PLAN FOR TODAY

Selectivity Estimation

Query Optimisation
**Selectivity Estimation**

To estimate the cost of a query, we add up the estimated costs of each operator in the query.

Need to know the size of the intermediate relations (generated from one operator and passed into another) in order to do this!

Need to know the selectivity of predicates - what % of tuples are selected by a predicate.

These are all estimates... if we don’t know, we make up a value for it (e.g., selectivity = 1/10)
## Selectivity Estimation: Equalities

<table>
<thead>
<tr>
<th>PREDICATE</th>
<th>SELECTIVITY</th>
<th>ASSUMPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = value</td>
<td>1 / (# distinct values of A in relation)</td>
<td>We know</td>
</tr>
<tr>
<td>A = value</td>
<td>1 / 10</td>
<td>We don’t know</td>
</tr>
<tr>
<td>A = B</td>
<td>1 / MAX (# distinct A-values, # distinct B-values)</td>
<td>We know</td>
</tr>
<tr>
<td>A = B</td>
<td>1 / (# distinct values of A)</td>
<td>We know</td>
</tr>
<tr>
<td>A = B</td>
<td>1 / 10</td>
<td>We don’t know</td>
</tr>
</tbody>
</table>

|column| = the number of distinct values for the column

If you have an index on column A, you can assume you know |A|, max(A), and min(A)

When using selectivity to compute # of tuples, round up the result (e.g. 245.7 → 246 tuples)
## Selectivity Estimation: Inequalities on Integers

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Selectivity</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A &lt; c$</td>
<td>$\left(\frac{c - \min(A)}{(\max(A) - \min(A) + 1)}\right)$</td>
<td>We know $\max(A)$ and $\min(A)$</td>
</tr>
<tr>
<td>$A \leq c$</td>
<td>$\left(\frac{c - \min(A) + 1}{(\max(A) - \min(A) + 1)}\right)$</td>
<td>$c$ is an integer</td>
</tr>
<tr>
<td>$A &lt; c$</td>
<td>$\frac{1}{3}$</td>
<td>We don’t know $\max(A)$ and $\min(A)$</td>
</tr>
<tr>
<td>$A \leq c$</td>
<td>$\frac{1}{3}$</td>
<td>$c$ is an integer</td>
</tr>
<tr>
<td>$A &gt; c$</td>
<td>$\left(\frac{\max(A) - c}{(\max(A) - \min(A) + 1)}\right)$</td>
<td>We know $\max(A)$ and $\min(A)$</td>
</tr>
<tr>
<td>$A \geq c$</td>
<td>$\left(\frac{\max(A) - c + 1}{(\max(A) - \min(A) + 1)}\right)$</td>
<td>$c$ is an integer</td>
</tr>
<tr>
<td>$A &gt; c$</td>
<td>$\frac{1}{3}$</td>
<td>We don’t know $\max(A)$ and $\min(A)$</td>
</tr>
<tr>
<td>$A \geq c$</td>
<td>$\frac{1}{3}$</td>
<td>$c$ is an integer</td>
</tr>
</tbody>
</table>

* We add 1 to the denominator in order for our [low, high] range to be inclusive
  
  E.g. range $[2, 4] = 2, 3, 4 \rightarrow (4 - 2) + 1 = 3$
## Selectivity Estimation: Inequalities on Floats

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Selectivity</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &lt; c</td>
<td>(c - min(A)) / (max(A) - min(A))</td>
<td>We know max(A) and min(A) c is a float</td>
</tr>
<tr>
<td>A ≤ c</td>
<td>1 / 3</td>
<td>We don’t know max(A) and min(A) c is a float</td>
</tr>
<tr>
<td>A &gt; c</td>
<td>(max(A) - c) / (max(A) - min(A))</td>
<td>We know max(A) and min(A) c is a float</td>
</tr>
<tr>
<td>A ≥ c</td>
<td>1 / 3</td>
<td>We don’t know max(A) and min(A) c is a float</td>
</tr>
</tbody>
</table>

* We don’t add 1 to the denominator (floats are continuous, integers are discrete)
  E.g. range [2.0, 4.0] = 2.0, 2.1, ..., 3.9, 4.0 → 4.0 - 2.0 = 2.0
**Selectivity Estimation: Connectives**

<table>
<thead>
<tr>
<th>PREDICATE</th>
<th>SELECTIVITY</th>
<th>ASSUMPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1 AND p2</td>
<td>sel(p1) · sel(p2)</td>
<td>Independent predicates</td>
</tr>
<tr>
<td>p1 OR p2</td>
<td>sel(p1) + sel(p2) – sel(p1) · sel(p2)</td>
<td>Independent predicates</td>
</tr>
<tr>
<td>NOT p</td>
<td>1 – sel(p)</td>
<td></td>
</tr>
</tbody>
</table>
Selectivity Estimation

How many tuples are selected by the following query?

```sql
SELECT * FROM R
```

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R
```

1000 tuples
(no predicates, select all)

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```sql
SELECT * FROM R
WHERE A = 42
```

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

\[
\text{SELECT } * \text{ FROM } R \\
\text{WHERE } A = 42
\]

50 unique values in A

\[
\frac{1}{50} \cdot (1000 \text{ tuples}) = 20 \text{ tuples}
\]

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
**Selectivity Estimation**

How many tuples are selected by the following query?

```sql
SELECT * FROM R
WHERE C = 42
```

<table>
<thead>
<tr>
<th>R(A,B,C) has 1000 tuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute A:</td>
</tr>
<tr>
<td>50 unique integers, uniformly distributed in the range [1, 50]</td>
</tr>
<tr>
<td>Attribute B:</td>
</tr>
<tr>
<td>100 unique floats, uniformly distributed in the range [1, 100]</td>
</tr>
</tbody>
</table>
**Selectivity Estimation**

How many tuples are selected by the following query?

```sql
SELECT * FROM R
WHERE C = 42
```

No information about C

\[
\frac{1}{10} \cdot (1000 \text{ tuples}) = 100 \text{ tuples}
\]

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range \([1, 50]\)

Attribute B:
100 unique floats, uniformly distributed in the range \([1, 100]\)
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A <= 25
```
Selectivity Estimation

How many tuples are selected by the following query?

\[
\text{SELECT} \ \ast \ \text{FROM} \ R \\
\text{WHERE} \ A \leq 25
\]

\[
\text{sel}(A \leq 25) = \\
= (25 - 1 + 1) / (50 - 1 + 1) \\
= 1/2
\]

\[
1/2 \cdot (1000 \text{ tuples}) = 500 \text{ tuples}
\]

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range \([1, 50]\)

Attribute B:
100 unique floats, uniformly distributed in the range \([1, 100]\)
How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE B <= 25
```

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE B <= 25
```

\[
\text{sel}(B \leq 25) = \frac{(25 - 1)}{(100 - 1)} = \frac{24}{99} = 0.2424...
\]

\[
\text{round}(0.2424... \cdot (1000 \text{ tuples})) = 242 \text{ tuples}
\]

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE C <= 25
```
**Selectivity Estimation**

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE C <= 25
```

No information about C

```
round(1/3 \cdot (1000 \text{ tuples})) = 333 \text{ tuples}
```

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A <= 25
AND B <= 25
```

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
**Selectivity Estimation**

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A <= 25
AND B <= 25
```

\[
\text{sel}(A \leq 25) \cdot \text{sel}(B \leq 25) = \frac{1}{2} \cdot \frac{24}{99} = \frac{12}{99}
\]

\[
= 0.1212... \\
\]

\[
\text{round}(0.1212... \cdot (1000 \text{ tuples})) = 121 \text{ tuples}
\]

R(A,B,C) has 1000 tuples

Attribute A:

50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:

100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A <= 25
  OR B <= 25
```
**Selectivity Estimation**

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A <= 25
  OR B <= 25
```

\[
\text{sel}(A \leq 25) + \text{sel}(B \leq 25) - \text{sel}(A \leq 25) \cdot \text{sel}(B \leq 25) \\
= \frac{1}{2} + \frac{24}{99} - \frac{1}{2} \cdot \frac{24}{99} = 0.62121...
\]

\[\text{round}(0.62121... \cdot (1000 \text{ tuples})) = 621 \text{ tuples}\]

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A = C
```
**Selectivity Estimation**

How many tuples are selected by the following query?

```
SELECT * FROM R
WHERE A = C
```

No information about C

\[
\frac{1}{50} \cdot (1000 \text{ tuples}) = 20 \text{ tuples}
\]

R(A,B,C) has 1000 tuples

Attribute A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute B:
100 unique floats, uniformly distributed in the range [1, 100]
Selectivity Estimation

How many tuples are selected by the following query?

```
SELECT * FROM R, S
WHERE R.A = S.D
```
**Selectivity Estimation**

How many tuples are selected by the following query?

```
SELECT * FROM R, S
WHERE R.A = S.D
```

Max output size = $|R| \cdot |S|$

$\text{sel}(R.A = S.D) = \frac{1}{\text{MAX}(50, 25)} = \frac{1}{50}$

$\frac{1}{50} \cdot (1000 \cdot 500) = 10,000$ tuples

R(A,B,C) has 1000 tuples

S(D, E) has 500 tuples

Attribute R.A:
50 unique integers, uniformly distributed in the range [1, 50]

Attribute S.D:
25 unique integers, uniformly distributed in the range [1, 25]
QUERY OPTIMISATION
We can represent relational algebra expressions as trees.

Order of operators affects I/Os and resource usage, but not necessarily output.

\[ \pi_{\text{sname}}(\sigma_{\text{bid}=100 \land \text{rating} > 5} (\text{Reserves} \bowtie \text{Sailors})) \]
Given a plan, some things we can do are:

**Push selections/projections down the tree**

- The earlier we reduce the size of input, the fewer I/Os are incurred as we traverse up the tree
- Reduces I/O cost if materialized

\[
\text{Reserves} \bowtie_{\text{sid} = \text{sid}} \pi_{\text{sname}} \sigma_{\text{bid}=100 \land \text{rating} > 5} \\
\text{Sailors} \bowtie_{\text{sid} = \text{sid}} \pi_{\text{sname}} \sigma_{\text{bid}=100 \land \text{rating} > 5}
\]
Query Optimisation – Alternate Plans

Given a plan, some things we can do are:

- Push selections/projections down the tree
- Materialize intermediate relations (write to a temp file)
  - Results in additional write I/Os, but is better in the long run
- Use indices (e.g., INLJ)
Table R consists of 50 pages
Table S consists of 100 pages
sel(S.age < 25) = 0.5

Without materializing, we are performing $\sigma_{\text{age} < 25}$ on the fly each time in PNLJ, and scanning the entire table S for each page of R

Cost = Scan R (50) + PNLJ (50 \cdot 100) = 5,050 I/Os
Table R consists of 50 pages
Table S consists of 100 pages
sel(S.age < 25) = 0.5

By materializing the intermediate relation, we are
applying $\sigma_{\text{age} < 25}$ before PNLJ, and performing
the join on the result of the selection

Cost = Scan R (50) + Scan S (100) +
Materialise (50) + PNLJ (50 \cdot 50) = 2,700 I/Os
A query optimiser takes in a query plan (e.g., one directly translated from a SQL query) and outputs a better (hopefully optimal) query plan

Works on and optimizes over a plan space (set of all plans considered)

Performs cost estimation on query plans

Uses a search algorithm to search through plan space to find plan with lowest cost estimate

May not be optimal (bad estimate or small plan space considered)
We will be looking at the System R optimiser (aka Selinger optimiser)

**Plan space**
- Only left-deep trees, avoid Cartesian products unless they are the only option
- **Left-deep trees** represent a plan where all new tables are joined one at a time from the right

**Cost estimation**
- Actual System R optimiser incorporates both CPU and I/O cost
- We will only use I/O cost in this course

**Search algorithm**
- Dynamic programming
Only consider left-deep plans

- **left-deep**
  $$\left(\left(R \bowtie S\right) \bowtie T\right) \bowtie U$$

- **bushy**
  $$\left(\left(R \bowtie S\right) \bowtie \left(T \bowtie U\right)\right)$$

- **right-deep**
  $$T \bowtie \left(\left(U \bowtie \left(S \bowtie R\right)\right)\right)$$
**Query Optimisation – System R**

Why only left-deep trees?

- Join new tables one at a time from the right
- Create an ordering in which to add tables to the query being executed
- Too many possible trees for joins
  - Using only left-deep trees: $N!$ different ways to order relations
  - Including all permutations tree layouts: A very large number of ways to parenthesize given an ordering (superexponential in $N$)

<table>
<thead>
<tr>
<th># of relations $n$</th>
<th># of different join trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>1,680</td>
</tr>
<tr>
<td>6</td>
<td>30,240</td>
</tr>
<tr>
<td>7</td>
<td>665,280</td>
</tr>
<tr>
<td>8</td>
<td>17,297,280</td>
</tr>
<tr>
<td>10</td>
<td>17,643,225,600</td>
</tr>
</tbody>
</table>
Search algorithm for System R: use dynamic programming

Based on the principle of optimality

Runtime drops from $n!$ to around $n \cdot 2^n$

To be considered, must be:

- Left deep
- No Cartesian products

(I.e. if we join R and S on <cond1> and we join S and T on <cond2>, we don’t consider joining R and T if there’s no condition between them)
Query Optimisation – System R

For $N$ relations joined, perform $N$ passes

On the $i$-th pass, output only the best plan for joining any $i$ of the $N$ relations

Also keep around plans that have higher cost but have an interesting order

Interesting orders are orderings on intermediate relations that may help reduce the cost of later operators (e.g., joins, sorting, hashing)

ORDER BY attributes

GROUP BY attributes

downstream join attributes
System R Optimisation: Example

Pass 1:
Find minimum cost access method for each (relation, interesting order) pair

   Index scan, full table scans

A toy example:

```
SELECT * FROM R, S, T
WHERE R.B = S.B
AND S.C = T.C
AND R.A <= 50
```
Pass 1:
Assume the single table access plans have the following IO costs:

```
SELECT * FROM R, S, T
WHERE R.B = S.B
AND S.C = T.C
AND R.A <= 50
```

<table>
<thead>
<tr>
<th>Operation</th>
<th>IO Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full scan on R</td>
<td>1000 I/Os</td>
</tr>
<tr>
<td>Index scan on R.A</td>
<td>200 I/Os</td>
</tr>
<tr>
<td>Index scan on R.B</td>
<td>1100 I/Os</td>
</tr>
<tr>
<td>Full scan on S</td>
<td>2000 I/Os</td>
</tr>
<tr>
<td>Index scan on S.B</td>
<td>2500 I/Os</td>
</tr>
<tr>
<td>Full scan on T</td>
<td>3000 I/Os</td>
</tr>
<tr>
<td>Index scan on T.C</td>
<td>3500 I/Os</td>
</tr>
<tr>
<td>Index scan on T.D</td>
<td>3500 I/Os</td>
</tr>
</tbody>
</table>
**System R Optimisation: Example**

Pass 1:
Which single table access plans advance to the next stage?

```
SELECT * FROM R, S, T
WHERE R.B = S.B
    AND S.C = T.C
    AND R.A <= 50
```

<table>
<thead>
<tr>
<th>Access Plan</th>
<th>I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full scan on R</td>
<td>1000</td>
</tr>
<tr>
<td>Index scan on R.A</td>
<td>200</td>
</tr>
<tr>
<td>(sorted on R.A)</td>
<td></td>
</tr>
<tr>
<td>Index scan on R.B</td>
<td>1100</td>
</tr>
<tr>
<td>(sorted on R.B)</td>
<td></td>
</tr>
<tr>
<td>Full scan on S</td>
<td>2000</td>
</tr>
<tr>
<td>Index scan on S.B</td>
<td>2500</td>
</tr>
<tr>
<td>(sorted on S.B)</td>
<td></td>
</tr>
<tr>
<td>Full scan on T</td>
<td>3000</td>
</tr>
<tr>
<td>Index scan on T.C</td>
<td>3500</td>
</tr>
<tr>
<td>(sorted on T.C)</td>
<td></td>
</tr>
<tr>
<td>Index scan on T.D</td>
<td>3500</td>
</tr>
<tr>
<td>(sorted on T.D)</td>
<td></td>
</tr>
</tbody>
</table>
Pass 1:
Which single table access plans advance to the next stage?

```
SELECT * FROM R, S, T
WHERE R.B = S.B
AND S.C = T.C
AND R.A <= 50
```
Pass 1:
Which single table access plans advance to the next stage?

- Index scan on R.A (best overall plan for R)
- Index scan on R.B (output sorted on R.B and R.B is used in a join)
- Full scan on S (best overall plan for S)
- Index scan on S.B (output sorted on S.B and S.B is used in a join)
- Full scan on T (best overall plan for T)
- Index scan on T.C (output sorted on T.C and T.C is used in a join)
**System R Optimisation: Example**

Assume the following join costs:

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>R BNLJ S</td>
<td>21,000 I/Os</td>
</tr>
<tr>
<td>R SMj S</td>
<td>3,600 I/Os</td>
</tr>
<tr>
<td>S BNLJ R</td>
<td>18,000 I/Os</td>
</tr>
<tr>
<td>S SMj R</td>
<td>3,000 I/Os</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>R BNLJ T</td>
<td>30,000 I/Os</td>
</tr>
<tr>
<td>R SMj T</td>
<td>40,000 I/Os</td>
</tr>
<tr>
<td>T BNLJ R</td>
<td>35,000 I/Os</td>
</tr>
<tr>
<td>T SMj R</td>
<td>20,000 I/Os</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>S BNLJ T</td>
<td>15,000 I/Os</td>
</tr>
<tr>
<td>S SMj T</td>
<td>10,000 I/Os</td>
</tr>
<tr>
<td>T BNLJ S</td>
<td>25,000 I/Os</td>
</tr>
<tr>
<td>T SMj S</td>
<td>30,000 I/Os</td>
</tr>
</tbody>
</table>

**SELECT * FROM R, S, T**

**WHERE**

- R.B = S.B
- S.C = T.C
- R.A <= 50

Which of these joins will actually be considered by the query optimiser on pass 2?
Assume the following join costs:

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R \bowtie_{BNLJ} S$</td>
<td>21,000 I/Os</td>
</tr>
<tr>
<td>$R \bowtie_{SMJ} S$</td>
<td>3,600 I/Os</td>
</tr>
<tr>
<td>$S \bowtie_{BNLJ} R$</td>
<td>18,000 I/Os</td>
</tr>
<tr>
<td>$S \bowtie_{SMJ} R$</td>
<td>3,000 I/Os</td>
</tr>
<tr>
<td>$R \bowtie_{BNLJ} T$</td>
<td>30,000 I/Os</td>
</tr>
<tr>
<td>$R \bowtie_{SMJ} T$</td>
<td>40,000 I/Os</td>
</tr>
<tr>
<td>$T \bowtie_{BNLJ} R$</td>
<td>35,000 I/Os</td>
</tr>
<tr>
<td>$T \bowtie_{SMJ} R$</td>
<td>20,000 I/Os</td>
</tr>
<tr>
<td>$S \bowtie_{BNLJ} T$</td>
<td>15,000 I/Os</td>
</tr>
<tr>
<td>$S \bowtie_{SMJ} T$</td>
<td>10,000 I/Os</td>
</tr>
<tr>
<td>$T \bowtie_{BNLJ} S$</td>
<td>25,000 I/Os</td>
</tr>
<tr>
<td>$T \bowtie_{SMJ} S$</td>
<td>30,000 I/Os</td>
</tr>
</tbody>
</table>

**SELECT * FROM** $R$, $S$, $T$

**WHERE** $R.B = S.B$

**AND** $S.C = T.C$

**AND** $R.A \leq 50$

Which of these joins will actually be considered by the query optimiser on Pass 2?

**Eliminate Cartesian products (joins between $R$ and $T$)**
System R Optimisation: Example

Assume the following join costs:

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>R ⋈_{BNLJ} S</td>
<td>21,000 I/Os</td>
</tr>
<tr>
<td>R ⋈_{SMJ} S</td>
<td>3,600 I/Os</td>
</tr>
<tr>
<td>S ⋈_{BNLJ} R</td>
<td>18,000 I/Os</td>
</tr>
<tr>
<td>S ⋈_{SMJ} R</td>
<td>3,000 I/Os</td>
</tr>
<tr>
<td>R ⋈_{BNLJ} T</td>
<td>30,000 I/Os</td>
</tr>
<tr>
<td>R ⋈_{SMJ} T</td>
<td>40,000 I/Os</td>
</tr>
<tr>
<td>T ⋈_{BNLJ} R</td>
<td>35,000 I/Os</td>
</tr>
<tr>
<td>T ⋈_{SMJ} R</td>
<td>20,000 I/Os</td>
</tr>
<tr>
<td>S ⋈_{BNLJ} T</td>
<td>15,000 I/Os</td>
</tr>
<tr>
<td>S ⋈_{SMJ} T</td>
<td>10,000 I/Os</td>
</tr>
<tr>
<td>T ⋈_{BNLJ} S</td>
<td>25,000 I/Os</td>
</tr>
<tr>
<td>T ⋈_{SMJ} S</td>
<td>30,000 I/Os</td>
</tr>
</tbody>
</table>

SELECT * FROM R, S, T
WHERE R.B = S.B
AND S.C = T.C
AND R.A <= 50

Which of these joins will advance to the next pass of the query optimiser?
System R Optimisation: Example

Assume the following join costs:

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost (I/Os)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (\bowtie) BNLJ S</td>
<td>21,000</td>
</tr>
<tr>
<td>R (\bowtie) SMJ S</td>
<td>3,600</td>
</tr>
<tr>
<td>S (\bowtie) BNLJ R</td>
<td>18,000</td>
</tr>
<tr>
<td>S (\bowtie) SMJ R</td>
<td>3,000</td>
</tr>
<tr>
<td>R (\bowtie) BNLJ T</td>
<td>30,000</td>
</tr>
<tr>
<td>R (\bowtie) SMJ T</td>
<td>40,000</td>
</tr>
<tr>
<td>T (\bowtie) BNLJ R</td>
<td>35,000</td>
</tr>
<tr>
<td>T (\bowtie) SMJ R</td>
<td>20,000</td>
</tr>
<tr>
<td>S (\bowtie) BNLJ T</td>
<td>15,000</td>
</tr>
<tr>
<td>S (\bowtie) SMJ T</td>
<td>10,000</td>
</tr>
<tr>
<td>T (\bowtie) BNLJ S</td>
<td>25,000</td>
</tr>
<tr>
<td>T (\bowtie) SMJ S</td>
<td>30,000</td>
</tr>
</tbody>
</table>

SELECT \* FROM R, S, T
WHERE R.B = S.B
AND S.C = T.C
AND R.A <= 50

Which of these joins will advance to the next pass of the query optimiser?

None of the joins produce an interesting order (no downstream joins, ORDER BY, GROUP BY). Only consider best join for each considered set of tables.
SYSTEM R OPTIMISATION: EXAMPLE

Assume the following join costs:

<table>
<thead>
<tr>
<th>Join</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>R ⋈ BNLJ S</td>
<td>21,000 I/Os</td>
</tr>
<tr>
<td>R ⋈ SMj S</td>
<td>3,600 I/Os</td>
</tr>
<tr>
<td>S ⋈ BNLJ R</td>
<td>18,000 I/Os</td>
</tr>
<tr>
<td>S ⋈ SMj R</td>
<td>3,000 I/Os</td>
</tr>
<tr>
<td>R ⋈ BNLJ T</td>
<td>30,000 I/Os</td>
</tr>
<tr>
<td>R ⋈ SMj T</td>
<td>40,000 I/Os</td>
</tr>
<tr>
<td>T ⋈ BNLJ R</td>
<td>35,000 I/Os</td>
</tr>
<tr>
<td>T ⋈ SMj R</td>
<td>20,000 I/Os</td>
</tr>
<tr>
<td>S ⋈ BNLJ T</td>
<td>15,000 I/Os</td>
</tr>
<tr>
<td>S ⋈ SMj T</td>
<td>10,000 I/Os</td>
</tr>
<tr>
<td>T ⋈ BNLJ S</td>
<td>25,000 I/Os</td>
</tr>
<tr>
<td>T ⋈ SMj S</td>
<td>30,000 I/Os</td>
</tr>
</tbody>
</table>

SELECT * FROM R, S, T
WHERE R.B = S.B
AND S.C = T.C
AND R.A <= 50

Will any of these remaining joins produce an interesting order?
**System R Optimisation: Example**

Assume the following join costs:

- $R \bowtie_{BNLJ} S \; 21,000 \text{ I/Os}$
- $R \bowtie_{SMJ} S \; 3,600 \text{ I/Os}$
- $S \bowtie_{BNLJ} R \; 18,000 \text{ I/Os}$
- $S \bowtie_{SMJ} R \; 3,000 \text{ I/Os}$
- $R \bowtie_{BNLJ} T \; 30,000 \text{ I/Os}$
- $R \bowtie_{SMJ} T \; 40,000 \text{ I/Os}$
- $T \bowtie_{BNLJ} R \; 35,000 \text{ I/Os}$
- $T \bowtie_{SMJ} R \; 20,000 \text{ I/Os}$
- $S \bowtie_{BNLJ} T \; 15,000 \text{ I/Os}$
- $S \bowtie_{SMJ} T \; 10,000 \text{ I/Os}$
- $T \bowtie_{BNLJ} S \; 25,000 \text{ I/Os}$
- $T \bowtie_{SMJ} S \; 30,000 \text{ I/Os}$

**SELECT** *
* FROM $R, S, T$
WHERE $R.B = S.B$
AND $S.C = T.C$
AND $R.A \leq 50$

Will any of these remaining joins produce an interesting order?

**No.** $R \bowtie_{SMJ} S$ is sorted on B (not interesting), and $T \bowtie_{SMJ} S$ is sorted on C (not interesting)
How could we modify the query so that $S \bowtie_{SMJ} R$ yields an interesting order?
SYSTEM R OPTIMISATION: EXAMPLE

How could we modify the query so that $S \bowtie_{SMj} R$ yields an interesting order?

$S \bowtie_{SMj} R$ will be sorted on column B, so we need B to be interesting. We could add ORDER BY B, GROUP BY B, or another join condition involving R.B or S.B to the query to make it interesting
**System R Optimisation: Example**

Will the query plan $T \bowtie_{BNLJ} (S \bowtie_{SMJ} R)$ be considered by the final pass of the query optimiser?
System R Optimisation: Example

Will the query plan $T \bowtie_{BNLJ} (S \bowtie_{SMJ} R)$ be considered by the final pass of the query optimiser?

No, this query plan is not left-deep (all join results must be on the left side of their parent join), so it is not considered in the final pass.
COMMIT

END TRANSACTION