Pessimistic Query Optimization: Tighter Upper Bounds for Intermediate Join Cardinalities

Walter Cai   Magdalena Balazinska   Dan Suciu

University of Washington

[walter,magda,suciu]@cs.washington.edu

April 23, 2019

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.
- Method for enumerating practical subset of bounding formulas.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.
- Method for enumerating practical subset of bounding formulas.
- Partition budgeting strategy to control the space complexity of our sketches, and the time complexity of our bound calculation.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.
- Method for enumerating practical subset of bounding formulas.
- Partition budgeting strategy to control the space complexity of our sketches, and the time complexity of our bound calculation.
- Demonstrate practicality on challenging real world benchmark.

Query Optimization

- Accepts queries.
- Picks “best” physical plan.
  - Could be millions of correct physical plans!
  - Conceptually, a tree with leaves as base relations.

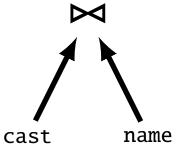
Query Optimization

- Accepts queries.
- Picks "best" physical plan.
  - Could be millions of correct physical plans!
  - Conceptually, a tree with leaves as base relations.

cast

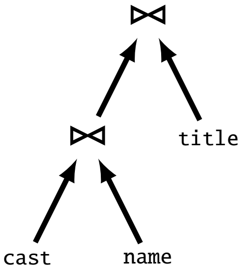
Query Optimization

- Accepts queries.
- Picks “best” physical plan.
  - Could be millions of correct physical plans!
  - Conceptually, a tree with leaves as base relations.



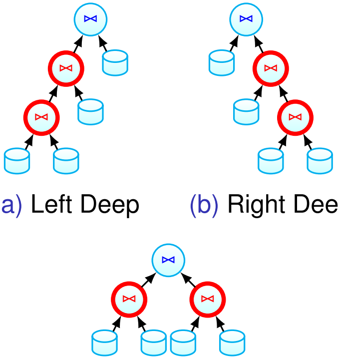
Query Optimization

- Accepts queries.
- Picks "best" physical plan.
  - Could be millions of correct physical plans!
  - Conceptually, a tree with leaves as base relations.



Query Optimization

- Accepts queries.
- Picks "best" physical plan.
  - Could be millions of correct physical plans!
  - Conceptually, a tree with leaves as base relations.



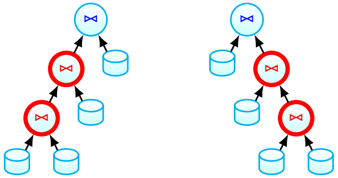
(a) Left Deep (b) Right Deep

(c) Bushy

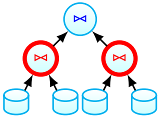
Figure: Join tree illustrations.

Query Optimization

- Cost-Based.
  - Large parameterized summation.
  - Sum over cost of each physical operator.



(a) Left Deep (b) Right Deep



(c) Bushy

Figure: Join tree illustrations.

Query Optimization

- Join Algorithms are generally binary so the DBMS will generate intermediate relations.
- **Cardinality Estimation**: how large will these intermediate relations be?

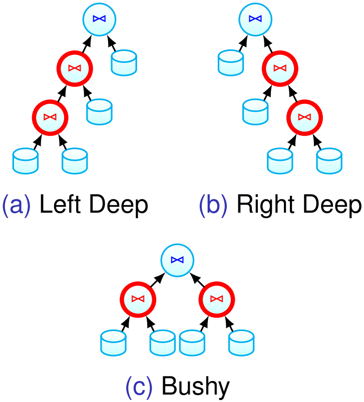


Figure: Join tree illustrations.

Cardinality Estimation Error

- Systems rely on strong assumptions about the underlying data.

Cardinality Estimation Error

- Systems rely on strong assumptions about the underlying data.
- Assume independence of attribute value distributions across columns.

Cardinality Estimation Error

- Systems rely on strong assumptions about the underlying data.
- Assume independence of attribute value distributions across columns.
- Leads to underestimation.
  - Real world data is correlated.
  - Underestimation is risky: leads to massive blow-up from poor join orderings/algorithm choice.

Join Order Benchmark (JOB)

- Built on the IMDb dataset.

Join Order Benchmark (JOB)

- Built on the IMDb dataset.
  - 113 queries.

Join Order Benchmark (JOB)

- Built on the IMDb dataset.
  - 113 queries.
  - 33 unique topoplogies.

Join Order Benchmark (JOB)

- Built on the IMDb dataset.
  - 113 queries.
  - 33 unique topoplogies.
  - Skew!

Join Order Benchmark (JOB)

- Built on the IMDb dataset.
  - 113 queries.
  - 33 unique topoplogies.
  - Skew!
  - Correlation!

Join Order Benchmark (JOB)

- Built on the IMDb dataset.
  - 113 queries.
  - 33 unique topoplogies.
  - Skew!
  - Correlation!
  - Complex selection predicates!

JOB Example Query

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

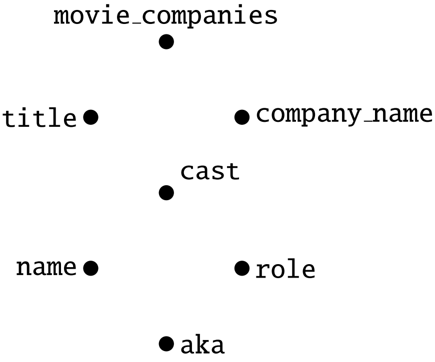


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

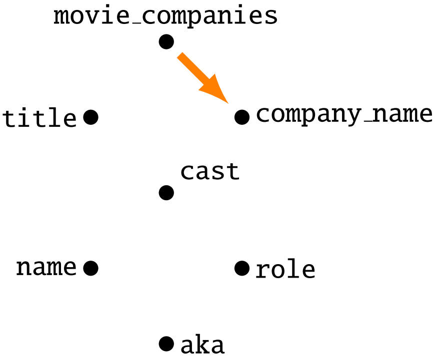


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

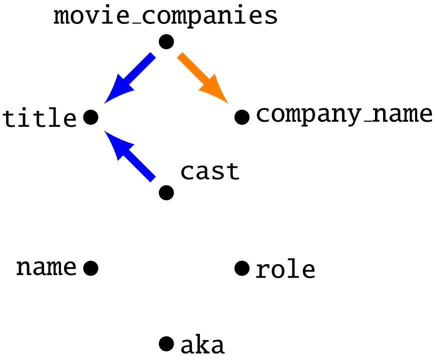


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

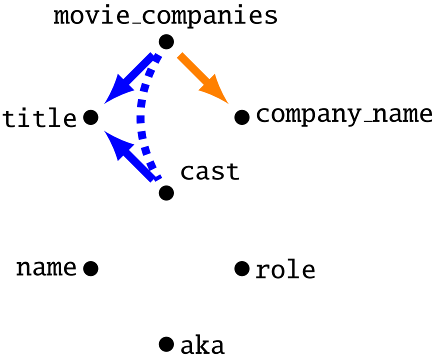


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

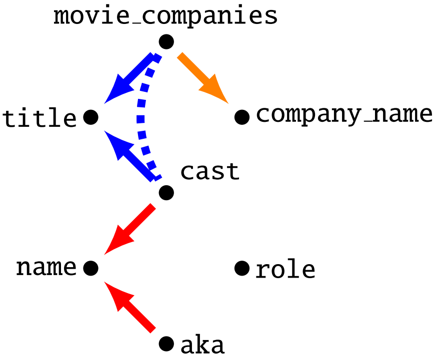


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

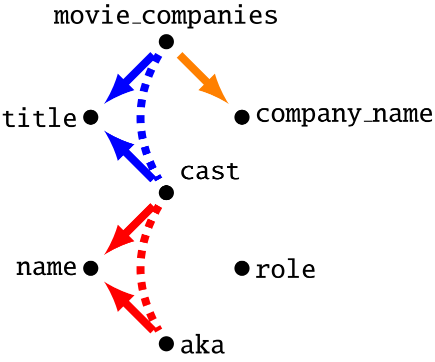


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

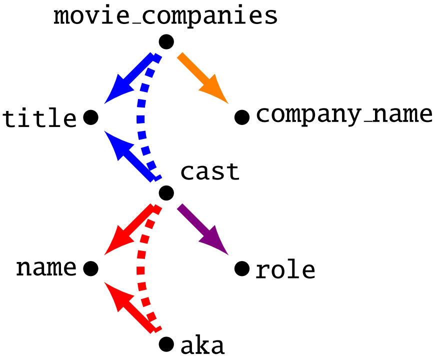


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

JOB Example Query

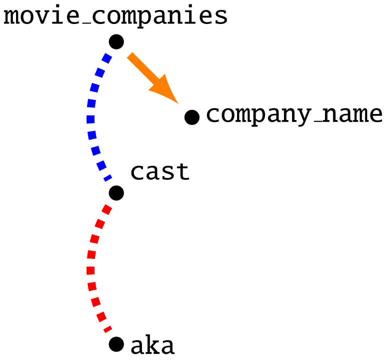


Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies
WHERE
    aka.person_id = cast.person_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id;
```

JOB Example Query

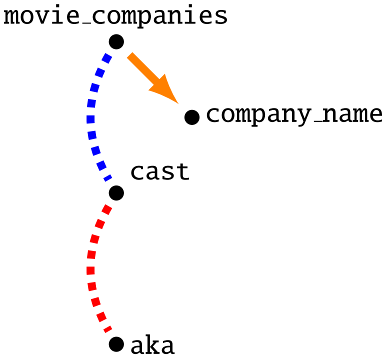


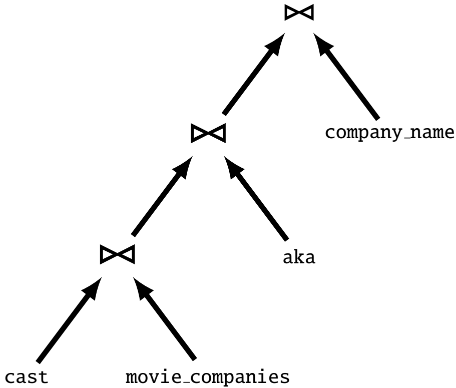
Figure: Join Graph.

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies
WHERE
    aka.person_id = cast.person_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id;
```

$$Q(x, y, z, w) :- \\ aka(x, y), \\ cast(y, z), \\ movie\_companies(z, w), \\ company\_name(w)$$

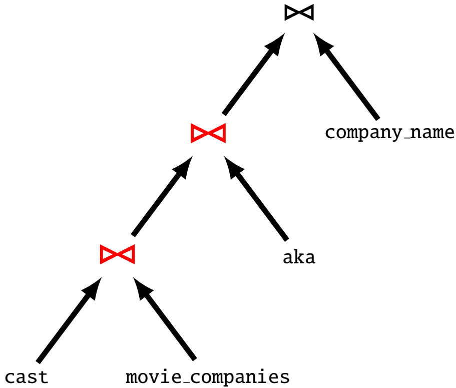
Worst Case Scenario

$$Q(x,y,z,w) \coloneq aka(x,y), cast(y,z), movie\_companies(z,w), company\_name(w)$$



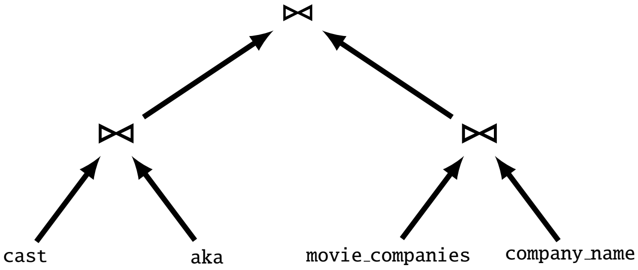
Worst Case Scenario

$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$



A Better Plan

$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$



Review: Entropy

Take random variable X:

$$h(X) = -\sum_{a} \mathbb{P}(X = a) \cdot \log(\mathbb{P}(X = a))$$

Multiple variables:

$$h(X, Y) = -\sum_{a,b} \mathbb{P}(X = a, Y = b) \cdot \log(\mathbb{P}(X = a, Y = b))$$

Conditional Entropy:

$$h(X|Y) = -\sum_{a,b} \mathbb{P}(X = a, Y = b) \cdot \log\left(\frac{\mathbb{P}(X = a, Y = b)}{\mathbb{P}(Y = b)}\right)$$

Review: Entropy

Let X be uniformly distributed on the space $\{a_1, a_2, \ldots, a_n\}$.

$$\begin{aligned}
h(X) &= -\sum_{i=1}^{n} \mathbb{P}(X = a_i) \cdot \log(\mathbb{P}(X = a_i)) \\
&= -\sum_{i=1}^{n} \frac{1}{n} \cdot \log\left(\frac{1}{n}\right) \\
&= -n\frac{1}{n} \cdot \log\left(\frac{1}{n}\right) \\
&= \log(n)
\end{aligned}$$

Connection to Entropy

$$Q(x, y, z, w) :\!-\ aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$

Connection to Entropy

$$Q(x, y, z, w) :\!-\; aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$

- Create a random variable for each of the attributes present in the query.

$$x \to X, \quad y \to Y, \quad z \to Z, \quad w \to W$$

Connection to Entropy

$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$

- Create a random variable for each of the attributes present in the query.

$$x \to X, \quad y \to Y, \quad z \to Z, \quad w \to W$$

- Let (X, Y, Z, W) be uniformly distributed over all tuples in the true output of Q.

$$\Big|Q(x, y, z, w)\Big| = \exp\Big(h(X, Y, Z, W)\Big)$$

Entropic Bounds

$$\left|Q(x, y, z, w)\right| = \exp\left(h(X, Y, Z, W)\right)$$

- Suffices to bound $h(X, Y, Z, W)$.

Entropic Bounds

$$\Big|Q(x, y, z, w)\Big| = \exp\Big(h(X, Y, Z, W)\Big)$$

- Suffices to bound $h(X, Y, Z, W)$.
- There are plenty of entropic bounds to choose from!

Entropic Bounds

	$h(X, Y, Z, W) \leq \ldots$
1	$h(X, Y) + h(Z\|Y) + h(W\|Z)$
2	$h(X, Y) + h(Z\|Y) + h(W)$
3	$h(X, Y) + h(Z, W)$
4	$h(X, Y) + h(Z\|W) + h(W)$
5	$h(X\|Y) + h(Y, Z) + h(W\|Z)$
6	$h(X\|Y) + h(Y, Z) + h(W)$
7	$h(X\|Y) + h(Y\|Z) + h(Z, W)$
8	$h(X\|Y) + h(Y\|Z) + h(Z\|W) + h(Z)$

(only a subset of all entropic bounding formulas)

Entropic Bounds

	$h(X, Y, Z, W) \leq \ldots$
1	$h(X, Y) + h(Z\|Y) + h(W\|Z)$
2	$h(X, Y) + h(Z\|Y) + h(W)$
3	$h(X, Y) + h(Z, W)$
4	$h(X, Y) + h(Z\|W) + h(W)$
5	$h(X\|Y) + h(Y, Z) + h(W\|Z)$
6	$h(X\|Y) + h(Y, Z) + h(W)$
7	$h(X\|Y) + h(Y\|Z) + h(Z, W)$
8	$h(X\|Y) + h(Y\|Z) + h(Z\|W) + h(Z)$

(only a subset of all entropic bounding formulas)

Entropic Bounds

$$|Q(x, y, z, w)| = \exp(h(X, Y, Z, W)) \leq \exp(h(X|Y) + h(Y, Z) + h(W|Z))$$

$$h(X|Y) \leq \log d^{y}_{\texttt{aka}}$$
$$h(Y, Z) \leq \log c_{\texttt{cast}}$$
$$h(W|Z) \leq \log d^{z}_{\texttt{movie\_companies}}$$

$$d^{y}_{\texttt{aka}} = \text{“Max Degree”}$$
$$= \text{Count of most common } \texttt{y} \text{ attribute value in } \texttt{aka\_name}.$$
$$c_{\texttt{cast}} = \text{“Count”}$$
$$= \text{Count of entire } \texttt{cast\_info} \text{ relation.}$$

Cardinality Bound

$$\begin{aligned}|Q(x,y,z,w)| &= \exp(h(X,Y,Z,W)) \\ &\le \exp(\underbrace{h(X|Y)}_{\le \log d^{y}_{\texttt{aka}}} + \underbrace{h(Y,Z)}_{\le \log c_{\texttt{cast}}} + \underbrace{h(W|Z)}_{\le \log d^{z}_{\texttt{movie\_companies}}}) \\ &\le d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot d^{z}_{\texttt{movie\_companies}}\end{aligned}$$

Bound Formula Generation

$$Q(x,y,z,w) \coloneq aka(x,y), cast(y,z), \underbrace{movie\_companies}_{\texttt{mc}}(z,w), \underbrace{company\_name}_{\texttt{cn}}(w)$$

x	y	z	w	entropic formula	bound formula
aka	aka	cast	mc	$h(X,Y)+h(Z\|Y)+h(W\|Z)$	$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$
aka	aka	cast	cn	$h(X,Y)+h(Z\|Y)+h(W)$	$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot c_{\texttt{cn}}$
aka	aka	mc	mc	$h(X,Y)+h(Z,W)$	$c_{\texttt{aka}} \cdot c_{\texttt{mc}}$
aka	aka	mc	cn	$h(X,Y)+h(Z\|W)+h(W)$	$c_{\texttt{aka}} \cdot d^{w}_{\texttt{mc}} \cdot c_{\texttt{cn}}$
aka	cast	cast	mc	$h(X\|Y)+h(Y,Z)+h(W\|Z)$	$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$
aka	cast	cast	cn	$h(X\|Y)+h(Y,Z)+h(W)$	$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot c_{\texttt{cn}}$
aka	cast	mc	mc	$h(X\|Y)+h(Y\|Z)+h(Z,W)$	$d^{y}_{\texttt{aka}} \cdot d^{z}_{\texttt{cast}} \cdot c_{\texttt{mc}}$
aka	cast	mc	cn	$h(X\|Y)+h(Y\|Z)+h(Z\|W)+h(Z)$	$d^{y}_{\texttt{aka}} \cdot d^{z}_{\texttt{cast}} \cdot d^{w}_{\texttt{mc}} \cdot c_{\texttt{cn}}$

Bound Formula Generation

$$Q(x,y,z,w) :\!-\ aka(x,y), cast(y,z), \underbrace{movie\_companies}_{\texttt{mc}}(z,w), \underbrace{company\_name}_{\texttt{cn}}(w)$$

x	y	z	w	entropic formula	bound formula
aka	aka	cast	mc	$h(X,Y) + h(Z\|Y) + h(W\|Z)$	$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$
aka	aka	cast	cn	$h(X,Y) + h(Z\|Y) + h(W)$	$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot c_{\texttt{cn}}$
aka	aka	mc	mc	$h(X,Y) + h(Z,W)$	$c_{\texttt{aka}} \cdot c_{\texttt{mc}}$
aka	aka	mc	cn	$h(X,Y) + h(Z\|W) + h(W)$	$c_{\texttt{aka}} \cdot d^{w}_{\texttt{mc}} \cdot c_{\texttt{cn}}$
aka	cast	cast	mc	$h(X\|Y) + h(Y,Z) + h(W\|Z)$	$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$
aka	cast	cast	cn	$h(X\|Y) + h(Y,Z) + h(W)$	$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot c_{\texttt{cn}}$
aka	cast	mc	mc	$h(X\|Y) + h(Y\|Z) + h(Z,W)$	$d^{y}_{\texttt{aka}} \cdot d^{z}_{\texttt{cast}} \cdot c_{\texttt{mc}}$
aka	cast	mc	cn	$h(X\|Y) + h(Y\|Z) + h(Z\|W) + h(Z)$	$d^{y}_{\texttt{aka}} \cdot d^{z}_{\texttt{cast}} \cdot d^{w}_{\texttt{mc}} \cdot c_{\texttt{cn}}$

Bound Formula Generation

$$Q(x,y,z,w) :- aka(x,y), cast(y,z), \underbrace{movie\_companies}_{\mathtt{mc}}(z,w), \underbrace{company\_name}_{\mathtt{cn}}(w)$$

x	y	z	w	entropic formula	bound formula
aka	aka	cast	mc	$h(X,Y)+h(Z\|Y)+h(W\|Z)$	$c_{\mathtt{aka}} \cdot d^{y}_{\mathtt{cast}} \cdot d^{z}_{\mathtt{mc}}$
aka	aka	cast	cn	$h(X,Y)+h(Z\|Y)+h(W)$	$c_{\mathtt{aka}} \cdot d^{y}_{\mathtt{cast}} \cdot c_{\mathtt{cn}}$
aka	aka	mc	mc	$h(X,Y)+h(Z,W)$	$c_{\mathtt{aka}} \cdot c_{\mathtt{mc}}$
aka	aka	mc	cn	$h(X,Y)+h(Z\|W)+h(W)$	$c_{\mathtt{aka}} \cdot d^{w}_{\mathtt{mc}} \cdot c_{\mathtt{cn}}$
aka	cast	cast	mc	$h(X\|Y)+h(Y,Z)+h(W\|Z)$	$d^{y}_{\mathtt{aka}} \cdot c_{\mathtt{cast}} \cdot d^{z}_{\mathtt{mc}}$
aka	cast	cast	cn	$h(X\|Y)+h(Y,Z)+h(W)$	$d^{y}_{\mathtt{aka}} \cdot c_{\mathtt{cast}} \cdot c_{\mathtt{cn}}$
aka	cast	mc	mc	$h(X\|Y)+h(Y\|Z)+h(Z,W)$	$d^{y}_{\mathtt{aka}} \cdot d^{z}_{\mathtt{cast}} \cdot c_{\mathtt{mc}}$
aka	cast	mc	cn	$h(X\|Y)+h(Y\|Z)+h(Z\|W)+h(Z)$	$d^{y}_{\mathtt{aka}} \cdot d^{z}_{\mathtt{cast}} \cdot d^{w}_{\mathtt{mc}} \cdot c_{\mathtt{cn}}$

Bound Formula Generation

$$Q(x,y,z,w) \coloneq aka(x,y), cast(y,z), \underbrace{movie\_companies}_{\texttt{mc}}(z,w), \underbrace{company\_name}_{\texttt{cn}}(w)$$

x	y	z	w	entropic formula	bound formula
aka	aka	cast	mc	$h(X,Y)+h(Z\|Y)+h(W\|Z)$	$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$
aka	aka	cast	cn	$h(X,Y)+h(Z\|Y)+h(W)$	$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot c_{\texttt{cn}}$
aka	aka	mc	mc	$h(X,Y)+h(Z,W)$	$c_{\texttt{aka}} \cdot c_{\texttt{mc}}$
aka	aka	mc	cn	$h(X,Y)+h(Z\|W)+h(W)$	$c_{\texttt{aka}} \cdot d^{w}_{\texttt{mc}} \cdot c_{\texttt{cn}}$
aka	cast	cast	mc	$h(X\|Y)+h(Y,Z)+h(W\|Z)$	$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$
aka	cast	cast	cn	$h(X\|Y)+h(Y,Z)+h(W)$	$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot c_{\texttt{cn}}$
aka	cast	mc	mc	$h(X\|Y)+h(Y\|Z)+h(Z,W)$	$d^{y}_{\texttt{aka}} \cdot d^{z}_{\texttt{cast}} \cdot c_{\texttt{mc}}$
aka	cast	mc	cn	$h(X\|Y)+h(Y\|Z)+h(Z\|W)+h(Z)$	$d^{y}_{\texttt{aka}} \cdot d^{z}_{\texttt{cast}} \cdot d^{w}_{\texttt{mc}} \cdot c_{\texttt{cn}}$

Entropic Bounds

Neat! But is it useful?

- Short answer: No. (Not yet, anyway)

Entropic Bounds

Neat! But is it useful?

- Short answer: No. (Not yet, anyway)
  - Bounds are still far too loose (overestimation).

Entropic Bounds

Neat! But is it useful?

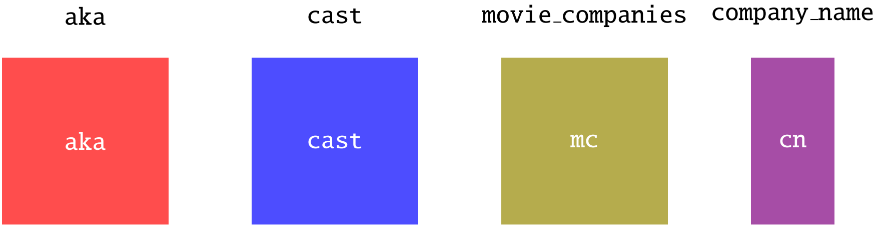
- Short answer: No. (Not yet, anyway)
  - Bounds are still far too loose (overestimation).
  - Need to tighten the bounds.

Entropic Bounds

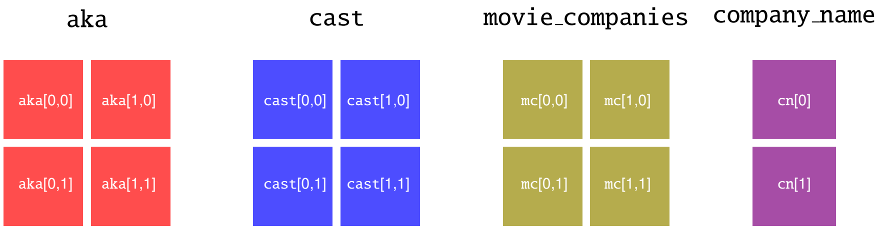
Neat! But is it useful?

- Short answer: No. (Not yet, anyway)
  - Bounds are still far too loose (overestimation).
  - Need to tighten the bounds.
- How to tighten? Partitioning.

$$Q(x, y, z, w) :\!-\ aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$



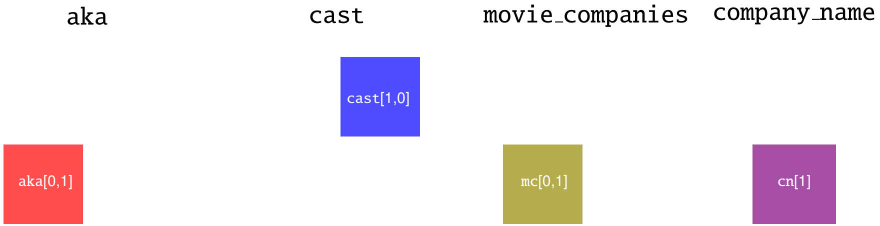
$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$



Hash the values of each tuple and bucketize on the hash values.

$$\mathtt{aka}[1, 0] = \left\{ t \in \mathtt{cast} \middle| \mathrm{hash}(t[y]) = 1 \ \wedge \ \mathrm{hash}(t[z]) = 0 \right\}$$

$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$

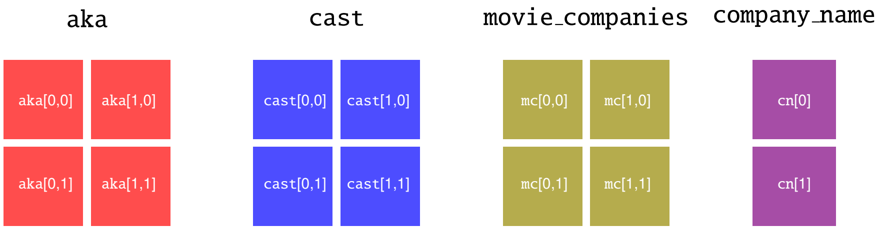


- Pick a hash value for each attribute in the query:

$$x, y, z, w \rightarrow [0, 1, 0, 1]$$

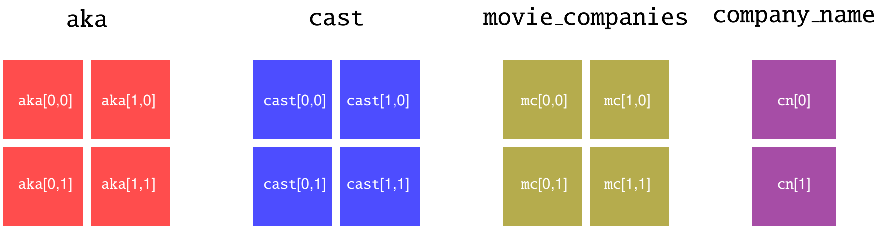
- The matching buckets from each relation is the partition $D[0, 1, 0, 1]$.

$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$



- $Q(D)$: query evaluated on database D.
- $Q(D[J])$: query evaluated on parition $D[J]$.

$$Q(x,y,z,w) :\!-\; aka(x,y), cast(y,z), movie\_companies(z,w), company\_name(w)$$



- Bound each partition $D[J]$.
- Sum will be a bound on the full database D.

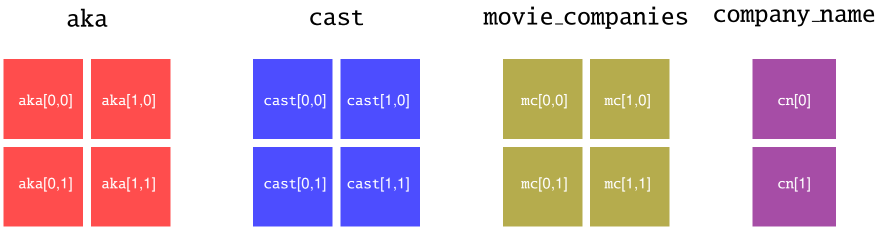
$$Q(D) = \bigcup_J Q(D[J])$$

$$|Q(D)| \leq \sum_J bound(Q(D[J]))$$

Partition Bounding

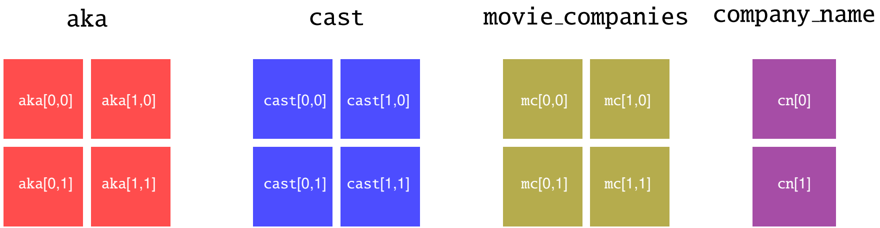
$$\left|Q(D)\right| \leq \sum_{J \in \{0,1\}^4} \min \begin{cases} c_{\texttt{aka}[J]} \cdot d^{y}_{\texttt{cast}[J]} \cdot d^{z}_{\texttt{mc}[J]} \\ c_{\texttt{aka}[J]} \cdot d^{y}_{\texttt{cast}[J]} \cdot c_{\texttt{cn}[J]} \\ c_{\texttt{aka}[J]} \cdot c_{\texttt{mc}[J]} \\ c_{\texttt{aka}[J]} \cdot d^{w}_{\texttt{mc}[J]} \cdot c_{\texttt{cn}[J]} \\ d^{y}_{\texttt{aka}[J]} \cdot c_{\texttt{cast}[J]} \cdot d^{z}_{\texttt{mc}[J]} \\ d^{y}_{\texttt{aka}[J]} \cdot c_{\texttt{cast}[J]} \cdot c_{\texttt{cn}[J]} \\ d^{y}_{\texttt{aka}[J]} \cdot d^{z}_{\texttt{cast}[J]} \cdot c_{\texttt{mc}[J]} \\ d^{y}_{\texttt{aka}[J]} \cdot d^{z}_{\texttt{cast}[J]} \cdot d^{w}_{\texttt{mc}[J]} \cdot c_{\texttt{cn}[J]} \end{cases}$$

The Bound Sketch



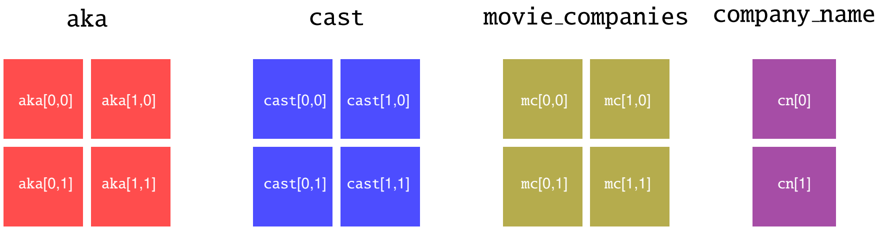
- One bound sketch per table.

The Bound Sketch



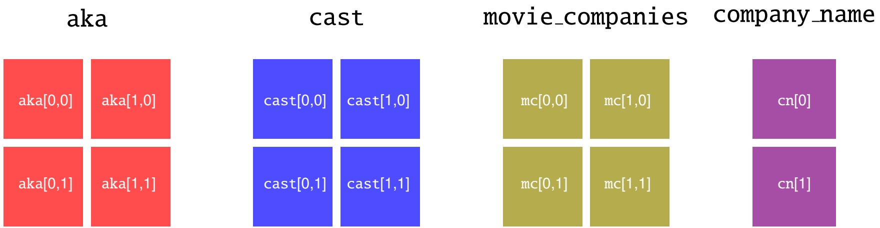
- One bound sketch per table.
- Need count and degree statistics.

The Bound Sketch



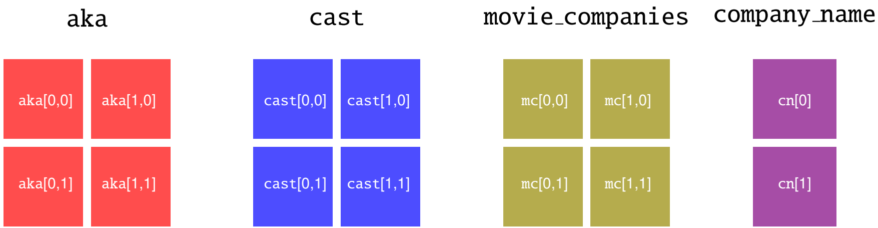
- One bound sketch per table.
- Need count and degree statistics.
- Some calculated offline, some at runtime.

The Bound Sketch



- One bound sketch per table.
- Need count and degree statistics.
- Some calculated offline, some at runtime.
- Like a richer randomized histogram.

The Bound Sketch



- One bound sketch per table.
- Need count and degree statistics.
- Some calculated offline, some at runtime.
- Like a richer randomized histogram.
- Restriction: this method is only suitable for equijoins.

Exponential Growth

- Sketch size (number of buckets) exponential in hash size.
  - Exponent is number of attributes in relation.

Exponential Growth

- Sketch size (number of buckets) exponential in hash size.
  - Exponent is number of attributes in relation.
- Number of elements to sum up exponential in hash size.
  - Exponent is number of attributes in entire query.

Exponential Growth

- Sketch size (number of buckets) exponential in hash size.
  - Exponent is number of attributes in relation.
- Number of elements to sum up exponential in hash size.
  - Exponent is number of attributes in entire query.
- Non-monotonic bound behavior

Tuning Bucket Allocation

- As the number of buckets increases, we get more information, and bounds tighten, right?

Tuning Bucket Allocation

- As the number of buckets increases, we get more information, and bounds tighten, right?
- When exclusively partitioning unconditionally covered attributes: yes.

Tuning Bucket Allocation

- As the number of buckets increases, we get more information, and bounds tighten, right?
- When exclusively partitioning unconditionally covered attributes: yes.
- When also partitioning conditionally covered attributes: not necessarily.

Tuning Bucket Allocation

- As the number of buckets increases, we get more information, and bounds tighten, right?
- When exclusively partitioning unconditionally covered attributes: yes.
- When also partitioning conditionally covered attributes: not necessarily.
    - Non-monotonic tradeoff space.

Example of Non-monotonic Behavior

$$Q\left(x,y,z,w\right) \text{:-} R\left(z,y\right), S\left(y,z\right), T\left(z,w\right)$$

$$\underbrace{\begin{array}{|c|c|}\hline x & y \\ \hline 0 & 0 \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline 1 & 1 \\ \hline \end{array}}_{R} \bowtie \underbrace{\begin{array}{|c|c|}\hline y & z \\ \hline 0 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline 3 & 1 \\ \hline \end{array}}_{S} \bowtie \underbrace{\begin{array}{|c|c|}\hline z & w \\ \hline 0 & 0 \\ \hline 1 & 1 \\ \hline 2 & 2 \\ \hline 3 & 3 \\ \hline \end{array}}_{T} = \underbrace{\begin{array}{|c|c|c|c|}\hline x & y & z & w \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array}}_{Q}$$

Example of Non-monotonic Behavior

$$\underbrace{\begin{array}{|c|c|}\hline x & y \\ \hline 0 & 0 \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline 1 & 1 \\ \hline \end{array}}_{R} \bowtie \underbrace{\begin{array}{|c|c|}\hline y & z \\ \hline 0 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline 3 & 1 \\ \hline \end{array}}_{S} \bowtie \underbrace{\begin{array}{|c|c|}\hline z & w \\ \hline 0 & 0 \\ \hline 1 & 1 \\ \hline 2 & 2 \\ \hline 3 & 3 \\ \hline \end{array}}_{T} = \underbrace{\begin{array}{|c|c|c|c|}\hline x & y & z & w \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array}}_{Q}$$

Example of Non-monotonic Behavior

$$\underbrace{\begin{array}{|c|c|}\hline x & y \\ \hline 0 & 0 \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline 1 & 1 \\ \hline \end{array}}_{R} \bowtie \underbrace{\begin{array}{|c|c|}\hline y & z \\ \hline 0 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline 3 & 1 \\ \hline \end{array}}_{S} \bowtie \underbrace{\begin{array}{|c|c|}\hline z & w \\ \hline 0 & 0 \\ \hline 1 & 1 \\ \hline 2 & 2 \\ \hline 3 & 3 \\ \hline \end{array}}_{T} = \underbrace{\begin{array}{|c|c|c|c|}\hline x & y & z & w \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array}}_{Q}$$

$$\left|Q(x,y,z,w)\right| \leq \min \begin{cases} c_R \cdot d_S^y \cdot d_T^z \\ d_R^y \cdot c_S \cdot d_T^z \\ d_R^y \cdot d_S^z \cdot c_T \\ c_R \cdot c_T \end{cases}$$

Example of Non-monotonic Behavior

$$\underbrace{\begin{array}{|c|c|}\hline x & y \\ \hline 0 & 0 \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline 1 & 1 \\ \hline \end{array}}_{R} \bowtie \underbrace{\begin{array}{|c|c|}\hline y & z \\ \hline 0 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline 3 & 1 \\ \hline \end{array}}_{S} \bowtie \underbrace{\begin{array}{|c|c|}\hline z & w \\ \hline 0 & 0 \\ \hline 1 & 1 \\ \hline 2 & 2 \\ \hline 3 & 3 \\ \hline \end{array}}_{T} = \underbrace{\begin{array}{|c|c|c|c|}\hline x & y & z & w \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array}}_{Q}$$

$$\left|Q(x,y,z,w)\right| \leq \min \begin{cases} c_R \cdot d_S^y \cdot d_T^z \\ d_R^y \cdot c_S \cdot d_T^z \\ d_R^y \cdot d_S^z \cdot c_T \\ c_R \cdot c_T \end{cases}$$

$$c_{R^{(0)}} \cdot d^y_{S^{(0,0)}} \cdot d^z_{T^{(0)}} = 4 \cdot 1 \cdot 1 = 4$$

- Define hash function $\text{hash}(u_i) = i\%2$.

$$\text{hash}(0) = \text{hash}(2) = 0$$
$$\text{hash}(1) = \text{hash}(3) = 1$$

Partitioned Relations

$$\text{hash}(y), \text{hash}(z) = \dots$$

$$0,0 \longrightarrow \begin{array}{|c|c|}\hline 0&0\\\hline 1&0\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 0&0\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 0&0\\\hline 2&2\\\hline\end{array} = \begin{array}{|c|c|c|c|}\hline 0&0&0&0\\\hline 1&0&0&0\\\hline\end{array}$$

$$0,1 \longrightarrow \begin{array}{|c|c|}\hline 0&0\\\hline 1&0\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 2&1\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 1&1\\\hline 3&3\\\hline\end{array} = \emptyset$$

$$1,0 \longrightarrow \begin{array}{|c|c|}\hline 0&1\\\hline 1&1\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 1&0\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 0&0\\\hline 2&2\\\hline\end{array} = \begin{array}{|c|c|c|c|}\hline 0&1&0&0\\\hline 1&1&0&0\\\hline\end{array}$$

$$1,1 \longrightarrow \begin{array}{|c|c|}\hline 0&1\\\hline 1&1\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 3&1\\\hline\end{array} \bowtie \begin{array}{|c|c|}\hline 1&1\\\hline 3&3\\\hline\end{array} = \emptyset$$

$$0,0 \longrightarrow \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 2 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}$$

$$0,1 \longrightarrow \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 2 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & 3 \\ \hline \end{array} = \emptyset$$

$$1,0 \longrightarrow \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 2 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$$1,1 \longrightarrow \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 3 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & 3 \\ \hline \end{array} = \emptyset$$

$$\sum_{\substack{i,j \\ \in\{0,1\}}} \min \begin{cases} c_{R^{(i)}} \cdot d^{y}_{S^{(i,j)}} \cdot d^{z}_{T^{(j)}} \\ d^{y}_{R^{(i)}} \cdot c_{S^{(i,j)}} \cdot d^{z}_{T^{(j)}} \\ d^{y}_{R^{(i)}} \cdot d^{z}_{S^{(i,j)}} \cdot c_{T^{(j)}} \\ c_{R^{(i)}} \cdot c_{T^{(j)}} \end{cases}$$

$$0,0 \longrightarrow \quad \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 2 & 2 \\ \hline\end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline\end{array}$$

$$0,1 \longrightarrow \quad \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 2 & 1 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & 3 \\ \hline\end{array} = \emptyset$$

$$1,0 \longrightarrow \quad \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 0 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 2 & 2 \\ \hline\end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline\end{array}$$

$$1,1 \longrightarrow \quad \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 3 & 1 \\ \hline\end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & 3 \\ \hline\end{array} = \emptyset$$

$$\sum_{\substack{i,j \\ \in\{0,1\}}} \min \begin{cases} c_{R^{(i)}} \cdot d^y_{S^{(i,j)}} \cdot d^z_{T^{(j)}} \\ d^y_{R^{(i)}} \cdot c_{S^{(i,j)}} \cdot d^z_{T^{(j)}} \\ d^y_{R^{(i)}} \cdot d^z_{S^{(i,j)}} \cdot c_{T^{(j)}} \\ c_{R^{(i)}} \cdot c_{T^{(j)}} \end{cases} = \sum_{\substack{i,j \\ \in\{0,1\}}} \min \begin{cases} 2 \cdot 1 \cdot 1 \\ 2 \cdot 1 \cdot 1 \\ 2 \cdot 1 \cdot 2 \\ 2 \cdot 2 \end{cases}$$

$$0,0 \longrightarrow \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 2 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}$$

$$0,1 \longrightarrow \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 2 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & 3 \\ \hline \end{array} = \emptyset$$

$$1,0 \longrightarrow \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 2 & 2 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$$1,1 \longrightarrow \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 3 & 1 \\ \hline \end{array} \bowtie \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 3 & 3 \\ \hline \end{array} = \emptyset$$

$$\sum_{\substack{i,j \\ \in\{0,1\}}} \min \begin{cases} c_{R(i)} \cdot d^y_{S(i,j)} \cdot d^z_{T(j)} \\ d^y_{R(i)} \cdot c_{S(i,j)} \cdot d^z_{T(j)} \\ d^y_{R(i)} \cdot d^z_{S(i,j)} \cdot c_{T(j)} \\ c_{R(i)} \cdot c_{T(j)} \end{cases} = \sum_{\substack{i,j \\ \in\{0,1\}}} \min \begin{cases} 2 \cdot 1 \cdot 1 \\ 2 \cdot 1 \cdot 1 \\ 2 \cdot 1 \cdot 2 \\ 2 \cdot 2 \end{cases} = \sum_{\substack{i,j \\ \in\{0,1\}}} 2 = 8$$

Non-Linearity of Degree Statistic

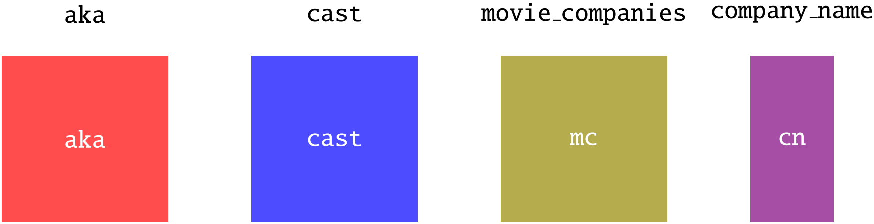
- Count is linear with respect to disjoint union!

$$\text{count}(A) + \text{count}(B) = \text{count}(A \cup B)$$

- Degree is not...

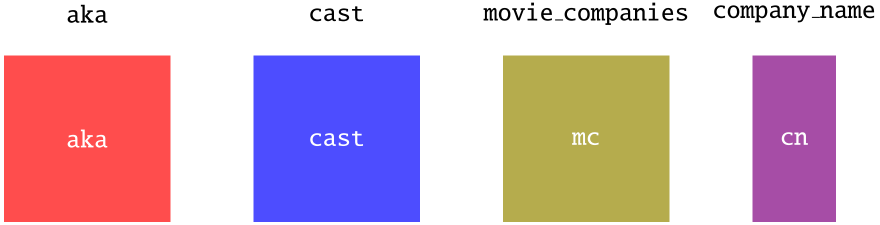
$$\text{degree}(A) + \text{degree}(B) \geq \text{degree}(A \cup B)$$

$$Q(x,y,z,w) :\!-\ aka(x,y), cast(y,z), movie\_companies(z,w), company\_name(w)$$



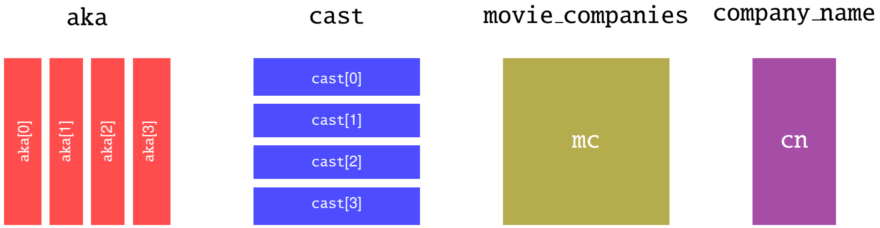
$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$

$$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$$



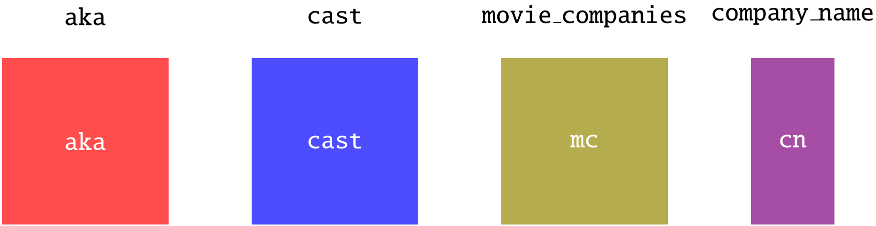
$$Q(x,y,z,w) :\!-\ aka(x,y), cast(y,z), movie\_companies(z,w), company\_name(w)$$

$$c_{\texttt{aka}} \cdot d^{y}_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$$



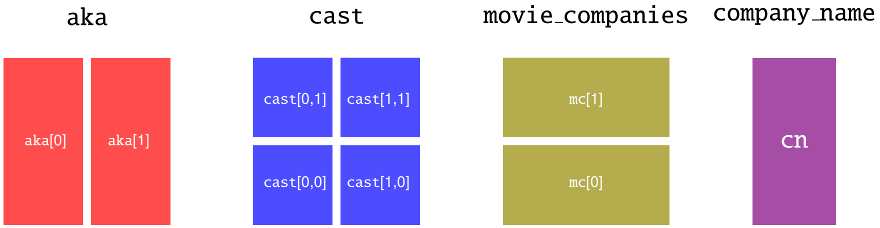
$$Q(x, y, z, w) :- aka(x, y), cast(y, z), movie\_companies(z, w), company\_name(w)$$

$$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$$



$$Q(x,y,z,w) :\!-\; aka(x,y), cast(y,z), movie\_companies(z,w), company\_name(w)$$

$$d^{y}_{\texttt{aka}} \cdot c_{\texttt{cast}} \cdot d^{z}_{\texttt{mc}}$$

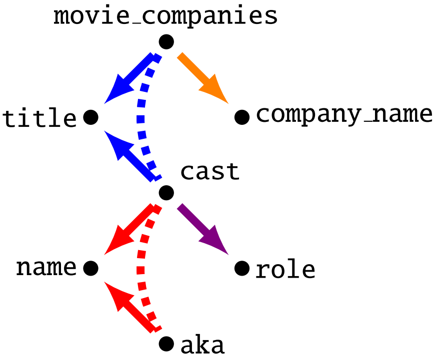


Bound Calculation

Calculate the minimal bound over all entropic bounds.

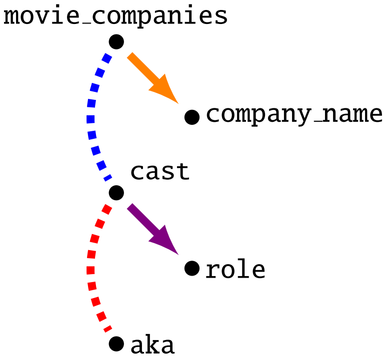
$$|Q(D)| \leq \min_{\substack{b \in \\ \text{bounding formulas}}} \left(\sum_{\substack{J \in \\ \text{partition indexes}}} b(Q(D[J])) \right)$$

Filter Predicate Analysis



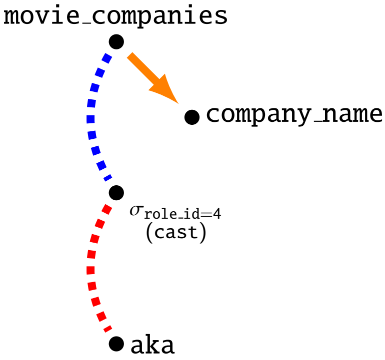

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

Filter Predicate Analysis



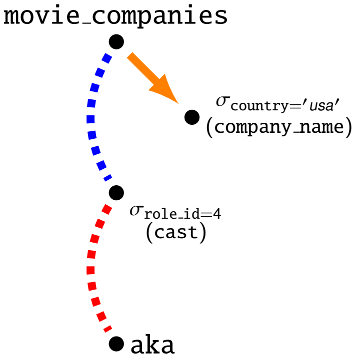

```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

Filter Predicate Analysis




```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

Filter Predicate Analysis




```
SELECT
    *
FROM
    aka,
    cast,
    company_name,
    movie_companies,
    name,
    role,
    title
WHERE
    company_name.country = 'usa' AND
    role.type = 'writer' AND
    aka.person_id = name.id AND
    cast.person_id = name.id AND
    aka.person_id = cast.person_id AND
    cast.movie_id = title.id AND
    movie_companies.movie = title.id_id AND
    cast.movie_id = movie_companies.movie_id AND
    movie_companies.company_id = company_name.id AND
    cast.role_id = role.id;
```

Filter Propagation

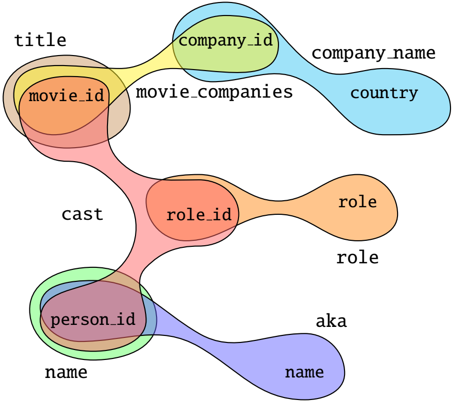


Figure: Original hypergraph represrentation.

Filter Propagation

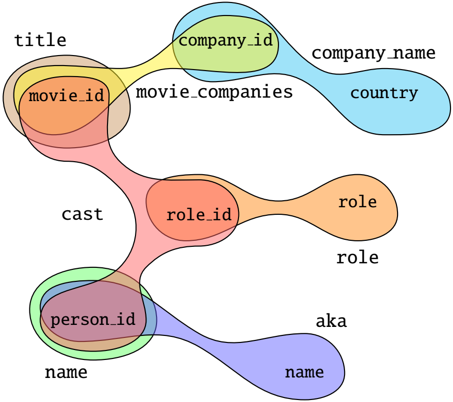


Figure: Original hypergraph represrentation.

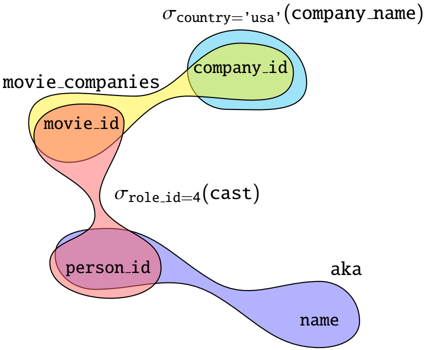


Figure: Hypergraph after selection propagation and elimination.

Table Scans During Optimization

Analysis of selection predicates can lead to:

Table Scans During Optimization

Analysis of selection predicates can lead to:

- Full propagation.
  - Highly selective predicate: yields fewer tuples than the hash size.
  - Scans on predicate relation and (most likely) on foreign key relation.

Table Scans During Optimization

Analysis of selection predicates can lead to:

- Full propagation.
  - Highly selective predicate: yields fewer tuples than the hash size.
  - Scans on predicate relation and (most likely) on foreign key relation.
- Updated the bound sketch.
  - Selective predicate but more tuples than hash size.
  - Scan on predicate relation.

Table Scans During Optimization

Analysis of selection predicates can lead to:

- Full propagation.
  - Highly selective predicate: yields fewer tuples than the hash size.
  - Scans on predicate relation and (most likely) on foreign key relation.
- Updated the bound sketch.
  - Selective predicate but more tuples than hash size.
  - Scan on predicate relation.
- Defaulting to unmodified bound sketch.
  - Non selective predicate.
  - Early exit during scan on predicate relation.

Table Scans During Optimization

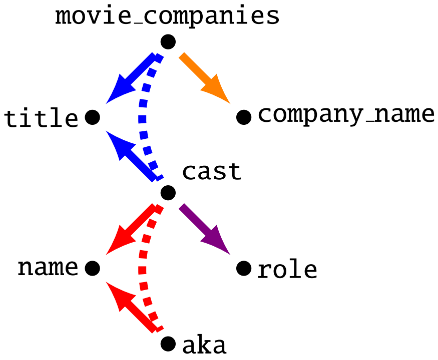
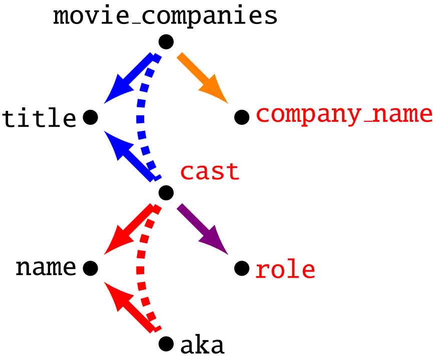


Table Scans During Optimization



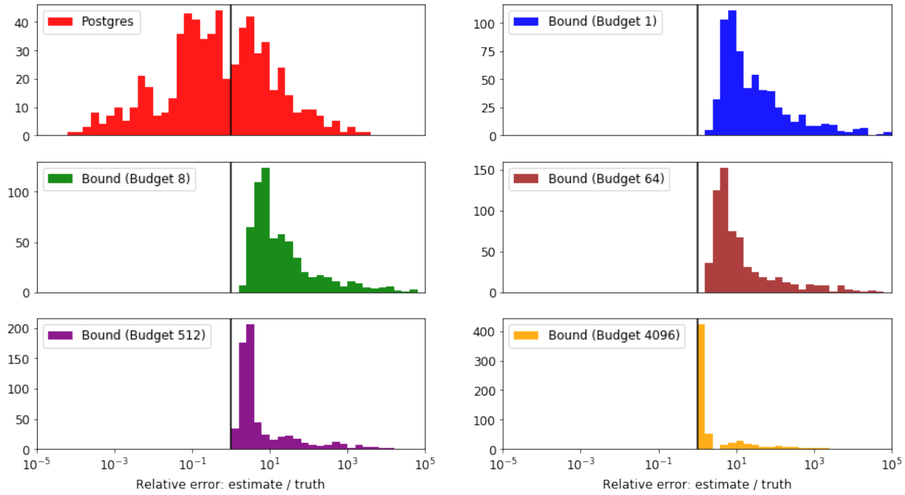
Googleplus Microbenchmark Examples

```
SELECT COUNT(*)
FROM
    community_44 AS t0,
    community_44 AS t1,
    community_44 AS t2,
    community_44 AS t3
WHERE
    t0.object = t1.subject AND
    t1.object = t2.subject AND
    t2.object = t3.subject AND
    t0.subject % 512 = 89 AND
    t3.object % 512 = 174;
```

Googleplus Microbenchmark Examples

```
SELECT COUNT(*)
FROM
    community_30 AS t0,
    community_30 AS t1,
    community_30 AS t2,
    community_30 AS t3,
    community_30 AS t4
WHERE
    t0.object = t1.subject AND
    t0.object = t2.subject AND
    t0.object = t3.subject AND
    t3.object = t4.subject AND
    t0.subject % 256 = 49 AND
    t1.object % 256 = 213 AND
    t2.object % 256 = 152 AND
    t4.object % 256 = 248;
    AND ci.movie_id = mc.movie_id;
```

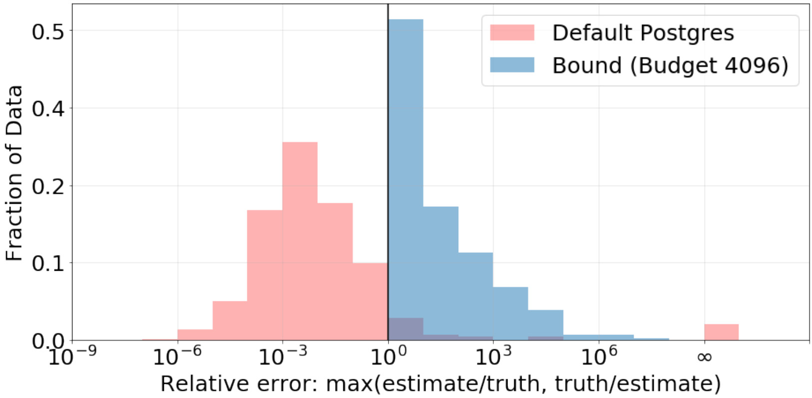
Googleplus Progressive Bound Tightness



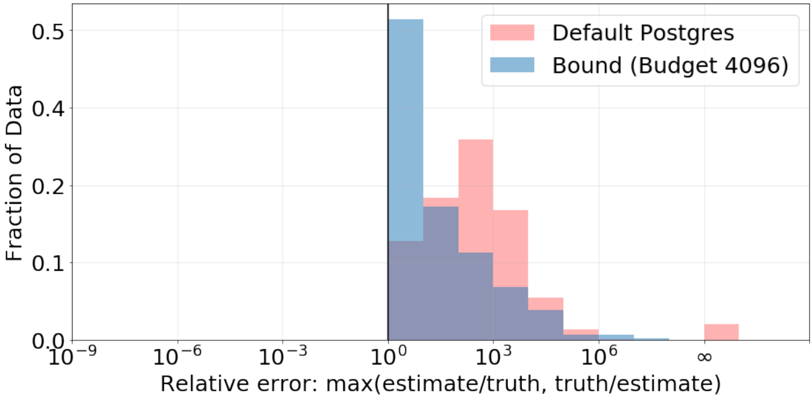
Join Order Benchmark

- Built on the IMDb dataset.
  - 113 queries.
  - 33 unique topoplogies.
  - Skew!
  - Correlation!
  - Complex selection predicates!
- *How Good Are Query Optimizers, Really?* Leis et al. VLDB 2015.

Bound Relative Error Versus Postgres Relative Error



Bound Q-Error Versus Postgres Q-Error



Plan Execution Runtime (With Foreign Keys Indexes)

Figure: Linear scale runtime improvements over JOB queries.

Plan Execution Runtime (With Foreign Keys Indexes)

Figure: Linear scale runtime improvements over JOB queries.

- Total runtime.
  - Postgres: 3,190 seconds.
  - Bound (4096 buckets): 1,832 seconds.

Plan Execution Runtime (No Foreign Key Indexes)

Figure: Linear scale plan execution time over JOB queries.

Plan Execution Runtime (No Foreign Key Indexes)

Figure: Linear scale plan execution time over JOB queries.

- Total runtime (including 1 hour cutoff for postgres).
  - Postgres: 21,125 seconds.
  - Bound (4096 buckets): 2,216 seconds.

Takeaways

- Significant gain for very slow queries.

Takeaways

- Significant gain for very slow queries.
- On par with fast queries.

Optimization Time

- Currently using naive enumeration and sketch construction approach.

Optimization Time

- Currently using naive enumeration and sketch construction approach.
- Approximation of degree statistics.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.
- Method for enumerating practical subset of bounding formulas.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.
- Method for enumerating practical subset of bounding formulas.
- Partition budgeting strategy to control the space complexity of our sketches, and the time complexity of our bound calculation.

Contributions

- Technique for tightening theoretically guaranteed join cardinality upper bounds.
- Method for enumerating practical subset of bounding formulas.
- Partition budgeting strategy to control the space complexity of our sketches, and the time complexity of our bound calculation.
- Demonstrate practicality on challenging real world benchmark.

Acknowledgements

- Thank you to Jenny, Tomer, Laurel, Brandon, Jingjing, Tobin, and Leilani!
- This research is supported by NSF grant AITF 1535565 and III 1614738.