Characteristic Sets:
Accurate Cardinality Estimation for RDF Queries with Multiple Joins

Thomas Neumann
Guido Moerkotte

Presented By: Pranjal Gupta
Recap.

- RDF is the underlying query language of the Semantic Web.
- Data is represented as the set of triple \((\text{subject}, \text{predicate}, \text{object})\).
- Single table (3 columns)
Recap.

- RDF is the underlying query language of the Semantic Web.
- Data is represented as the set of triple \((\text{subject}, \text{predicate}, \text{object})\).
- Single table (3 columns)
- Query graph is made up of sequence of query patterns.

```sql
SELECT DISTINCT ?e
WHERE { ?e <author> "Jane Austen" , ?e <title> ?b , ?e <year> ?y }
```
Recap.

- RDF is the underlying query language of the Semantic Web.
- Data is represented as the set of triple (subject, predicate, object).
- Single table (3 columns)
- Query graph is made up of sequence of query patterns.

```
SELECT DISTINCT ?e
WHERE { ?e <author> "Jane Austen" , ?e <title> ?b, ?e <year> ?y }
```

- Multiple self joins -> need for query optimizer that produces efficient query plans that has optimal join ordering.
Star queries.

- Quite a common feature in queries.
- Characterized by sequence of query patterns having a common subject.
Star queries.

- Quite a common feature in queries.
- Characterized by sequence of query patterns having a common subject.

```
SELECT DISTINCT ?e
WHERE {
?e <author> "Jane Austen" ,
?e <title> ?b , ?e <year> ?y
}
```
Objectives.

- Highly accurate cardinality estimation for Star Queries.
  - By using Characteristic sets.

- Extending the use of characteristic sets to calculate the cardinality of general queries.

- Using cardinality estimator with query optimizer.
Challenges.

1. Lack of explicit schema based on the structure. Cannot partition the data for estimation, since all data looks the same.

2. Predicates are correlated and hence, cardinality cannot be estimated using single-bucket histograms.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$sel(\sigma_P=\text{isCitizenOf})$</td>
<td>$1.06 \times 10^{-4}$</td>
</tr>
<tr>
<td>$sel(\sigma_O=\text{United States})$</td>
<td>$6.41 \times 10^{-4}$</td>
</tr>
<tr>
<td>$sel(\sigma_P=\text{isCitizenOf} \land O=\text{United States})$</td>
<td>$4.86 \times 10^{-5}$</td>
</tr>
<tr>
<td>$sel(\sigma_P=\text{isCitizenOf}) \land sel(\sigma_O=\text{United States})$</td>
<td>$6.80 \times 10^{-8}$</td>
</tr>
</tbody>
</table>

3. RDF predicates are usually string values -> histograms are deemed inappropriate for estimation.

4. RDF-3X’s solution.
1. RDF data does not have a fixed schema

2. The outgoing “predicate” edges gives an idea about the “class” of the entity.
   
   e.g. - Artist, City, Country.

3. A “soft” schema hence occur in data, based on the predicates of a subject.
Characteristic set

\[ S_C(s) := \{ p | \exists o : (s, p, o) \in R \} \]

Set of all predicates that have at least one tuple with the subject
Characteristic set

\[ S_C(s) := \{ p | \exists o : (s, p, o) \in R \} \]

Set of all predicates that have at least one tuple with the subject

\[ S_C(\text{“Google”}) = \{ \text{“product”}, \text{“founder”}, \text{“founded_in”}, \text{“CEO”}, \text{“website”} \} \]
Set of characteristic set

\[ S_C(R) := \{ S_C(s) | \exists p, o : (s, p, o) \in R \} \]

Set of characteristic sets of all subject s give that there exists atleast one pair of predicate p and object o
Set of characteristic sets of all subject s give that there exists at least one pair of predicate p and object o

\[ S_C(R) := \{ S_C(s) | \exists p, o : (s, p, o) \in R \} \]

Set of characteristic sets of all subject s give that there exists at least one pair of predicate p and object o

\{ “Author”, “Title”, “Publisher”, “ISBN”, “Year”, “Language” \}

\{ “Founder”, “Founded In”, “CEO”, “CFO”, “Product”, “Revenue”, “Profit” \}

\{ “Country”, “Province”, “Population”, “latitude”, “longitude” \}

“The girl with a dragon tattoo”

“Namesake”

“Tell me your Dreams”

“Amazon”

“Google”

“Tesla”

“New York”

“Mumbai”

“Toronto”
Calculating simple cardinality

- Star-shaped edge structures are also present in queries.
- Each triple describes only one characteristic of the subject.
- Hence, queries have multiple triple patterns with one subject variable.
Calculating simple cardinality

- Star-shaped edge structures are also present in queries.
- Each triple describes only one characteristic of the subject.
- Hence, queries have multiple triple patterns with one subject variable.

```
SELECT DISTINCT ?e
WHERE { ?e <author> ?a , ?e <title> ?b }
```
Calculating simple cardinality

Q =
SELECT DISTINCT ?e
WHERE { ?e <author> ?a , ?e <title> ?b }

$S_c(Q) = \{ \text{“title”}, \text{“author”} \}$

SOLUTION

$\sum_{S \in \{S | S \in S_c(R) \land \{\text{author, title}\} \subseteq S\}} \text{count}(S)$

Sum of cardinalities of all the supersets of query characteristic sets in $S_c(R)$
Occurrence annotations

Limitation of previous calculations:

- Only works if there is a DISTINCT in the selection clause
Limitation of previous calculations:

- Only works if there is a DISTINCT in the selection clause

\[ S_c(\text{ent 416}) = \{ \text{“title”}, \text{“author”} \} \]

\[ \text{count} = 1 \]
Occurrence annotations

Limitation of previous calculations:

- Only works if there is a DISTINCT in the selection clause

Let it Snow

\[ S_c(<ent\ 416>) = \{ \text{“title”, “author”} \} \]

count = 1

SELECT DISTINCT ?e
WHERE { ?e <author> ?a , ?e <title> ?b }
Occurrence annotations

Predicate Annotations!

- Number of occurrences for each predicate in the characteristic set is also stored

\[
S = \{ \text{p1, p2, p3} \ldots \}
\]

| distinct | \(|\{s \mid \exists p, o : (s, p, o) \in R \land S_C(s) = S\}\)| |
|----------|--------------------------------------------------|
| count\(p_1\) | \(|\{(s, p_1, o) \mid (s, p_1, o) \in R \land S_C(s) = S\}\}| |
| count\(p_2\) | \(|\{(s, p_2, o) \mid (s, p_2, o) \in R \land S_C(s) = S\}\}| |
| \ldots | \ldots |
Occurrence annotations

\[ Q = \]
SELECT DISTINCT ?e
WHERE { ?e <author> ?a , ?e <title> ?b }

\[ S_c(Q) = \{ "title", "author" \} \]
Occurrence annotations

Q =
SELECT DISTINCT ?e
WHERE { ?e <author> ?a , ?e <title> ?b }

$S_c(Q) = \{ "title", "author" \}$

$S = \{ "title", "author", "year" \}$

<table>
<thead>
<tr>
<th>distinct</th>
<th>author</th>
<th>title</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2300</td>
<td>1010</td>
<td>1090</td>
</tr>
</tbody>
</table>

avg. author
= 2300/1000 = 2.3

avg. title
= 1010/1000 = 1.01

2323, not 1000

- There can be a loss of precision
Queries with bounded objects

- We stored the count of predicate for each characteristic set it appeared in -> correlation b/w subject and predicate.
- Opt the same strategy for storing the correlation b/w subject predicate and object? INEFFICIENT
Queries with bounded objects

- We stored the count of predicate for each characteristic set it appeared in -> correlation b/w subject and predicate.
- Opt the same strategy for storing the correlation b/w subject predicate and object? INEFFICIENT

OBSERVATION

- Subjects of a characteristic set follow similar behavior.
- In each characteristic set there is one predicate that is least selective -> key of a relational table.
- Other predicates follow the “key” predicate.
Queries with bounded objects

- Out of the multiple object bounded patterns, take the one most selective.
- Other object-bound is assumed to have soft functional dependency.
- Overestimation.

\[
\text{sel}(?o = x | ?p = p) \in \left[ \frac{1}{p_d}, 1 \right].
\]
Cardinality of Star Joins

Complete Algorithm

\text{STARJOIN Cardinality}(S_C, Q = \{ (?, s, p_1, o_1), \ldots, (?, s, p_n, o_n) \})

S_Q = \{ p_1, \ldots, p_n \}

\text{card} = 0

\text{for each } S \in S_C : S_Q \subseteq S

\quad m = 1

\quad o = 1

\quad \text{for } i = 1 \text{ to } n

\quad \quad \text{if } ?o_i \text{ is bound to a value } o_i

\quad \quad \quad o = \min(o, sel(?o_i = o_i | ?p = p_i))

\quad \quad \text{else}

\quad \quad \quad m = m * \frac{S\.count(p_i)}{S\.distinct}

\quad \text{card} = \text{card} + S\.distinct * m * o

\text{return } \text{card}
Cardinality of Star Joins

Complete Algorithm

```
\text{STARJOIN\textsc{CARDINALITY}}(S_C, Q = \{(\?s, p_1, \?o_1), \ldots, \\
(\?s, p_n, \?o_n)\})

S_Q = \{p_1, \ldots, p_n\}
\text{card} = 0

\text{for each } S \in S_C : S_Q \subseteq S
    m = 1
    o = 1
    \text{for } i = 1 \text{ to } n
        \text{if } \?o_i \text{ is bound to a value } o_i
            o = \min(o, \text{sel}(\?o_i = o_i \mid \?p = p_i))
        \text{else}
            m = m \cdot \frac{S.\text{count}(p_i)}{S.\text{distinct}}
    \text{card} = \text{card} + S.\text{distinct} \cdot m \cdot o
\text{return card}
```

Loops over all the characteristic sets in $S_C$ that is the super-set of the Query characteristic set.
Cardinality of Star Joins

**Complete Algorithm**

\[
\text{STARJOINCARDINALITY}(S_C, Q = \{(s, p_1, o_1), \ldots, (s, p_n, o_n)\})
\]

\[
S_Q = \{p_1, \ldots, p_n\}
\]

\[
card = 0
\]

**for each** \(S \in S_C : S_Q \subseteq S\)

\[
m = 1
\]

\[
o = 1
\]

**for** \(i = 1\) **to** \(n\)

**if** \(?o_i\) is bound to a value \(o_i\)

\[
o = \min(o, \text{sel}(?o_i = o_i | ?p = p_i))
\]

**else**

\[
m = m \times \frac{S.\text{count}(p_i)}{S.\text{distinct}}
\]

\[
card = card + S.\text{distinct} \times m \times o
\]

**return** \(card\)

Loops over all the triples that appear in the query
Cardinality of Star Joins

Complete Algorithm

\text{STARJOINCARDINALITY}(\mathcal{S}_C,Q = \{(s, p_1, o_1), \ldots, (s, p_n, o_n)\})

\begin{align*}
S_Q &= \{p_1, \ldots, p_n\} \\
\text{card} &= 0 \\
\text{for each } S &\in \mathcal{S}_C : S_Q \subseteq S \\
&\quad m = 1 \\
&\quad o = 1 \\
&\quad \text{for } i = 1 \text{ to } n \\
&\quad \quad \text{if } o_i \text{ is bound to a value } o_i \\
&\quad \quad \quad o = \min(o, \text{sel}(o_i = o_i | p = p_i)) \\
&\quad \quad \text{else} \\
&\quad \quad \quad m = m \times \frac{S.\text{count}(p_i)}{S.\text{distinct}} \\
&\quad \quad \text{card} = \text{card} + S.\text{distinct} \times m \times o \\
\text{return } \text{card}
\end{align*}

if object is bounded, take the minimum of the selectivity lower bound among all object-bounded triples in query
Cardinality of Star Joins

Complete Algorithm

\[
\text{STARJOIN\textsc{CARDINALITY}}(S_C, Q = \{(?s, p_1, ?o_1), \ldots, \linebreak[2] (?s, p_n, ?o_n)\})
\]

\[S_Q = \{p_1, \ldots, p_n\}\]

\[\text{card} = 0\]

\textbf{for each} \(S \in S_C : S_Q \subseteq S\)

\[m = 1\]

\[o = 1\]

\textbf{for} \(i = 1\) \textbf{to} \(n\)

\textbf{if} \(?o_i\) is bound to a value \(o_i\)

\[o = \min(o, \text{sel}(?o_i = o_i | ?p = p_i))\]

\textbf{else}

\[m = m \ast \frac{S.\text{count}(p_i)}{S.\text{distinct}}\]

\[\text{card} = \text{card} + S.\text{distinct} \ast m \ast o\]

\text{return} \(\text{card}\)

else, update the cumulative selectivity \(m\)
Cardinality of Star Joins

Complete Algorithm

\texttt{STARJOIN\textsc{CARDINALITY}}(S_C, Q = \{(?s, p_1, ?o_1), \ldots, 
(?s, p_n, ?o_n)\} )

\( S_Q = \{p_1, \ldots, p_n\} \)
\( \text{card} = 0 \)

\textbf{for each} \( S \in S_C : S_Q \subseteq S \)
\( m = 1 \)
\( o = 1 \)
\textbf{for} \( i = 1 \) \textbf{to} \( n \)
\textbf{if} \( ?o_i \) is bound to a value \( o_i \)
\( o = \min(o, sel(?o_i = o_i | ?p = p_i)) \)
\textbf{else}
\( m = m \times \frac{S.\text{count}(p_i)}{S.\text{distinct}} \)
\( \text{card} = \text{card} + S.\text{distinct} * m * o \)

\textbf{return} \( \text{card} \)

Calculate the cardinality in current characteristic set and add to global cardinality
Handling diverse sets

- The number of characteristic sets in a data can be very large.
- Keeps only the most frequent 10,000 characteristic sets.
- Merge the others with the most frequent ones.
Handling diverse sets

- The number of characteristic sets in a data can be very large.
- Keeps only the most frequent 10,000 characteristic sets.
- Merge the others with the most frequent ones.

Merging solutions:

\[
\begin{align*}
S_1 &= \{(\text{author}, 120), 100\} \\
S_2 &= \{(\text{title}, 230), 200\} \\
S_3 &= \{(\text{author}, 2300), (\text{title}, 1001), (\text{year}, 1000), 1000 \}\} \\
S_4 &= \{(\text{author}, 30), (\text{title}, 20), 20\}
\end{align*}
\]
Handling diverse sets

- The number of characteristic sets in a data can be very large.
- Keeps only the most frequent 10,000 characteristic sets.
- Merge the others with the most frequent ones.

MERGING SOLUTIONS

\[ S_1 = \{(\text{author}, 150), 120\} \]
\[ S_2 = \{(\text{title}, 250), 140\} \]
\[ S_3 = \{(\text{author}, 2300), (\text{title}, 1001), (\text{year}, 1000), 1000\} \]
\[ S_4 = \{(\text{author}, 30), (\text{title}, 20), 20\} \]

- UNDERESTIMATION
Handling diverse sets

- The number of characteristic sets in a data can be very large.
- Keeps only the most frequent 10,000 characteristic sets.
- Merge the others with the most frequent ones.

**MERGING SOLUTIONS**

\[ S_1 = \{(author, 120), 100\} \]
\[ S_2 = \{(title, 230), 200\} \]
\[ S_3 = \{(author, 2300), (title, 1001), (year, 1000), 1000\} \]
\[ S_4 = \{(author, 30), (title, 20), 20\} \]

**OVERESTIMATION**
Handling diverse sets

- The number of characteristic sets in a data can be very large.
- Keeps only the most frequent 10,000 characteristic sets.
- Merge the others with the most frequent ones.

\[ S_3 = \{(\text{author}, 2330), (\text{title}, 1021), (\text{year}, 1000), (\text{year}, 1020)\} \]

Merging solutions

- Prefer overestimations.
- Increases only small error, but gives correct upper bound in computation.

Overestimation
Merging algo

\begin{align*}
\text{MERGECHARACTERISTICSETS}(S_C, S) \\
\tilde{S} &= \{ S' \mid S' \in S_C \land S \subseteq S' \} \\
\text{if } \tilde{S} \neq \emptyset & \\
\tilde{S}' &= \{ S' \mid S' \in \tilde{S} \land |S'| = \min(\{|S'| \mid S' \in \tilde{S}\}) \} \\
\text{merge } S \text{ into } \arg \max_{S' \in \tilde{S}'} S'.\text{distinct} \\
\text{else} & \\
\tilde{S} &= \{ S' \mid S' \subset S \land \exists S'' \in S_C : S' \subseteq S'' \} \\
S_1 &= \arg \max_{S' \in \tilde{S}} |S'|, \quad S_2 = S \setminus S_1 \\
\text{if } S_1 \neq \emptyset & \\
\text{MERGECHARACTERISTICSETS}(S_C, S_1) \quad \text{MERGECHARACTERISTICSETS}(S_C, S_2)
\end{align*}

Set of all characteristic sets that are superset of S.
Merging algo

\textbf{M}erge\textbf{C}haracteristic\textbf{S}ets($S_C$, $S$)

$\bar{S} = \{S' | S' \in S_C \land S \subseteq S'\}$

\textbf{if} $\bar{S} \neq \emptyset$

\begin{align*}
\bar{S}' &= \{S' | S' \in \bar{S} \land |S'| = \min(\{|S'| | S' \in \bar{S}\})\} \\
\text{merge } S \text{ into } \arg\max_{S' \in \bar{S}'} S'.\text{distinct}
\end{align*}

\textbf{else}

$\bar{S} = \{S' | S' \subset S \land \exists S'' \in S_C : S' \subseteq S''\}$

$S_1 = \arg\max_{S' \in \bar{S}} |S'|, S_2 = S \setminus S_1$

\textbf{if} $S_1 \neq \emptyset$

\begin{align*}
\text{M}erge\text{C}haracteristic\text{S}ets(S_C, S_1) \\
\text{M}erge\text{C}haracteristic\text{S}ets(S_C, S_2)
\end{align*}

$\bar{S}' = \text{Set of all characteristic sets which have the least elements in } \bar{S}$

merge $S$ with the one which has the maximum distinct
Merging algo

\[
\text{MERGEC\text{\textsc{HARACTERISTICSETS}}}(S_C, S) \\
\hat{S} = \{S' \mid S' \in S_C \land S \subseteq S'\} \\
\text{if } \hat{S} \neq \emptyset \\
\quad \hat{S}' = \{S' \mid S' \in \hat{S} \land |S'| = \min(\{|S'| \mid S' \in \hat{S}\})\} \\
\quad \text{merge } S \text{ into } \arg \max_{S' \in \hat{S}'} S'.\text{distinct} \\
\text{else} \\
\quad \hat{S} = \{S' \mid S' \subset S \land \exists S'' \in S_C : S' \subseteq S''\} \\
\quad S_1 = \arg \max_{S' \in \hat{S}} |S'|, S_2 = S \setminus S_1 \\
\quad \text{if } S_1 \neq \emptyset \\
\quad \quad \text{MERGEC\text{\textsc{HARACTERISTICSETS}}}(S_C, S_1) \\
\quad \quad \text{MERGEC\text{\textsc{HARACTERISTICSETS}}}(S_C, S_2)
\]

Else, break S into $S_1$ and $S_2$, such that $S_1$ is the maximal subset of a characteristic set in $S_C$

Merge $S_1$ and $S_2$
Merging algo

\[
\text{MERGECHARACTERISTICSETS}(S_C, S) \\
\bar{S} = \{S' | S' \in S_C \land S \subseteq S'\} \\
\text{if } \bar{S} \neq \emptyset \\
\bar{S}' = \{S' | S' \in \bar{S} \land |S'| = \min(\{|S'| | S' \in \bar{S}\})\} \\
\text{merge } S \text{ into } \arg\max_{S' \in \bar{S}'} S'.\text{distinct} \\
\text{else} \\
\bar{S} = \{S' | S' \subset S \land \exists S'' \in S_C : S' \subseteq S''\} \\
S_1 = \arg\max_{S' \in \bar{S}} |S'|, S_2 = S \setminus S_1 \\
\text{if } S_1 \neq \emptyset \\
\text{MERGECHARACTERISTICSETS}(S_C, S_1) \\
\text{MERGECHARACTERISTICSETS}(S_C, S_2)
\]

Else, break \( S \) into \( S_1 \) and \( S_2 \), such that \( S_1 \) is the maximal subset of a characteristic set in \( S_C \)

Merge \( S_1 \) and \( S_2 \)
Using characteristic sets

Principles for using characteristic set based cardinality estimator into the plan generator:
Using characteristic sets

Principles for using characteristic set based cardinality estimator into the plan generator:

#1

Calculate cardinality estimate once per equivalent query plans

- Cardinality is independent of the plan structure
- It should not change by changing the ordering of operators.
Using characteristic sets

Principles for using characteristic set based cardinality estimator into the plan generator:

#2 Use maximum amount of consistent correlation information

- A typical query graph has a lot of joins, we can have consistent information for only a few portions of the graph.
- We use characteristic sets to estimate to the maximum portion of the graph, before starting to use join estimates.
Using characteristic sets

Principles for using characteristic set based cardinality estimator into the plan generator:

#3

Assume independence if no correlation information is available.

- If no consistent info available, we assume independence to calculate estimates using general join stats.
- It introduces error.
- Error is relatively low, since independence is being assumed very “late” in cost estimation.
SELECT ?a ?t
         ?p <name> ?”ACM” }
General Query

SELECT ?a ?t
• Bottom-up Dynamic Programming approach. At each step, match one of the query patterns.

• We use the already calculated cardinality for the query subgraph from the DP table, if available.

• Else, we calculate the cardinality for the part of graph using the ESTIMATE QUERY CARDINALITY function
Estimation Algorithm

**ESTIMATE_QUERY_CARDINALITY**

\[
\text{card}(Q) = \prod_{R \in V} |R| \prod_{p \in E} \text{sel}(p)
\]
**ESTIMATEQUERYCARDINALITY**$(Q)$

Q$^R$ = RDF query graph derived from Q

card = 1

mark all nodes and edges in Q and Q$^R$ as uncovered

while uncovered Q$^R$ contains subject star joins

S = largest subject star join in the uncovered part of Q$^R$

mark S as covered in Q and Q

\[ \text{card} = \text{card} \times \text{STARJOINCARDINALITY}(S_C, S) \]

while uncovered Q$^R$ contains object star joins

S = largest object star join in the uncovered part of Q$^R$

mark S as covered in Q and Q

\[ \text{card} = \text{card} \times \text{STARJOINCARDINALITY}(S^O, S) \]

\[ \text{card} = \text{card} \times \prod_{R \in \text{uncovered Q}} |R| \times \prod_{\land_p \in \text{uncovered Q}} \text{sel}(\land_p) \]

return card

Selects the largest subject star join (S) from the uncovered region of Q$^R$ and calculates the cardinality of that star.

marks S in Q$^R$. 
Estimation Algorithm

\text{ESTIMATEQUERYCARDINALITY}(Q)

\begin{align*}
Q^R &= \text{RDF query graph derived from } Q \\
\text{card} &= 1 \\
\text{mark all nodes and edges in } Q \text{ and } Q^R \text{ as uncovered} \\
\text{while} \text{ uncovered } Q^R \text{ contains subject star joins} \\
\quad S &= \text{largest subject star join in the uncovered part of } Q^R \\
\quad \text{mark } S \text{ as covered in } Q^R \text{ and } Q \\
\quad \text{card} &= \text{card} \times \text{STARJOINCARDINALITY}(S_C, S) \\
\text{while} \text{ uncovered } Q^R \text{ contains object star joins} \\
\quad S &= \text{largest object star join in the uncovered part of } Q^R \\
\quad \text{mark } S \text{ as covered in } Q^R \text{ and } Q \\
\quad \text{card} &= \text{card} \times \text{STARJOINCARDINALITY}(S^O_C, S) \\
\text{card} &= \text{card} \times \prod_{R \in \text{uncovered } Q} |R| \times \prod_{\exists p \in \text{uncovered } Q} \text{sel}(\times p) \\
\text{return card}
\end{align*}

Selects the largest object star join (S) from the uncovered region of $Q^R$ and calculates the cardinality of that star. Marks S in $Q^R$. 
Estimation Algorithm

\textbf{ESTIMATE}\textbf{QUERY}\textbf{CARDINALITY}(Q)

\[ Q^R = \text{RDF query graph derived from } Q \]
\[ \text{card} = 1 \]

mark all nodes and edges in \( Q \) and \( Q^R \) as uncovered

\textbf{while} uncovered \( Q^R \) contains subject star joins

\( S = \text{largest subject star join in the uncovered part of } Q^R \)

mark \( S \) as covered in \( Q^R \) and \( Q \)

\[ \text{card} = \text{card} \times \text{STARJOIN}\text{CARDINALITY}(S_C, S) \]

\textbf{while} uncovered \( Q^R \) contains object star joins

\( S = \text{largest object star join in the uncovered part of } Q^R \)

mark \( S \) as covered in \( Q^R \) and \( Q \)

\[ \text{card} = \text{card} \times \text{STARJOIN}\text{CARDINALITY}(S_Q, S) \]

\[ \text{card} = \text{card} \times \prod_{R \in \text{uncovered } Q} |R| \times \prod_{\nabla_p \in \text{uncovered } Q} \text{sel}(\nabla_p) \]

\textbf{return} \( \text{card} \)

Uses independence assumption for all the nodes and edges left in the \( Q^R \) for estimation
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY

* uncovered nodes/edges
# Evaluations

## Systems:
- RDF-3X with Characteristic sets estimator
- RDF-3X original
- Commercial system: DB A
- Commercial system: DB B
- Commercial system: DB C
- Stocker et al. (Stocker)
- Maduko et al. (Maduko)

## Datasets:
- Yago
- LibraryThing
Single Join queries

- $q \text{-error} = \max(\ c^\land/c \ , \ c/c^\land \ ), \ \text{bucketed}$
- queries of the form : \{ (?s p1 ?a) . (?s p2 ?b) \}
- YAGO : 1751 queries, LibraryThing : 19,062,990
Single Join queries

- \( q\text{-error} = \max\left( \frac{c^}{c}, \frac{c}{c^} \right) \), bucketed
- queries of the form: \{ (?s p1 ?a) . (?s p2 ?b) \}
- YAGO: 1751 queries, LibraryThing: 19,062,990
**Single Join queries**

- q-error = max( c^/c , c/c^ ), bucketed
- queries of the form : { (?s p1 ?a) . (?s p2 ?b) }
- YAGO : 1751 queries, LibraryThing : 19,062,990

<table>
<thead>
<tr>
<th>q-error</th>
<th>DB A</th>
<th>DB B</th>
<th>DB C</th>
<th>Stocker</th>
<th>Maduko</th>
<th>RDF-3X</th>
<th>CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 2</td>
<td>15.0</td>
<td>23.2</td>
<td>22.9</td>
<td>0</td>
<td>26.7</td>
<td>30.2</td>
<td>100</td>
</tr>
<tr>
<td>≤ 5</td>
<td>10.5</td>
<td>27.3</td>
<td>27.8</td>
<td>0</td>
<td>40.6</td>
<td>30.9</td>
<td>0</td>
</tr>
<tr>
<td>≤ 10</td>
<td>10.3</td>
<td>17.2</td>
<td>17.7</td>
<td>0</td>
<td>19.7</td>
<td>16.6</td>
<td>0</td>
</tr>
<tr>
<td>≤ 100</td>
<td>35.3</td>
<td>28.8</td>
<td>27.8</td>
<td>0.1</td>
<td>12.0</td>
<td>19.9</td>
<td>0</td>
</tr>
<tr>
<td>≤ 1000</td>
<td>20.7</td>
<td>3.1</td>
<td>3.3</td>
<td>0</td>
<td>0.8</td>
<td>2.2</td>
<td>0</td>
</tr>
<tr>
<td>&gt; 1000</td>
<td>8.1</td>
<td>0.4</td>
<td>0.6</td>
<td>99.9</td>
<td>0.1</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>max</td>
<td>28367552</td>
<td>1416363</td>
<td>7140611</td>
<td>1.3 * 10^{15}</td>
<td>17471</td>
<td>2909310</td>
<td>1.01</td>
</tr>
</tbody>
</table>
Complex Join queries

- Upto 6 joins, with object constraints.
Complex Join queries

- Upto 6 joins, with object constraints.

<table>
<thead>
<tr>
<th></th>
<th>card (g.mean)</th>
<th>median</th>
<th>error max</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact</td>
<td>26347</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>our</td>
<td>13730</td>
<td>0.77</td>
<td>11.34</td>
<td>1.86</td>
</tr>
<tr>
<td>RDF-3X</td>
<td>83</td>
<td>180.56</td>
<td>395397.00</td>
<td>46506.80</td>
</tr>
<tr>
<td>Stocker</td>
<td>1</td>
<td>15863.00</td>
<td>6.45*10^6</td>
<td>994426.00</td>
</tr>
<tr>
<td>Maduko</td>
<td>3</td>
<td>20591.00</td>
<td>6.50*10^6</td>
<td>953590.00</td>
</tr>
<tr>
<td>DB A</td>
<td>1</td>
<td>15863.00</td>
<td>6.45*10^6</td>
<td>994426.00</td>
</tr>
<tr>
<td>DB B</td>
<td>71</td>
<td>1464.81</td>
<td>2.37*10^6</td>
<td>1.29*10^6</td>
</tr>
<tr>
<td>DB C</td>
<td>2</td>
<td>7826.75</td>
<td>2.37*10^6</td>
<td>1.61*10^6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>card (g.mean)</th>
<th>median</th>
<th>error max</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact</td>
<td>1741</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>our</td>
<td>1244</td>
<td>0.17</td>
<td>12.60</td>
<td>1.83</td>
</tr>
<tr>
<td>RDF-3X</td>
<td>20</td>
<td>64.72</td>
<td>768.86</td>
<td>235.01</td>
</tr>
<tr>
<td>Stocker</td>
<td>1</td>
<td>1333.00</td>
<td>520293.00</td>
<td>83145.00</td>
</tr>
<tr>
<td>Maduko</td>
<td>75</td>
<td>29.00</td>
<td>3491.55</td>
<td>451.25</td>
</tr>
<tr>
<td>DB A</td>
<td>3</td>
<td>336.00</td>
<td>278266.00</td>
<td>31548.10</td>
</tr>
<tr>
<td>DB B</td>
<td>722</td>
<td>469.80</td>
<td>70539.20</td>
<td>40098.80</td>
</tr>
<tr>
<td>DB C</td>
<td>35</td>
<td>29.85</td>
<td>11547.00</td>
<td>41453.70</td>
</tr>
</tbody>
</table>
Other datasets

- UniProt data :: >800M triples, <1000 characteristic sets
  - strong schema
  - Very good cardinality estimates
- Billion Triples data :: >1B triples, ~500K characteristic sets
  - Merging

<table>
<thead>
<tr>
<th></th>
<th>≤ 2</th>
<th>≤ 5</th>
<th>≤ 10</th>
<th>≤ 100</th>
<th>≤ 1000</th>
<th>&gt; 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>full CS</td>
<td>99.2</td>
<td>0.4</td>
<td>0.1</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>merged CS</td>
<td>91.7</td>
<td>1.7</td>
<td>0.9</td>
<td>1.3</td>
<td>0.2</td>
<td>4.1</td>
</tr>
</tbody>
</table>
end.