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 (subject, predicate, object).**
- **Single table (3 columns)**

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 }

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.

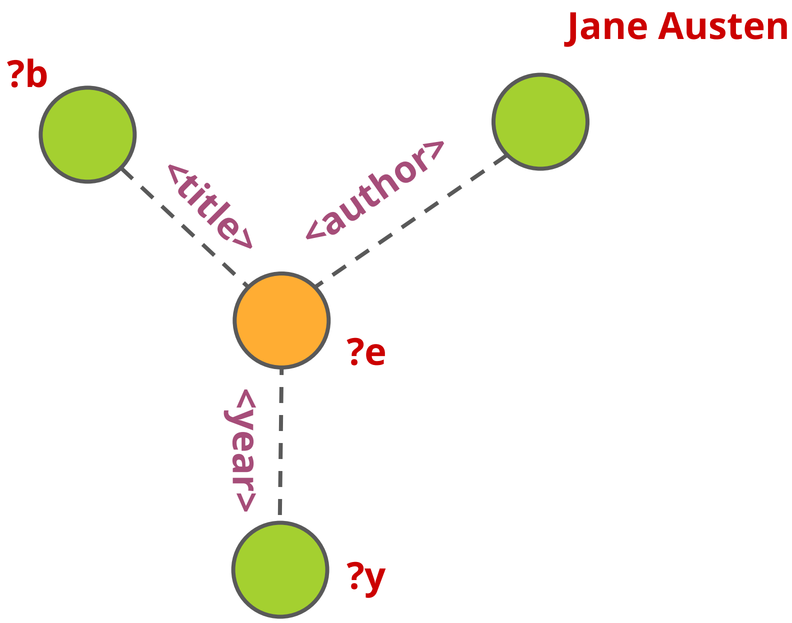
SELECT DISTINCT ?e

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.**

$sel(\sigma_{P=\text{isCitizenOf}})$	$1.06 * 10^{-4}$
$sel(\sigma_{O=\text{United\_States}})$	$6.41 * 10^{-4}$
$sel(\sigma_{P=\text{isCitizenOf} \wedge O=\text{United\_States}})$	$4.86 * 10^{-5}$
$sel(\sigma_{P=\text{isCitizenOf}}) * sel(\sigma_{\text{O=United\_States}})$	$6.80 * 10^{-8}$

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

4. **RDF-3X’s solution.**

Characteristic set

IDEA

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

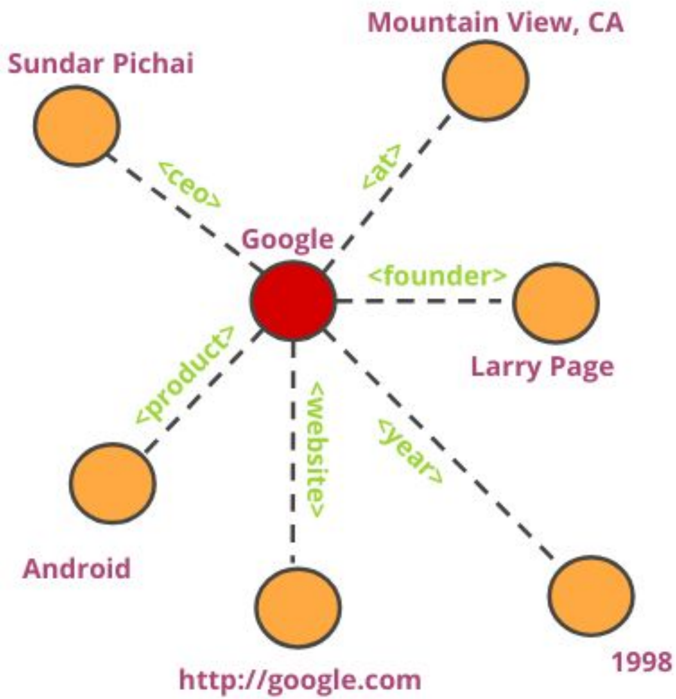
$$S_C(s) := \{p | \exists o : (s, p, o) \in R\}$$

Set of all predicates that have atleast one tuple with the subject

Characteristic set

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

Set of all predicates that have atleast one tuple with the subject



S_C("Google") = {
"product",
"founder",
"founded_in",
"CEO",
"website"
}

Set of characteristic set

$$\mathcal{S}_C(R) := \{S_C(s) | \exists p, o : (s, p, o) \in R\}$$

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

Set of characteristic 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

{ “Author”, “Title”, “Publisher”, “ISBN”, “Year”, “Language” } → “The girl with a dragon tattoo”, “Namesake”, “Tell me your Dreams”

{ “Founder”, “Founded In”, “CEO”, “CFO”, “Product”, “Revenue”, “Profit” } → “Amazon”, “Google”, “Tesla”

{ “Country”, “Province”, “Population”, “latitude”, “longitude” } → “New York”, “Mumbai”, “Toronto”

Calculating simple cardinality

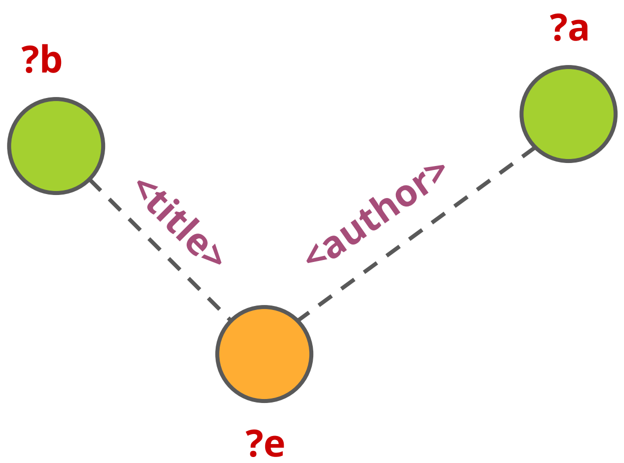
- **Star-shaped edge structures are also present in queries.**
- **Each triple describes only one characteristic of the subject.**
- **Hence, queries have multiple triple patterns with one subject variable.**

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 }

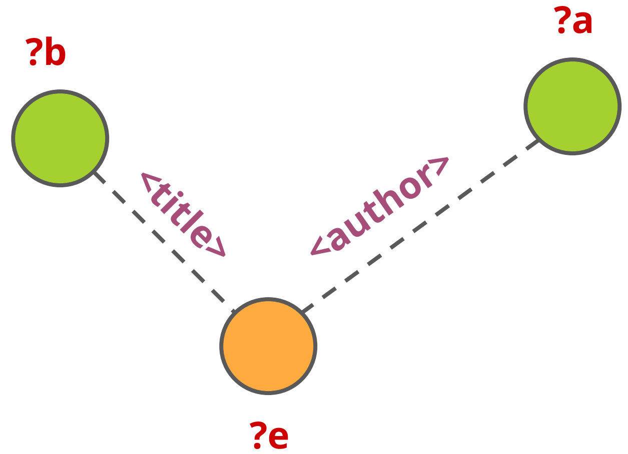


Calculating simple cardinality

Q =

SELECT DISTINCT ?e

WHERE { ?e <author> ?a , ?e <title> ?b }



$S_c(Q)$ = { “title”, “author” }

SOLUTION

$$\sum_{S \in \{S | S \in \mathcal{S}_C(R) \wedge \{author, title\} \subseteq S\}} count(S)$$

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

Occurrence annotations

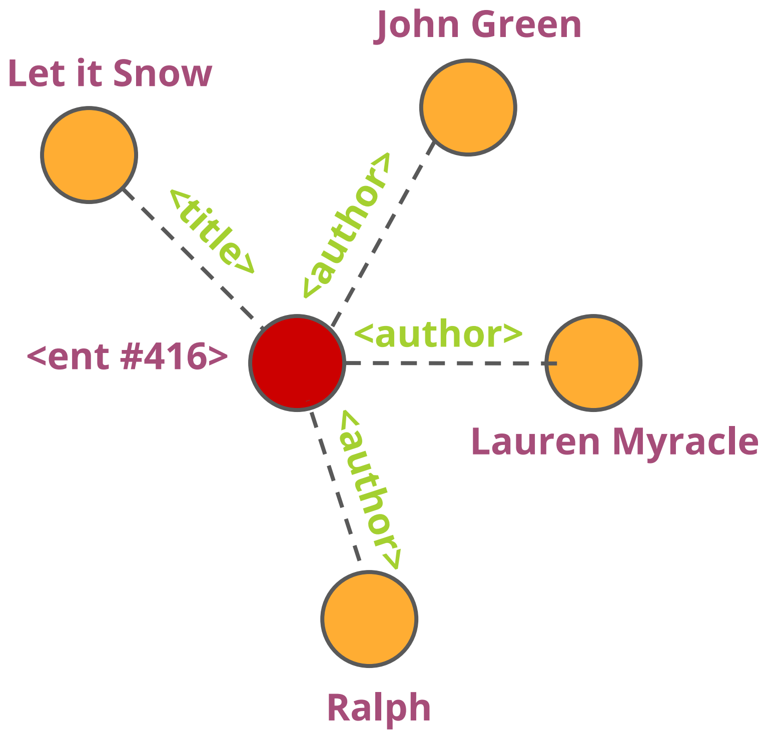
Limitation of previous calculations :

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

Occurrence annotations

Limitation of previous calculations :

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



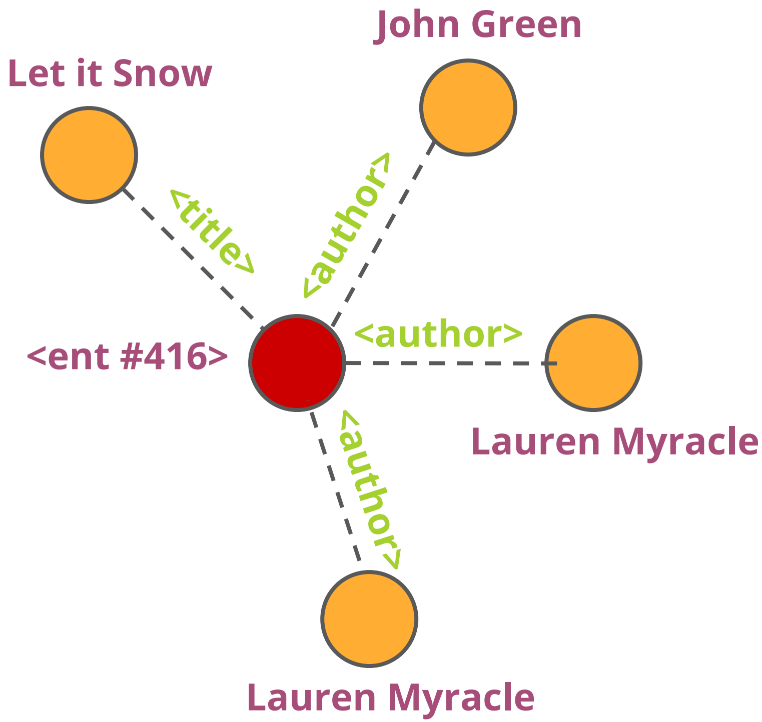
S_C(<ent 416>) = { “title”, “author” }

count = 1

Occurrence annotations

Limitation of previous calculations :

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



S_C(<ent 416>) = { “title”, “author” }

count = 1

SELECT ~~DISTINCT~~ ?e

WHERE { ?e <author> ?a , ?e <title> ?b }

3, not 1

Occurrence annotations

Predicate Annotations !

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

eg. S = { p1, p2, p3 … }

distinct	$\lvert\{s \mid \exists p, o : (s,p,o) \in R \wedge S_C(s) = S\}\rvert$
count(p_1)	$\lvert\{(s,p_1,o) \mid (s,p_1,o) \in R \wedge S_C(s) = S\}\rvert$
count(p_2)	$\lvert\{(s,p_2,o) \mid (s,p_2,o) \in R \wedge S_C(s) = S\}\rvert$
$\ldots$	$\ldots$

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" }

2323, not 1000

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

distinct	author	title	year
1000	2300	1010	1090

avg. author

= 2300/1000 = 2.3

avg. title

= 1010/1000 = 1.01

- 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.

$$sel(?o = x | ?p = p) \in [\frac{1}{p_d}, 1].$$

Cardinality of Star Joins

Complete Algorithm

$$\textsc{StarJoinCardinality}(\mathcal{S}_C, Q = \{(?s, p_1, ?o_1), \ldots, (?s, p_n, ?o_n)\})$$

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

$card = 0$

for each $S \in \mathcal{S}_C : S_Q \subseteq S$

  $m = 1$

  $o = 1$

  for $i = 1$ **to** n

    if $?o_i$ is bound to a value o_i

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

    else

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

  $card = card + S.distinct * m * o$

return $card$

Cardinality of Star Joins

Complete Algorithm

$\textsc{StarJoinCardinality}(\mathcal{S}_C, Q = \{(?s, p_1, ?o_1), \ldots, (?s, p_n, ?o_n)\})$

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

$\textsc{StarJoinCardinality}(\mathcal{S}_C, Q = \{(?s, p_1, ?o_1), \ldots, (?s, p_n, ?o_n)\})$

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

Loops over all the triples that appear in the query

Cardinality of Star Joins

Complete Algorithm

$$\textsc{StarJoinCardinality}(\mathcal{S}_C, Q = \{(?s, p_1, ?o_1), \ldots, (?s, p_n, ?o_n)\})$$

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

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

$\textsc{StarJoinCardinality}(\mathcal{S}_C, Q = \{(?s, p_1, ?o_1), \ldots, (?s, p_n, ?o_n)\})$

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

$card = 0$

for each $S \in \mathcal{S}_C : S_Q \subseteq S$

  $m = 1$

  $o = 1$

  for $i = 1$ **to** n

    if $?o_i$ is bound to a value o_i

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

    else

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

  $card = card + S.distinct * m * o$

return $card$

else, update the cummulative selectivity (m)

Cardinality of Star Joins

Complete Algorithm

$$\textsc{StarJoinCardinality}(\mathcal{S}_C, Q = \{(?s, p_1, ?o_1), \ldots, (?s, p_n, ?o_n)\})$$

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

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

S_1 = {(author, 120), 100}

S_2 = {(title, 230), 200}

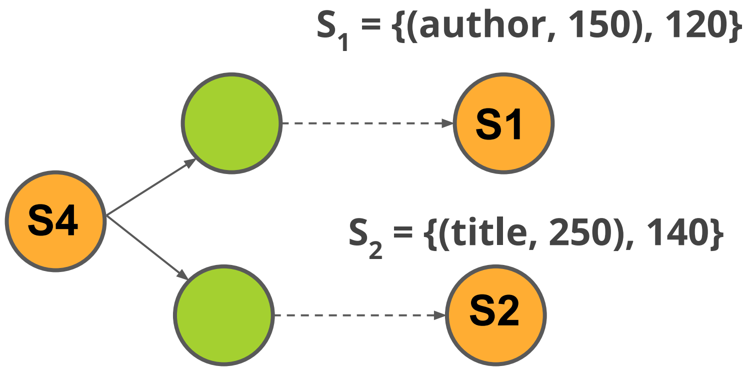
S_3 = {(author, 2300), (title, 1001), (year, 1000), 1000 }

S_4 = {(author, 30), (title, 20), 20}

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



- UNDERESTIMATION

S_1 = {(author, 120), 100}

S_2 = {(title, 230), 200}

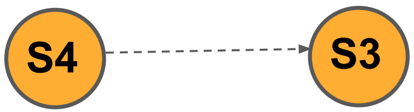
S_3 = {(author, 2300), (title, 1001), (year, 1000), 1000 }

S_4 = {(author, 30), (title, 20), 20}

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_3 = {(author, 2330), (title, 1021), (year, 1000), 1020 }

- OVERESTIMATION

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}

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

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

S_3 = {(author, 2330), (title, 1021), (year, 1000), 1020 }

- OVERESTIMATION

Merging algo

$\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S)$
$\bar{S} = \{S' | S' \in \mathcal{S}_C \wedge S \subseteq S'\}$
if $\bar{S} \neq \emptyset$
  $\bar{S}' = \{S' | S' \in \bar{S} \wedge |S'| = min(\{|S'|| S' \in \bar{S}\})\}$
  merge S into $\arg\max_{S' \in \bar{S}'} S'.distinct$
else
  $\bar{S} = \{S' | S' \subset S \wedge \exists S'' \in \mathcal{S}_C : S' \subseteq S''\}$
  $S_1 = \arg\max_{S' \in \bar{S}} |S'|, S_2 = S \setminus S_1$
  if $S_1 \neq \emptyset$
    $\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S_1)$
    $\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S_2)$

Set of all characteristic sets that are superset of S.

Merging algo

$\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S)$

$\bar{S} = \{S' | S' \in \mathcal{S}_C \wedge S \subseteq S'\}$

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

$\bar{S}' = \{S' | S' \in \bar{S} \wedge |S'| = min(\{|S'| | S' \in \bar{S}\})\}$

merge S into $\arg\max_{S' \in \bar{S}'} S'.distinct$

else

$\bar{S} = \{S' | S' \subset S \wedge \exists S'' \in \mathcal{S}_C : S' \subseteq S''\}$

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

if $S_1 \neq \emptyset$

$\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S_1)$

$\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S_2)$

$\bar{S}'$ = 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

$\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S)$
$\bar{S} = \{S' | S' \in \mathcal{S}_C \wedge S \subseteq S'\}$
if $\bar{S} \neq \emptyset$
  $\bar{S}' = \{S' | S' \in \bar{S} \wedge |S'| = min(\{|S'| | S' \in \bar{S}\})\}$
  merge S into $\arg\max_{S' \in \bar{S}'} S'.distinct$
else
  $\bar{S} = \{S' | S' \subset S \wedge \exists S'' \in \mathcal{S}_C : S' \subseteq S''\}$
  $S_1 = \arg\max_{S' \in \bar{S}} |S'|, S_2 = S \setminus S_1$
  if $S_1 \neq \emptyset$
    $\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S_1)$
    $\textsc{MergeCharacteristicSets}(\mathcal{S}_C, S_2)$

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

$\text{MERGECHARACTERISTICSETS}(\mathcal{S}_C, S)$
$\bar{S} = \{S' | S' \in \mathcal{S}_C \wedge S \subseteq S'\}$
if $\bar{S} \neq \emptyset$
$\quad \bar{S}' = \{S' | S' \in \bar{S} \wedge |S'| = min(\{|S'| | S' \in \bar{S}\})\}$
$\quad$ merge S into $\arg\max_{S' \in \bar{S}'} S'.distinct$
else
$\quad \bar{S} = \{S' | S' \subset S \wedge \exists S'' \in \mathcal{S}_C : S' \subseteq S''\}$
$\quad S_1 = \arg\max_{S' \in \bar{S}} |S'|, S_2 = S \setminus S_1$
$\quad$ **if** $S_1 \neq \emptyset$
$\quad\quad \text{MERGECHARACTERISTICSETS}(\mathcal{S}_C, S_1)$
$\quad\quad \text{MERGECHARACTERISTICSETS}(\mathcal{S}_C, S_2)$

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.**

General Query

SELECT ?a ?t

WHERE { ?b <author> ?a , ?b <title> ?t, ?b <year> ‘2009’, ?b <published_by> ?p,

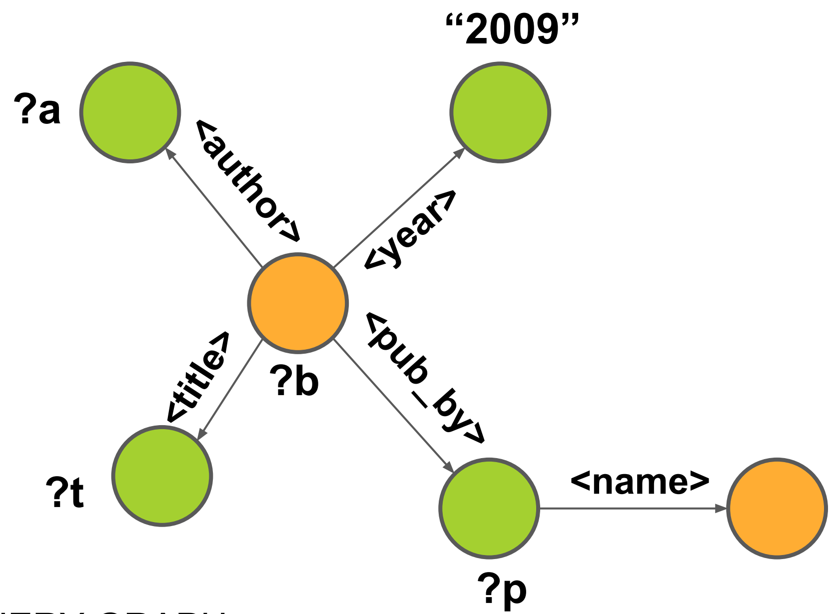
?p <name> ? ”ACM” }

General Query

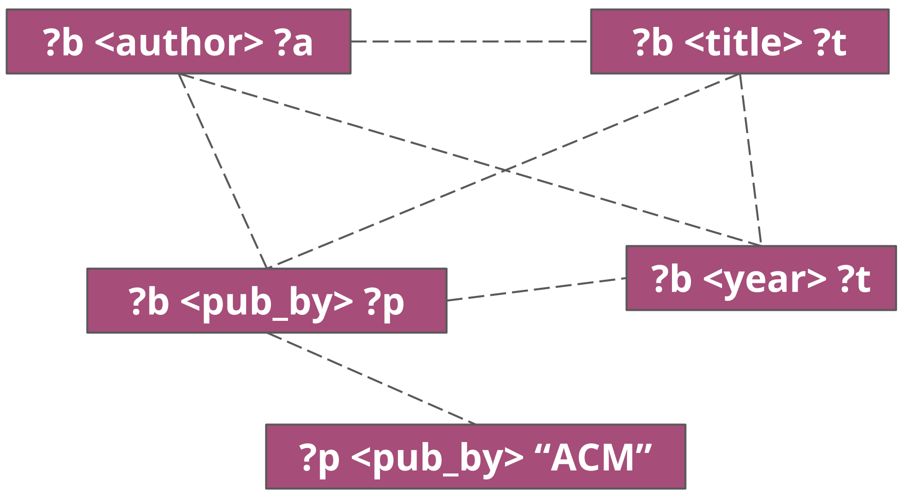
SELECT ?a ?t

WHERE { ?b <author> ?a , ?b <title> ?t, ?b <year> ‘2009’, ?b <published_by> ?p,

?p <name> ? ”ACM” }



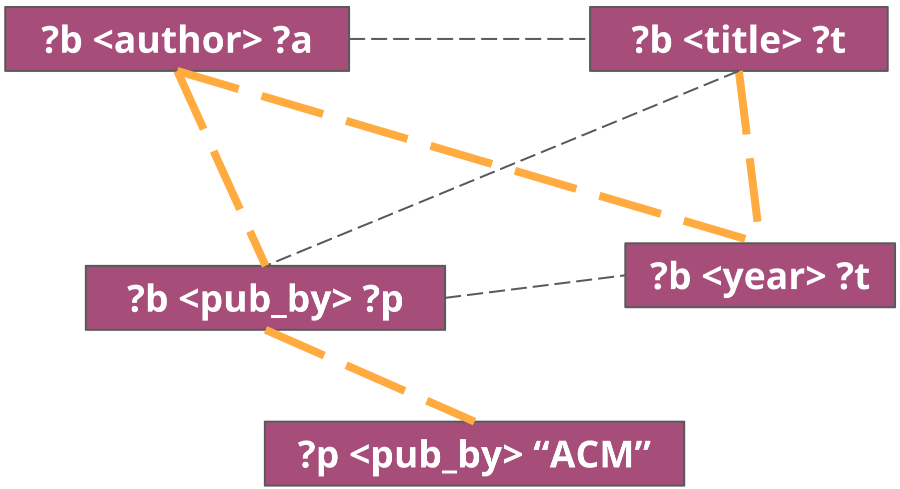
QUERY GRAPH



JOIN GRAPH

Join Tree

- **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**



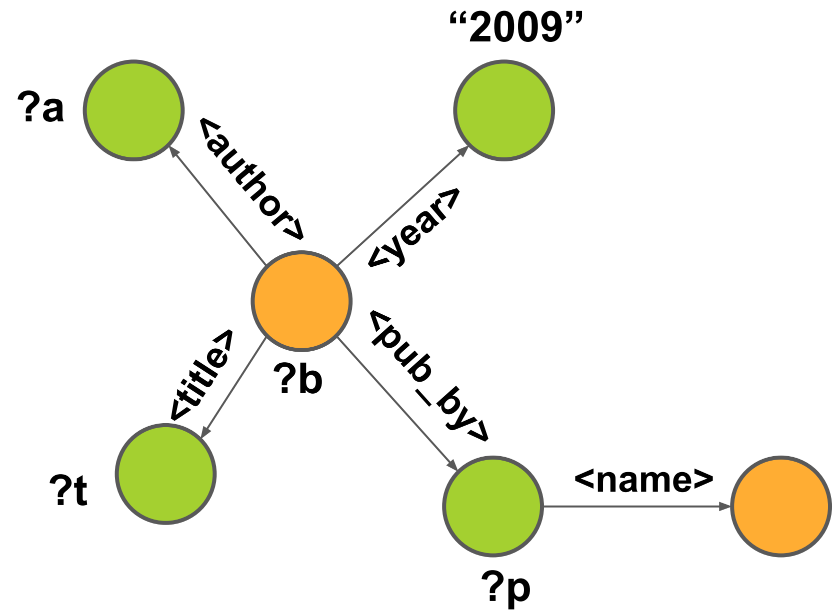
optimal join tree

Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY

WORST CASE

$$card(Q) = \prod_{R \in V} |R| \prod_{p \in E} sel(p)$$



Estimation Algorithm

$\textsc{EstimateQueryCardinality}(Q)$
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^R and Q
  $card = card * \textsc{StarJoinCardinality}(\mathcal{S}_C, S)$
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^R and Q
  $card = card * \textsc{StarJoinCardinality}(\mathcal{S}_C^O, S)$
$card = card * \prod_{R \in uncoveredQ} |R| * \prod_{\bowtie_p \in uncoveredQ} sel(\bowtie_p)$
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

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^R and Q
    $card = card *$ STARJOINCARDINALITY($\mathcal{S}_C$,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^R and Q
    $card = card *$ STARJOINCARDINALITY($\mathcal{S}_C^O$,S)
$card = card * \prod_{R \in uncoveredQ} |R| * \prod_{\bowtie_p \in uncoveredQ} sel(\bowtie_p)$
return $card$

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.

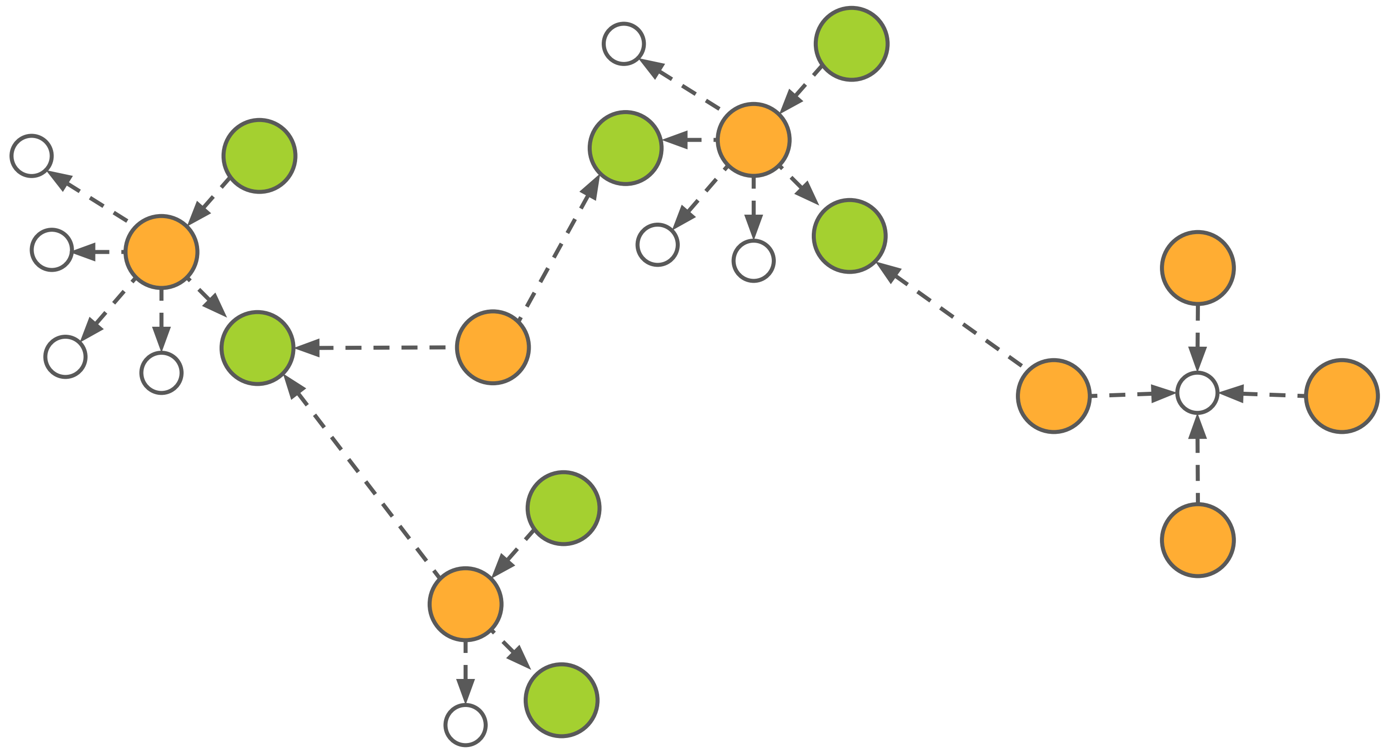
Estimation Algorithm

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^R and Q
    $card = card * \text{StarJoinCardinality}(\mathcal{S}_C, S)$
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^R and Q
    $card = card * \text{StarJoinCardinality}(\mathcal{S}_C^O, S)$
$card = card * \prod_{R \in uncoveredQ} |R| * \prod_{\bowtie_p \in uncoveredQ} sel(\bowtie_p)$
return $card$

Uses independence assumption for all the nodes and edges left in the Q_R for estimation

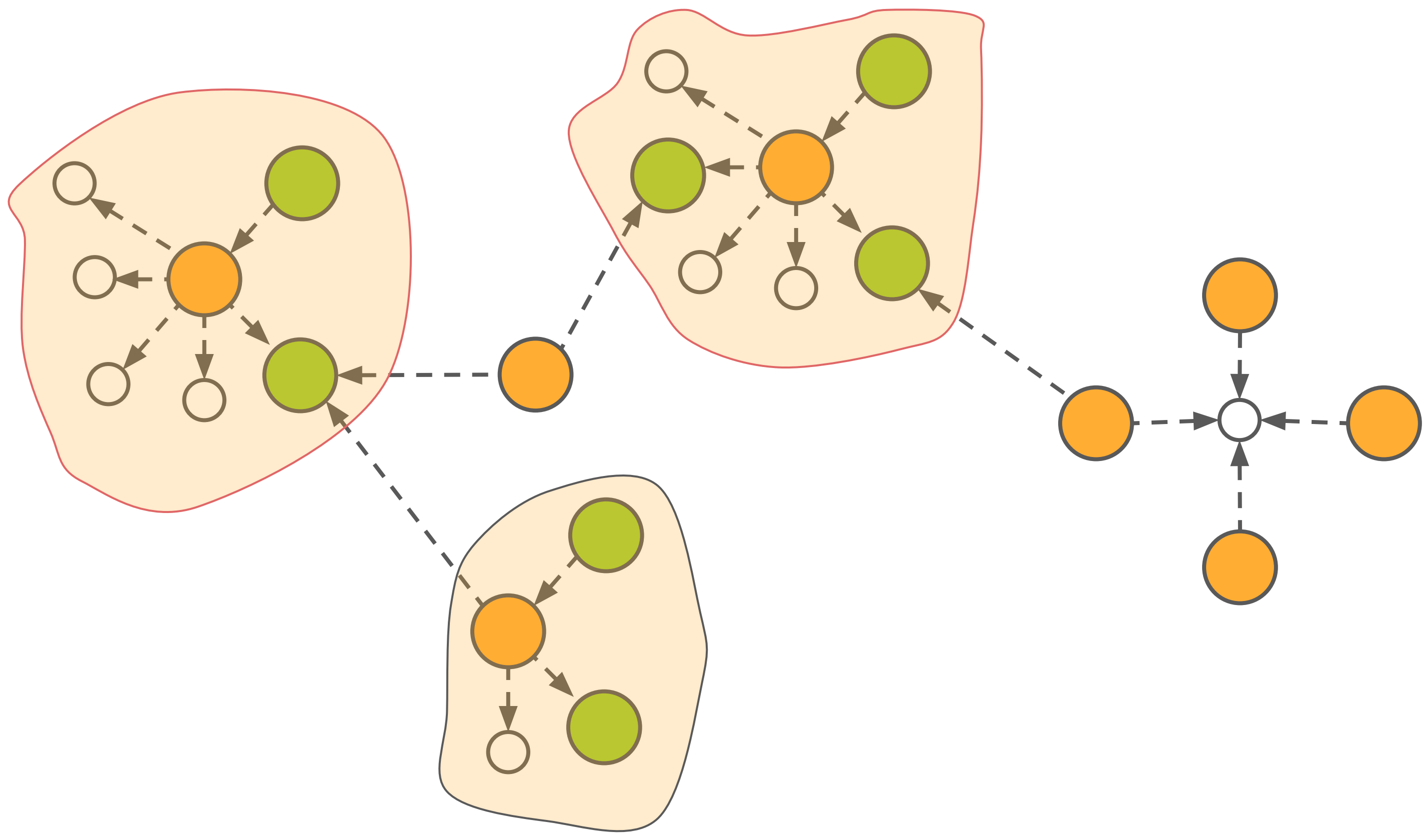
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY



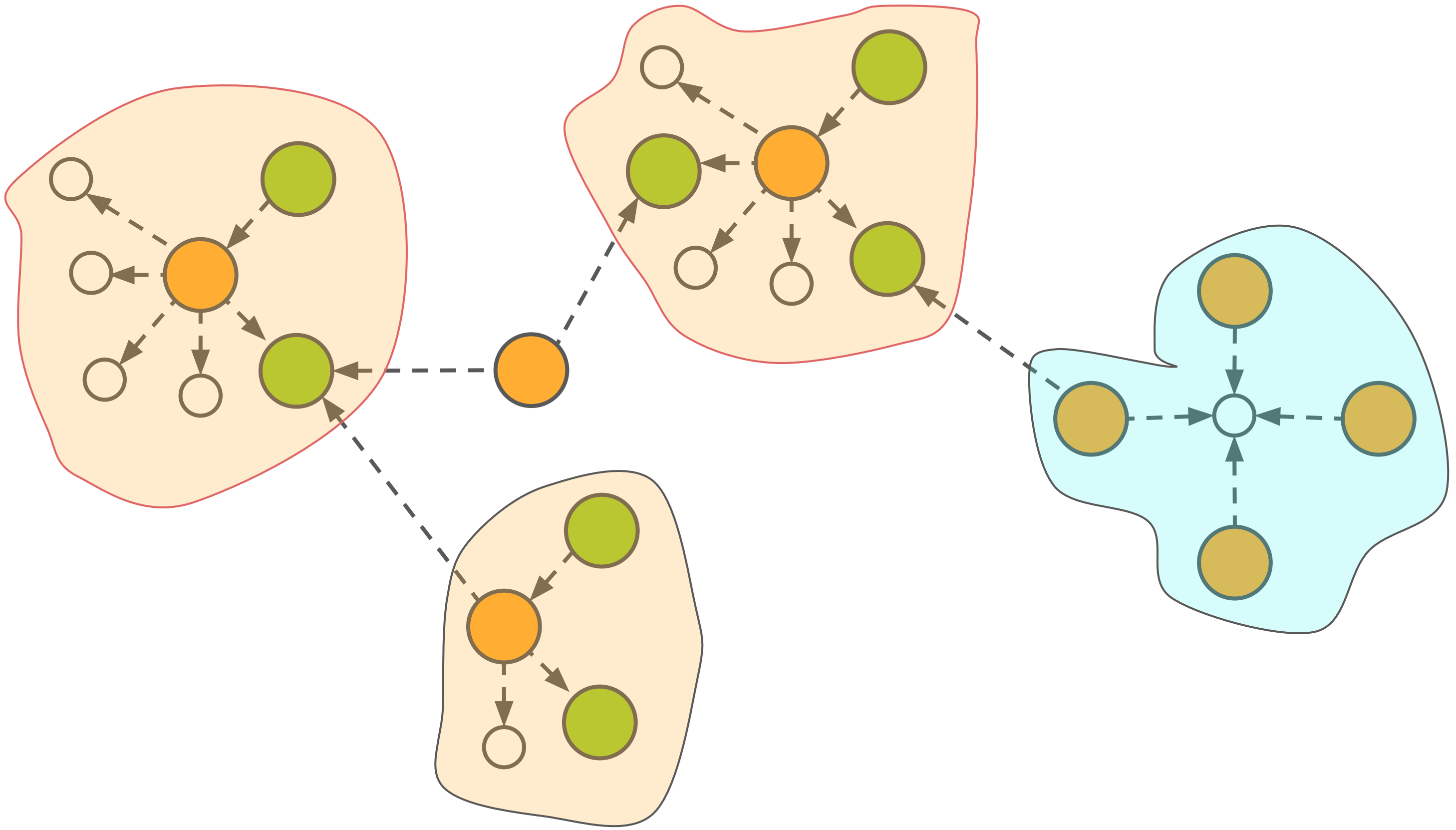
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY



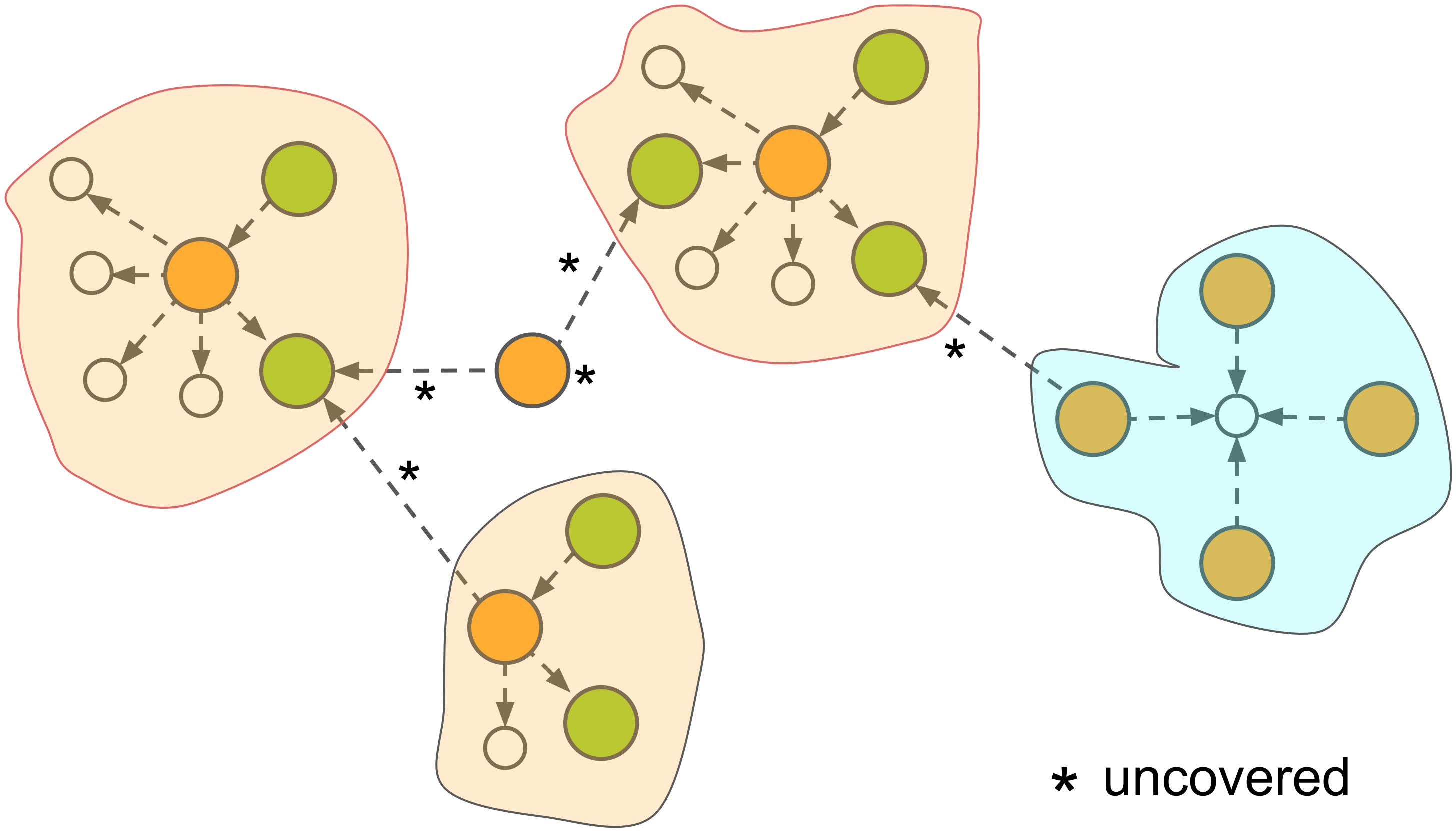
Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY



Estimation Algorithm

ESTIMATE_QUERY_CARDINALITY



* uncovered nodes/edges

Evaluations

Systems :

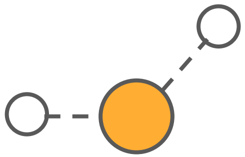
- RDF-3X with Characteristic sets estimator
- RDF-3X original
- Commercial system: DB A
- Commercial system: DB B
- Commercial system: DB C
- Stocker et al. (Stocker)
- Maduko et al. (Maduko)

Datasets :

- Yago
- LibraryThing

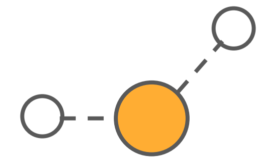
Single Join queries

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



Single Join queries

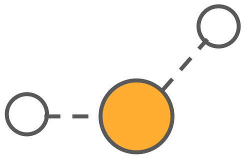
- q-error = max(c^/c , c/c^), bucketed
- queries of the form : { (?s p1 ?a) . (?s p2 ?b) }
- YAGO : 1751 queries, LibraryThing : 19,062,990



	Yago						
q-error	DB A	DB B	DB C	Stocker	Madoku	RDF-3X	CS
≤ 2	16.6	23.4	25.6	0	100[1]	14.9	99.9
≤ 5	12.2	16.0	16.4	0	0	20.7	0.1
≤ 10	7.6	10.2	10.0	2.2	0	16.0	0
≤ 100	40.6	21.4	21.7	1.1	0	38.5	0
≤ 1000	19.7	14.4	14.0	3.2	0	8.9	0
> 1000	3.3	14.6	12.2	93.5	0	0.9	0
max	314275	1731400	783276	$3.8 * 10^{14}$	1[1]	7779527	2.97

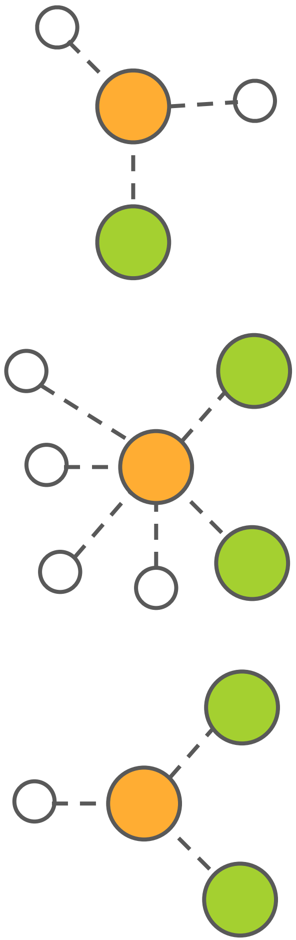
Single Join queries

- 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



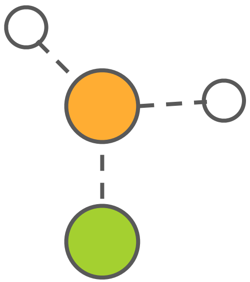
	LibraryThing						
q-error	DB A	DB B	DB C	Stocker	Maduko	RDF-3X	CS
≤ 2	15.0	23.2	22.9	0	26.7	30.2	100
≤ 5	10.5	27.3	27.8	0	40.6	30.9	0
≤ 10	10.3	17.2	17.7	0	19.7	16.6	0
≤ 100	35.3	28.8	27.8	0.1	12.0	19.9	0
≤ 1000	20.7	3.1	3.3	0	0.8	2.2	0
> 1000	8.1	0.4	0.6	99.9	0.1	0.2	0
max	28367552	1416363	7140611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

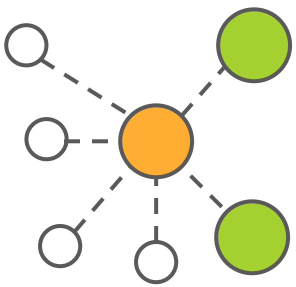
Complex Join queries

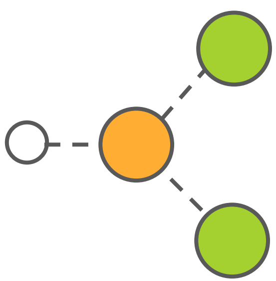


- **Upto 6 joins, with object constraints.**

Complex Join queries







- **Upto 6 joins, with object constraints.**

LibraryThing

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

Yago

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

Other datasets

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

	≤ 2	≤ 5	≤ 10	≤ 100	≤ 1000	> 1000
full CS	99.2	0.4	0.1	0.3	0.0	0.0
merged CS	91.7	1.7	0.9	1.3	0.2	4.1

end.