Which one
is more
efficient?

Another way
how to get it



$\Pi_{P.Name}$

$\bowtie_{P.UserID=R.UserID}$

$\sigma_{P.Job='TA'}$

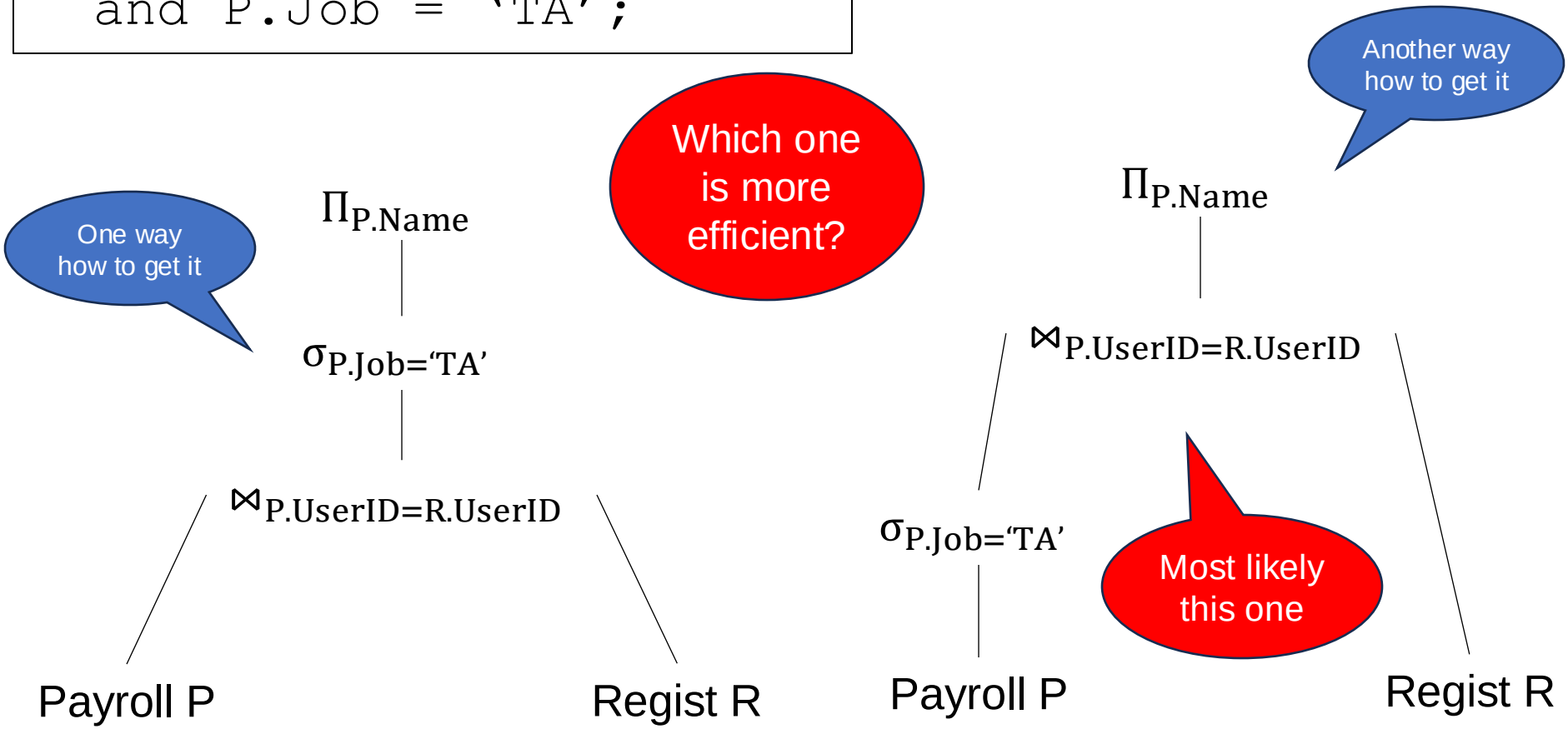
Payroll P

Regist R

Query Plan: Execution Order

```
SELECT P.Name
FROM Payroll P, Regist R
WHERE P.UserID = R.UserID
  and P.Job = 'TA';
```

We say what we want,
not how to get it



Discussion

- Database system converts a SQL query to a Relational Algebra Plan

Discussion

- Database system converts a SQL query to a Relational Algebra Plan

- Then it optimizes the plan by exploring equivalent plans, using simple algebraic identities:

$$R \bowtie S = S \bowtie R$$
$$R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$
$$\sigma_\theta(R \bowtie S) = \sigma_\theta(R) \bowtie S$$

… many others*

*over 500 rules in SQL Server

Discussion

- Database system converts a SQL query to a Relational Algebra Plan

- Then it optimizes the plan by exploring equivalent plans, using simple algebraic identities:

$$R \bowtie S = S \bowtie R$$
$$R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$$
$$\sigma_\theta(R \bowtie S) = \sigma_\theta(R) \bowtie S$$

… many others*

- Next lecture: how to convert SQL to RA plan

*over 500 rules in SQL Server