Cardinality Estimation in Databases
Logistics

- Exam on Wednesday
- Material covered today won't be on the exam
Roadmap

What is Cardinality Estimation / Where is it used in DBMSs?

Histograms in PostgreSQL

Handling Correlated Columns

Other Operators and Estimation Methods

Estimating Size of Joins

Effect of Cardinality Estimation Errors on Query Optimization
What is Cardinality Estimation?

Estimate Size of any SQL expression

---
Aggregate (cost=251.24..251.25 rows=1 width=8)
  -> Hash Join (cost=14.89..244.63 rows=2643 width=0)
    Hash Cond: (rail_ridership.station_id = stations.station_id)
    -> Seq Scan on rail_ridership (cost=0.00..197.18 rows=982 width=11)
      Filter: (total_ons >= 40000)
    -> Hash (cost=10.85..10.85 rows=323 width=22)
      -> Hash Join (cost=3.70..10.85 rows=323 width=22)
        Hash Cond: (station_orders.station_id = stations.station_id)
        -> Seq Scan on station_orders (cost=0.00..6.26 rows=326 width=11)
          -> Hash (cost=2.20..2.20 rows=120 width=11)
            -> Seq Scan on stations (cost=0.00..2.20 rows=120 width=11)
  (11 rows)
Similarity to Probability

\[ P(X = x_1) = \frac{4}{10} \]
Similarity to Probability

- Data:
  - x1
  - x2
  - x1
  - x2
  - x1
  - x3
  - x3
  - x4

- Selectivity:
  - \( P(X = x_1) = \frac{4}{10} \)
  - \( P(X = x_1 \text{ OR } X = x_2) = \frac{7}{10} \)
How many cardinality estimates are needed for Query Optimization?

- Query optimizer potentially needs estimate for every possible join
- With large join graphs (e.g., 8-16 joins), this can be 100-1000s of estimates
- Therefore, estimator needs to be very fast (~milliseconds)
- Many simple assumptions / trade-offs because of this
Roadmap

What is Cardinality Estimation / Where is it used in DBMSs?

Histograms in PostgreSQL

Handling Correlated Columns

Other Operators and Estimation Methods

Estimating Size of Joins

Effect of Cardinality Estimation Errors on Query Optimization
Statistics Collected by PostgreSQL

Table: `rail_ridership`

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>season</td>
<td>text</td>
</tr>
<tr>
<td>line_id</td>
<td>text</td>
</tr>
<tr>
<td>direction</td>
<td>integer</td>
</tr>
<tr>
<td>time_period_id</td>
<td>text</td>
</tr>
<tr>
<td>station_id</td>
<td>text</td>
</tr>
<tr>
<td><strong>total_ons</strong></td>
<td>integer</td>
</tr>
<tr>
<td>total_offs</td>
<td>integer</td>
</tr>
<tr>
<td>number_service_days</td>
<td>integer</td>
</tr>
<tr>
<td>average_ons</td>
<td>integer</td>
</tr>
<tr>
<td>average_offs</td>
<td>integer</td>
</tr>
<tr>
<td>average_flow</td>
<td>integer</td>
</tr>
</tbody>
</table>

For every column:
- Histograms
- Most Common Values
We want to summarize this information with small storage / inference costs
Histogram (Equal Width)

Divide data into N equal sized bins (e.g., 100 bins)
y axis = fraction of data in that bins; Total = 7854
Histogram (Equal Width)

Divide data into N equal sized bins (e.g., 100 bins)
y axis = fraction of data in that bin

<table>
<thead>
<tr>
<th>Value Bins</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5K</td>
<td>0.61</td>
</tr>
<tr>
<td>5K-10K</td>
<td>0.09</td>
</tr>
<tr>
<td>10K-15K</td>
<td>0.05</td>
</tr>
<tr>
<td>15K-20K</td>
<td>0.036</td>
</tr>
<tr>
<td>20K-25K</td>
<td>0.02</td>
</tr>
<tr>
<td>25K-30K</td>
<td>0.015</td>
</tr>
<tr>
<td>...</td>
<td>0.0001</td>
</tr>
<tr>
<td>480K-485K</td>
<td></td>
</tr>
</tbody>
</table>

Filter

total_ons <= 5000

Estimate

0.61 * 7854 = 4796

Actual

4796
Histogram (Equal Width)

Divide data into N equal sized bins (e.g., 100 bins)
y axis = fraction of data in that bin

Assume within a bucket, values uniformly distributed

<table>
<thead>
<tr>
<th>Value Bins</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-5K</td>
<td>0.61</td>
</tr>
<tr>
<td>5K-10K</td>
<td>0.09</td>
</tr>
<tr>
<td>10K-15K</td>
<td>0.05</td>
</tr>
<tr>
<td>15K-20K</td>
<td>0.036</td>
</tr>
<tr>
<td>20K-25K</td>
<td>0.02</td>
</tr>
<tr>
<td>25K-30K</td>
<td>0.015</td>
</tr>
<tr>
<td>...</td>
<td>0.0001</td>
</tr>
<tr>
<td>480K-485K</td>
<td></td>
</tr>
</tbody>
</table>

Filter: total_ons <= 7500
Estimate: \((0.61 + 0.045) \times 7854\)
Actual: 5259
SELECT histogram_bounds
FROM pg_stats
WHERE tablename = 'rail_ridership' AND attname = 'total_ons';
Histogram (Equal Depth)

100 bins with ~similar #values

<table>
<thead>
<tr>
<th>Value Bins</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000 - 30</td>
<td>0.000</td>
</tr>
<tr>
<td>30 - 96</td>
<td>0.008</td>
</tr>
<tr>
<td>96 - 128</td>
<td>0.008</td>
</tr>
<tr>
<td>128 - 155</td>
<td>0.008</td>
</tr>
<tr>
<td>155 - 177</td>
<td>0.008</td>
</tr>
<tr>
<td>177 - 204</td>
<td>0.008</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>813 - 860</td>
<td>0.008</td>
</tr>
<tr>
<td>860 - 1000</td>
<td>0.010</td>
</tr>
<tr>
<td>1000 - 128</td>
<td>0.008</td>
</tr>
<tr>
<td>128 - 155</td>
<td>0.008</td>
</tr>
<tr>
<td>155 - 177</td>
<td>0.008</td>
</tr>
<tr>
<td>177 - 204</td>
<td>0.008</td>
</tr>
<tr>
<td>204 - 242</td>
<td>0.008</td>
</tr>
<tr>
<td>242 - 248.5</td>
<td>0.008</td>
</tr>
<tr>
<td>248.5 - 485</td>
<td>0.008</td>
</tr>
</tbody>
</table>

Filter | Estimate | Actual |
-------|----------|--------|
total_ons >= 195.5K | (0.008+0.008) * 7854 = 126 | 128 |
Histogram (Equal Depth)

100 bins with ~similar #values

<table>
<thead>
<tr>
<th>Value Bins</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>30-96</td>
<td>0.008</td>
</tr>
<tr>
<td>96-128</td>
<td>0.008</td>
</tr>
<tr>
<td>128-155</td>
<td>0.008</td>
</tr>
<tr>
<td>155-177</td>
<td>0.008</td>
</tr>
<tr>
<td>177-204</td>
<td>0.008</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>195.5K-242K</td>
<td>0.008</td>
</tr>
<tr>
<td>242K - 485K</td>
<td>0.008</td>
</tr>
</tbody>
</table>

Filter

<table>
<thead>
<tr>
<th>total_ons &gt;= 364,172</th>
<th>(0.004) * 7854 = 33</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

Assumption: **Uniformity** within a bucket; i.e., half values in half of the bucket size

Aggregate (cost=197.26..197.27 rows=1 width=8) -> Seq Scan on rail_ridership r (cost=0.00..197.18 rows=33 width=0) Filter: (total_ons >= 364172)
What if there are some extreme outliers in these ranges?
Histogram (Equal Depth) + Most Common Values

100 bins with ~similar #values

Value Bins

<table>
<thead>
<tr>
<th>Value Bins</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>30-96</td>
<td>0.01</td>
</tr>
<tr>
<td>96-128</td>
<td>0.008</td>
</tr>
<tr>
<td>128-155</td>
<td>0.01</td>
</tr>
<tr>
<td>155-177</td>
<td>0.008</td>
</tr>
<tr>
<td>177-204</td>
<td>0.01</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>813-860</td>
<td>0.01</td>
</tr>
<tr>
<td>195.5K-242K</td>
<td>0.01</td>
</tr>
<tr>
<td>242K-485K</td>
<td>0.01</td>
</tr>
</tbody>
</table>

100 most common values

<table>
<thead>
<tr>
<th>Value</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>1</td>
<td>0.0048</td>
</tr>
<tr>
<td>2</td>
<td>0.00407</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>0.0002</td>
</tr>
<tr>
<td>850</td>
<td>0.0007</td>
</tr>
</tbody>
</table>

SELECT most_common_vals, most_common_freqs FROM pg_stats
WHERE tablename = 'rail_ridership'
AND attname = 'total_ons';
Histogram (Equal Depth) + Most Common Values

100 bins with ~similar #values

100 most common values

<table>
<thead>
<tr>
<th>Value</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>1</td>
<td>0.0048</td>
</tr>
<tr>
<td>2</td>
<td>0.00407</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>0.0002</td>
</tr>
<tr>
<td>850</td>
<td>0.0007</td>
</tr>
<tr>
<td></td>
<td>0.14</td>
</tr>
</tbody>
</table>

Filter | Estimate | Actual
-------|----------|--------
total_ons = 0 | (0.07) * 7854 = 550 | 550
Combining Histograms and Most Common Values

100 bins with ~similar #values

Filter

812 <= total_ons < 860

Estimate

(0.008 + 0.0007) * 7854 = 68.45

Actual

70

100 most common values

<table>
<thead>
<tr>
<th>Value</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>1</td>
<td>0.0048</td>
</tr>
<tr>
<td>2</td>
<td>0.00407</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>0.0002</td>
</tr>
<tr>
<td>850</td>
<td>0.0007</td>
</tr>
<tr>
<td></td>
<td>0.14</td>
</tr>
</tbody>
</table>
**Multiple-Column Example**

```
SELECT COUNT(*)
from rail_ridership AS rr
WHERE rr.average_ons >= 1000
```

- Estimated Selectivity: \( \frac{1}{7} \)
- Estimate: 1111
- True: 1111

```
SELECT COUNT(*)
from rail_ridership AS rr
WHERE rr.total_ons >= 242507
```

- Estimated Selectivity: \( \frac{1}{120} \)
- Estimate: 64
- True: 65

```
SELECT COUNT(*)
from rail_ridership AS rr
WHERE rr.total_ons >= 242507
AND rr.average_ons >= 1000
```

- Estimated Selectivity: \( \frac{1}{7} \times \frac{1}{120} = \frac{1}{840} \)
- Estimate: 10
- True: 65

*Underestimate because filters assumed independent!*
Assumptions Recap

- Uniformity
  - e.g., within a bucket, uniform distribution
- Independence
  - (filters on different) columns are independent
Roadmap

What is Cardinality Estimation / Where is it used in DBMSs?

Histograms in PostgreSQL

Handling Correlated Columns

Other Operators and Estimation Methods

Estimating Size of Joins

Effect of Cardinality Estimation Errors on Query Optimization
Beyond Independence: DBMS heuristics

SQL Server added this recently
Idea: Assume filters are correlated (not necessarily true)

```
SELECT COUNT(*)
from rail_ridership AS rr
WHERE rr.total_ons >= 242507
AND rr.average_ons >= 1000
```

Estimated Selectivity: \((1/120) \times \sqrt{1/7}\) = 0.0032
Estimate: 25
True: 65
### Beyond Independence: Multi-Column Common Values

<table>
<thead>
<tr>
<th>total_ons</th>
<th>average_ons</th>
<th>Selectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.07</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.0048</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Not used by default, because it's not practical to have all pairwise columns considered
- Can be helpful to create for strongly correlated columns
Beyond Independence: Multi-Column Histograms

Can create multi-column histograms in PostgreSQL, but not on by default.

Challenges:
(1) Need equi-depth bins over 2 (or more dimensions)
Beyond Independence: Multi-Column Histograms

Can create multi-column histograms in PostgreSQL, but not on by default.

Challenges:

1. Need equi-depth bins over 2 (or more dimensions)
2. Which columns should the histograms be over?
   e.g., rail_ridership has ~8 columns ==> $O(100^{(8)})$ space + slow inference
Beyond Independence: Graphical Models

A, B, C

Filter example: A = 'a1', B = 'b1', C = 'c1'

3 independent histograms
Space: 3 |N|

\[ P(A)P(B)P(C) \]

3d histogram
Space: |N|^3

\[ P(A, B, C) \]

BayesCard by Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, Jingren Zhou
Beyond Independence: Graphical Models

A, B, C

Filter example: A = 'a1', B = 'b1', C = 'c1'

\[ P(A)P(B)P(C) \]

3 independent histograms
Space: 3 \(|N|\)

\[ P(C | A)P(A) \quad P(B | A)P(A) \]

2 independent 2d histograms
Space: 2 \(|N|^2\)

\[ P(A, B, C) \]

3d histogram
Space: \(|N|^3\)

BayesCard by Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, Jingren Zhou
Graphical Models: rail_ridership

- season
- total_ons
- average_ons
- total_offs
- average_offs
- station_ids
- time_period
- average_flows

$O(8 \times 10^2)$ vs $O(100^8)$
Roadmap

What is Cardinality Estimation / Where is it used in DBMSs?

Histograms in PostgreSQL

Handling Correlated Columns

Other Operators and Estimation Methods

Estimating Size of Joins

Effect of Cardinality Estimation Errors on Query Optimization
Other Filter Kinds

LIKEs
- SELECT COUNT(*) from rail_ridership as rr WHERE rr.station_id LIKE 'a%';
- SELECT COUNT(*) from rail_ridership as rr WHERE rr.station_id LIKE '%place%';

- Used to be that DBMS systems would just assume some constant selectivity (e.g., always 1/10th)
- Prefix trees
- Even w/ improved methods, much bigger errors even on single table cardinalities

User Defined Functions
- SELECT COUNT(*) from rail_ridership as rr WHERE check_hash(rr.station_id);
- Quite common, eg., for processing video frames
- Basically, impossible to estimate using statistics

Note: LIKE or UDFs can lead to bad query performance, because the optimizer will have much worse estimates!
Sampling

- Keep a small sample of each table (e.g., 1%) in memory
- Run the filter on the small sample, and estimate selectivity

Benefits
- Any kind of operator is ok
- No strong assumptions

Drawbacks
- Can take more time / or less accurate
- Does not scale to joins
Sampling over Joins

<table>
<thead>
<tr>
<th>eid</th>
<th>employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arun</td>
</tr>
<tr>
<td>2</td>
<td>Sho</td>
</tr>
<tr>
<td>3</td>
<td>Justin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>eid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vivaan</td>
</tr>
<tr>
<td>1</td>
<td>Mark</td>
</tr>
<tr>
<td>2</td>
<td>Sy</td>
</tr>
<tr>
<td>3</td>
<td>Jack</td>
</tr>
</tbody>
</table>

1/3 Sample Emp  
<table>
<thead>
<tr>
<th>eid</th>
<th>employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Sho</td>
</tr>
</tbody>
</table>

1/3 Sample Kids  
<table>
<thead>
<tr>
<th>eid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vivaan</td>
</tr>
</tbody>
</table>
Roadmap

What is Cardinality Estimation / Where is it used in DBMSs?

Histograms in PostgreSQL

Handling Correlated Columns

Other Operators and Estimation Methods

Estimating Size of Joins

Effect of Cardinality Estimation Errors on Query Optimization
### Joins: Simple Examples

**SQL Query:**

```sql
SELECT * FROM emp, dept, kids
WHERE emp.dno = dept.dno
AND emp.eid = kids.eid
```

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Kids</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>employee</td>
<td>dept</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>Arun</td>
<td>EECS</td>
</tr>
<tr>
<td>2</td>
<td>Sho</td>
<td>EECS</td>
</tr>
<tr>
<td>3</td>
<td>Justin</td>
<td>Math</td>
</tr>
</tbody>
</table>

\[
| emp \bowtie dept | = 3 \\
| emp \bowtie kids | = 4 \\
| emp \bowtie dept \bowtie kids | = 4 \\
\]
Joins: Simple Examples

```
SELECT * FROM emp, dept, kids
WHERE
emp.dno = dept.dno
AND emp.eid = kids.eid
```

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Kids</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>employee</td>
<td>dept</td>
</tr>
<tr>
<td>1</td>
<td>Arun</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Sho</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Justin</td>
<td>3</td>
</tr>
</tbody>
</table>

\[| \text{emp} \bowtie \text{dept}| = 3\]

\[| \text{emp} \bowtie \text{kids}| = 4\]

\[| \text{emp} \bowtie \text{dept} \bowtie \text{kids}| = 4\]

Easy because
- one-to-many relationship
- no filters
Joins: Simple Examples

```
SELECT * FROM emp, dept, kids
WHERE
  dept.dept = 'Math'
  emp.dno = dept.dno
  AND emp.eid = kids.eid
```

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Kids</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>employee</td>
<td>dept</td>
</tr>
<tr>
<td>1</td>
<td>Arun</td>
<td>EECS</td>
</tr>
<tr>
<td>2</td>
<td>Sho</td>
<td>EECS</td>
</tr>
<tr>
<td>3</td>
<td>Justin</td>
<td>Math</td>
</tr>
</tbody>
</table>

Correlations across multiple tables!

\[|emp \bowtie dept| = 1\]

\[|emp \bowtie kids| = 4\]

\[|emp \bowtie dept \bowtie kids| = 1\]
PostgreSQL Join Cardinality Estimation

**Tables** \( R_1 \) \( \bowtie \) \( R_2 \)

Join

\( R_1 . \text{rid} = R_2 . \text{rid} \)

\( |R_1 \bowtie R_2| \approx \min(d_1, d_2) \)

Number of rows for each unique value of rid (Uniformity)

\( |R_1| \) \( |R_2| \)

Number of unique rid values in result

\( d_1 \) \( d_2 \)

Number of rows for each unique value of rid in the join

\( |R_1 \bowtie R_2| \approx \frac{|R_1| \cdot |R_2|}{\max(d_1, d_2)} \)

\( |R_1| \cdot |R_2| \) will depend on the filters used on \( R_1, R_2 \).

\( d_1, d_2 \) Estimate using single table estimation methods
Bigger Joins

\[ |R_1 \bowtie R_2 \bowtie R_3| \]

Treat \((R_1 \bowtie R_2)\) as a new table;

Then estimate \(|(R_1 \bowtie R_2) \bowtie R_3|\)

Challenge is that we will have worse statistics on the two-table join than on individual tables;
Thus, estimates for \(|R_1 \bowtie R_2|\ d_{R_1 \bowtie R_2}\) will be much worse; And these errors accumulate quickly.
SELECT * FROM emp, dept, kids
WHERE
emp.dno = dept.dno
AND emp.eid = kids.eid

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Kids</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>employee</td>
<td>eid</td>
</tr>
<tr>
<td>1</td>
<td>Arun</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Sho</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Justin</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
| R_1 \bowtie R_2 | = \frac{|R_1||R_2|}{\max(d_1,d_2)}
\]

\[
| emp \bowtie dept | = \frac{(3 \times 3)}{3} = 3
\]

\[
| emp \bowtie kids | = \frac{(3 \times 4)}{3} = 4
\]

\[
|(emp \bowtie dept) \bowtie kids| = \frac{3 \times 4}{3} = 4
\]

\[
|emp| = 3, |dept| = 3, |kids| = 4
\]

\[
d_{e,eid} = 3, d_{d,eid} = 3, d_{k,eid} = 3
\]
**Joins: Simple Examples**

```sql
SELECT * FROM emp, dept, kids
WHERE
department.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid
```

$$|R_1 \bowtie R_2| = \frac{|R_1| \cdot |R_2|}{\text{max}(d_1, d_2)}$$

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
<th>Kids</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>employee</td>
<td>dept</td>
</tr>
<tr>
<td>1</td>
<td>Arun</td>
<td>EECS</td>
</tr>
<tr>
<td>2</td>
<td>Sho</td>
<td>EECS</td>
</tr>
<tr>
<td>3</td>
<td>Justin</td>
<td>Math</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$|\text{emp} \bowtie \text{dept}| = \frac{3 \times 1}{3} = 1$$

$$|\text{emp} \bowtie \text{kids}| = \frac{3 \times 4}{3} = 4$$

$$|\text{(emp} \bowtie \text{dept}) \bowtie \text{kids}| = \frac{1 \times 4}{3} = 1.3$$

<table>
<thead>
<tr>
<th>emp</th>
<th>dept</th>
<th>kids</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Because $\text{eid}$ is not uniform in $\text{kids}$!
SELECT COUNT(*)
FROM rail_ridership r, lines l
WHERE r.line_id = l.line_id
AND l.line_name = 'Blue Line';

\[ |R_1 \bowtie R_2| = \frac{|R_1||R_2|}{\max(d_1,d_2)} \]

| \( rr \) | = 7854, \( d_1 = 5 \)
| \( l \) | = 1, \( d_2 = 1 \)

\[ \frac{7854 \cdot 1}{5} = 1570.8 \]

**Why over-estimate?**
*Because blue line is not uniformly distributed in rail_ridership*

-> Hash Join (cost=1.07..216.71 rows=1571 width=0)
   Hash Cond: (r.line_id = l.line_id)
   -> Seq Scan on rail_ridership r (cost=0.00..177.54 rows=7854 width=5)
   -> Hash (cost=1.06..1.06 rows=1 width=6)
       -> Seq Scan on lines l (cost=0.00..1.06 rows=1 width=6)
          Filter: (line_name = 'Blue Line'::text)
SELECT COUNT(*)
FROM rail_ridership r, lines l
WHERE r.line_id = l.line_id
AND l.line_id = 'blue';

Exactly correct answer because of MCV list on rail_ridership

<table>
<thead>
<tr>
<th>Aggregate  (cost=208.14..208.15 rows=1 width=8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested Loop  (cost=0.00..206.16 rows=792 width=0)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

There was no filter on rail ridership before!
Under-Estimates v/s Over-Estimates

DBMS assumptions can lead to both under-estimates or over-estimates

Under-estimates are often a bigger challenge for Query Optimization:
e.g., DBMS estimates small size, and plans in memory nested loop join.
But, results much larger ==> $O(n^2)$ join with disk spills

Under-estimates are also very common due to the independence assumption!
Roadmap

What is Cardinality Estimation / Where is it used in DBMSs?

Histograms in PostgreSQL

Handling Correlated Columns

Other Operators and Estimation Methods

Estimating Size of Joins

Effect of Cardinality Estimation Errors on Query Optimization
Cardinality Estimation from Query Feedback

<table>
<thead>
<tr>
<th>Queries</th>
<th>TRUE</th>
<th>Estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT COUNT(*) FROM rail_ridership r, lines l WHERE r.line_id = l.line_id AND l.line_id = 'blue';</td>
<td>750</td>
<td>1500</td>
</tr>
<tr>
<td>SELECT COUNT(*) FROM rail_ridership r, lines l WHERE r.line_id = l.line_id AND l.line_id = 'red';</td>
<td>1200</td>
<td>1500</td>
</tr>
<tr>
<td>SELECT COUNT(*) from rail_ridership as rr WHERE rr.station_id ILIKE '%place%';</td>
<td>7854</td>
<td>400</td>
</tr>
</tbody>
</table>

**Is this a good estimator?**

**Loss (True, Estimator)**

What loss should the model optimize for? e.g., Mean Squared Error
Cardinality Estimation and Query Optimization

```sql
SELECT * FROM A, B, C
WHERE A.b1 = B.b1
  A.c1 = C.c1
  A.a IN ('A1')
  B.b >= 50
```

### Plan 1

```
A
  ┌──────┐
  │      │
  ├──┬───┘
  │  │
  ├──┼───
  │  │
  └───┘
B  C
```

### Plan 2

```
A
  ┌──────┐
  │      │
  └──────┘
  │      │
  ┌──────┐
  │      │
  └──────┘
B  C
```

<table>
<thead>
<tr>
<th>TRUE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>A ⨝ B</td>
</tr>
<tr>
<td></td>
<td>A ⨝ C</td>
</tr>
<tr>
<td>Plan 1</td>
<td>13</td>
</tr>
<tr>
<td>Plan 2</td>
<td>18</td>
</tr>
</tbody>
</table>

Final Evaluation: Runtimes by executing plan on DBMS

\[
\text{Cost}_1 = (|A| + |B|) + (|A \bowtie B| + |C|) = 13
\]

\[
\text{Cost}_2 = (|A| + |C|) + (|A \bowtie C| + |B|) = 18
\]
SELECT * FROM A, B, C
WHERE A.b1 = B.b1
A.c1 = C.c1
A.a IN ('A1')
B.b >= 50

<table>
<thead>
<tr>
<th>TRUE</th>
<th>Est1</th>
<th>Est2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A × B</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>A × C</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Plan 1</td>
<td>13</td>
<td>18</td>
</tr>
<tr>
<td>Plan 2</td>
<td>18</td>
<td>24</td>
</tr>
</tbody>
</table>

Plan 1

Cost₁ = (|A| + |B|) + (|A × B| + |C|) = 13

Plan 2

Cost₂ = (|A| + |C|) + (|A × C| + |B|) = 18

Final Evaluation: Runtimes by executing plan on DBMS
SELECT * FROM emp, dept, kids
WHERE
dept.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid
SELECT * FROM emp, dept, kids
WHERE
dept.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid
Plan Graph: Example Plan

SELECT * FROM emp, dept, kids
WHERE
department.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid

Join Graph

Example Plan

Query Optimization ↔ Plan Graph
Sub-plans ↔ Nodes
Joins ↔ Edges
Left deep query plans ↔ Paths (Bottom to Top)
SELECT * FROM emp, dept, kids
WHERE
dept.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid

- Cost of each edge depends on estimated sizes
- Shortest Path from bottom to top == best plan
  (equivalent to Selinger's Dynamic Programming)
Plan Graph

SELECT * FROM emp, dept, kids
WHERE
department.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid

Cardinality Estimation for Query Optimization
Estimate size of each node (subplan)
Sizes used to estimate cost of each edge (join)
How do errors in estimating sizes affect plans?
Plan Graph

SELECT * FROM emp, dept, kids
WHERE dept.dept = 'Math'
emp.dno = dept.dno
AND emp.eid = kids.eid

Join Graph

emp  ⋈ dept  ⋈ kids

Cardinality Estimation for Query Optimization
Estimate size of each node (subplan)
Sizes used to estimate cost of each edge (join)
How do errors in estimating sizes affect plans?

Demo
Thanks!
Plan Cost

$Y$ True
$
\hat{Y}$ Estimates

$P$ Query Plan

$P^*(Y) = \arg\min_P \sum_{e \in P} C(e, Y)$ \quad Best plan for $Y$
Plan Costs

\( Y \quad \text{True} \)

\( \hat{Y} \quad \text{Estimates} \)

P Query Plan

\[ P^*(Y) = \arg\min_P \sum_{e \in P} C(e, Y) \quad \text{Best plan for } Y \]

Cost of that plan using True cardinalities

\[ \text{P-Cost}(\hat{Y}, Y) = \sum_{e \in P^*(\hat{Y})} C(e, Y) \]

Choose best plan for estimates
Q-Error: Bounding P-Cost

\[ Q_{Error} = \max \left( \frac{\hat{Y}}{Y}, \frac{Y}{\hat{Y}} \right) \]

\( \hat{Y} = [5, 10, 20] \)
\( Y = [10, 1, 40] \)

\[ Q_{Error}(\hat{Y}, Y) = [2, 10, 2] \]

\( q = \max(Q_{Error}(\hat{Y}, Y)) \)
Q-Error: Bounding P-Cost

\[ P-Cost(\hat{Y}, Y) \leq \alpha P-Cost(Y, Y) \]

\[ q = \max(Q\text{Error}(\hat{Y}, Y)) \]

\[ \text{Cost}(q|A|, q|B|) \leq q^2 \text{Cost}(|A|, |B|) \]

\[ P-Cost(\hat{Y}, Y) \leq q^2 P-Cost(\hat{Y}, \hat{Y}) \]
\[ \leq q^2 P-Cost(Y, \hat{Y}) \]
\[ \leq q^4 P-Cost(Y, Y) \]

Intuition: Multiplicative Error more important, because even small selectivity error on single tables can become much bigger after joins.

SELECT COUNT(*)
FROM lines l
WHERE
line.line_name ILIKE '%e%';

True Selectivity: 5/5
Assume, you estimate: Selectivity = 3/5;

SELECT COUNT(*)
FROM lines l,
rail_ridership as rr
WHERE l.line_name ILIKE '%e%'
AND rr.line_id = l.line_id

Estimate: 3/5 \times 7854 \quad \text{----> much bigger error.}
**QO as Shortest Paths**

**Join Graph**

- kt
- t
- ci
- n
- rt

**Shortest Path**

\[
\begin{align*}
\min_F & \quad \sum_{e \in E} C_e F_e \\
\text{s.t.} & \quad \sum_{e \in \text{Out}(S)} F_e = \sum_{e \in \text{In}(D)} F_e = 1 \\
& \quad \sum_{e \in \text{Out}(V)} F_e = \sum_{e \in \text{In}(V)} F_e
\end{align*}
\]

(send 1 unit of flow from S to D)

(each node sends as much flow as it receives)
Plan Graph: Electric Flows

Join Graph

Electric Flows

\[
\min_F \sum_{e \in E} C_e F_e^2
\]

s.t.

\[
\sum_{e \in \text{Out}(S)} F_e = \sum_{e \in \text{In}(D)} F_e = 1
\]

\[
\sum_{e \in \text{Out}(V)} F_e = \sum_{e \in \text{In}(V)} F_e
\]

Edge width is proportional to the flow on each path

Closed form analytical solution; approximates shortest path

Almost no flow on all red edges

https://arxiv.org/abs/2101.04964
Pessimistic Query Optimization

Idea: instead of estimating cardinality, give an upper bound for cardinality

Why might this be helpful?