Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation

Giorgio Stefanoni
Bloomberg L.P., London, UK

Boris Motik
University of Oxford, UK

Egor V. Kostylev
University of Oxford, UK

ABSTRACT

Estimating the cardinality (i.e., the number of answers) of conjunctive queries is particularly difficult in RDF systems: queries over RDF data are navigational and thus tend to involve many joins. We present a new, principled cardinality estimation technique based on graph summarisation. We interpret a summary of an RDF graph using a possible world semantics and formalise the estimation problem as computing the expected cardinality over all RDF graphs represented by the summary, and we present a closed-form formula for computing the expectation of arbitrary queries. We also discuss approaches to RDF graph summarisation. Finally, we show empirically that our cardinality technique is more accurate and more consistent, often by orders of magnitude, than the state of the art.

ACM Reference Format:
Giorgio Stefanoni, Boris Motik, and Egor V. Kostylev. 2018. Estimating the Cardinality of Conjunctive Queries over RDF Data Using Graph Summarisation. In *Proceedings of The 2018 Web Conference (WWW 2018).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186003

1 INTRODUCTION

The Resource Description Framework (RDF) [23] data model is often used in applications where developing a rigid schema is either infeasible or undesirable. An RDF graph is a set of triples of the form $\langle s, p, o \rangle$, where s, p, and o are objects called *resources*; by viewing triples as directed labelled edges, an RDF graph corresponds to an ordinary directed labelled graph. SPARQL [18] is the standard RDF query language. *Conjunctive queries* (also known as *basic graph patterns*) are the basic type of SPARQL queries, and estimating their cardinality (i.e., the number of answers) is an important problem. For example, estimates are used to determine the cost of query plans, so estimation accuracy directly influences plan quality [26]. Thus, cardinality estimators are key components of most RDF systems.

Cardinality estimation has a long tradition in databases. It is usually solved by summarising a database to different kinds of *synopses* that can be used to accurately estimate the cardinality of certain queries. One-dimensional synopses, such as one-dimensional histograms [20, 33] and wavelets [28], can summarise one attribute of one relation, so they can be used on queries involving one selection on a single relation. Multidimensional synopses, such as multidimensional histograms [1, 6, 16, 34], multidimensional wavelets [8, 13], discrete cosine transforms [24], and kernel methods [15], can summarise several attributes of one relation, so they can be used on queries involving several selections on a single relation. Schema-level synopses, such as graphical models [14, 40], sampling [2, 41], TuG synopses [36], and statistical views [5, 11, 38] can summarise several attributes over different relations, but they often require a *join schema* that identifies the supported joins.

Queries whose cardinality cannot be estimated using the available synopses are typically broken into subqueries that can be estimated, and partial estimates are combined using ad hoc assumptions [4, 12]: the *independence assumption* assumes that each selection or join affects the query answers independently, the *preservation assumption* assumes that each attribute value of a relation participating in a join is present in the join result, and the *containment assumption* assumes that, for each pair of joined attributes, all values of one attribute are contained in the other attribute. However, these assumptions usually do not hold in practice and thus introduce estimation errors that compound exponentially with the number of joins [21]. Due to the graph-like nature of RDF, queries in RDF often navigate over paths and thus contain many (self-)joins (ten or more are not rare). Techniques that aim to address these specifics of RDF either just adapt relational approaches [19, 31, 37] or focus on particular types of queries (e.g., star or chain queries) [30] and fall back to the ad hoc assumptions for other types of queries. Hence, as we show in Section 6, accurately estimating the cardinality of complex RDF queries remains challenging.

To address these issues, in Section 3 we propose a new, principled technique for estimating the cardinality of conjunctive queries over RDF data. Our technique is based on graph summarisation—the process of compressing a graph by merging its vertices. Summaries have already been used for graph exploration [29, 39], fast approximate graph analytics [25, 35], and query processing [7]. In Section 3, we introduce a specific kind of summary that we use as a schema-level synopsis for RDF data. Following LeFevre and Terzi [25], we interpret a summary using a *possible world semantics* as a family of RDF graphs represented by the summary. We formalise the cardinality estimation problem as computing the expectation of the query cardinality across this family, and so our approach does not require any ad hoc assumptions. Finally, we show how to determine the probability that the estimation error exceeds a given bound. To the best of our knowledge, this is the first approach that can provide such guarantees for arbitrary queries.

In Section 4 we present formulas for computing the expected cardinality and its variance. Our formulas can handle queries of the form `SELECT * WHERE { BGP }` for `BGP` a basic graph pattern, which form the core of SPARQL. Note that, without `DISTINCT`, projecting variables in the `SELECT` clause does not affect the query cardinality.

To apply our framework in practice, effective graph summarisation algorithms are needed. Any summarisation algorithm can be used with our technique in principle, but some will fit our framework better than others: we intuitively expect that a good summary

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186003

should merge vertices participating in similar connections. Hence, algorithms for unlabelled graphs [25, 29, 35] are unlikely to work well on RDF, where links are inherently labelled. We experimented with an adaptation of the SNAP [39] technique to RDF, but it produced very large summaries. A recently proposed RDF-specific summarisation technique seems very promising, but its computational complexity seems very high [7]. Thus, in Section 5 we present a new graph summarisation approach. In the first step, we use a simple notion of similarity based on the labels on the edges that resources occur in. If the resulting summary is not sufficiently small, in the second step we further merge resources based on the similarity of the connections resources participate in.

In Section 6 we present the results of an extensive evaluation on LUBM, WatDiv, and DBLP benchmarks with 55 M to 109 M triples, and 15 to 103 queries. We compared the accuracy of our technique against RDF-3X [31] and the *characteristic sets technique* [30] (both specifically targeting RDF data), PostgreSQL (which uses one-dimensional histograms), and SystemX (a prominent commercial RDBMS that combines histograms with dynamic sampling); we considered both triple table and vertical partitioning for storing RDF in RDBMSs. We show that our technique is generally more accurate than the related approaches, often by orders of magnitude.

The proofs of all of our technical results are given in the extended version of the paper at http://arxiv.org/abs/1801.09619.

2 DEFINITIONS AND NOTATION

We now recall the relevant definitions and notation. RDF vocabulary consist of *resources*, which are URIs (i.e., abstract entities) or *literals* (i.e., concrete values described by datatypes such as *xsd*:*string* or *xsd*:*integer*). A *term* is a resource or a variable. An *(RDF) atom* is a triple of the form $\langle s, p, o\rangle$, where s, p, and o are terms called the *subject*, *predicate*, and *object*, respectively. An *(RDF) triple* is a variable-free atom. An *(RDF) graph* is a finite set of triples.

A *substitution* π is a mapping of variables to terms; the domain and range of π are $\mathsf{dom}(\pi)$ and $\mathsf{rng}(\pi)$, respectively; $\mathsf{varrng}(\pi)$ is the set of all variables in $\mathsf{rng}(\pi)$; and π is *empty* if $\mathsf{dom}(\pi) = \emptyset$. Let Z be a term, an atom, or a set of atoms. Then, $\pi(Z)$ is obtained from Z by replacing each occurrence of $x \in \mathsf{dom}(\pi)$ in Z with $\pi(x)$; if Z is a set of atoms, then $\pi(Z)$ is also a set and does not contain duplicates. Moreover, $\mathsf{res}(Z)$, $\mathsf{var}(Z)$, and $\mathsf{term}(Z)$ are the sets of resources, variables, and terms occurring in Z, respectively.

SPARQL [18] is a standard query language for RDF. Its syntax is very verbose, so we use a more compact notation that captures basic SPARQL queries of the form `SELECT * WHERE { BGP }`. SPARQL variables are written as ?x, but we drop the question mark and reserve (possibly indexed) letters x, y, and z for variables. A *conjunctive query* (*CQ*) q is a finite set of atoms. A substitution π is an *answer* to q on an RDF graph G if $\mathsf{dom}(\pi) = \mathsf{var}(q)$ and $\pi(q) \subseteq G$; moreover, $[\![q]\!]_G$ is the set of all answers to q on G, and the *cardinality* of q on G is the size of $[\![q]\!]_G$. For technical reasons, we allow q to be empty, in which case $[\![q]\!]_G$ contains just one empty answer. These definitions are compatible with the ones by Pérez et al. [32]; however, since we consider only CQs without variable projection, we simplify the notation and treat the query as a finite set of atoms.

The q-error is often used to measure precision of cardinality estimates [30]: if a query returns N answers and its cardinality is estimated as E, the *q-error* of the estimate is $e = \max(N'/E', E'/N')$, where $N' = \max(N, 1)$ and $E' = \max(E, 1)$. Intuitively, the q-error shows the difference in the orders of magnitude between a real cardinality N and its estimation E, regardless of whether E over- or undershoots N. We use N' and E' instead of N and E in the definition of q-error in order to avoid division by zero, as well as to prevent artificially high q-error values when $E < 1$.

Let $n \geq m > 0$ be integers. Then, $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ is the *binomial coefficient*, $(n)_m = n(n-1)\dots(n-m+1)$ is the *falling factorial*, and $[m, n]$ is the set of all integers between m and n (inclusive).

3 CARDINALITY ESTIMATION FRAMEWORK

To estimate the cardinality of a query without ad hoc assumptions, we must 'compress' our input RDF graph while preserving as much of its structure as possible. Graph summarisation has already been used to this end in related settings [7, 25, 29, 35, 39], and in our work we use it to estimate CQ cardinality in a principled way.

Consider the input graph G in Figure 1 where employees are connected to their cars and managers, and resources are assigned to classes (using the *rdf*:*type* predicate) as shown in the legend. A summary S of G is obtained by merging the resources of G into *buckets* as shown using dotted boxes; for example, employees e_1 and e_2 are replaced by bucket b_1, and e_3 and e_4 by another bucket b_3. In contrast to the existing approaches, in our approach each edge in S is labelled with the number of edges of G that collapse due to merging. For example, the *manages* edge from e_1 to e_2 collapses into a self-loop on b_1 of weight 1, and the edges from e_1 to e_3 and from e_2 to e_4 collapse into an edge from b_1 to b_3 of weight 2.

Our objective is to estimate the cardinality of a CQ q on G using S, rather than G. To this end, we interpret S using the 'possible worlds' semantics [25]: we assume that S represents with equal probability each graph G' that summarises as S—that is, if we replace the resources of G' with buckets in the same way as when we constructed S from G, we obtain S. Thus, S represents the input graph G, but, due to the loss of information in summarisation, it represents other graphs as well. In Figure 1, in addition to G, summary S also represents graphs G_1 and G_2. We then estimate the cardinality of q on G as the expected cardinality of q over all graphs represented by S. Instead of the assumptions mentioned in Section 1, we thus use a consistent semantic interpretation of S.

We next formalise our notion of a summary. We do not talk of a 'summary of an input graph G': a summary is a synopsis that represents a family of graphs.

Definition 3.1. A summary $S = \langle H, w, \mu\rangle$ *is a triple where* H *is an RDF graph called the* summarisation graph, $w : H \to \mathbb{N}$ *is a* weight function *assigning a positive natural number to each triple in* H*, and* μ *is a surjective* summarisation function *from a finite set of resources* $\mathsf{dom}(\mu)$ *to* $\mathsf{res}(H)$*. The resources of* $\mathsf{res}(H)$ *are called* buckets.

For Z *a term, an atom, or a set of atoms,* $\mu(Z)$ *is obtained from* Z *by replacing each resource* $d \in \mathsf{res}(Z) \cap \mathsf{dom}(\mu)$ *with* $\mu(d)$*; if* Z *is a set, then* $\mu(Z)$ *is a set as well (i.e., duplicates are removed). The* S-size *of a bucket* $b \in \mathsf{res}(H)$ *is defined as* $s[b] = |\{d \in \mathsf{dom}(\mu) \mid \mu(d) = b\}|$*, and the* S-size *of a triple* $\langle b_1, b_2, b_3\rangle \in \mathsf{res}(H) \times \mathsf{res}(H) \times \mathsf{res}(H)$ *is defined as* $s[\langle b_1, b_2, b_3\rangle] = s[b_1] \times s[b_2] \times s[b_3]$.

While buckets in histograms record value ranges, in our approach μ explicitly associates buckets with resources; this is needed

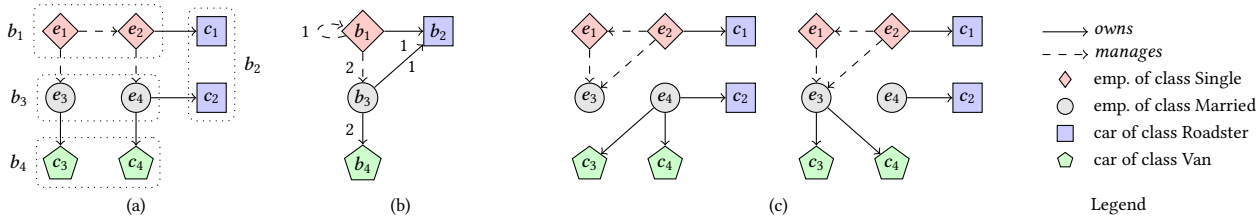

Figure 1: An example RDF graph G (a), its summary S (b), and example graphs G_1 and G_2 that also summarise as S (c)

as there is no natural notion of a range of RDF resources. In Figure 1, we have $\mu(e_1) = \mu(e_2) = b_1$, $\mu(c_1) = \mu(c_2) = b_2$, $\mu(e_3) = \mu(e_4) = b_3$, $\mu(c_3) = \mu(c_4) = b_4$, and μ is identity on all remaining resources, all of which occur in G as a predicate or the object of a triple whose predicate is *rdf:type* (such as *owns*, *rdf:type*, or *Van*).

The size of a bucket b is the number of resources mapped to b; in our example, $s[b_i] = 2$ for $i \in [1, 4]$, but $s[rdf{:}type] = s[Van] = 1$. For $f \in G$ and $h \in H$ where $\mu(f) = h$, we say that f is *summarised* as h, and that h can be *expanded into* f. Clearly, h can be expanded into at most $s[h]$ triples. In our example, we have

$$s[\langle b_1, manages, b_1\rangle] = s[b_1] \times s[manages] \times s[b_1] \;\; = 4,$$
$$s[\langle b_4, rdf{:}type, Van\rangle] = s[b_4] \times s[rdf{:}type] \times s[Van] = 2.$$

Definition 3.2 formalises the 'possible worlds' semantics of S.

Definition 3.2. A summary $S = \langle H, w, \mu\rangle$ represents *a graph G if (i) $\mathrm{res}(G) \subseteq \mathrm{dom}(\mu)$, (ii) $H = \mu(G)$, and (iii) for each $h \in H$, it is the case that $w[h] = |\{f \in G \mid \mu(f) = h\}|$. Moreover, R_S is the set of all graphs that S represents. Finally, summary S is* consistent *if $R_S \neq \emptyset$.*

At most $s[h]$ triples can be summarised as $h \in H$, so $w[h] \leq s[h]$ must hold for S to be consistent. Also, a graph in R_S is obtained by expanding each $h \in H$ arbitrarily into $w[h]$ triples if $w[h] \leq s[h]$ holds. Thus, we can check the consistency of S as in Proposition 3.3.

Proposition 3.3. *A summary $S = \langle H, w, \mu\rangle$ is consistent if and only if $w[h] \leq s[h]$ holds for each $h \in H$.*

We now estimate the cardinality of a CQ q as the expected cardinality of q over all graphs represented by S. We also introduce the related, natural notion of cardinality variance.

Definition 3.4. The expectation $E_{q,S}$ *and the* variance $\sigma^2_{q,S}$ *of the cardinality of a CQ q on a summary S are defined as*

$$E_{q,S} = \sum_{G \in R_S} \frac{|[\![q]\!]_G|}{|R_S|} \quad \text{and} \quad \sigma^2_{q,S} = \sum_{G \in R_S} \frac{(|[\![q]\!]_G| - E_{q,S})^2}{|R_S|}.$$

Please note that $E_{q,S}$ is not necessarily a natural number. For example, if q is a variable-free query, then $|[\![q]\!]_G|$ is equal to the number of graphs in R_S that contain q; since $|[\![q]\!]_G| \leq |R_S|$ is always satisfied, $E_{q,S}$ is a rational number satisfying $0 \leq E_{q,S} \leq 1$.

Using $E_{q,S}$ and $\sigma^2_{q,S}$, we can bound from above the probability $\mathsf{P}(e \geq \varepsilon)$ that the q-error of the estimate exceeds a given amount ε as shown in Theorem 3.5. Intuitively, for a graph G' chosen from R_S uniformly at random, $\mathsf{P}(e \geq \varepsilon)$ is the probability that $E_{q,S}$ over- or underestimates $|[\![q]\!]_{G'}|$ by a factor larger than ε; we can equivalently express this by saying that $\mathsf{P}(e \geq \varepsilon) \cdot |R_S|$ is the number of such graphs G'. We show in Section 6.6 that $\mathsf{P}(e \geq \varepsilon)$ is often small even for ε as low as 10, so in such cases there is good chance that the cardinality of q on our input graph is close to $E_{q,S}$.

Theorem 3.5. *Let q be a CQ and let S be a summary such that $E_{q,S} \geq 1$. Then, for each $\varepsilon > 1$ and for e the q-error of estimating the cardinality of q as $E_{q,S}$, we have*

$$\mathsf{P}(e \geq \varepsilon) \leq \left(\frac{\varepsilon \cdot \sigma_{q,S}}{(\varepsilon - 1) \cdot E_{q,S}}\right)^2.$$

If $\varepsilon \geq E_{q,S}$ holds additionally, then the numerator $\varepsilon \cdot \sigma_{q,S}$ on the right-hand side of the inequality can be replaced with $\sigma_{q,S}$.

4 COMPUTING THE EXPECTATION

Given a CQ q and a summary $S = \langle H, w, \mu\rangle$, we can compute $E_{q,S}$ by iterating over all graphs in R_S, but this is impractical since the number of such graphs is exponential in $|H|$. In this section, we present formulas that compute $E_{q,S}$ in time polynomial in $|H|$.

4.1 Intuition

Our formulas are quite complex, so we first discuss the intuitions using the summary S for Figure 1 and several example queries.

We first explain how to compute $E_{q,S}$ for variable-free queries, which is the basic element of our approach. Let $q_1 = \{f_1, f_2\}$ where $f_1 = \langle e_1, manages, e_3\rangle$ and $f_2 = \langle e_3, owns, c_3\rangle$. Since q_1 is variable-free, for each $G' \in R_S$, the cardinality $[\![q_1]\!]_{G'}$ of q_1 on G' is one if $q_1 \subseteq G'$, and it is zero otherwise; thus, the numerator of $E_{q_1,S}$ is equal to the number of graphs in R_S that contain all atoms of q_1 (i.e., that contain q_1 as a subset). Triples f_1 and f_2 of the query are summarised as $h_1 = \langle b_1, manages, b_3\rangle$ and $h_2 = \langle b_3, owns, b_4\rangle$; for readability, let $w_i = w[h_i]$ and $s_i = s[h_i]$ for $i \in \{1, 2\}$. Now every expansion of h_1 is determined by a choice of w_1 triples from s_1 possibilities, so there are a total of $\binom{s_1}{w_1}$ expansions of h_1. Moreover, each expansion of h_1 that contains f_1 is obtained by choosing f_1 and the remaining $w_1 - 1$ triples from $s_1 - 1$ possibilities; thus, a total of $\binom{s_1-1}{w_1-1}$ expansions of h_1 contain f_1. Analogously, h_2 has $\binom{s_2}{w_2}$ expansions, of which $\binom{s_2-1}{w_2-1}$ contain f_2. Now a key observation is that f_1 and f_2 are summarised as distinct triples; consequently, the expansions of h_1 and h_2 are independent, and so the total number of expansions of h_1 and h_2, and the number of expansions containing both f_1 and f_2, are given by the product of the above factors. Finally, the expansions of each triple $h \in H \setminus \{h_1, h_2\}$ are independent from and thus irrelevant to q_1, so they do not affect $E_{q_1,S}$. Hence, we get

$$E_{q_1,S} = \frac{\binom{s_1-1}{w_1-1}}{\binom{s_1}{w_1}} \cdot \frac{\binom{s_2-1}{w_2-1}}{\binom{s_2}{w_2}} = \frac{w_1}{s_1} \cdot \frac{w_2}{s_2} = \frac{2}{2 \cdot 2} \cdot \frac{2}{2 \cdot 2} = 0.25.$$

Next, we demonstrate how to handle queries with variables. Let $q_2 = \{\langle x, \mathit{manages}, y\rangle, \langle y, \mathit{owns}, z\rangle\}$. For each $G' \in R_S$ and each $\pi \in [\![q_2]\!]_{G'}$, query $\pi(q_2)$ does not contain variables, so we can compute $E_{\pi(q_2),S}$ just as for q_1. We can thus compute $E_{q_2,S}$ by summing the contribution of each such π, but enumerating all π would be inefficient. However, for each $\pi \in [\![q_2]\!]_{G'}$, there exists $\tau \in [\![\mu(q_2)]\!]_H$ such that $\mu(\pi(x)) = \tau(x)$ for all $x \in \mathrm{dom}(\pi)$; we say that τ *expands into* π. Thus, for each τ, we can compute $E_{\pi(q_2),S}$ for just one prototypical substitution π that τ expands into, and multiply $E_{\pi(q_2),S}$ by the number of the expansions of τ. On our example, evaluating $\mu(q_2)$ on H produces substitutions $\tau_1 = \{x \mapsto b_1, y \mapsto b_3, z \mapsto b_4\}$, $\tau_2 = \{x \mapsto b_1, y \mapsto b_1, z \mapsto b_2\}$, $\tau_3 = \{x \mapsto b_1, y \mapsto b_3, z \mapsto b_2\}$. For τ_1, we can select, say, $\pi = \{x \mapsto e_1, y \mapsto e_3, z \mapsto c_3\}$ as a prototypical expansion of τ_1; then $\pi(q_2) = q_1$, and so $E_{\pi(q_2),S} = 0.25$. Moreover, since τ_1 maps both atoms of q_2 to distinct triples in H, each expansion π of τ_1 does the same; thus, we can obtain all such π by expanding each variable in $\mathrm{dom}(\tau_1)$ independently, and so there are $s[\tau_1(x)] \cdot s[\tau_1(y)] \cdot s[\tau_1(z)] = 2 \cdot 2 \cdot 2 = 8$ expansions of τ_1. Thus, τ_1 contributes to $E_{q_2,S}$ by $8 \cdot 0.25 = 2$. We compute the contributions of τ_2 and τ_3 analogously as $2^3 \cdot \frac{1\cdot1}{4\cdot4} = 0.5$ and $2^3 \cdot \frac{2\cdot1}{4\cdot4} = 1$. By summing all contributions, we get $E_{q_2,S} = 2 + 0.5 + 1 = 3.5$.

Generalising to any CQ q, formula (1) sums the contribution of each $\tau \in [\![\mu(q)]\!]_H$: the first factor counts the expansions of τ to a prototypical π, and the second one counts the contribution of π.

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \prod_{x \in \mathrm{var}(q)} s[\tau(x)] \cdot \prod_{a \in q} \frac{w[\tau(\mu(a))]}{s[\tau(\mu(a))]} \qquad (1)$$

However, formula (1) is correct only if each answer to $\mu(q)$ in H maps all atoms of $\mu(q)$ to distinct triples in H. Consider a query $q_3 = \{\langle e_3, \mathit{owns}, x\rangle, \langle e_3, \mathit{owns}, y\rangle\}$, so $[\![\mu(q_3)]\!]_H$ contains answers

$$\tau_1 = \{x \mapsto b_2, y \mapsto b_4\}, \quad \tau_2 = \{x \mapsto b_4, y \mapsto b_2\},$$
$$\tau_3 = \{x \mapsto b_4, y \mapsto b_4\}, \quad \tau_4 = \{x \mapsto b_2, y \mapsto b_2\}.$$

Answers τ_1 and τ_2 map the atoms of $\mu(q_3)$ to distinct triples in H, so their contributions are given by (1) as $2 \cdot 2 \cdot \frac{2}{2\cdot2} \cdot \frac{1}{2\cdot2} = 0.5$. In contrast, τ_3 maps both atoms of $\mu(q_3)$ to $h_3 = \langle b_3, \mathit{owns}, b_4\rangle$, so τ_3 expands into two kinds of π. First, π can map both atoms of q_3 to the same triple (e.g., $\langle e_3, \mathit{owns}, c_4\rangle \in G_2$). Then $\pi(q_3)$ contains just one atom, so $E_{\pi(q_3),S} = w[h_3]/s[h_3] = 2/4 = 0.5$; also, x and y are mapped to the same value in each such π, so the number of such π is $N_1 = s[\tau_3(x)] = 2$. Second, π can map the atoms of q_3 to distinct triples. Then, each expansion of h_3 containing $\pi(q_3)$ corresponds to a choice of $w[h_3] - 2$ triples out of $s[h_3] - 2$ possibilities, so

$$E_{\pi(q_3),S} = \frac{\binom{s[h_3]-2}{w[h_3]-2}}{\binom{s[h_3]}{w[h_3]}} = \frac{(w[h_3])_2}{(s[h_3])_2} = \frac{(2)_2}{(4)_2} = \frac{2\cdot1}{4\cdot3} \approx 0.167.$$

Moreover, there are $N_2 = s[\tau_3(x)] \cdot s[\tau_3(y)] = 2 \cdot 2 = 4$ expansions of τ_3, and $N_1 = 2$ of these map x and y to the same resource; thus, $N_2 - N_1 = 2$ expansions of τ_3 map x and y to distinct resources; and so τ_3 contributes to $E_{q_3,S}$ by $N_1 \cdot 0.5 + (N_2 - N_1) \cdot 0.167 = 1.33$. Finally, answer τ_4 maps both atoms of $\mu(q_3)$ to the same triple $h_4 = \langle b_3, \mathit{owns}, b_2\rangle$, but no expansion of τ_4 contains two triples since $w[h_4] = 1$; thus, we consider only $s[\tau_4(x)] = 2$ expansions π of τ_4, each mapping both atoms of q_3 to the same triple and thus contributing by $E_{\pi(q_3),S} = 1/2 = 0.5$; hence, τ_4 contributes to $E_{q_3,S}$ by $2 \cdot 0.5 = 1$. Finally, we get $E_{q_3,S} \approx 0.5 + 0.5 + 1.33 + 1 = 3.33$.

4.2 Formalisation

For this section, we fix a consistent summary $S = \langle H, w, \mu\rangle$, a CQ q with $\mathrm{res}(q) \subseteq \mathrm{dom}(\mu)$, and an arbitrary total order $\preceq$ on terms where $d \preceq x$ holds for each resource d and each variable x.

We first introduce a notion of a partition of q. Intuitively, a partition groups the atoms of q into disjoint sets, and it describes the 'type' of answer to q on graphs in R_S where all atoms from each such group are mapped to the same triple.

Definition 4.1. A partition *of q is a set P of mutually disjoint nonempty subsets u_i of q such that $q = \bigcup_{u_i \in P} u_i$.*

The term graph $\mathcal{G}_P$ *for P contains all terms from* $\mathrm{term}(q)$ *as vertices, and undirected edges between terms s_1 and s_2, p_1 and p_2, and o_1 and o_2 for each $u \in P$ and all atoms $\langle s_1, p_1, o_1\rangle$ and $\langle s_2, p_2, o_2\rangle$ in u. Partition P is* unifiable *if no two distinct resources from* $\mathrm{res}(q)$ *are reachable in $\mathcal{G}_P$, in which case substitution γ_P maps each variable $x \in \mathrm{var}(q)$ to the $\preceq$-least term reachable from x in $\mathcal{G}_P$.*

The partition base *B for q is the set of all unifiable partitions of q. The partial order $\preceq$ on B is defined so that, for $P, P' \in B$, relationship $P \preceq P'$ holds iff, for each $u \in P$, there exists $u' \in P'$ with $u \subseteq u'$.*

For $P, P' \in B$ with $P \preceq P'$, a chain from P to P' of length $\ell \geq 0$ *is a sequence $P = P_0, \ldots, P_\ell = P'$ of partitions from B where $P_{i-1} \prec P_i$ holds for $i \in [1, \ell]$; moreover, $C^e(P, P')$ and $C^o(P, P')$ are the numbers of chains from P to P' of even and odd length, respectively; finally, $K(P, P') = C^e(P, P') - C^o(P, P')$.*

Note that $\emptyset$ is the only unifying partition of $q = \emptyset$. Moreover, $C^e(P, P) = 1$ and $C^o(P, P) = 0$ for each $P \in B$, so $K(P, P) = 1$.

Let $q_3 = \{a_1, a_2\}$ for $a_1 = \langle e_3, \mathit{owns}, x\rangle$ and $a_2 = \langle e_3, \mathit{owns}, y\rangle$ be as in Section 4.1. Then, $P_1 = \{\{a_1\}, \{a_2\}\}$ and $P_2 = \{\{a_1, a_2\}\}$ are the unifiable partitions of q_3: partition P_1 represents the answers to q_3 on graphs in R_S that map a_1 and a_2 to distinct triples, and P_2 represents the answers that map a_1 and a_2 to the same triple; the latter is captured by $\gamma_{P_2} = \{y \mapsto x\}$ (assuming $x \preceq y$). Also, $P_1 \prec P_2$ and $K(P_1, P_2) = -1$ show that the answers to q_3 satisfying P_2 are exactly the answers to q_3 minus the answers to q_3 that satisfy P_1.

For an answer τ to $\mu(q)$ on H and an appropriate partition P, we define coefficients $N_\tau(P)$ and $F_\tau(P)$ as follows.

Definition 4.2. Let τ be an answer to $\mu(q)$ on H. A partition $P \in B$ is satisfied *by τ if, for each $x \in \mathrm{var}(q)$, (i) $\gamma_P(x) \in \mathrm{var}(q)$ implies $\tau(x) = \tau(\gamma_P(x))$, and (ii) $\gamma_P(x) \in \mathrm{res}(q)$ implies $\tau(x) = \mu(\gamma_P(x))$. Set B_τ contains all partitions of B satisfied by τ. For $P \in B_\tau$, let*

$$N_\tau(P) = \prod_{x \in \mathrm{varrng}(\gamma_P)} s[\tau(x)] \qquad \text{and}$$

$$F_\tau(P) = \begin{cases} \prod_{h \in \tau(\mu(q))} \frac{(w[h])_{\#_h}}{(s[h])_{\#_h}} & \text{if } \#_h \leq w[h]\ \forall h \in \tau(\mu(q)) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where } \#_h = |\{u \in P \mid \tau(\mu(u)) = \{h\}\}|.$$

In our example, $\gamma_{P_2}(x) = \gamma_{P_2}(y)$ and $\tau_1(x) \neq \tau_1(y)$ imply that P_2 is not satisfied by τ_1, which intuitively means that no expansion of τ_1 can match the atoms of q_3 as required by P_2. Moreover, $N_{\tau_3}(P_1) = s[\tau_3(x)] \cdot s[\tau_3(y)] = 4$ and $N_{\tau_3}(P_2) = s[\tau_3(x)] = 2$ count the expansions of τ_3 that comply with γ_{P_1} and γ_{P_2}, respectively; thus, the number of expansions of τ_3 complying with just P_1 is

$$K(P_1, P_1) \cdot N_{\tau_3}(P_1) + K(P_1, P_2) \cdot N_{\tau_3}(P_2) = 1 \cdot 4 - 1 \cdot 2 = 2.$$

The contribution of each expansion of τ_3 matching the atoms of q_3 to distinct triples is $F_{\tau_3}(P_1)$: set $\tau_3(\mu(u))$ contains just h_3 for each $u \in P_1$, so $\#_{h_3} = 2$, and $F_{\tau_3}(P_1) \approx 0.167$.

Thus, to compute $E_{q,S}$, we evaluate $\mu(q)$ in H and, for each answer τ, we combine $F_\tau(P)$ and $N_\tau(P')$ as in Theorem 4.3. Enumerating all τ corresponds to query answering and is thus polynomial in $|H|$, and the size of B_τ does not depend on H; hence $E_{q,S}$ can be computed in time polynomial in $|H|$. Moreover, Theorem 4.4 shows how to compute the variance of the cardinality of q.

Theorem 4.3. *The following identity holds:*

$$E_{q,S} = \sum_{\tau \in [\![\mu(q)]\!]_H} \sum_{P \in B_\tau} F_\tau(P) \cdot \sum_{P' \in B_\tau, P \preceq P'} K(P, P') \cdot N_\tau(P').$$

Theorem 4.4. *Identity $\sigma^2_{q,S} = E_{q \cup \rho(q),S} - E^2_{q,S}$ holds, where ρ is a substitution mapping each variable in* var(q) *to a fresh variable.*

We next introduce the notion of *μ-unification-free queries*, where no two distinct atoms can ever be mapped to the same triple. Proposition 4.6 shows that, for such queries, the simpler formula (1) correctly computes the expectation $E_{q,S}$.

Definition 4.5. Atoms a_1 and a_2 are unifiable *if a substitution κ exists such that $\kappa(a_1) = \kappa(a_2)$. Query q is* μ-unification-free *if no two distinct atoms of $\mu(q)$ are unifiable; otherwise, query q is* μ-unifiable.

Proposition 4.6. *Formula* (1) *correctly computes $E_{q,S}$ when q is a μ-unification-free query.*

Queries q_1 and q_2 from Section 4.1 are μ-unification-free, while q_3 is not. Moreover, query $q_4 = \{\langle e_3, owns, x\rangle, \langle e_4, owns, y\rangle\}$ is also *not* μ-unification-free: its atoms unify *after* applying μ to q_4. Note that queries used in practice often consist of atoms of the form $\langle t, r, t'\rangle$ and $\langle t, rdf{:}type, r\rangle$ where t and t' are terms and r is a distinct resource; then, such queries are unification-free whenever each such resource r is assigned to a distinct bucket.

If q is μ-unification-free, then $P = \{\{a_i\} \mid a_i \in q\}$ is the only unifiable partition of q, so the estimation formula simplifies to (1). In particular, P' in the last sum in Theorem 4.3 can in such cases only be P, so the summation reduces to $N_\tau(P)$—the first product of (1). Moreover, the atoms of $\mu(q)$ cannot be equated, so $\#_h = 1$ for each $h \in \tau(\mu(q))$; thus, $F_\tau(P)$ reduces to the second product of (1).

5 SUMMARISING RDF GRAPHS

As we have mentioned in Section 1, reusing existing graph summarisation techniques proved challenging: some produced summaries that did not lead to accurate cardinality estimates, and others could not be efficiently applied to large graphs. We thus developed a new summarisation approach consisting of two steps we describe next.

5.1 Typed Summary

To obtain a coarse measure of resource similarity, we assign to each resource d in an RDF graph G a *type* $\mathrm{t}_d = \langle C(d), O(d), I(d), P(d)\rangle$, which is a tuple consisting of four components described next.

The most important component of t_d is the *class type* $C(d)$ of d, which is the set defined as follows:

$$C(d) = \begin{cases} \{o \in \mathrm{res}(G) \mid \langle d, rdf{:}type, o\rangle \in G\} & \text{if } d \text{ is a URI,} \\ \{\text{the datatype of } d\} & \text{if } d \text{ is a literal.} \end{cases}$$

One can intuitively expect that a summary whose buckets contain, say, both people and cars, is unlikely to provide good cardinality estimates. Thus, our summarisation approaches will put resources d_1 and d_2 into the same bucket only if $C(d_1) = C(d_2)$ holds. A similar idea has been used in query answering over ontologies [9].

Sets $O(d)$ and $I(d)$ describe the outgoing and incoming connections of d. Let $\langle p_1, \ldots, p_k\rangle$ be an arbitrary ordering of the predicates in G different from *rdf*:*type*; then, we could capture the outgoing connections of d by a vector $\langle n_1, \ldots, n_k\rangle$, where each n_i is the number of the outgoing connections from d for predicate p_i. This, however, is likely to create too many types: resources in an RDF graph are unlikely to have exactly the same outgoing connections with the same cardinalities. We thus represent the cardinality information using histograms. For each predicate p_i, we compute the set out_{p_i} that, for each resource d_j occurring in G in subject position, contains a pair $\langle d_j, |\{o \in \mathrm{res}(G) \text{ such that } \langle d_j, p_i, o\rangle \in G\}|\,\rangle$ associating d_j with the number of resources that d_j is connected to via property p_i. We sort the pairs in out_{p_i} by the value of their second component into a list $\langle d_1, n_1\rangle, \ldots, \langle d_N, n_N\rangle$. Finally, given a predetermined number J_i of histogram buckets (not to be confused with summary buckets) for p_i, we create a histogram for out_{p_i}: for $w = \lceil N/J_i \rceil$, the first bucket contains resources $d_1, \ldots, d_w$ and has ID 1, the second bucket contains resources $d_{w+1}, \ldots, d_{2w}$ and has ID 2, and so on. Thus, each bucket contains resources with similar outgoing frequencies for p_i, and the difference between bucket IDs is indicative of the bucket similarity. Then, the *outgoing type* $O(d)$ of a resource d is the vector $\langle O_1, \ldots, O_k\rangle$, where each O_i is the ID of the bucket that d was assigned to in the histogram for p_i. The *incoming type* $I(d)$ of d is defined analogously. The numbers of buckets can be selected independently for each predicate and direction: it is reasonable to use a larger number of buckets if the relevant frequencies vary significantly across resources.

Finally, we also identify highly connected groups of resources: one can intuitively expect that such resources should be assigned to the same bucket. To this end, we divide the resources into a predetermined number of partitions of roughly the same size while minimising the number of edges between partitions using the algorithms by Karypis and Kumar [22]. The *partition type* $P(d)$ of a resource d is the partition to which d was assigned to.

Thus, if the types t_{d_1} and t_{d_2} of resources d_1 and d_2 are the same, then d_1 and d_2 occur in the same classes (or datatypes), have comparable outgoing and incoming frequencies for all predicates, and belong to the same highly connected component of the RDF graph; hence, we can expect d_1 and d_2 to be similar. Thus, we define the *typed summary* $S = \langle H, w, \mu\rangle$ of an RDF G as follows: for each resource $d \in \mathrm{res}(G)$, if d occurs in G in triples of the form $\langle s, d, o\rangle$ or $\langle s, rdf{:}type, d\rangle$, we let $\mu(d) = d$; otherwise, we let $\mu(d) = b_{\mathrm{t}_d}$, where b_{t_d} is a distinct bucket uniquely associated with the type t_d of d. Finally, we define $H = \mu(G)$ and adjust the function w accordingly.

5.2 Summary Refinement by MinHashing

If a typed summary is large, we compress it by merging similar buckets and types. To identify similar buckets, we define the *vicinity* $V_S(b)$ of a bucket $b \in \mathrm{res}(H)$ as the union of the following sets:

$$V^{out}_S(b) = \{\langle b_p, b_o\rangle \mid \langle b, b_p, b_o\rangle \in H \wedge b_p \neq \mu(rdf{:}type)\},$$
$$V^{in}_S(b) = \{\langle b_i, b_p\rangle \mid \langle b_i, b_p, b\rangle \in H \wedge b_p \neq \mu(rdf{:}type)\}.$$

Since vicinity $V_S(b)$ describes the connections of b, given buckets $b_1, b_2 \in \text{res}(H)$, the degree of commonality between $V_S(b_1)$ and $V_S(b_2)$ is indicative of the similarity of b_1 and b_2. This is captured by the *Jaccard index* [27] of b_1 and b_2, which is defined as

$$\mathcal{J}_S(b_1, b_2) = \frac{|V_S(b_1) \cap V_S(b_2)|}{|V_S(b_1) \cup V_S(b_2)|}$$

if $V_S(b_1) \cup V_S(b_2) \neq \emptyset$, and otherwise $\mathcal{J}_S(b_1, b_2) = 1$. Thus, we can naïvely reduce the size of S by repeatedly merging the pair of buckets b_1 and b_2 with maximal $\mathcal{J}_S(b_1, b_2)$, but this is impractical for two reasons. First, computing $\mathcal{J}_S(b_1, b_2)$ requires iterating over $V_S(b_1)$ and $V_S(b_2)$, which can be large. Second, the number of pairs of b_1 and b_2 can be large in even moderately sized summaries.

We address the first problems using MinHashing [27], which uses fixed-sized signatures to approximate $\mathcal{J}_S(b_1, b_2)$. Given two integer parameters m and n, we generate a *MinHash scheme* $\mathcal{F}$ *of size* $m \times n$, which is an $m \times n$ matrix of hash functions chosen uniformly at random with replacement. Each $\mathcal{F}[i, j]$ maps elements of $V_S(b)$ (i.e., pairs of resources) to natural numbers. The *signature* of a bucket $b \in \text{res}(H)$ on $\mathcal{F}$ and S is the $m \times n$ matrix $\mathcal{M}^b$ where

$$\mathcal{M}^b[i, j] = \min_{\alpha \in V_S(b)} \mathcal{F}[i, j](\alpha)$$

if $V_S(b) \neq \emptyset$, and otherwise $\mathcal{M}^b[i, j] = \infty$. It is known that

$$\hat{\mathcal{J}}_S(b_1, b_2) = \frac{|\{\langle i, j\rangle \mid \mathcal{M}^{b_1}[i, j] = \mathcal{M}^{b_2}[i, j]\}|}{m \cdot n}$$

is a good approximation of the Jaccard index of buckets b_1 and b_2 [27]. For $i \in [1, m]$, we write $\mathcal{M}^b[i]$ for the i-th row of $\mathcal{M}^b$.

We address the second problem using locality sensitive hashing [27]: we generate a hash function LSH for $\mathcal{M}^b[i]$ and use it to assign each bucket b to a bin $\mathit{Bins}[\text{LSH}(\mathcal{M}^b[i])]$. For each $\mathit{Bin} \in \mathit{Bins}$ and all buckets $b_1, b_2 \in \mathit{Bin}$, it is known that $\hat{\mathcal{J}}_S(b_1, b_2)$ is likely to be high, so all pairs of buckets in Bin are likely to be similar [27].

Algorithm 1 uses these ideas to compute a summary of a graph G. It takes as arguments a set of types T covering all resources of G. The algorithm first converts G into a trivial summary S (line 3) where $H = G$ and $\mu(d) = d$ for each $d \in \text{res}(G)$, and then it computes the sets $\mathcal{B}_\mathsf{t}$ of buckets of type t for each $\mathsf{t} \in \mathsf{T}$ (line 4). The algorithm next enters a loop (lines 5–21) that merges buckets of the same type. It returns S if the number of edges in H is in the required space budget (line 6); otherwise, it considers each type $\mathsf{t} \in \mathsf{T}$ (lines 8–16) and computes the signature of each bucket $b \in \mathcal{B}_\mathsf{t}$ (line 9). For each row i of the signature (lines 10–16), the algorithm applies locality sensitive hashing to each bucket $b \in \mathcal{B}_\mathsf{t}$ (line 12) and adds b to a bin. The number of bins is not predetermined, which reduces collisions. Finally, for all bins containing at least two buckets, all buckets in the bin are scheduled to be merged into one designated bucket b (lines 13–16). Locality sensitive hashing ensures that similar buckets are likely to end in the same bin, but this is not guaranteed so the standard algorithm would merge b' and b only if the estimate $\hat{\mathcal{J}}_S(b, b')$ is above a particular threshold. We skip this step for two reasons: the quadratic step of computing $\hat{\mathcal{J}}_S(b, b')$ for all pairs of buckets in a bin can be prohibitive, and the chance that dissimilar buckets are assigned to the same bin is small since we do not limit the number of bins. After all signature rows have been considered, the merges are applied to the summary (line 18). Finally, if the size of S has not been reduced sufficiently, then the types in T are too

Algorithm 1 MinHash summarisation algorithm

Inputs: G : an RDF graph
T : a set of types
$m \times n$: size of the MinHash scheme
target : target size of the summary

1: Generate a MinHash scheme $\mathcal{F}$ of size $m \times n$
2: Generate a locality sensitive hash LSH : $\mathbb{N}^n \to \mathbb{N}$
3: Let $S := \langle H, w, \mu\rangle$ be the trivial summary of G
4: **for all** $\mathsf{t} \in \mathsf{T}$ **do** $\mathcal{B}_\mathsf{t} := \{b \in \text{res}(H) \mid b \text{ is of type } \mathsf{t}\}$
5: **loop**
6: **if** $|H| \leq$ *target* **then return** S
7: *queue* $:= \emptyset$
8: **for all** $\mathsf{t} \in \mathsf{T}$ **do**
9: **for all** $b \in \mathcal{B}_\mathsf{t}$ **do** Compute $\mathcal{M}^b$ using $\mathcal{F}$ and S
10: **for** $i := 1$ **to** m **do**
11: *Bins* $:= \emptyset$
12: **for all** $b \in \mathcal{B}_\mathsf{t}$ **do** Add b to $\mathit{Bins}[\text{LSH}(\mathcal{M}^b[i])]$
13: **for** $\mathit{Bin} \in \mathit{Bins}$ such that $|\mathit{Bin}| \geq 2$ **do**
14: Choose some $b \in \mathit{Bin}$
15: **for** $b' \in \mathit{Bin}$ with $b \neq b'$ **do**
16: Add $\langle b', b\rangle$ to *queue* and remove b' from $\mathcal{B}_\mathsf{t}$
17: *sizeBefore* $:= |H|$
18: **for all** $\langle b', b\rangle \in$ *queue* **do** Merge b' into b in S
19: **if** *sizeBefore* $- |H| \leq (|H| - \mathit{target}) \cdot 0.01$ **then**
20: Merge similar types in T
21: **if** no merge has happened **then return** S

specific to allow further merges, so the algorithm merges similar types in T (line 20) to allow further reducing in the size of S.

Let t and t' be types whose outgoing and incoming types are $\langle O_1, \ldots, O_k\rangle$ and $\langle O'_1, \ldots, O'_k\rangle$, and $\langle I_1, \ldots, I_k\rangle$ and $\langle I'_1, \ldots, I'_k\rangle$. Then, t and t' can be similar only if their class and partition types coincide, and their similarity $\mathcal{J}(\mathsf{t}, \mathsf{t}')$ is the average of the generalised Jaccard indexes [27] of their outgoing and incoming types:

$$\mathcal{J}(\mathsf{t}, \mathsf{t}') = \frac{1}{2}\left(\sum_i \frac{\min(O_i, O'_i)}{\max(O_i, O'_i)} + \sum_i \frac{\min(I_i, I'_i)}{\max(I_i, I'_i)}\right).$$

Merging t and t' produces t'' with the class and partition types same as t and t', outgoing type $\langle (O_1 + O'_1)/2, \ldots, (O_k + O'_k)/2\rangle$, and an analogous incoming type; we also set $\mathcal{B}_{\mathsf{t}''} = \mathcal{B}_\mathsf{t} \cup \mathcal{B}_{\mathsf{t}'}$.

To merge the types in line 20, the algorithm first groups the types by their class and partition type. Ideally, we would merge the most similar pairs within each such group, but computing the similarity of each pair in the group would be prohibitively expensive. Instead, the algorithm selects 500 pairs at random, computes their similarity, and merges the most similar pair if their similarity is at least 50%. This process is repeated until the size of T is reduced by 20% (if possible), which then allows for further merging of buckets.

5.3 Choosing the Parameters

To recapitulate, our algorithm is parametrised by (i) the number of buckets for the incoming and outgoing histograms per predicate, (ii) the number of partitions, (iii) the target size of the summary,

and (iv) the size of the MinHash scheme $m \times n$ (if used). Standard techniques can be used to choose the number of buckets, such as the Freedman–Diaconis rule [10]. We found it reasonable to partition a graph into 10 partitions, and fall back to just one partition if more than 20% of the edges are cut. We also identified $m = 20$, $n = 2$, and target sizes in the order of tens of thousands as reasonable defaults.

6 EXPERIMENTAL EVALUATION

We implemented our techniques in a prototype system called SumRDF.[1] The system is written in Java. It evaluates queries over graphs stored in RAM using left-deep index nested loop joins. We evaluated our system with three distinct objectives. First, we compared the accuracy of the estimates produced by SumRDF and several state-of-the-art RDF and relational systems. Second, we investigated the impact of the correct handling of μ-unifiable queries; these correspond to self-joins and are difficult to estimate. Third, we evaluated the usefulness of the bounds provided by Theorem 3.5. We ran our experiments on a Dell computer with 96 GB of RAM, Fedora 22, and two Xeon X5667 processors with 16 physical cores.

6.1 Test Data

We used three well-known RDF benchmarks shown in Table 1. We group queries by their shape as linear, star, snowflake (joined stars), and complex (e.g., cyclic). For each group, we show the number of all queries, the number of μ-unifiable queries, and the minimal and maximal numbers of atoms in a query. Each query is assigned an ID. All datasets and queries are available on the SumRDF Web site.

LUBM [17] is a synthetic benchmark comprising a data generator, an OWL 2 DL ontology, and 14 test queries. The ontology must be taken into account for the queries to produce results, so we generated an RDF graph with 500 universities and extended it with triples implied by the ontology, thus obtaining 91.1 M triples. The 14 queries are relatively simple, and are all μ-unification-free, so we additionally handcrafted 24 queries, five of which are μ-unifiable.

WatDiv [3] is another synthetic benchmark designed to stress-test RDF systems. We used its data generator to obtain an RDF graph with 108.9 M triples. The benchmark also includes three queries and 17 query templates with one parameter, and a generator that replaces the parameter with resources from the RDF graph. We instantiated each query template into five concrete queries and, if the parameter was not in the class position, we obtained one more query by replacing the parameter with a fresh variable. We thus obtained 103 queries overall, all of which are μ-unification-free.

DBLP is a bibliography database that was converted into an RDF graph of 55.5 M triples. We used six queries already been considered in the literature [42], and we handcrafted nine additional queries. We thus obtained 15 queries, eight of which are μ-unifiable.

6.2 Constructing the Summaries

LUBM can be easily partitioned into ten partitions, and the numbers of connections do not vary much across resources so we used just one bucket per histogram. The typed summary for LUBM contained fewer than 15,000 edges, so the refinement step was not needed. In contrast, WatDiv and DBLP could not be effectively partitioned, so we used just one partition type, and we manually selected between one and 64 histogram buckets per predicate. The typed summaries were large so we further refined them using $m = 20$, $n = 2$, and *target* of 45,000 for WatDiv and 30,000 for DBLP. Table 1 shows that each graph was compressed by a factor more than 1,000.

6.3 Comparison Systems

We evaluated our system against a portfolio of the following four systems, which represent the state of the art in open-source and commercial (RDF) data management.

RDF-3X [31] v0.3.7 is a state of the art RDF system whose cardinality estimator was specifically tailored to RDF and was shown to outperform the related approaches.

C-SET is the *characteristic sets* technique [30] that specifically targets RDF and pays particular attention to star queries. There is no publicly available implementation of this technique, so we reimplemented it ourselves. It was shown to outperform the approach by Stocker et al. [37], so we did not test the latter separately.

PostgreSQL v9.4.5 uses a conventional cardinality estimator with one-dimensional histograms. Statistics collection was automatic.

SystemX is a commercial RDBMS.[2] Its highly tuned cardinality estimator operates in two modes: in the *static mode* the system uses statistics collected automatically prior to estimation, and in the *dynamic mode* it can sample the data during estimation if the estimate accuracy is low. We used the maximum sampling level.

To store RDF data into RDBMSs, we replaced all resources with unique integers, and then we considered two storage modes. First, we stored all triples in one ternary *triple table*, so all queries amount to self-joins over this table. Second, we *vertically partitioned* the data by storing $\langle s, rdf{:}type, o\rangle$ as tuple $\langle s\rangle$ in a unary relation o, and $\langle s, p, o\rangle$ with $p \neq rdf{:}type$ as tuple $\langle s, o\rangle$ in a binary relation p. Using multiple tables generally allows for more accurate statistics collection. We thus obtained a total of eight comparison systems: RDF-3X, C-SET, two (P-T and P-V) PostgreSQL variants, and two static (XS-T and XS-V) and two dynamic (XD-T and XD-V) SystemX variants. For systems other than C-SET, we obtained their cardinality estimates using their `EXPLAIN` facility.

6.4 Accuracy Experiments

Figure 2 shows the q-error distribution per dataset and system, thus summarising the results of our accuracy experiments on all 156 test queries. Each bar chart shows the percentages of the queries with q-error in the intervals from the legend. Each box plot groups the q-errors per system: the box shows the lower and upper quartiles of the q-error, the line dividing the box shows the median, the line whiskers show the minimum and maximum, and the diamond mark shows the average. Please note that the maximum q-errors sometimes fall beyond the shown portion of the y axis.

As one can see from the figure, SumRDF outperforms the other systems by several orders of magnitude on LUBM and WatDiv. On DBLP, it exhibits the best maximum and average q-errors, and SystemX with dynamic sampling is the only system that provides comparable performance. We conjecture the latter to be due to

[1] http://www.cs.ox.ac.uk/isg/tools/SumRDF/

[2] Publishing benchmarks with the system named is prohibited by the system's license.

Table 1: Dataset, summary, and query statistics

Benchmark	Original graph G		Summary S				Queries by type															
							Linear				Star				Snowflake				Complex			
	Nr resources	Nr triples	Nr buckets	Nr triples	Reduction factor	Construction time	total	μ-un.	min	max	total	μ-un.	min	max	total	μ-un.	min	max	total	μ-un.	min	max
LUBM	16.4 M	91.1 M	313	14,926	6.1 k	6 min	16	3	1	6	2	0	4	10	6	0	4	10	14	2	4	9
WatDiv	10.2 M	108.9 M	5,504	43,350	2.5 k	2 h	30	0	2	3	40	0	2	9	30	0	5	9	3	0	6	10
DBLP	25.4 M	55.5 M	4,765	28,631	1.9 k	45 min	4	0	1	4	1	1	3	3	2	1	6	7	8	6	5	11

Table 2: Estimates confidence for varying q-errors

Q-Error ε	10					100					1000				
The % bound on P$(e \geq \varepsilon)$ from Theorem 3.5	[0,1]	(1, 5]	(5, 10]	(10, 25]	> 25	[0,1]	(1, 5]	(5, 10]	(10, 25]	> 25	[0,1]	(1, 5]	(5, 10]	(10, 25]	> 25
LUBM (29)	20 (2)	0 (0)	6 (1)	2 (0)	1 (0)	25 (3)	0 (0)	4 (0)	0 (0)	0 (0)	28 (1)	1 (0)	0 (0)	0 (0)	0 (0)
WatDiv (48)	16 (1)	6 (1)	1 (0)	10 (2)	15 (3)	42 (0)	1 (0)	0 (0)	0 (0)	5 (0)	43 (0)	0 (0)	0 (0)	0 (0)	5 (0)
DBLP (3)	2 (0)	1 (0)	0 (0)	0 (0)	0 (0)	3 (0)	0 (0)	0 (0)	0 (0)	0 (0)	3 (0)	0 (0)	0 (0)	0 (0)	0 (0)

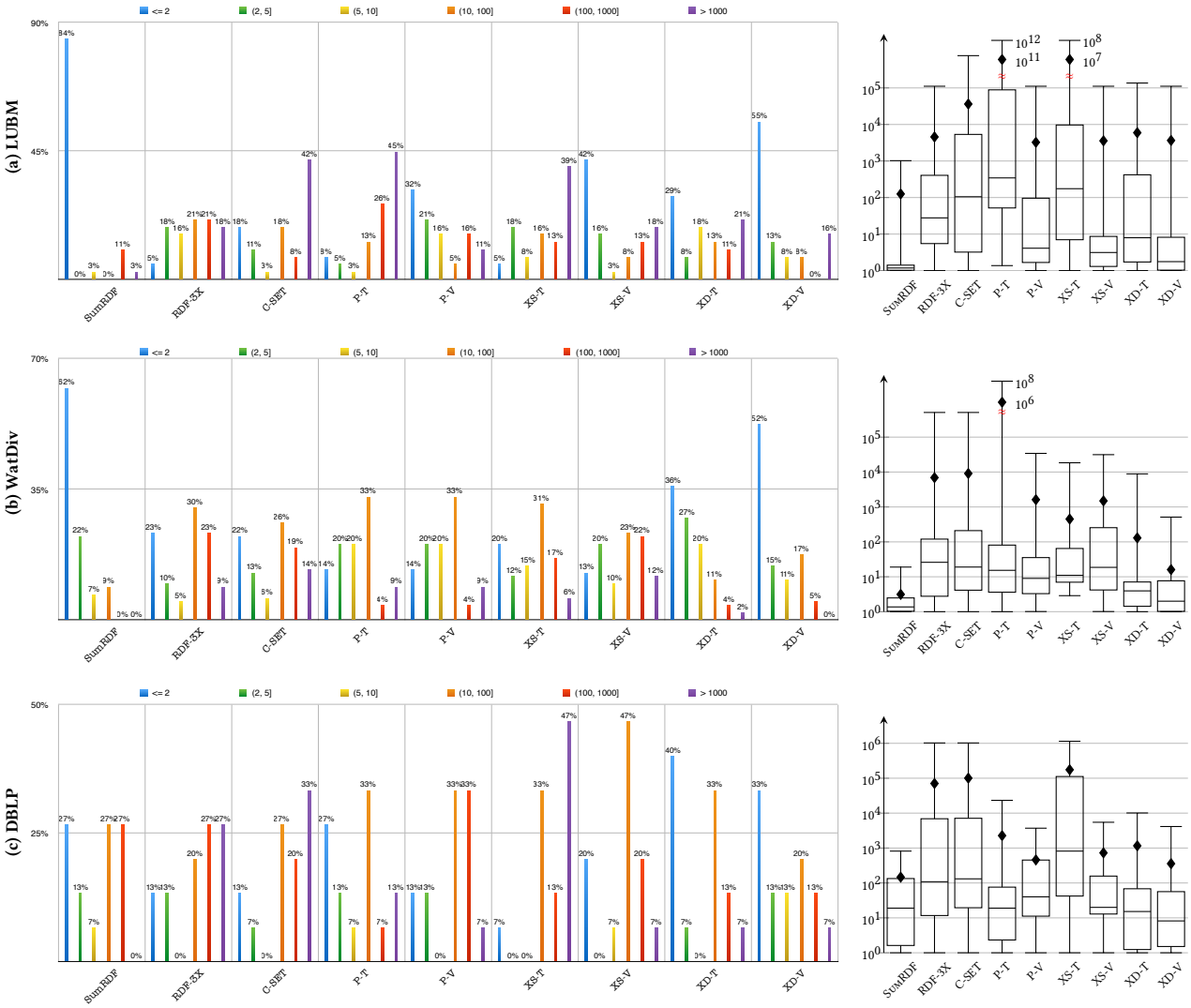

Figure 2: Histograms and box plots of the distribution of the q-error on test datasets

Table 3: The q-error for μ-unifiable queries

	ID	Eq. (1)	SumRDF	RDF-3X	C-SET	P-T	P-V	XS-T	XS-V	XD-T	XD-V
LUBM	U1	1.4 k	769.8	71.3	205.2 k	3.5 T	122.5	26.3 G	2.6 k	17.5 k	3.4 k
	U2	111.0 k	926.3	111.0 k	111.0 k	7.9 T	111.0 k	12.3 M	111.0 k	982.0	111.0 k
	U3	1.1	1.4	254.5	67.5 k	25 G	2.7	87.3 M	1.2	4.8 k	1.0
	U4	1.2	1.8	455.4	763.9 k	650 G	9.8	49.2 M	1.1	2.7 k	1.2
	U5	1.1	1.0	5.9	1.3	11.2 k	1.6	10.2 k	1.5	12.2 k	1.5
DBLP	D1	1.2	1.2	9.9 k	440.8 k	106.3	3.4	220.4 k	1.4	2.8	2.8
	D2	2.1	2.1	1.0 M	1.0 M	23.2 k	40.0	1.0 M	8.8	15.2	3.2
	D3	238.0	19.9	11.9 k	238.0	45.4	238.0	238.0	238.0	79.3	238.0
	D4	320.2	320.2	3.9 k	29.7 k	432.1	1.0 k	213.5 k	5.5 k	6.3 k	55.8
	D5	721.3	721.3	313.3	4.1 k	3.9	344.4	4.1 k	4.1 k	2.0	4.1 k
	D6	57.0	57.0	57.0	57.0	2.7	57.0	57.0	57.0	57.0	57.0
	D7	210.3	210.3	10.1 k	10.1 k	10.1 k	561.3	10.1 k	76.5	10.1 k	8.2
	D8	817.0	817.0	817.0	817.0	38.9	817.0	817.0	817.0	817.0	817.0
min		1.1	1.0	5.9	1.3	2.7	1.6	57.0	1.1	2.0	1.0
med		210.3	57.0	817.0	29.7 k	10.1 k	122.5	213.5 k	76.6	982.1	55.8
avg		8.8 k	296.1	90.1 k	204.3 k	940.8 G	8.8 k	31.8 M	9.5 k	4.3 k	9.2 k
max		111.0 k	926.2	1.0 M	1.0 M	7.9 T	111.0 k	263.4 M	111.0 k	17.6 k	111.0 k

selections in queries: if a query contains a resource that has not been captured accurately using histograms, more accurate statistics are obtained on demand. Overall, our technique considerably improves the state of the art in cardinality estimation of RDF queries.

We noticed no pattern in the systems' behaviour on queries with q-error above 10: each system undershoots on some, but overshoots on other queries. Different variants of the same system often differ greatly on the same query; for example, P-T might overshoot while P-V would undershoot, and similarly for SystemX. Also, for each dataset and system, there is a query where the system is most accurate, but SumRDF is never the least accurate.

Atoms of the form $\langle x, rdf{:}type, C\rangle$, whose presence in the query does not affect the query answers, were a common source of estimation errors. In C-SET, P-T, and XS-T, such atoms impose a selection on the *rdf:type* property that is combined using independence assumption, which usually leads to a significant underestimation.

We further analysed *difficult queries*, where SumRDF exhibited a q-error above 10. Five queries of LUBM have q-errors between 750 and 1,000; three are triangular and two are cyclic. Nine queries of WatDiv have q-error between 10 to 20, and all of these contain one or two selections. Finally, on DBLP, three queries have q-errors between 10 and 20, and additional five have q-errors above 20; all are either cyclic or have up to three selections. Thus, queries containing cycles and/or selections are sometimes hard for SumRDF. However, most systems exhibited a q-error above 10 on these queries as well. SystemX with dynamic sampling performed better on acyclic queries with selections: the system would additionally sample whenever the statistics were insufficiently precise.

6.5 Impact of μ-Unification

Formula (1) for μ-unification-free queries does not require computing the partition base for the query, so the formula should be easier to evaluate than the general formula from Theorem 4.3. Hence, we investigated whether using the general formula improves estimation accuracy. Table 3 compares the q-error of the estimate computed using formula (1) with the q-errors of all other systems (including SumRDF) for all μ-unifiable queries. WatDiv does not occur in the table since all of its queries are μ-unification-free.

As one can see, μ-unifiable queries tend to be difficult: for each query, at least one system exhibits a q-error above 900. However, handling μ-unifiable queries correctly reduces the maximum error by two orders of magnitude, and it also considerably improves the median and the average. Finally, SumRDF considerably outperforms all other systems on all aggregate metrics, apart from the median of XD-V, whereas the aggregate metrics of formula (1) seem similar to P-V. Thus, our principled approach to μ-unifiable queries contributes substantially to estimation accuracy.

6.6 Probabilistic Error Bounds

For ε equal to 10, 100, and 1,000, Table 2 shows the bound on the probability of q-error exceeding ε computed as stated in Theorem 3.5. The probabilities (expressed as percentages) are grouped into five intervals. Each table cell shows the number of queries in the interval, and in parentheses the number of queries where the q-error indeed exceeds ε. For example, on LUBM, there are 20 queries for which the probability of q-error being larger than 10 is less than 1%, and for two of these the q-error is larger than 10.

We could not compute $\sigma_{q,S}$ in several cases: query $q \cup \rho(q)$ from Theorem 4.4 is always μ-unifiable and tends to be very complex. Moreover, Theorem 3.5 does not apply to queries with $E_{q,S} < 1$. The numbers in parentheses next to each dataset name show the number of queries on which we could compute $\mathrm{P}(e \geq \varepsilon)$.

Already for $\varepsilon = 10$, we can correctly bound with 99% certainty the estimates for 18 out of 29 LUBM queries, 15 out of 48 WatDiv queries, and two out of three DBLP queries. With $\varepsilon = 100$, we cover 75% of LUBM, 87% of WatDiv, and all of the DBLP queries. These results suggest that the confidence bounds provided by Theorem 3.5 are indeed very useful. In our future work, we shall incorporate them into query planning algorithms so that a planner can prefer plans with higher confidence cardinality estimates.

7 CONCLUSION AND OUTLOOK

We presented a new approach for estimating the cardinality of CQs on RDF graphs. Our approach is based on graph summarisation and it formalises the estimation problem based on a precise 'possible worlds' semantics. Our technique can also provide statistical confidence for an estimate, even for arbitrarily shaped, complex CQs. Finally, we showed experimentally that our approach outperforms state of the art RDF and relational cardinality estimators.

We see many exciting opportunities for future work. On the theoretical side, supporting range queries should be easy, but adding `DISTINCT` and aggregation is likely to be more involved. We shall also try to incorporate further information about the data and thus reduce the number of graphs represented by a summary. On the practical side, we shall further analyse the sources of errors and develop ways of computing more precise summaries, possibly by incorporating variants of dynamic sampling. We shall also investigate how to update the summary without recomputing it from scratch when the underlying graph is updated.

ACKNOWLEDGMENTS

The research presented in this paper was supported by the EPSRC projects MaSI^3 (EP/P025943/1), DBOnto (EP/L012138/1), and ED^3 (EP/N014359/1).

REFERENCES

[1] Ashraf Aboulnaga and Surajit Chaudhuri. 1999. Self-tuning Histograms: Building Histograms Without Looking at Data. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1999)*. ACM, Philadelphia, PA, USA, 181–192.

[2] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. Join Synopses for Approximate Query Answering. *SIGMOD Record* 28, 2 (1999), 275–286.

[3] Günes Aluç, Olaf Hartig, M. Tamer Özsu, and Khuzaima Daudjee. 2014. Diversified Stress Testing of RDF Data Management Systems. In *Proc. of the 13th Int. Semantic Web Conference (ISWC 2014) (LNCS)*, Vol. 8796. Springer, Riva del Garda, Italy, 197–212.

[4] Nicolas Bruno. 2003. *Statistics on Query Expressions in Relational Database Management Systems.* Ph.D. Dissertation. Columbia University.

[5] Nicolas Bruno and Surajit Chaudhuri. 2004. Conditional Selectivity for Statistics on Query Expressions. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2004)*. ACM, Paris, France, 311–322.

[6] Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. 2001. STHoles: A Multidimensional Workload-aware Histogram. *SIGMOD Record* 30, 2 (2001), 211–222.

[7] Šejla Čebirić, François Goasdoué, and Ioana Manolescu. 2015. Query-oriented Summarization of RDF Graphs. *P-VLDB* 8, 12 (2015), 2012–2015.

[8] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 2001. Approximate query processing using wavelets. *VLDB Journal* 10, 2 (2001), 199–223.

[9] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Aaron Kershenbaum, Edith Schonberg, Kavitha Srinivas, and Li Ma. 2007. Scalable Semantic Retrieval through Summarization and Refinement. In *Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI 2007)*. AAAI Press, Vancouver, BC, Canada, 299–304.

[10] David Freedman and Persi Diaconis. 1981. On the Histogram as a Density Estimator: L_2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 57, 4 (1981), 453–476.

[11] César A. Galindo-Legaria, Milind Joshi, Florian Waas, and Ming-Chuan Wu. 2003. Statistics on Views. In *Proc. of the 29th Int. Conf. on Very Large Databases (VLDB 2003)*. Morgan Kaufmann, Berlin, Germany, 952–962.

[12] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. 2000. *Database System Implementation.* Prentice-Hall, Upper Saddle River, NJ, USA.

[13] Minos Garofalakis and Phillip B. Gibbons. 2002. Wavelet Synopses with Error Guarantees. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2002)*. ACM, Madison, WI, USA, 476–487.

[14] Lise Getoor, Benjamin Taskar, and Daphne Koller. 2001. Selectivity Estimation Using Probabilistic Models. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2001)*. ACM, Santa Barbara, CA, USA, 461–472.

[15] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. 2000. Approximating Multi-dimensional Aggregate Range Queries over Real Attributes. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2000)*. ACM, Dallas, TX, USA, 463–474.

[16] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. 2005. Selectivity estimators for multidimensional range queries over real attributes. *VLDB Journal* 14, 2 (2005), 137–154.

[17] Y. Guo, Z. Pan, and J. Heflin. 2005. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics* 3, 2–3 (2005), 158–182.

[18] Steve Harris and Andy Seaborne. 2013. SPARQL 1.1 Query Language, W3C Recommendation. (March 21 2013).

[19] Hai Huang and Chengfei Liu. 2011. Estimating Selectivity for Joined RDF Triple Patterns. In *Proc. of the 20th ACM Conf. on Information and Knowledge Management (CIKM 2011)*. ACM, Glasgow, United Kingdom, 1435–1444.

[20] Yannis Ioannidis. 2003. The History of Histograms (Abridged). In *Proc. of the 29th Int. Conf. on Very Large Databases (VLDB 2003)*. Morgan Kaufmann, Berlin, Germany, 19–30.

[21] Yannis E. Ioannidis and Stavros Christodoulakis. 1991. On the Propagation of Errors in the Size of Join Results. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1991)*. ACM, Denver, CO, USA, 268–277.

[22] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1998), 359–392.

[23] Graham Klyne, Jeremy J. Carroll, and Brian McBride. 2014. RDF 1.1: Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/. (February 25 2014).

[24] Ju-Hong Lee, Deok-Hwan Kim, and Chin-Wan Chung. 1999. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1999)*. ACM, Philadelphia, PA, USA, 205–214.

[25] Kristen LeFevre and Evimaria Terzi. 2010. GraSS: Graph Structure Summarization. In *Proc. of the SIAM Int. Conf. on Data Mining (SDM 2010)*. SIAM, Columbus, OH, USA, 454–465.

[26] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *P-VLDB* 9, 3 (2015), 204–215.

[27] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press, New York, NY, USA.

[28] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. 1998. Wavelet-based Histograms for Selectivity Estimation. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1998)*. ACM, Seattle, WA, USA, 448–459.

[29] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph Summarization with Bounded Error. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2008)*. ACM, Vancouver, BC, Canada, 419–432.

[30] Thomas Neumann and Guido Moerkotte. 2011. Characteristic Sets: Accurate Cardinality Estimation for RDF Queries with Multiple Joins. In *Proc. of the 27th IEEE Int. Conf. on Data Engineering (ICDE 2011)*. IEEE Computer Society, Hannover, Germany, 984–994.

[31] Thomas Neumann and Gerhard Weikum. 2010. The RDF-3X engine for scalable management of RDF data. *VLDB Journal* 19, 1 (2010), 91–113.

[32] J. Pérez, M. Arenas, and C. Gutierrez. 2009. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems* 34, 3 (2009), 1–45.

[33] Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. 1996. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 1996)*. ACM, Montreal, QC, Canada, 294–305.

[34] Viswanath Poosala and Yannis E. Ioannidis. 1997. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proc. of the 23rd Int. Conf. on Very Large Databases (VLDB 1997)*. Morgan Kaufmann, Athens, Greece, 486–495.

[35] Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2017. Graph summarization with quality guarantees. *Data Mining and Knowledge Discovery* 31, 2 (2017), 314–349.

[36] Joshua Spiegel and Neoklis Polyzotis. 2009. TuG Synopses for Approximate Query Answering. *ACM Transactions on Database Systems* 34, 1 (2009), 3:1–3:56.

[37] Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, and Dave Reynolds. 2008. SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation. In *Proc. of the 17th Int. Conf. on World Wide Web (WWW 2008)*. ACM, Beijing, China, 595–604.

[38] Pawel Terlecki, Hardik Bati, César A. Galindo-Legaria, and Peter Zabback. 2009. Filtered statistics. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2009)*. ACM, Providence, RI, USA, 897–904.

[39] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Efficient Aggregation for Graph Summarization. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2008)*. ACM, Vancouver, BC, Canada, 567–580.

[40] Kostas Tzoumas, Amol Deshpande, and Christian S. Jensen. 2013. Efficiently Adapting Graphical Models for Selectivity Estimation. *VLDB Journal* 22, 1 (2013), 3–27.

[41] Feng Yu, Wen-Chi Hou, Cheng Luo, Dunren Che, and Mengxia Zhu. 2013. CS2: A New Database Synopsis for Query Estimation. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2013)*. ACM, New York, NY, USA, 469–480.

[42] Lei Zou, M. Tamer Özsu, Lei Chen, Xuchuan Shen, Ruizhe Huang, and Dongyan Zhao. 2014. gStore: A Graph-based SPARQL Query Engine. *VLDB Journal* 23, 4 (2014), 565–590.