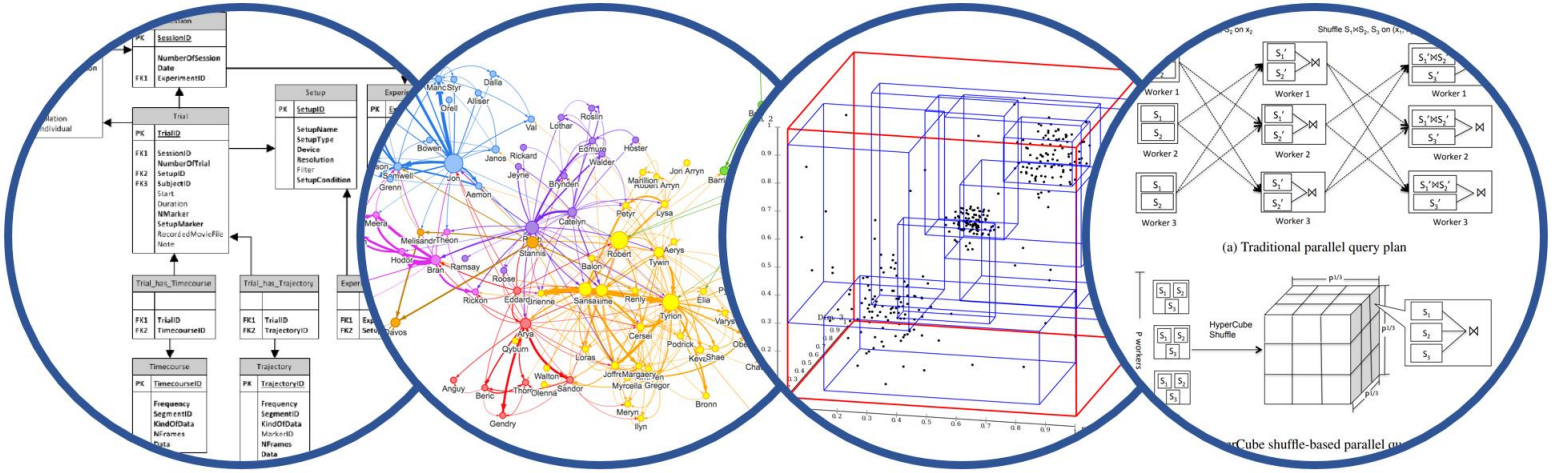


Introduction to Data Management

Cardinality Estimation

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

Announcements

- HW6: Part 2 due on Wednesday
- Lecture on Wednesday, Nov. 27:
  - No in-person lecture
  - Recording only
- Monday, Dec. 9, Final Review session
  - CSE2 G10
  - 10:30 – 12:20 (we may finish earlier)

Access Path Selection Wrapup

Multiple Indexes

- DB administrator can create multiple indexes
- But only one can be clustered…
- … and a selection can only use only one index

Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

Payroll:

Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

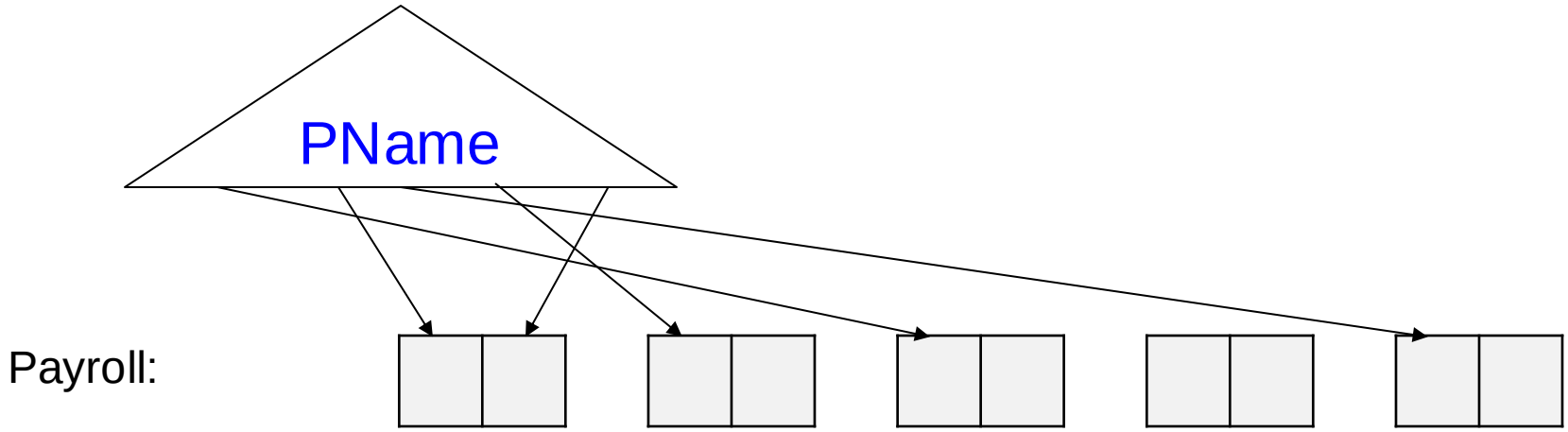
```
CREATE INDEX Pname on Payroll(name);
```

Payroll:

Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

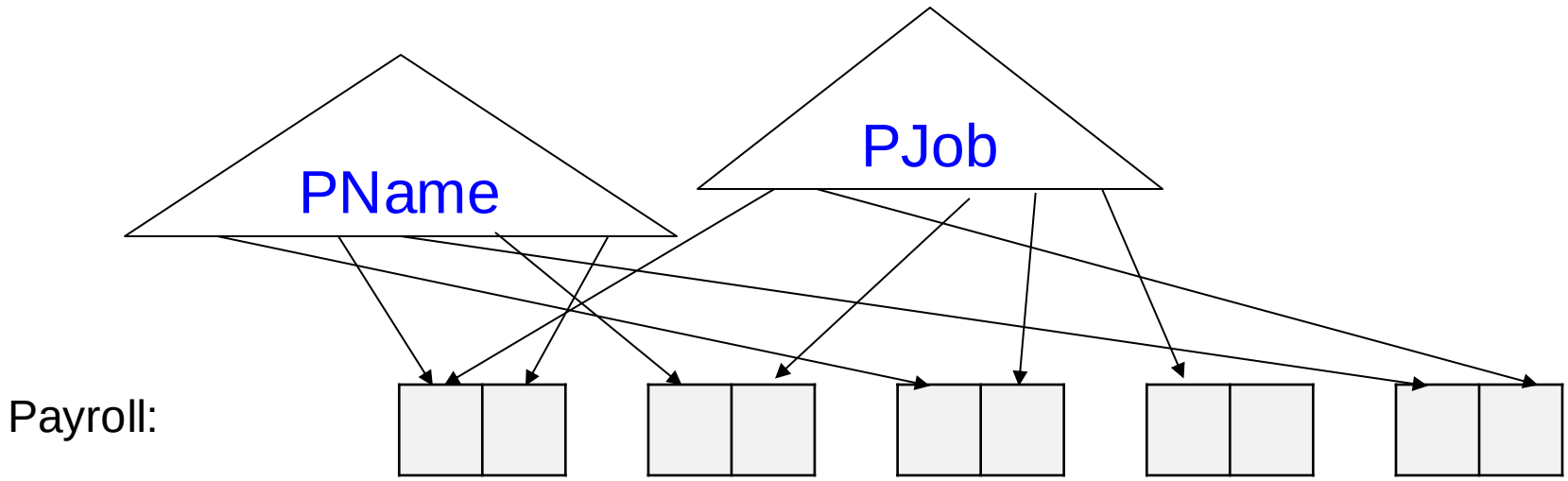
```
CREATE INDEX Pname on Payroll(name);
```



Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

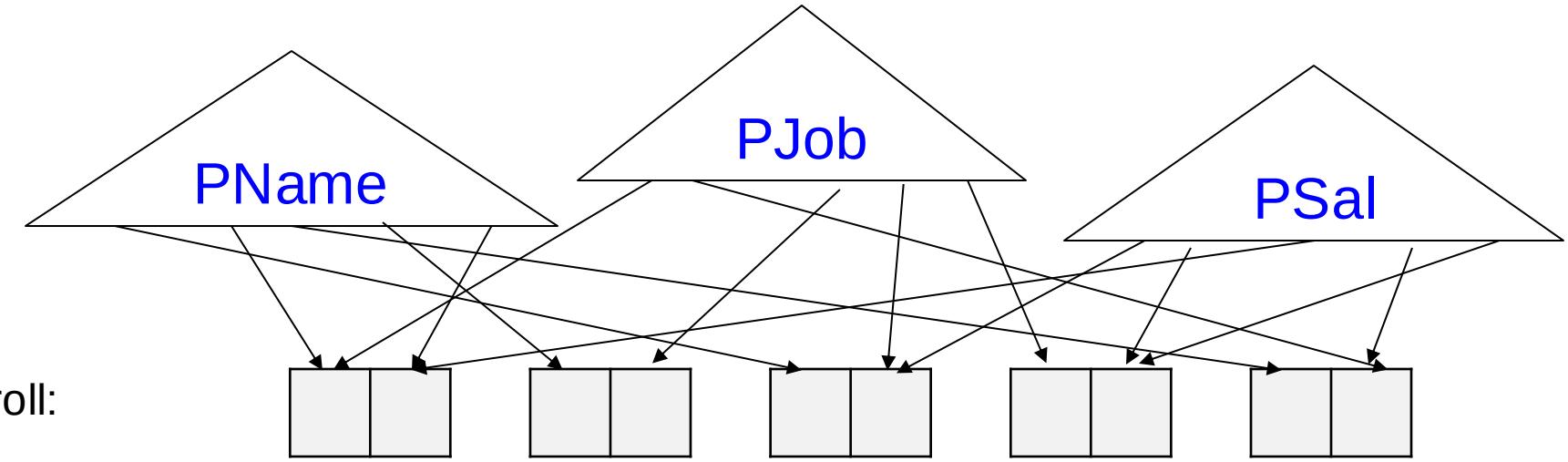
```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
```



Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

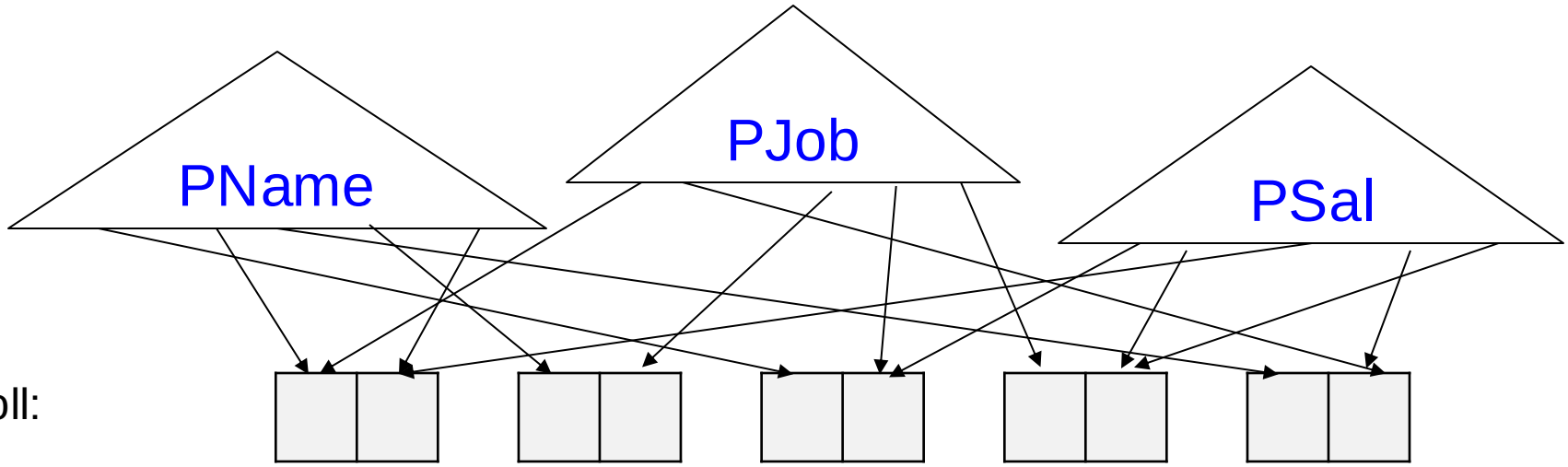


Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary=50000;
```



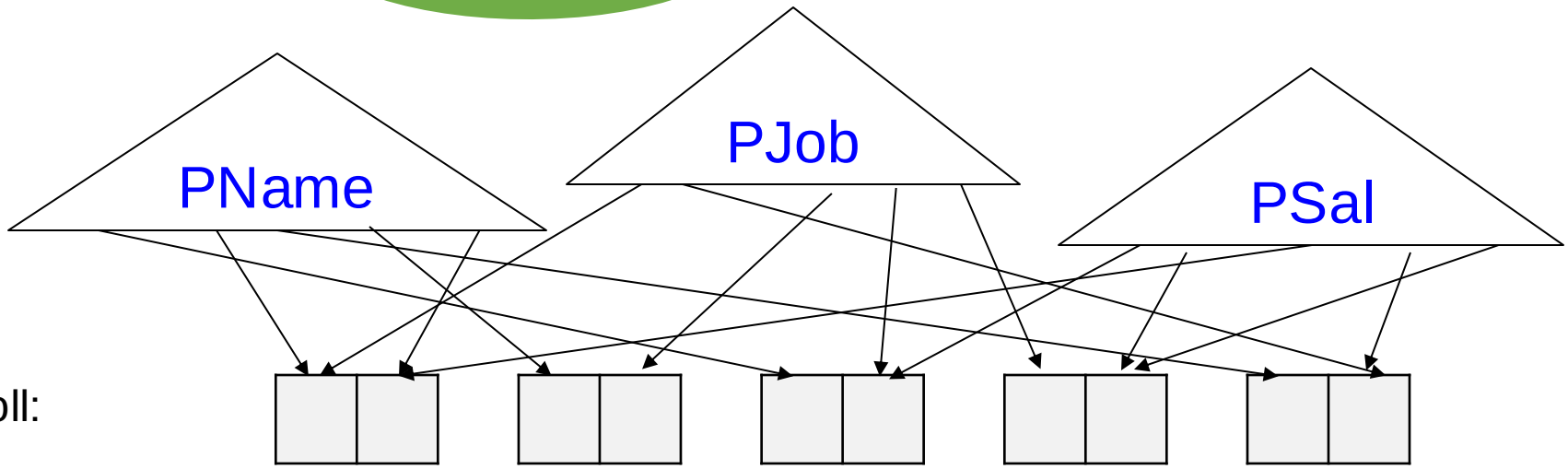
Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary=50000;
```

Access path:
use Pjob index



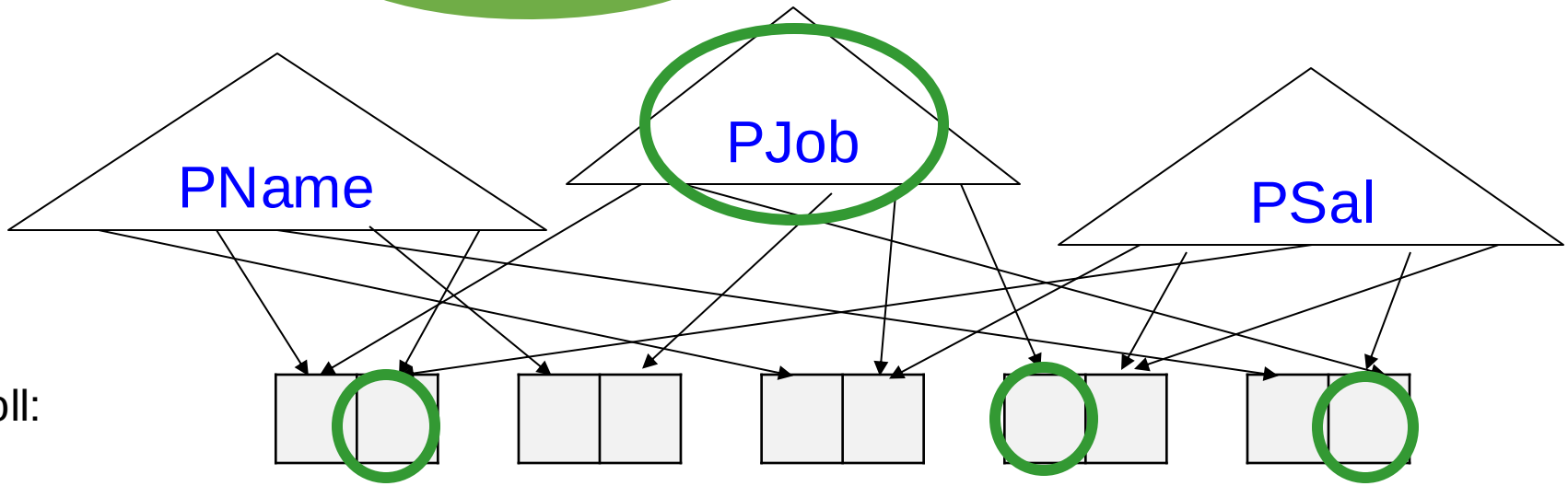
Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary=50000;
```

Access path:
use Pjob index



Access Path Selection

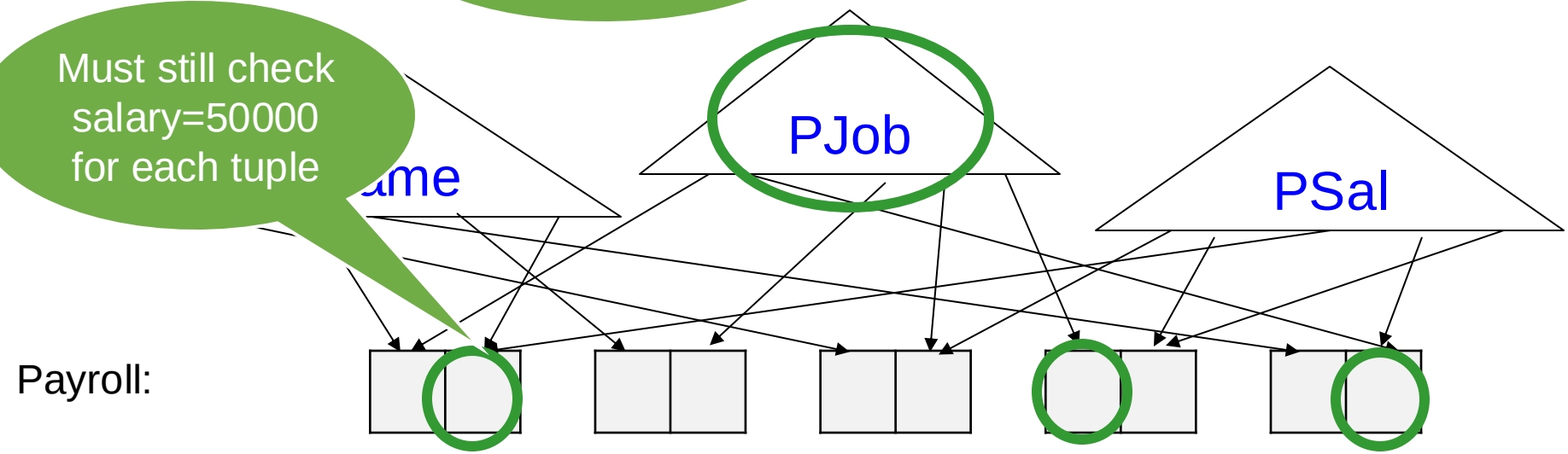
```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary=50000;
```

Access path:
use Pjob index

Must still check
salary=50000
for each tuple



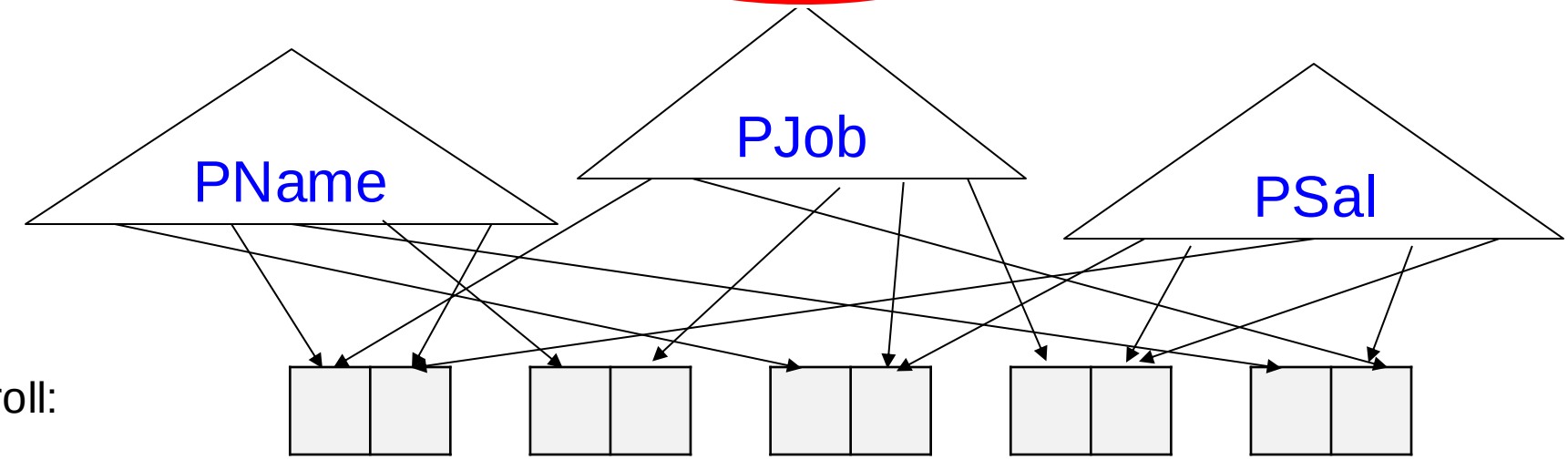
Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary=50000;
```

Acces path:
use Psal index



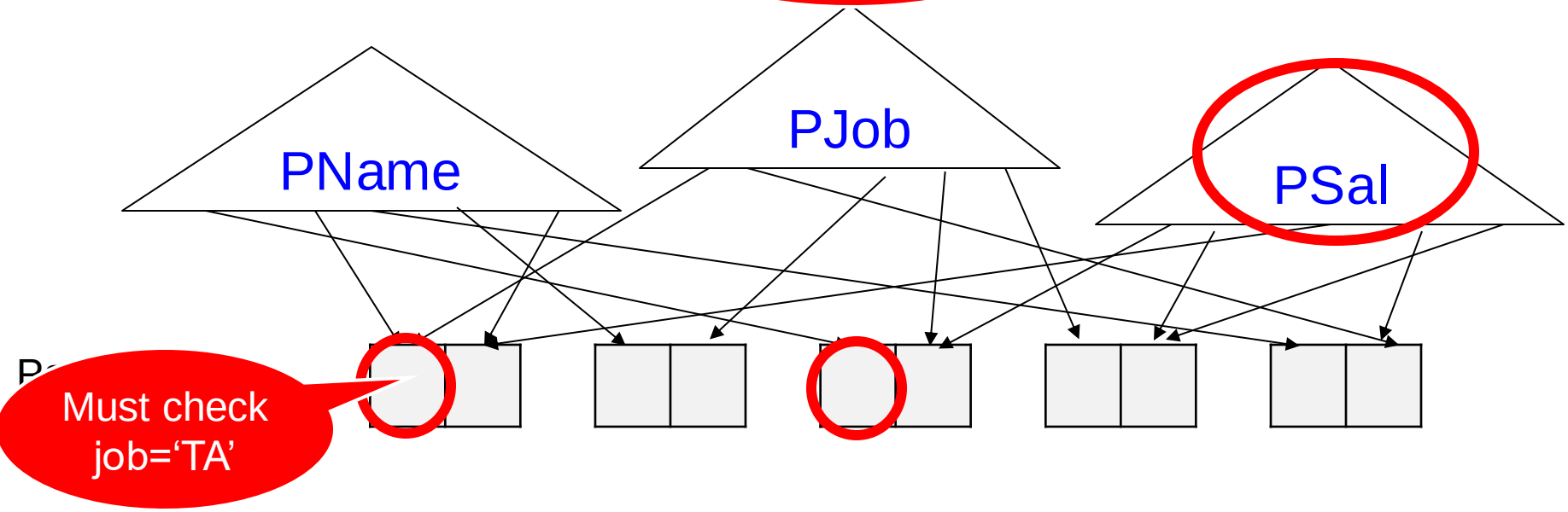
Access Path Selection

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX Pname on Payroll(name);
CREATE INDEX Pjob  on Payroll(job);
CREATE INDEX Psal  on Payroll(salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary=50000;
```

Acces path:
use Psal index



Multi-attribute Index

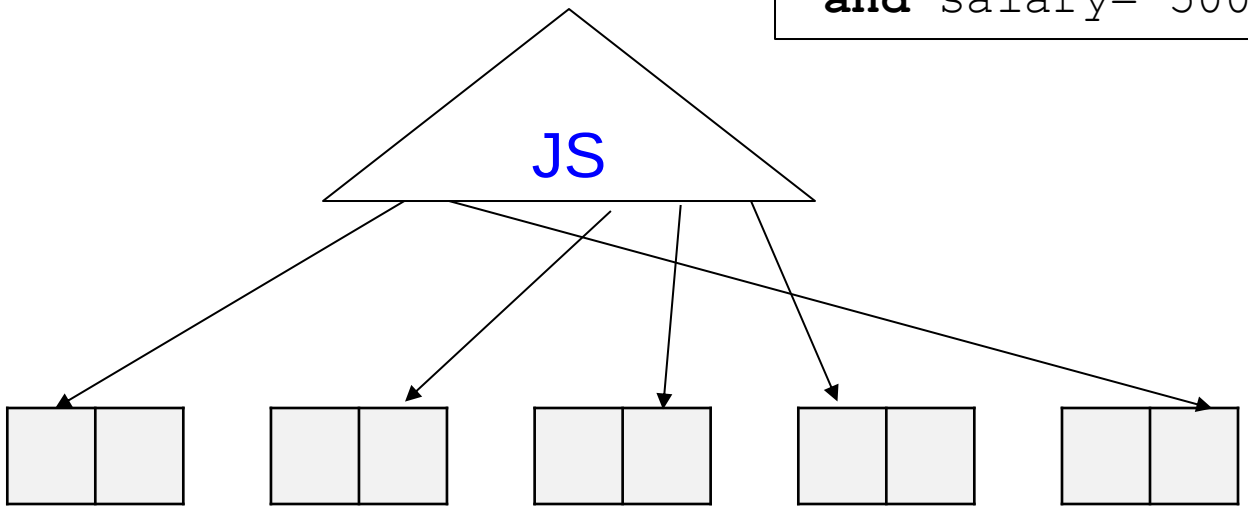
- An index can be on multiple attributes: A, B, C, …
  - Create index on R(B,A)
  - Create index on R(A,B,C)
  - Create index on R(C,A,B)
- Values are concatenated, then sorted
- Attribute order matters

Multi-attribute Index

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX JS on Payroll(job,salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
 and salary='50000';
```



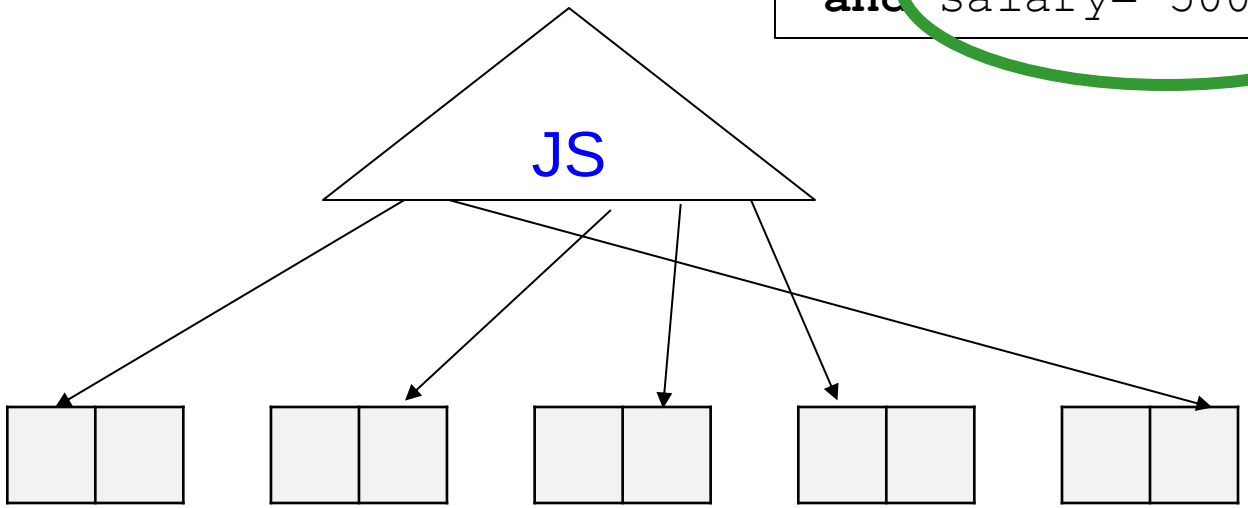
Payroll:

Multi-attribute Index

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX JS on Payroll(job,salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
  and salary='50000';
```



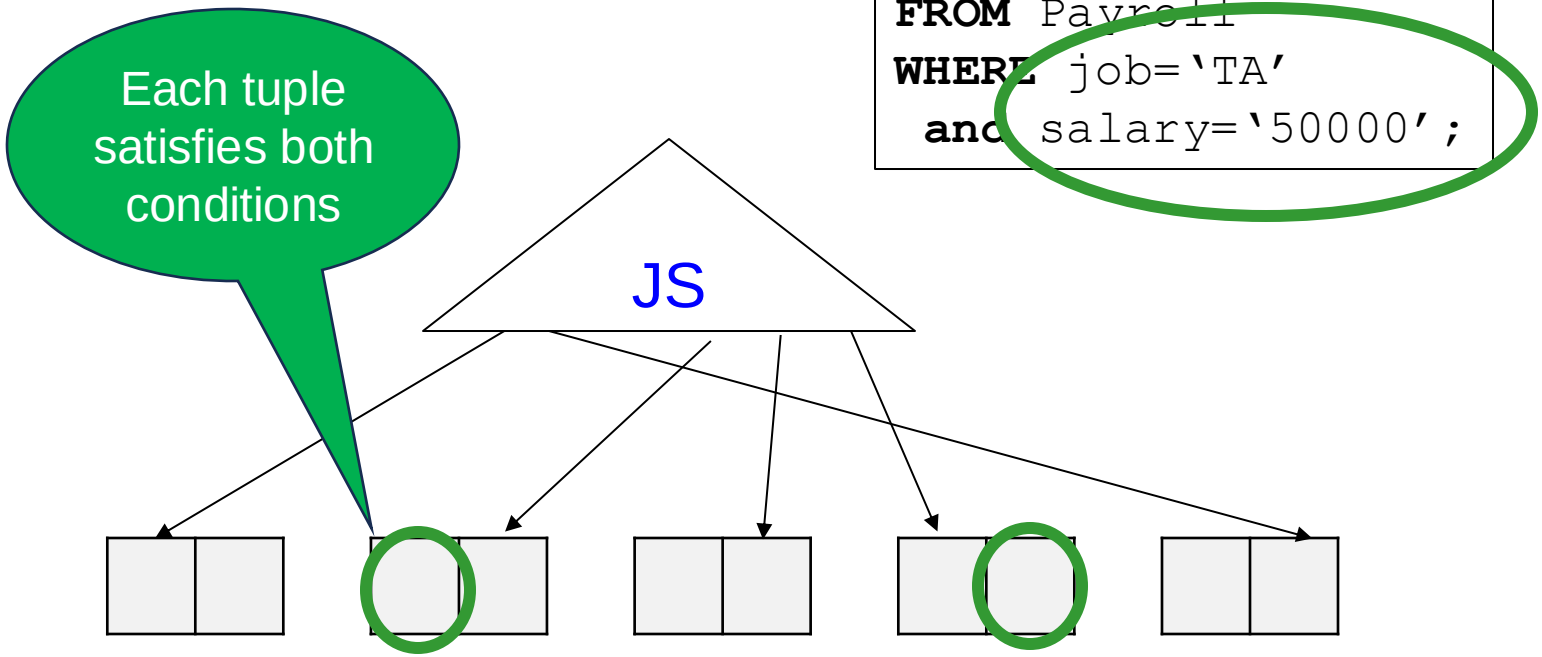
Payroll:

Multi-attribute Index

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX JS on Payroll(job,salary);
```

```
SELECT *
FROM Payroll
WHERE job='TA'
  and salary='50000';
```



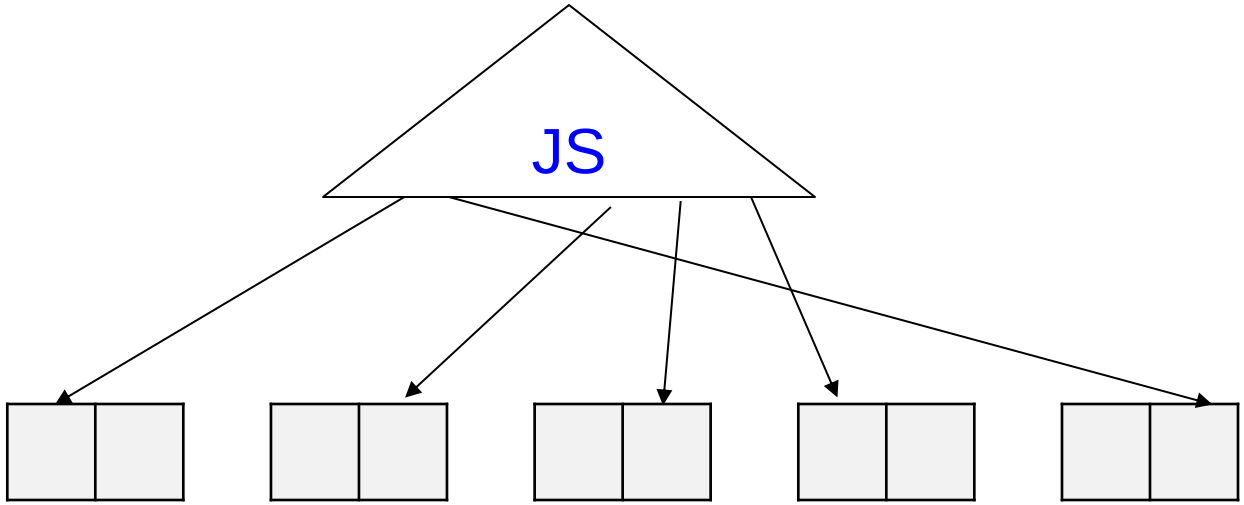
Multi-attribute Index

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX JS on Payroll(job,salary);
```

Can we still use the index?

```
SELECT *
FROM Payroll
WHERE job='TA';
```



Payroll:

Multi-attribute Index

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

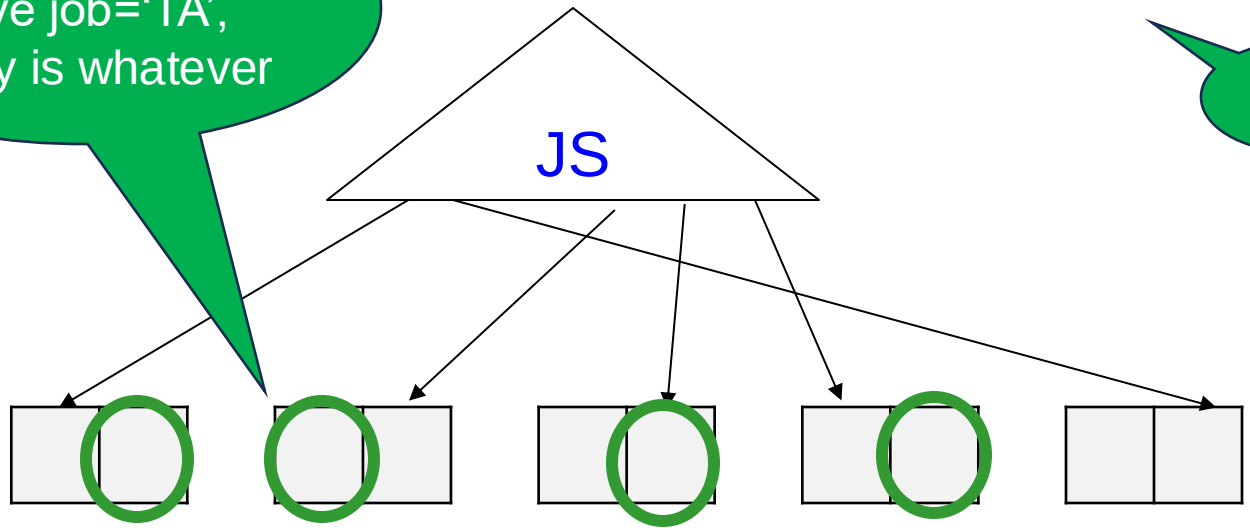
```
CREATE INDEX JS on Payroll(job,salary);
```

Can we still use the index?

```
SELECT *
FROM Payroll
WHERE job='TA';
```

YES

All these tuples have job='TA', salary is whatever



Payroll:

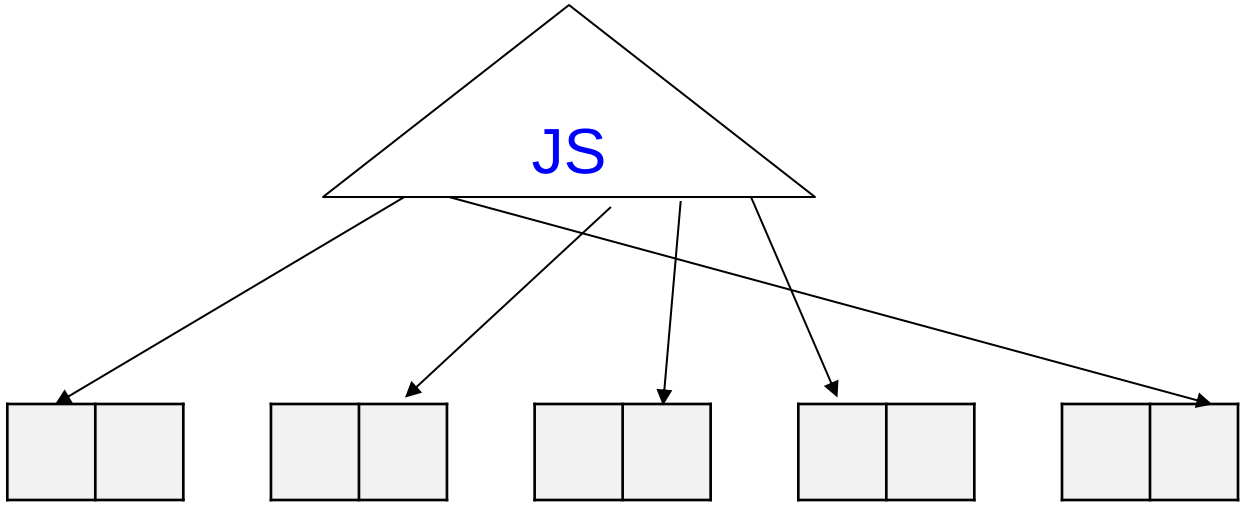
Multi-attribute Index

```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX JS on Payroll(job,salary);
```

Can we still use the index?

```
SELECT *
FROM Payroll
WHERE salary='50000';
```



Payroll:

Multi-attribute Index

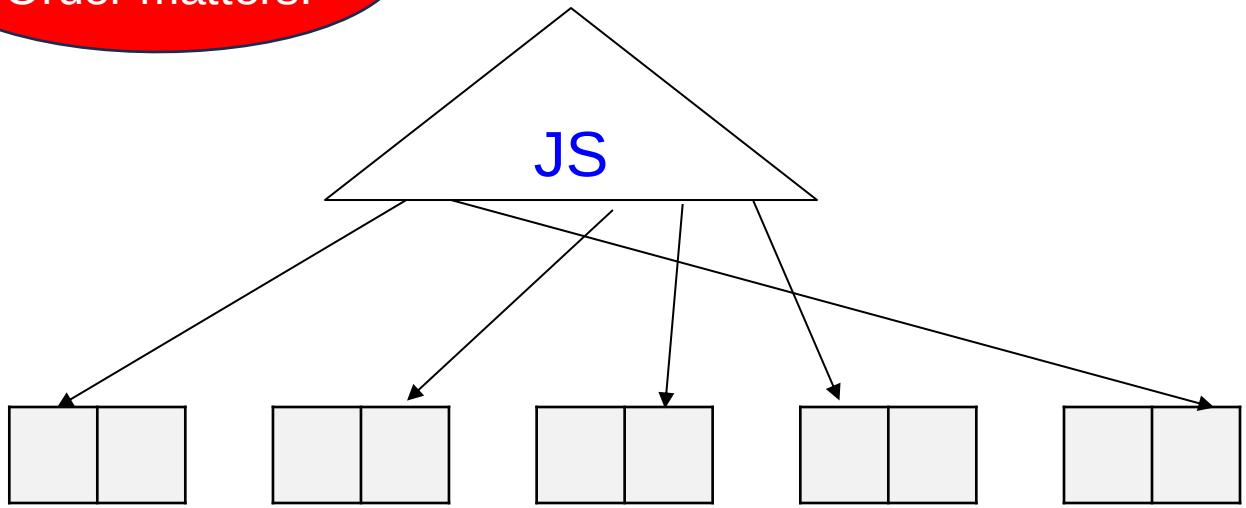
```
CREATE TABLE Payroll(UserID int, name text, job text, salary int)
```

```
CREATE INDEX JS on Payroll(job,salary);
```

NO
Order matters!

Can we still
use the index?

```
SELECT *
FROM Payroll
WHERE salary='50000';
```



Payroll:

Multi-attribute Index

- A pair ordered lexicographically:
  - (‘Director’,200000) < (‘Prof’,50000)<(‘Prof’,200000)<…
- Only 1st attribute is known: range seach
  - Find (‘Prof’, *)
- Only 2nd attribute is known: impossible
  - Find (*, 200000)

Cardinality Estimation

Overview

- DBMS keeps stats for each relation R(A,B,C,…):
  - Cardinality of R: T(R)
  - Number of distinct values in R.A: V(R,A)
    V(R, B), V(R, C), …
  - Histograms

- Cardinality estimation:
  Given only these stats, estimate output size.

Cardinality Estimation

Recursively on the query plan

Cardinality Estimation

Recursively on the query plan

- For a base table: Est(R) = T(R)

Cardinality Estimation

Recursively on the query plan

- For a base table: Est(R) = T(R)
- For an operator:
  $Est(\sigma_{pred}(R)) = \theta_{pred} * Est(R)$

Cardinality Estimation

Recursively on the query plan

- For a base table: Est(R) = T(R)

- For an operator:

$$Est(\sigma_{pred}(R)) = \theta_{pred} * Est(R)$$

$$Est(R \bowtie_{A=B} S) = \theta_{A=B} * Est(R) * Est(S)$$

Cardinality Estimation

Recursively on the query plan

- For a base table: Est(R) = T(R)

- For an operator:

$$Est(\sigma_{pred}(R)) = \theta_{pred} * Est(R)$$
$$Est(R \bowtie_{A=B} S) = \theta_{A=B} * Est(R) * Est(S)$$

θ is called the *selectivity factor*

Selectivity Factor

For $\sigma_{A=const}$ the selectivity factor is $\theta = \frac{1}{V(R,A)}$

Uniformity assumption

Selectivity Factor

For $\sigma_{A=const}$ the selectivity factor is $\theta = \frac{1}{V(R,A)}$

Uniformity assumption

T(Payroll) = 10000
V(Payroll, job) = 20

$\text{EST}\left[\sigma_{job='TA'}(\text{Payroll})\right] = ??$

Selectivity Factor

For $\sigma_{A=const}$ the selectivity factor is $\theta = \frac{1}{V(R,A)}$

Uniformity assumption

T(Payroll) = 10000
V(Payroll, job) = 20

$$EST\left[\sigma_{job='TA'}(Payroll)\right] = \frac{1}{20}10000 = 500$$

Selectivity Factor

For $\sigma_{p \text{ and } q}$ the sel. factor is $\theta_{p \text{ and } q} = \theta_p \times \theta_q$

Independence assumption

Selectivity Factor

For $\sigma_{p \text{ and } q}$ the sel. factor is $\theta_{p \text{ and } q} = \theta_p \times \theta_q$

Independence assumption

T(Payroll) = 10000
V(Payroll, job) = 20
V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{job}=\text{'TA'} \text{ and } \text{salary}=20000}(\text{Payroll})\right] = ??$$

Selectivity Factor

For $\sigma_{p \text{ and } q}$ the sel. factor is $\theta_{p \text{ and } q} = \theta_p \times \theta_q$

Independence assumption

T(Payroll) = 10000
V(Payroll, job) = 20
V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{job}=\text{'TA'} \text{ and } \text{salary}=20000}(\text{Payroll})\right] = \frac{1}{20}\frac{1}{50}10000 = 10$$

Selectivity Factor

For $\sigma_{p \text{ and } q}$ the sel. factor is $\theta_{p \text{ and } q} = \theta_p \times \theta_q$

Independence assumption

T(Payroll) = 10000
V(Payroll, job) = 20
V(Payroll, salary) = 50

This is likely an underestimate

$$\text{EST}\left[\sigma_{\text{job}=\text{'TA'} \text{ and } \text{salary}=20000}(\text{Payroll})\right] = \frac{1}{20}\frac{1}{50}10000 = 10$$

Selectivity Factor

$R \bowtie_{A=B} S$ the sel factor is $\theta = \frac{1}{\max(V(R,A),V(S,B))}$

Containment of values assumption

Selectivity Factor

$R \bowtie_{A=B} S$ the sel factor is $\theta = \frac{1}{\max(V(R,A),V(S,B))}$

Containment of values assumption

Justification:

- If $V(R,A) \leq V(S,B)$ then assume $R.A \subseteq S.B$
- 1 tuple in R joins $\frac{1}{V(S,B)} T(S)$ tuples in S
- Total output: $\frac{1}{V(S,B)} T(R)T(S)$

Selectivity Factor

$R \bowtie_{A=B} S$ the sel factor is $\theta = \frac{1}{\max(V(R,A),V(S,B))}$

Containment of values assumption

Justification:

- If $V(R,A) \leq V(S,B)$ then assume $R.A \subseteq S.B$
- 1 tuple in R joins $\frac{1}{V(S,B)} T(S)$ tuples in S
- Total output: $\frac{1}{V(S,B)} T(R)T(S)$

T(Payroll) = 10000
V(Payroll, UserID) = 10000

T(Regist) = 3000
V(Regist, UserID) = 2000

$EST[Payroll \bowtie Regist] =$??

Selectivity Factor

$R \bowtie_{A=B} S$ the sel factor is $\theta = \frac{1}{\max(V(R,A),V(S,B))}$

Containment of values assumption

Justification:

- If $V(R,A) \leq V(S,B)$ then assume $R.A \subseteq S.B$
- 1 tuple in R joins $\frac{1}{V(S,B)} T(S)$ tuples in S
- Total output: $\frac{1}{V(S,B)} T(R)T(S)$

T(Payroll) = 10000
V(Payroll, UserID) = 10000

T(Regist) = 3000
V(Regist, UserID) = 2000

$$EST[Payroll \bowtie Regist] = \frac{1}{\max(10000,2000)} 10000 \cdot 3000 = 3000$$

Summary of Assumptions

- Uniformity
- Independence
- Containment of values
- Preservation of values

Computing the Cost of a Plan

- Estimate cardinalities bottom-up
- Estimate cost by using estimated cardinalities

Histograms

- Histogram on R.A refines T(R), V(R,A)
- Each bucket contains: T(bucket), V(bucket, A)

Histograms

T(Payroll) = 10000, V(Payroll, salary) = 50

$$\mathrm{EST}\left[\sigma_{\mathrm{salary}=25\mathrm{k}}(\mathrm{Payroll})\right] = ???$$

Histograms

T(Payroll) = 10000, V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = ??? \qquad = \frac{1}{50} 10000 = 10$$

Histograms

T(Payroll) = 10000, V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = ??? \qquad = \frac{1}{50}10000 = 10$$

Salary:	0..20k	20k..29k	30k-39k	40k-49k	50k-59k	> 60k
T =	80	320	2000	4800	2600	200
V =	5	10	10	15	5	5

Histograms

T(Payroll) = 10000, V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = ??? \qquad = \frac{1}{50} 10000 = 10$$

Salary:	0..20k	20k..29k	30k-39k	40k-49k	50k-59k	> 60k
T =	80	320	2000	4800	2600	200
V =	5	10	10	15	5	5

Histograms

T(Payroll) = 10000, V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = ??? \qquad = \frac{1}{50}10000 = 10$$

Salary:	0..20k	20k..29k	30k-39k	40k-49k	50k-59k	> 60k
T =	80	320	2000	4800	2600	200
V =	5	10	10	15	5	5

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = \frac{1}{10}320 = 32$$

Histograms

T(Payroll) = 10000, V(Payroll, salary) = 50

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = ??? \qquad = \frac{1}{50}10000 = 10$$

Salary:	0..20k	20k..29k	30k-39k	40k-49k	50k-59k	> 60k
T =	80	320	2000	4800	2600	200
V =	5	10	10	15	5	5

$$\text{EST}\left[\sigma_{\text{salary}=25\text{k}}(\text{Payroll})\right] = \frac{1}{10}320 = 32$$

A better estimate

Recap: How a Query Engine Works

- Each RA operator has multiple physical operators
- Indexes: speed up some queries slow down others
- Rewrite rules transform one query plan to another
- Cardinality estimators used to estimate the cost; they are often wrong

Real Database Engines

Leis et al., *How Good Are Query Optimizers, Really?*, VLDB 2015

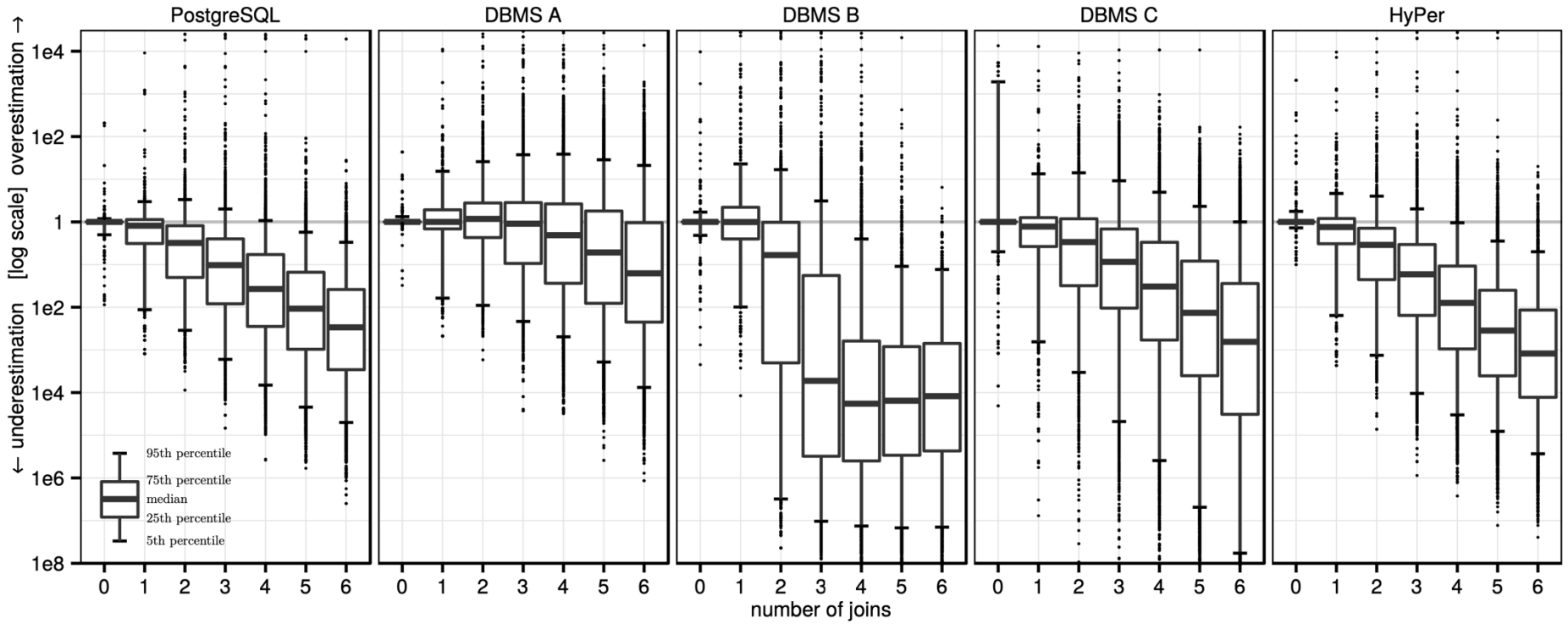


Figure 3: Quality of cardinality estimates for multi-join queries in comparison with the true cardinalities. Each boxplot summarizes the error distribution of all subexpressions with a particular size (over all queries in the workload)

Real Database Engines

Leis et al., *How Good Are Query Optimizers, Really?*, VLDB 2015

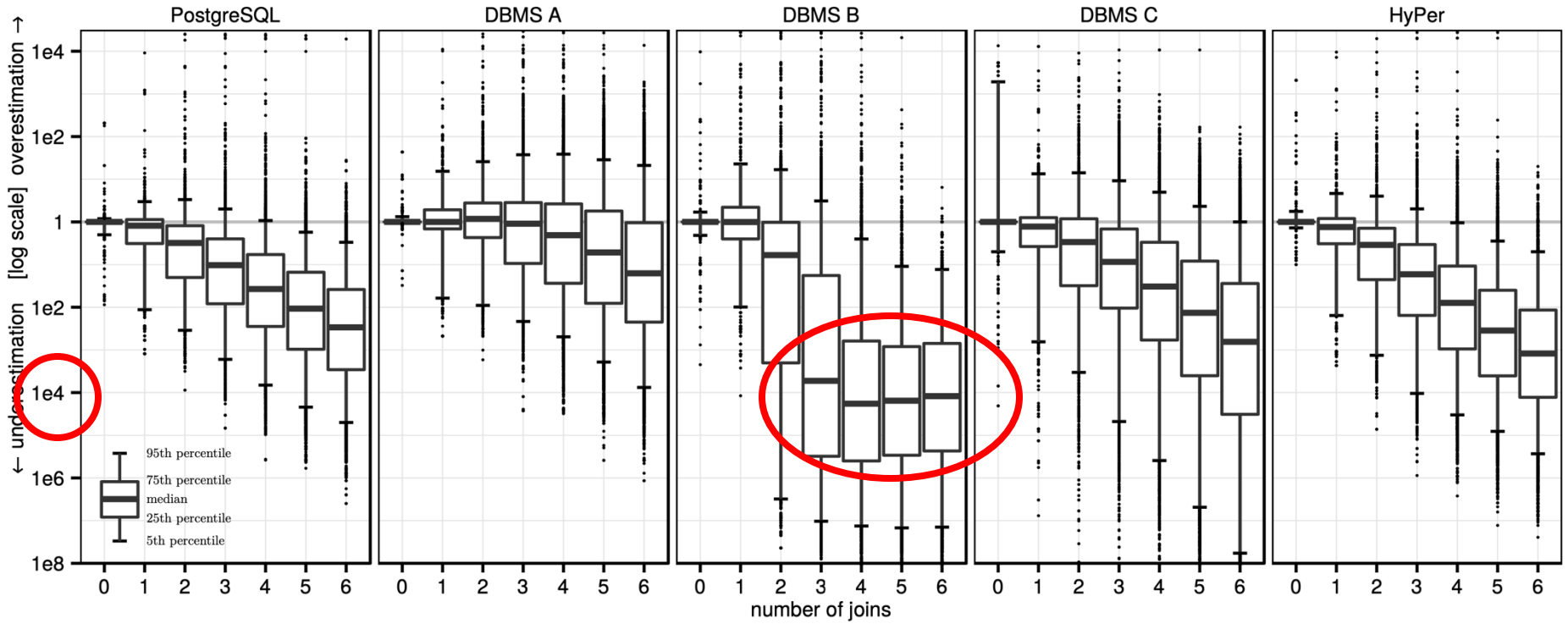


Figure 3: Quality of cardinality estimates for multi-join queries in comparison with the true cardinalities. Each boxplot summarizes the error distribution of all subexpressions with a particular size (over all queries in the workload)