Online Resource 1 for paper: "SRX: Efficient Management of Spatial RDF Data"

Konstantinos Theocharidis[1,5], John Liagouris[2], Nikos Mamoulis[3], Panagiotis Bouros[4], and Manolis Terrovitis[5]

[1]University of Peloponnese, Greece
[2]ETH Zürich, Switzerland
[3]University of Ioannina, Greece
[4]Johannes Gutenberg University Mainz, Germany
[5]IMSI 'Athena', Greece
ktheocharid@uop.gr, john.liagouris@inf.ethz.ch, nikos@cs.uoi.gr
bouros@uni-mainz.de, mter@imis.athena-innovation.gr

This supplementary document provides additional information on the datasets and benchmarks used in the experimental evaluation of the paper *SRX: Efficient Management of Spatial RDF Data* submitted to the VLDB Journal.

The material is provided for reproducibility purposes and consists of three parts. First, we give the spatial distribution of geometries in the two real datasets we used for query evaluation. Second, we provide the exact queries for each system, including Virtuoso, GraphDB, and Strabon. Third, we give details on the deltas extracted between different versions of the two datasets, which we used to generate update workloads.

To the best of our knowledge, the benchmark we present here is the first one for spatial RDF stores on real data. It includes: (i) a large collection of queries with different spatial predicates and selectivities, and (ii) a configurable update workload based on real changes in the two datasets over time.

1 Spatial distribution of geometries

Figures 1 and 2 show the density plots for the spatial distribution of geometries in the following two datasets, which have been used in Section 9.2 of the paper:

1. LinkedGeodata (LGD), as retrieved from `https://tinyurl.com/yc4lxqdv` on July 3, 2019.
2. YAGO2s `2.5.3` (YAGO), as retrieved from `https://tinyurl.com/y7ukhge3` on July 3, 2019.

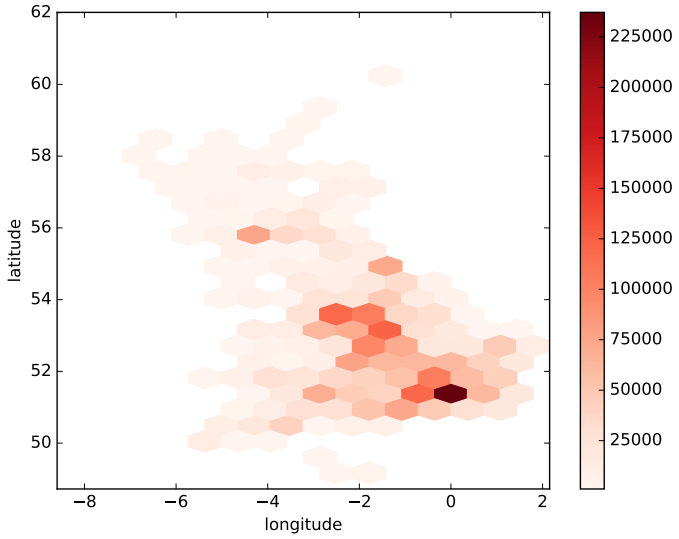

Figure 1: Spatial distribution of geometries in LGD. LGD contains geometries for entities in the United Kingdom with highest density in the area around London.

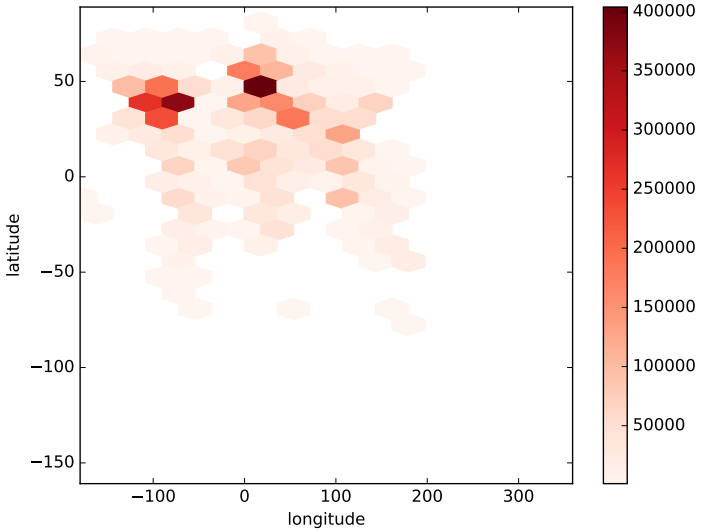

Figure 2: Spatial distribution of geometries in YAGO. YAGO's geometries spread all over the globe with highest density in North America and Europe.

2 Queries

In this section we describe the exact range, join, and kNN queries used in Section 9.2 of the paper. Each query is uniquely identified by a QueryID, as used in the paper. All queries can be found at: `https://web.imsi.athenarc.gr/SRX`.

2.1 Spatial range queries

The spatial range queries for LGD follow the template:

Select ?*s*
Where ?*s* name ?*n* .?*s* label ?*l* .
?*s* type **[TYPE]** . ?*s* hasGeometry ?*g* .
Filter WITHIN(?*g*, "RECTANGLE(**[MBR]**)")

where **[TYPE]** and **[MBR]** are instantiated for each query as follows:

QueryID	[TYPE]	[MBR]
LGD.SL1	police	-5,50,0,55
LGD.SL2	bus_stop	-10,50,0,60
LGD.SL3*	park	-5,50,0,55
LGD.LS1	pub	-5,45,0,50
LGD.LS2	bus_stop	-10,45,-5,50
LGD.LS3*	road	-10,45,-5,50
LGD.SS1	restaurant	-5,45,0,50
LGD.SS2*	park	-5,45,0,50
LGD.SS3*	road	-5,45,0,50
LGD.LL1	bus_stop	-5,55,0,60

The spatial range queries for YAGO are given in Table 1:

YAGO.SL1* Select ?*gn* ?*fn* ?*pr* Where ?*p* hasGivenName ?*gn* . ?*p* hasFamilyName ?*fn* . ?*p* hasWonPrize ?*pr* . ?*p* diedIn ?*c* . ?*c* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-100, 20, -80, 40)")	**YAGO.SL2*** Select ?*gn* ?*fn* Where ?*p* hasGivenName ?*gn* . ?*p* hasFamilyName ?*fn* . ?*p* a Wordnet_scientist_110560637 . ?*p* wasBornIn ?*c* . ?*c* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-95, 40, -90, 45)")
YAGO.LS1* Select ?*p* ?*w* Where ?*p* hasAcademicAdvisor ?*a* . ?*a* worksAt ?*w* . ?*w* isLocatedIn ?*l* . ?*l* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-160, -50, -150, -40)")	**YAGO.LS2*** Select ?*e* ?*c* Where ?*e* happenedIn ?*l* . ?*l* a ?*c* . ?*c* subClassOf Wordnet_city_108524735 . ?*l* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-130, 40, -120, 50)")
YAGO.SS1* Select ?*gn* ?*fn* Where ?*p* hasGivenName ?*gn* . ?*p* hasFamilyName ?*fn* . ?*p* a Wordnet_scientist_110560637 . ?*p* wasBornIn ?*c* . ?*c* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-105, 45, -100, 50)")	**YAGO.SS2*** Select ?*p* ?*w* Where ?*p* graduatedFrom ?*u* . ?*p* worksAt ?*w* . ?*u* isLocatedIn ?*l* . ?*l* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-110, 50, -100, 60)")
YAGO.LL1* Select ?*e* ?*c* Where ?*e* happenedIn ?*l* . ?*l* a ?*c* . ?*c* subClassOf Wordnet_city_108524735 . ?*l* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-90, 30, -80, 40)")	**YAGO.LL2*** Select ?*p* Where ?*p* hasArea ?*a* . ?*p* isLocatedIn ?*l* . ?*l* hasGeometry ?*g* . Filter WITHIN(?*g*, "RECTANGLE(-100, 30, -90, 40)")

Table 1: Spatial range queries for YAGO

2.2 Spatial distance join queries

The spatial distance join queries for LGD are given in Table 2:

LGD.J1 (point-point) Select ?$s1$?$s2$ Where ?$s1$ type hotel . ?$s1$ hasGeometry ?$g1$. ?$s2$ type hotel . ?$s2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.003"	**LGD.J2** (point-point) Select ?$s1$?$s2$?$l1$?$l2$ Where ?$s1$ name ?$l1$. ?$s1$ label ?$b1$. ?$s1$ type police . ?$s1$ hasGeometry ?$g1$. ?$s2$ name ?$l2$. ?$s2$ label ?$b2$. ?$s2$ type police . ?$s2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.01"
LGD.J3 (point-point) Select ?$s1$?$s2$ Where ?$s1$ name ?$l1$. ?$s1$ label ?$b1$. ?$s1$ type pub . ?$s1$ hasGeometry ?$g1$. ?$s2$ name ?$l2$. ?$s2$ label ?$b2$. ?$s2$ type police . ?$s2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.02"	**LGD.J4*** (polygon-polygon) Select ?$s1$?$s2$ Where ?$s1$ type park . ?$s1$ hasGeometry ?$g1$. ?$s2$ type park . ?$s2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.05"
LGD.J5* (point-polygon) Select ?$s1$?$s2$ Where ?$s1$ label ?$b1$. ?$s1$ type police . ?$s1$ hasGeometry ?$g1$. ?$s2$ label ?$b2$. ?$s2$ type park . ?$s2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.01"	**LGD.J6*** (point-line), **[EPS]** $\in \{0.01,0.001,0.0005\}$ Select ?$s1$?$s2$ Where ?$s1$ label ?$b1$. ?$s1$ type hotel . ?$s1$ hasGeometry ?$g1$. ?$s2$ label ?$b2$. ?$s2$ type road . ?$s2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "**[EPS]**"

Table 2: Spatial distance join queries for LGD

The spatial distance join queries for YAGO are given in Table 3:

YAGO.J1* Select ?$c1$?$c2$ Where ?$a1$ hasAirportCode ?$c1$. ?$a1$ hasGeometry ?$g1$. ?$a2$ hasAirportCode ?$c2$. ?$a2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"	**YAGO.J2*** Select ?$p1$?$p2$ Where ?$p1$ hasGivenName ?$gn1$. ?$p1$ hasFamilyName ?$fn1$. ?$p1$ hasWonPrize ?$pr1$. ?$p1$ wasBornIn ?$c1$. ?$c1$ hasGeometry ?$g1$. ?$p2$ hasGivenName ?$gn2$. ?$p2$ hasFamilyName ?$fn2$. ?$p2$ hasWonPrize ?$pr2$. ?$p2$ wasBornIn ?$c2$. ?$c2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"
YAGO.J3* Select ?p ?$c1$?$c2$ Where ?p hasGivenName ?gn . ?p hasFamilyName ?fn . ?p actedIn ?m . ?m isLocatedIn ?$c1$. ?$c1$ hasGeometry ?$g1$. ?p wasBornIn ?$c2$. ?$c2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"	**YAGO.J4*** Select ?$p1$?$p2$ Where ?$p1$ hasFamilyName ?$fn1$. ?$p1$ wasBornIn ?$c1$. ?$c1$ hasGeometry ?$g1$. ?$p1$ isMarriedTo ?$p2$. ?$p2$ wasBornIn ?$c2$. ?$c2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"
YAGO.J5* Select ?p Where ?p hasFamilyName ?fn . ?p livesIn ?$c1$. ?$c1$ hasGeometry ?$g1$. ?p worksAt ?$c2$. ?$c2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"	**YAGO.J6*** Select ?$p1$?$p2$ Where ?$p1$ hasGivenName ?$gn1$. ?$p1$ hasFamilyName ?$fn1$. ?$p1$ a Wordnet_scientist_110560637 . ?$p1$ wasBornIn ?$c1$. ?$c1$ hasGeometry ?$g1$. ?$p2$ hasGivenName ?$gn2$. ?$p2$ hasFamilyName ?$fn2$. ?$p2$ a Wordnet_scientist_110560637 . ?$p2$ diedIn ?$c2$. ?$c2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"
YAGO.J7* Select ?$p1$?$p2$ Where ?$p1$ graduatedFrom ?$u1$. ?$u1$ hasGeometry ?$g1$. ?$p2$ actedIn ?$m2$. ?$m2$ isLocatedIn ?$l2$. ?$l2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "0.1"	**YAGO.J8***, **[EPS]** $\in \{0.1,0.01,0.001\}$ Select ?$p1$?$p2$ Where ?$p1$ worksAt ?$w1$. ?$w1$ hasGeometry ?$g1$. ?$p2$ worksAt ?$w2$. ?$w2$ hasGeometry ?$g2$. Filter DISTANCE(?$g1$,?$g2$) < "**[EPS]**"

Table 3: Spatial distance join queries for YAGO

2.3 Spatial kNN queries for Encoding, Baseline, and Basic

The spatial kNN queries for LGD have the following template:

Select ?s
Where ?s name ?n .?s label ?l .
?s type **[TYPE]** . ?s hasGeometry ?g .
Filter kNN(?g, "POINT(**[COORDS]**)", k)

where $k \in \{5, 10, 20, 50, 100\}$, and **[TYPE]**, **[COORDS]** are instantiated for each query as follows:

QueryID	[TYPE]	[COORDS]
LGD.SL1	police	-2.5, 52.5
LGD.SL2	bus_stop	-5, 55
LGD.SL3*	park	-2.5, 52.5
LGD.LS1	pub	-2.5, 47.5
LGD.LS2	bus_stop	-7.5, 47.5
LGD.LS3*	road	-7.5, 47.5
LGD.SS1	restaurant	-2.5, 47.5
LGD.SS2*	park	-2.5, 47.5
LGD.SS3*	road	-2.5, 47.5
LGD.LL1	bus_stop	-2.5, 57.5

The spatial kNN queries for YAGO, where $k \in \{5, 10, 20, 50, 100\}$, are given in Table 4:

YAGO.SL1* Select ?gn ?fn ?pr Where ?p hasGivenName ?gn . ?p hasFamilyName ?fn . ?p hasWonPrize ?pr . ?p diedIn ?c . ?c hasGeometry ?g . Filter kNN(?g, "POINT(-90, 30)", k)	**YAGO.SL2*** Select ?gn ?fn Where ?p hasGivenName ?gn . ?p hasFamilyName ?fn . ?p a Wordnet_scientist_110560637 . ?p wasBornIn ?c . ?c hasGeometry ?g . Filter kNN(?g, "POINT(-92.5, 42.5)", k)
YAGO.LS1* Select ?p ?w Where ?p hasAcademicAdvisor ?a . ?a worksAt ?w . ?w isLocatedIn ?l . ?l hasGeometry ?g . Filter kNN(?g, "POINT(-155, -45)", k)	**YAGO.LS2*** Select ?e ?c Where ?e happenedIn ?l . ?l a ?c . ?c subClassOf Wordnet_city_108524735 . ?l hasGeometry ?g . Filter kNN(?g, "POINT(-125, 45)", k)
YAGO.SS1* Select ?gn ?fn Where ?p hasGivenName ?gn . ?p hasFamilyName ?fn . ?p a Wordnet_scientist_110560637 . ?p wasBornIn ?c . ?c hasGeometry ?g . Filter kNN(?g, "POINT(-102.5, 47.5)", k)	**YAGO.SS2*** Select ?p ?w Where ?p graduatedFrom ?u . ?p worksAt ?w . ?u isLocatedIn ?l . ?l hasGeometry ?g . Filter kNN(?g, "POINT(-105, 55)", k)
YAGO.LL1* Select ?e ?c Where ?e happenedIn ?l . ?l a ?c . ?c subClassOf Wordnet_city_108524735 . ?l hasGeometry ?g . Filter kNN(?g, "POINT(-85, 35)", k)	**YAGO.LL2*** Select ?p Where ?p hasArea ?a . ?p isLocatedIn ?l . ?l hasGeometry ?g . Filter kNN(?g, "POINT(-95, 35)", k)

Table 4: Spatial kNN queries for YAGO

2.4 Spatial kNN queries for Virtuoso

Virtuoso `7.2.5-rc1.3217-pthreads` does not support nearest neighbor queries per se, however, some of these queries can be expressed using a combination of distance filters and ordering. The spatial kNN queries we used for Virtuoso involve only

point queries – other geometries are not supported by the distance function in this version of the system. In Virtuoso's query language these queries for LGD follow the template:

Select ?s DISTANCE(?g, "POINT(**[COORDS]**)")
Where ?s name ?n .?s label ?l .
?s type **[TYPE]** . ?s hasGeometry ?g .
Filter WITHIN(?g, "POINT(**[COORDS]**)", **[RANGE]**)
Order By ASC 2
Limit k

where $k \in \{5, 10, 20, 50, 100\}$, and **[TYPE]**, **[COORDS]** are instantiated for each query as in Section 2.3 of this document. **[RANGE]** is measured in kilometers and is the minimum distance so that each query returns 100 results. **[RANGE]** is set for each query as follows:

QueryID	[RANGE]
LGD.SL1	110
LGD.SL2	20
LGD.LS1	170
LGD.LS2	190
LGD.SS1	180
LGD.LL1	80

2.5 Spatial kNN queries for GraphDB and Strabon

Similarly to Virtuoso, GraphDB `Free 8.6` and Strabon `3.3.2` do not support nearest neighbor queries per se, however, both systems can express them using a combination of distance filters and ordering. The spatial kNN queries for LGD in GraphDB's and Strabon's query language follow the same template:

Select ?s
Where ?s name ?n .?s label ?l .
?s type **[TYPE]** . ?s hasGeometry ?g .
Order By ASC
DISTANCE(?g, "POINT(**[COORDS]**)")
Limit k

where $k \in \{5, 10, 20, 50, 100\}$, and **[TYPE]**, **[COORDS]** are instantiated for each query as in Section 2.3 of this document.

The spatial kNN queries for YAGO in GraphDB's query language are given in Table 5 with $k \in \{5, 10, 20, 50, 100\}$:

YAGO.SL1* Select ?*gn* ?*fn* ?*pr* Where ?*p* hasGivenName ?*gn* . ?*p* hasFamilyName ?*fn* . ?*p* hasWonPrize ?*pr* . ?*p* diedIn ?*c* . ?*c* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-90, 30)") Limit *k*	**YAGO.SL2*** Select ?*gn* ?*fn* Where ?*p* hasGivenName ?*gn* . ?*p* hasFamilyName ?*fn* . ?*p* a Wordnet_scientist_110560637 . ?*p* wasBornIn ?*c* . ?*c* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-92.5, 42.5)") Limit *k*
YAGO.LS1* Select ?*p* ?*w* Where ?*p* hasAcademicAdvisor ?*a* . ?*a* worksAt ?*w* . ?*w* isLocatedIn ?*l* . ?*l* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-155, -45)") Limit *k*	**YAGO.LS2*** Select ?*e* ?*c* Where ?*e* happenedIn ?*l* . ?*l* a ?*c* . ?*c* subClassOf Wordnet_city_108524735 . ?*l* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-125, 45)") Limit *k*
YAGO.SS1* Select ?*gn* ?*fn* Where ?*p* hasGivenName ?*gn* . ?*p* hasFamilyName ?*fn* . ?*p* a Wordnet_scientist_110560637 . ?*p* wasBornIn ?*c* . ?*c* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-102.5, 47.5)") Limit *k*	**YAGO.SS2*** Select ?*p* ?*w* Where ?*p* graduatedFrom ?*u* . ?*p* worksAt ?*w* . ?*u* isLocatedIn ?*l* . ?*l* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-105, 55)") Limit *k*
YAGO.LL1* Select ?*e* ?*c* Where ?*e* happenedIn ?*l* . ?*l* a ?*c* . ?*c* subClassOf Wordnet_city_108524735 . ?*l* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-85, 35)") Limit *k*	**YAGO.LL2*** Select ?*p* Where ?*p* hasArea ?*a* . ?*p* isLocatedIn ?*l* . ?*l* hasGeometry ?*g* . Order By ASC DISTANCE(?*g*, "POINT(-95, 35)") Limit *k*

Table 5: Spatial kNN queries for YAGO in GraphDB

3 Deltas between different dataset versions

In this section we provide the exact differences (in number of triples) between versions:

1. `2013_04_29` and `2015_11_02` of LGD, as retrieved from `https://tinyurl.com/ydbscsxf` on July 3, 2019.

2. `2.5.3` and `3.0.2` of YAGO, as retrieved from `https://tinyurl.com/y7ukhge3` on July 3, 2019.

which we used to generate the update workloads of Section 9.4 of the paper. The workloads can be found at: `https://web.imsi.athenarc.gr/SRX`.

Columns "HasGeometry" in Tables 6 and 7 show the numbers of $\langle s, p, o \rangle$ triples where $p =$ "hasGeometry". These triples contain entity geometries as objects. The respective numbers for the rest of the triples are given in columns "Other".

For the update experiments with Strabon, which we present in Fig. 12 of paper, we created two different subsets of LGD (`2013_04_29`) that we used in Table 6. The first one corresponds to the initial base of Fig. 12a, while the second is a superset of the first and corresponds to the initial base of Fig. 12b.

To form the initial base **(a)** in Fig. 12a, we initially took the first 250 point (*Nodes*) and 250 linestring (*Ways*) 'HasGeometry' triples, including all their related 'Other' triples, from each of the deleted, updated, and inserted LGD deltas in Table 6. These

Table 6: Deltas between LGD versions `2013_04_29` (14.7GB) and `2015_11_02` (21.6GB) used for the update benchmark

Category	Deleted triples		Updated triples		Inserted triples	
	HasGeometry	**Other**	**HasGeometry**	**Other**	**HasGeometry**	**Other**
AerialwayThing_Nodes	2,265	19,761	12,640	61,878	22,799	198,796
AerialwayThing_Ways	1,126	15,329	9,419	38,345	9,486	94,979
AerowayThing_Nodes	7,860	95,947	7,366	53,177	18,488	174,201
AerowayThing_Ways	7,972	84,099	41,772	131,235	87,970	816,948
Amenity_Nodes	411,396	4,694,204	383,260	3,126,323	1,958,451	20,917,360
Amenity_Ways	168,547	2,381,609	465,889	1,913,576	1,705,501	19,771,641
Craft_Nodes	571	8,322	831	6,477	19,720	220,220
Craft_Ways	333	8,973	1,070	5,709	9,483	134,084
EmergencyThing_Nodes	4,996	107,124	9,631	197,532	369,059	4,031,998
EmergencyThing_Ways	3,002	62,098	9,412	42,782	26,342	442,821
HistoricThing_Nodes	11,783	241,258	20,013	191,571	170,642	1,459,176
HistoricThing_Ways	5,703	98,317	16,870	94,209	89,578	1,211,885
MilitaryThing_Nodes	472	5,968	583	6,345	11,282	147,547
MilitaryThing_Ways	732	12,412	2,155	9,376	10,062	122,245
Place_Nodes	72,941	10,117,063	183,472	5,211,302	930,603	10,494,621
Place_Ways	21,435	337,257	67,907	433,830	169,437	1,932,685
Shop_Nodes	79,362	948,495	115,006	999,194	678,149	7,271,373
Shop_Ways	21,683	1,300,070	59,325	205,847	242,059	906,075
TourismThing_Nodes	47,322	523,433	62,653	466,938	375,669	4,065,277
TourismThing_Ways	14,438	238,691	35,340	178,622	121,136	1,741,557
Total	**883,939**	**21,300,430**	**1,504,614**	**13,374,268**	**7,025,916**	**76,155,489**

Table 7: Deltas between YAGO versions `2.5.3` (18.1GB) and `3.0.2` (33.4GB) used for the update benchmark

Category	Deleted triples		Updated triples		Inserted triples	
	HasGeometry	**Other**	**HasGeometry**	**Other**	**HasGeometry**	**Other**
yagoDBpediaClasses	0	381,887	0	0	0	500,640
yagoDBpediaInstances	0	85,949	0	0	0	2,637,748
yagoFacts	0	1,198,399	0	325	0	2,379,436
yagoGeonamesClasses	0	585	0	0	0	586
yagoGeonamesClassIds	0	0	0	0	0	1
yagoGeonamesData	473,999	6,661,873	1,693,055	0	2,973,857	21,919,394
yagoGeonamesEntityIds	0	106,133	0	0	0	117,579
yagoGeonamesGlosses	0	0	0	0	0	1
yagoImportantTypes	0	2,723,628	0	0	0	0
yagoLabels	0	6,941,073	0	1,601,561	0	36,642,854
yagoLiteralFacts	0	2,533,216	0	650	0	1,584,304
yagoMetaFacts	0	820,616	0	0	0	2,295,716
yagoMultilingualClassLabels	0	0	0	0	0	1
yagoMultilingualInstanceLabels	0	8,164,316	0	0	0	0
yagoSchema	0	10	0	24	0	133
yagoSimpleTaxonomy	0	5,670	0	0	0	474,817
yagoSimpleTypes	0	4,334,474	0	0	0	15,824,332
yagoSources	0	61,838,095	0	0	0	190,482,753
yagoStatistics	0	7	0	88	0	5
yagoTaxonomy	0	381,893	0	0	0	500,645
yagoTransitiveType	0	11,145,280	0	0	0	45,085,647
yagoTypes	0	8,274,938	0	0	0	16,182,189
yagoWikipediaInfo	0	16,347,220	0	0	0	36,167,000
yagoWordnetDomains	0	172	0	0	0	1
yagoWordnetIds	0	0	0	0	0	1
Total	**473,999**	**131,945,434**	**1,693,055**	**1,602,648**	**2,973,857**	**372,795,783**

are the `deltas_a` to apply on base **(a)** in Fig. 12a. Then, we kept from the initial version of LGD (`2013_04_29`) only the triples related to any spatial entities found in

`deltas_a`, and we formed the base **(a)** that contains 10K triples.

The initial base **(b)** in Fig. 12b contains 100K triples, and is created similarly. In this case, we picked the first 2500 point (*Nodes*) and 2500 linestring (*Ways*) 'HasGeometry' triples, including all their related 'Other' triples, from each of the LGD deltas.