4th Workshop on
Scalable Semantic Web Knowledge Base Systems
(SSWS2008)

Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints

Christian Bizer
Andreas Schultz

Freie Universität Berlin

Karlsruhe, Germany. October 27, 2008

Overview

1. **Design of the Berlin SPARQL Benchmark**
   - The dataset and the query mix
2. **Benchmark Experiment and Results**
   - Which store is the best one?

Existing Benchmarks for Semantic Web Technologies

Lehigh University Benchmark (LUBM)

- benchmark for comparing the performance OWL reasoning engines
  - does not test specific SPARQL features like OPTIONALS, UNION, …
  - does not employ parameterized queries, concurrent clients, warm-up

DBpedia Benchmark

- uses DBpedia as benchmark dataset
- 5 queries that were relevant for DBpedia Mobile
  - very specific queries, benchmark dataset not scalable

SP²Bench

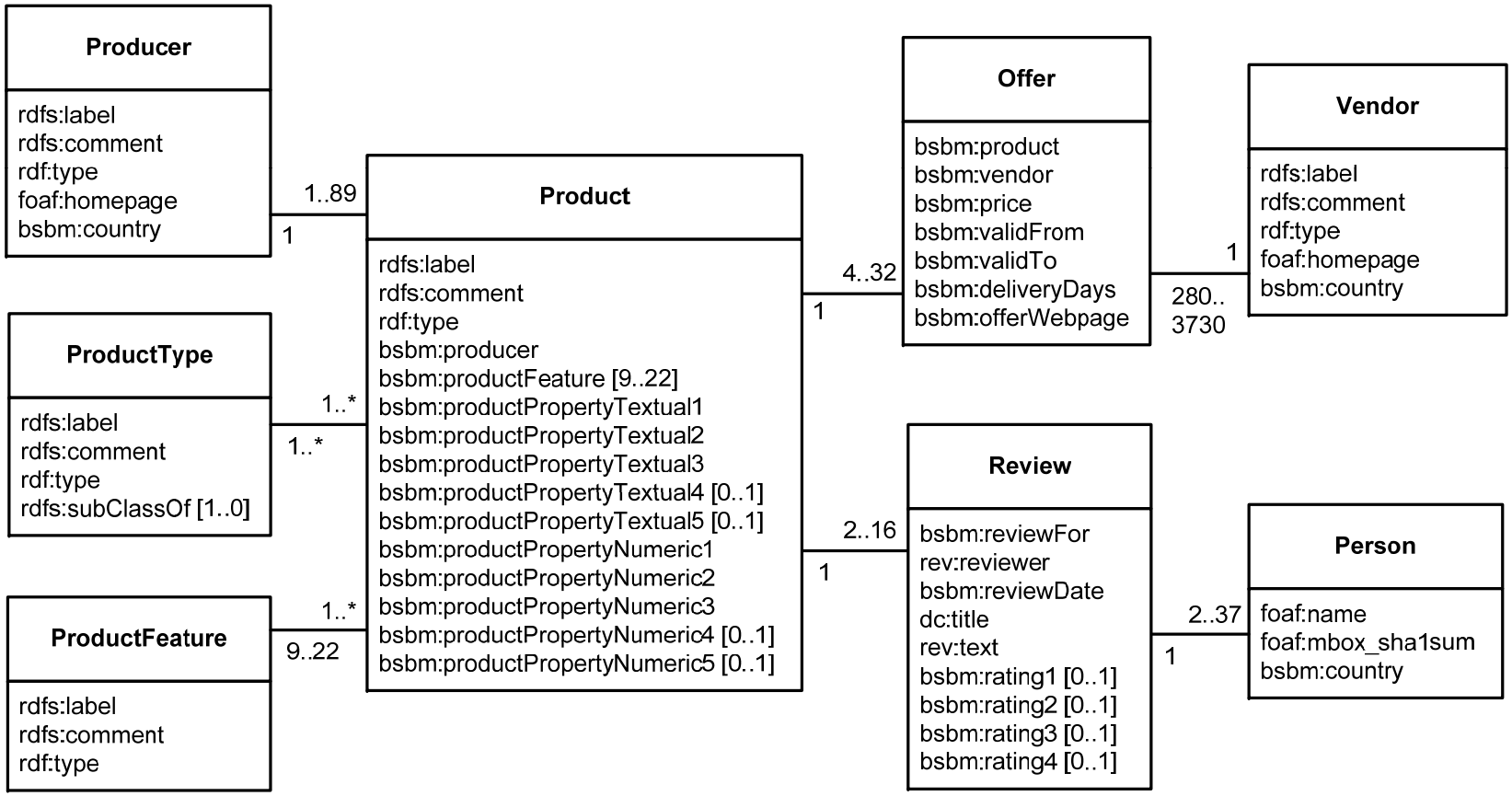
- from the databases group at Freiburg University, Germany
- uses an synthetic, scalable version of the DBLP bibliography dataset
- queries are designed for the comparison of different RDF store layouts
  - not designed towards realistic workloads of parameterized queries, no concurrent clients, no warm-up

2. Design Goals of the Berlin SPARQL Benchmark

1. **be SPARQL-specific**
   - test SPARQL-specific features
   - execute queries over the SPARQL protocol
2. **benchmark SUTs within an enterprise scenario where multiple clients execute realistic workloads of use case motivated queries.**
   - execute query mixes, instead of single queries
   - query parameterization
   - simulation of multiple clients
   - system warm-up
3. **allow the comparison of storage systems across internal data models.**
   - native RDF stores
   - relational database to RDF wrappers
   - relational databases
4. **do not require complex reasoning but measure query performance against large amounts of RDF data.**

Benchmark Dataset

- **The benchmark is built around an e-commerce use case, where a set of products is offered by different vendors and consumers have posted reviews about products.**



Data Generator

- **supports the creation of arbitrarily large datasets using the number of products as scale factor.**
- **produces**
  - RDF representation
  - relational representation
- **output formats**
  - Turtle
  - MySQL dump
- **The data generations is deterministic.**

Number of Instances in BSBM Datasets of different Sizes

Total number of triples	250K	1M	25M	100M
Number of products	666	2,785	70,812	284,826
Number of product features	2,860	4,745	23,833	47,884
Number of product types	55	151	731	2011
Number of producers	14	60	1422	5,618
Number of vendors	8	34	722	2,854
Number of offers	13,320	55,700	1,416,240	5,696,520
Number of reviewers	339	1432	36,249	146,054
Number of reviews	6,660	27,850	708,120	2,848,260
Total number of instances	**23,922**	**92,757**	**2,258,129**	**9,034,027**

The Benchmark Query Mix

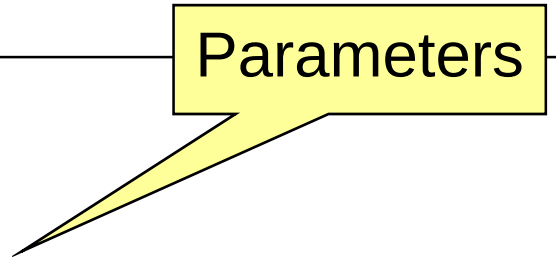
1. Query 1: Find products for a given set of generic features.
2. Query 2: Retrieve basic information about a specific product for display purposes.
3. Query 2: Retrieve basic information about a specific product for display purposes.
4. Query 3: Find products having some specific features and not having one feature.
5. Query 2: Retrieve basic information about a specific product for display purposes.
6. Query 2: Retrieve basic information about a specific product for display purposes.
7. Query 4: Find products matching two different sets of features.
8. Query 2: Retrieve basic information about a specific product for display purposes.
9. Query 2: Retrieve basic information about a specific product for display purposes.
10. Query 5: Find products that are similar to a given product.
11. Query 7: Retrieve in-depth information about a product including offers and reviews.
12. Query 7: Retrieve in-depth information about a product including offers and reviews.
13. Query 6: Find products having a label that contains a specific string.
14. Query 7: Retrieve in-depth information about a product including offers and reviews.
15. Query 7: Retrieve in-depth information about a product including offers and reviews.
16. Query 8: Give me recent English language reviews for a specific product.
17. Query 9: Get information about a reviewer.
18. Query 9: Get information about a reviewer.
19. Query 8: Give me recent English language reviews for a specific product.
20. Query 9: Get information about a reviewer.
21. Query 9: Get information about a reviewer.
22. Query 10: Get cheap offers which fulfill the consumer's delivery requirements.
23. Query 10: Get cheap offers which fulfill the consumer's delivery requirements.
24. Query 11: Get all information about an offer.
25. Query 12: Export information about an offer into another schema.

Representations of the Query Mix

1. SPARQL
2. SQL

Query 1: Find Products for a generic set of features

```
SELECT DISTINCT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product rdf:type %ProductType% .
    ?product bsbm:productFeature %ProductFeature1% .
    ?product bsbm:productFeature %ProductFeature2% .
    ?product bsbm:productPropertyNumeric1 ?value1 .
  FILTER (?value1 > %x%)}
ORDER BY ?label
LIMIT 10
```



Query Properties

- Small number of patterns
- Simple filters
- Uses ORDER BY and LIMIT

Query 2: Retrieve basic information about a specific product for display purposes

```
SELECT ?label ?comment ?producer ?productFeature ?propertyTextual1
    ?propertyTextual2 ?propertyTextual3 ?propertyNumeric1
    ?propertyNumeric2 ?propertyTextual4 ?propertyTextual5
    ?propertyNumeric4
WHERE {
    %ProductXYZ% rdfs:label ?label .
    %ProductXYZ% rdfs:comment ?comment .
    %ProductXYZ% bsbm:producer ?p .
    ?p rdfs:label ?producer .
    %ProductXYZ% dc:publisher ?p .
    %ProductXYZ% bsbm:productFeature ?f .
    ?f rdfs:label ?productFeature .
    %ProductXYZ% bsbm:productPropertyTextual1 ?propertyTextual1 .
    %ProductXYZ% bsbm:productPropertyTextual2 ?propertyTextual2 .
    %ProductXYZ% bsbm:productPropertyTextual3 ?propertyTextual3 .
    %ProductXYZ% bsbm:productPropertyNumeric1 ?propertyNumeric1 .
    %ProductXYZ% bsbm:productPropertyNumeric2 ?propertyNumeric2 .
  OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual4 ?propertyTextual4 }
  OPTIONAL { %ProductXYZ% bsbm:productPropertyTextual5 ?propertyTextual5 }
  OPTIONAL { %ProductXYZ% bsbm:productPropertyNumeric4 ?propertyNumeric4 }}
```

Query Properties

- large number of patterns
- Uses OPTIONAL

Query 3: Find products having some specific features and not having one feature

```
SELECT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product rdf:type %ProductType% .
    ?product bsbm:productFeature %ProductFeature1% .
    ?product bsbm:productPropertyNumeric1 ?p1 .
  FILTER ( ?p1 > %x% )
    ?product bsbm:productPropertyNumeric3 ?p3 .
  FILTER (?p3 < %y% )
  OPTIONAL {
      ?product bsbm:productFeature %ProductFeature2% .
      ?product rdfs:label ?testVar }
    FILTER (!bound(?testVar)) }
ORDER BY ?label
LIMIT 10
```

Query Properties

- Uses negation

Query 4: Find products matching two different sets of features

```
SELECT ?product ?label
WHERE {
    { ?product rdfs:label ?label .
      ?product rdf:type %ProductType% .
      ?product bsbm:productFeature %ProductFeature1% .
      ?product bsbm:productFeature %ProductFeature2% .
      ?product bsbm:productPropertyNumeric1 ?p1 .
    FILTER ( ?p1 > %x% )
} UNION {
      ?product rdfs:label ?label .
      ?product rdf:type %ProductType% .
      ?product bsbm:productFeature %ProductFeature1% .
      ?product bsbm:productFeature %ProductFeature3% .
      ?product bsbm:productPropertyNumeric2 ?p2 .
    FILTER ( ?p2> %y% ) }}
ORDER BY ?label
LIMIT 10 OFFSET 10
```

Query Properties

- Uses UNION

Query 5: Find products that are similar to a given product

```
SELECT DISTINCT ?product ?productLabel
WHERE {
    ?product rdfs:label ?productLabel .
    %ProductXYZ% rdf:type ?prodtype.
    ?product rdf:type ?prodtype .
  FILTER (%ProductXYZ% != ?product)
    %ProductXYZ% bsbm:productFeature ?prodFeature .
    ?product bsbm:productFeature ?prodFeature .
    %ProductXYZ% bsbm:productPropertyNumeric1 ?origProperty1 .
    ?product bsbm:productPropertyNumeric1 ?simProperty1 .
  FILTER (?simProperty1 < (?origProperty1 + 150) &&
          ?simProperty1 > (?origProperty1 – 150))
    %ProductXYZ% bsbm:productPropertyNumeric2 ?origProperty2 .
    ?product bsbm:productPropertyNumeric2 ?simProperty2 .
  FILTER (?simProperty2 < (?origProperty2 + 220) &&
          ?simProperty2 > (?origProperty2 – 220)) }
ORDER BY ?productLabel
LIMIT 5
```

Query Properties

- Uses complex filters

Query 6: Find products having a label that contains a specific word

```
SELECT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product rdf:type bsbm:Product .
  FILTER regex(?label, "%word1%")}
```

Query Properties

- Uses a regular expression to emulate free-text search

Query 7: Retrieve in-depth information about a product including offers and reviews

```
SELECT ?productLabel ?offer ?price ?vendor ?vendorTitle ?review
    ?revTitle ?reviewer ?revName ?rating1 ?rating2
WHERE {
    %ProductXYZ% rdfs:label ?productLabel .
  OPTIONAL {
      ?offer bsbm:product %ProductXYZ% .
      ?offer bsbm:price ?price .
      ?offer bsbm:vendor ?vendor .
      ?vendor rdfs:label ?vendorTitle .
      ?vendor bsbm:country
        <http://downlode.org/rdf/iso-3166/countries#DE>.
      ?offer dc:publisher ?vendor .
      ?offer bsbm:validTo ?date .
    FILTER (?date > %currentDate% ) }
  OPTIONAL {
      ?review bsbm:reviewFor %ProductXYZ% .
      ?review rev:reviewer ?reviewer .
      ?reviewer foaf:name ?revName .
      ?review dc:title ?revTitle .
    OPTIONAL { ?review bsbm:rating1 ?rating1 . }
    OPTIONAL { ?review bsbm:rating2 ?rating2 . } } }
```

Query 8: Give me recent English language reviews for a specific product

```
SELECT ?title ?text ?reviewDate ?reviewer ?reviewerName ?rating1
    ?rating2 ?rating3 ?rating4
WHERE {
    ?review bsbm:reviewFor %ProductXYZ% .
    ?review dc:title ?title .
    ?review rev:text ?text .
  FILTER langMatches( lang(?text), "EN" )
    ?review bsbm:reviewDate ?reviewDate .
    ?review rev:reviewer ?reviewer .
    ?reviewer foaf:name ?reviewerName .
  OPTIONAL { ?review bsbm:rating1 ?rating1 . }
  OPTIONAL { ?review bsbm:rating2 ?rating2 . }
  OPTIONAL { ?review bsbm:rating3 ?rating3 . }
  OPTIONAL { ?review bsbm:rating4 ?rating4 . } }
ORDER BY DESC(?reviewDate)
LIMIT 20
```

Query Properties

- Uses langMatches() function
- Uses OPTIONAL

Query 9: Get information about a reviewer

```
DESCRIBE ?x
WHERE {
    %ReviewXYZ% rev:reviewer ?x }
```

Query Properties

- Uses DESCRIBE

Query 10: Get cheap offers which fulfill the consumer's delivery requirements

```
SELECT DISTINCT ?offer ?price
WHERE {
    ?offer bsbm:product %ProductXYZ% .
    ?offer bsbm:vendor ?vendor .
    ?offer dc:publisher ?vendor .
    ?vendor bsbm:country %CountryXYZ% .
    ?offer bsbm:deliveryDays ?deliveryDays .
  FILTER (?deliveryDays <= 3)
    ?offer bsbm:price ?price .
    ?offer bsbm:validTo ?date .
  FILTER (?date > %currentDate% ) }
ORDER BY ?price
LIMIT 10
```

Query Properties

- Medium amount of patterns
- Medium amount of filters

Query 11: Get all information about an offer

```
SELECT ?property ?hasValue ?isValueOf
WHERE {
   { %OfferXYZ% ?property ?hasValue }
UNION
   { ?isValueOf ?property %OfferXYZ% }
}
```

Query Properties

- query contains unbound predicates
- uses UNION

Query 12: Export information about an offer into another schemata

```
CONSTRUCT
 { %OfferXYZ% bsbm-export:product ?productURI .
   %OfferXYZ% bsbm-export:productlabel ?productlabel .
   %OfferXYZ% bsbm-export:vendor ?vendorname .
   %OfferXYZ% bsbm-export:vendorhomepage ?vendorhomepage .
   %OfferXYZ% bsbm-export:offerURL ?offerURL .
   %OfferXYZ% bsbm-export:price ?price .
   %OfferXYZ% bsbm-export:deliveryDays ?deliveryDays .
   %OfferXYZ% bsbm-export:validuntil ?validTo }
WHERE
 { %OfferXYZ% bsbm:product ?productURI .
   ?productURI rdfs:label ?productlabel .
   %OfferXYZ% bsbm:vendor ?vendorURI .
   ?vendorURI rdfs:label ?vendorname .
   ?vendorURI foaf:homepage ?vendorhomepage .
   %OfferXYZ% bsbm:offerWebpage ?offerURL .
   %OfferXYZ% bsbm:price ?price .
   %OfferXYZ% bsbm:deliveryDays ?deliveryDays .
   %OfferXYZ% bsbm:validTo ?validTo }
```

Query Properties

- Uses CONSTRUCT

The Test Driver

- **Executes sequences of parameterized**
  - SPARQL queries over the SPARQL protocol
  - SQL queries over JDBC
- **The query sequence is deterministic.**
- **The test driver can simulate multiple clients.**
- **Test driver configuration**
  - SPARQL endpoint URL
  - Number of warmup query mixes
  - Number of benchmark query mixes
  - Number of clients
- **The data generator and test driver are available under GPL license.**

3. Benchmark Experiment and Results

- **We were interested in comparing the performance of**
  - relational database to RDF wrappers and
  - native RDF stores.
- **3 RDF stores**
  - **Jena TDB** Version 0.53 and Joseki Version 3.2.1 as HTTP interface.
  - **Sesame** Version 2.2 (Native Storage) over Tomcat Version
  - **Virtuoso** Triple Store v5.0.9
- **2 SPARQL-to-SQL rewriters**
  - **D2R Server** Version 0.4 with MySQL Version 5.1.26 as underlying RDBMS.
  - **Virtuoso RDF Views** with Virtuoso v5.0.9 as underlying RDBMS.
- **2 RDBMS**
  - **MySQL Version** 5.1.26
  - **Virtuoso** v5.0.9

Setup of the Benchmark Experiment

- **Dataset sizes: 250,000 to 100,000,000 triples**
- **1 to 16 concurrent clients**
- **Test run**
  - Warm-up 32 BSBM query mixes (altogether 800 queries)
  - Performance measurement:128 BSBM query mixes (altogether 3200 queries)
  - Test driver and SUT on same machine
- **Machine**
  - DELL workstation
  - Processor: Intel Core 2 Quad Q9450 2.66GHz
  - Memory: 8GB DDR2 667
  - Hard disks: 160GB (10,000 rpm) SATA2, 750GB (7,200 rpm) SATA2
  - Operating system: Ubuntu 8.04 64-bit

Load Times

	250K	1M	25M	100M
Jena TDB	00:00:13	00:00:41	00:16:05	01:33:48
Sesame Native	00:00:19	00:03:33	12:43:22	3:06:48:45
Virtuoso TS	00:00:05	00:00:25	00:40:27	08:15:16
MySQL	00:00:02	00:00:16	00:02:09	00:10:28
Virtuoso SQL	00:00:09	00:00:33	00:15:31	00:58:59

- values in d:hh:mm:ss
- for loading the Turtle and relational representation

Overall Query Execution Time for a Single Client

	Jena TDB	Sesame Native	Virtuoso TS	D2R Server	Virtuoso RV	MySQL SQL	Virtuoso SQL
250 K	72.3	20.5	20.8	313.7	17.4	1.6	3.0
1 M	112.0	24.1	24.7	709.5	18.5	2.9	3.5
25 M	2886.8	512.3	179.7	n/a*	31.6	24.2	23.7
100 M	9562.8	1831.5	775.8	n/a*	163.0	85.1	127.6

* as D2R Server timed out on query 5.

- **in seconds**
- **128 Query mixes = 3200 queries**
- **Performance of SPARQL-to-SQL rewriters and RDF stores is similar for small datasets.**
- **Virtuoso RDF Views outperforms fastest triple store for 100M dataset by factor 5.**
- **Individual query times vary widely.**
  - 25% - 90 % of the overall time spend on query 5 (complex filter expression) and query 6 (regex).

Overall Query Execution Time Single Client without Q5 and Q6

	Jena TDB	Sesame Native	Virtuoso TS	D2R Server	Virtuoso RV	MySQL SQL	Virtuoso SQL
250K	34.3	15.2	19.3	64.1	16.5	1.2	2.6
1M	36.9	10.2	21.5	62.0	17.2	1.6	2.8
25M	297.9	194.2	112.3	61.1	22.1	2.5	13.7
100M	937.8	606.0	511.0	253.5	130.7	2.9	106.0

- **Sesame is the fastest triple store for small datasets.**
- **Virtuoso TS is the fastest triple store for large datasets.**
- **SPARQL-to-SQL rewriters outperform triple stores for large datasets.**
- **Relational databases clearly outperform triple stores.**

Performance by Individual Query (queries per second)

Query 1: Find products for a given set of generic features.

	250K	1M	25M	100M
Jena TDB	72.47	34.53	2.77	0.62
Sesame Native	**541.80**	**532.69**	67.13	13.65
Virtuoso TS	309.54	307.94	201.91	**86.54**
D2R Server	309.60	273.32	**231.17**	85.24
Virtuoso RV	273.99	266.41	215.90	72.23
MySQL SQL	*2738.69*	*2378.56*	*1058.55*	*563.28*
Virtuoso SQL	*1071.14*	*1004.35*	*764.07*	*157.89*

Query 2: Retrieve basic information about a specific product for display purposes.

	250K	1M	25M	100M
Jena TDB	48.06	57.47	46.09	37.41
Sesame Native	**198.96**	**195.12**	90.85	35.52
Virtuoso TS	85.57	76.53	59.45	45.21
D2R Server	24.35	25.58	26.87	29.96
Virtuoso RV	116.04	111.45	**101.86**	**54.34**
MySQL SQL	*2768.51*	*3269.26*	*3125.51*	*3701.70*
Virtuoso SQL	*1215.22*	*919.28*	*900.50*	*707.56*

Performance by Individual Query (queries per second)

Query 3: Find products having some specific features and not having one feature.

	250K	1M	25M	100M
Jena TDB	69.80	33.79	2.70	0.66
Sesame Native	**414.53**	**427.05**	57.37	12.57
Virtuoso TS	250.55	243.45	195.12	**83.80**
D2R Server	168.17	116.06	42.38	18.28
Virtuoso RV	261.91	256.80	**231.26**	62.60
MySQL SQL	*2638.38*	*2194.83*	*980.28*	*536.34*
Virtuoso SQL	*1029.19*	*900.23*	*778.75*	*154.46*

Query 4: Find products matching two different sets of features.

	250K	1M	25M	100M
Jena TDB	36.28	17.40	1.40	0.36
Sesame Native	**425.53**	**269.64**	49.03	10.21
Virtuoso TS	178.02	176.30	72.31	38.60
D2R Server	192.42	179.15	**142.08**	**68.48**
Virtuoso RV	166.44	165.54	139.21	49.99
MySQL SQL	*2425.36*	*2111.28*	*840.49*	*544.58*
Virtuoso SQL	*962.56*	*913.64*	*719.98*	*142.73*

Performance by Individual Query (queries per second)

Query 5: Find products that are similar to a given product.

	250K	1M	25M	100M
Jena TDB	3.38	1.72	0.06	0.02
Sesame Native	43.25	26.83	1.54	0.52
Virtuoso TS	153.27	111.19	13.07	5.64
D2R Server	0.52	0.20	timeout	timeout
Virtuoso RV	**225.69**	**132.91**	**28.54**	**8.40**
MySQL SQL	*733.45*	*343.71*	*24.77*	*9.25*
Virtuoso SQL	*345.11*	*185.63*	*14.52*	*7.72*

Query 6: Find products having a label that contains specific words.

	250K	1M	25M	100M
Jena TDB	153.25	85.15	0.32	0.08
Sesame Native	49.37	12.99	0.54	0.13
Virtuoso TS	174.24	56.91	2.20	0.53
D2R Server	35.81	16.46	1.58	0.33
Virtuoso RV	**401.24**	**264.18**	**23.53**	**6.04**
MySQL SQL	*551.70*	*111.49*	*7.57*	*1.86*
Virtuoso SQL	*2448.12*	*1341.07*	*95.81*	*25.05*

Performance by Individual Query (queries per second)

Query 7: Retrieve in-depth information about a product including offers and reviews.

	250K	1M	25M	100M
Jena TDB	84.67	151.82	10.26	5.74
Sesame Native	66.76	**237.83**	4.84	1.63
Virtuoso TS	122.85	115.41	18.01	2.08
D2R Server	44.77	63.22	55.80	3.90
Virtuoso RV	**134.16**	127.10	**79.18**	**8.05**
MySQL SQL	*1460.24*	*774.39*	*714.51*	*724.81*
Virtuoso SQL	*623.25*	*687.16*	*72.66*	*8.40*

Query 8: Give me recent English language reviews for a specific product.

	250K	1M	25M	100M
Jena TDB	143.33	139.98	11.28	9.50
Sesame Native	**308.24**	**330.65**	12.82	3.76
Virtuoso TS	185.77	173.88	42.03	4.68
D2R Server	31.33	26.26	29.96	**11.12**
Virtuoso RV	199.95	190.17	**124.44**	8.80
MySQL SQL	*2634.21*	*2712.51*	*582.99*	*414.54*
Virtuoso SQL	*1287.91*	*1241.05*	*76.28*	*8.85*

Performance by Individual Query (queries per second)

Query 9: Get information about a reviewer.

	250K	1M	25M	100M
Jena TDB	421.66	420.42	99.40	49.23
Sesame Native	**755.91**	**854.57**	52.87	15.63
Virtuoso TS	509.84	495.50	159.52	18.83
D2R Server	73.74	67.18	77.58	62.16
Virtuoso RV	617.14	610.13	**512.14**	**78.53**
MySQL SQL	*2757.45*	*3392.31*	*2750.76*	*2855.80*
Virtuoso SQL	*1730.49*	*1874.73*	*323.55*	*48.19*

Query 10: Get cheap offers which fulfill the consumer's delivery requirements.

	250K	1M	25M	100M
Jena TDB	355.41	358.71	18.63	8.60
Sesame Native	**382.46**	**379.98**	6.78	2.14
Virtuoso TS	212.85	142.24	5.12	1.72
D2R Server	133.91	148.45	148.89	4.95
Virtuoso RV	301.81	294.78	**238.37**	**116.84**
MySQL SQL	*3133.67*	*3378.01*	*2071.27*	*3353.90*
Virtuoso SQL	*1573.72*	*1519.07*	*1491.70*	*144.61*

Performance by Individual Query (queries per second)

Query 11: Get all information about an offer.

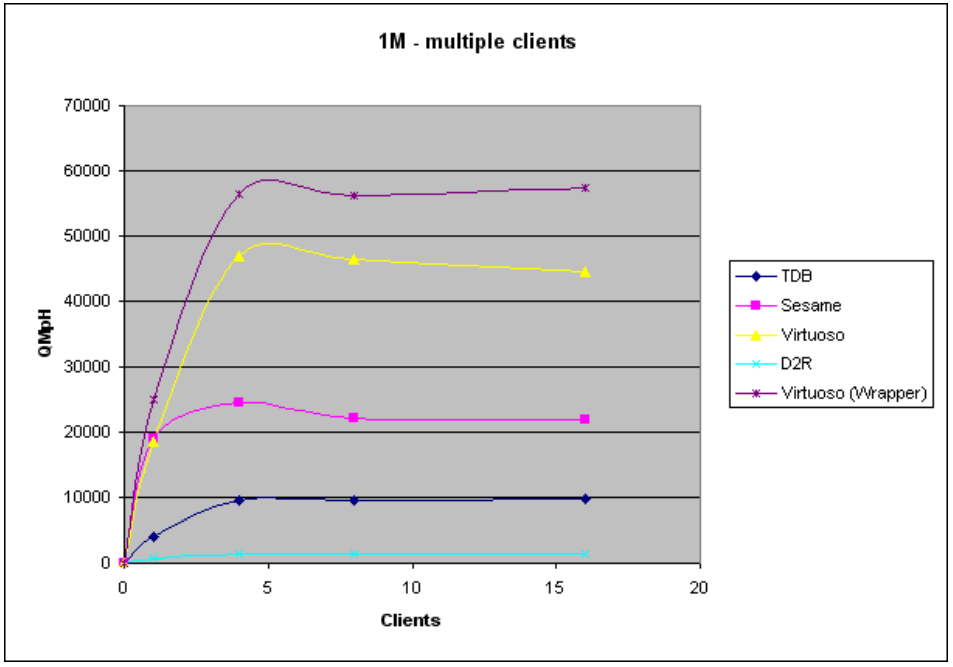
	250K	1M	25M	100M
Jena TDB	417.44	474.70	84.68	30.56
Sesame Native	*713.72*	*797.33*	49.75	13.23
Virtuoso TS	415.76	414.33	31.24	27.62
D2R Server	231.12	239.10	**205.61**	**114.21**
Virtuoso RV	156.04	156.92	135.54	41.54
MySQL SQL	*5144.62*	*5867.07*	*2215.38*	*7262.50*
Virtuoso SQL	*2193.30*	*2198.16*	*2438.44*	*2123.33*

Query 12: Export information about an offer into another schema.

	250K	1M	25M	100M
Jena TDB	284.36	290.30	71.64	41.11
Sesame Native	**588.37**	**672.37**	54.20	17.86
Virtuoso TS	209.12	208.00	31.70	23.68
D2R Server	202.75	192.52	205.02	**91.26**
Virtuoso RV	220.55	223.50	**208.99**	78.16
MySQL SQL	*4389.91*	*5120.33*	*4274.09*	*4998.41*
Virtuoso SQL	*1978.78*	*1899.96*	*2194.64*	*2402.51*

Results for Multiple Client (1M dataset)

Dataset size 1M	Number of clients			
	1	4	8	16
Jena TDB	4,114	9,566	9,479	9,651
Sesame Native	19,097	24,477	22,079	21,793
Virtuoso TS	18,604	46,987	46,479	44,685
D2R Server	649	1,354	1,312	1,359
Virtuoso RV	24,826	56,431	56,339	57,425
MySQL SQL	*155,823*	*362,323*	*382,366*	*389,798*
Virtuoso SQL	*129,849*	*216,349*	*261,576*	*293,339*



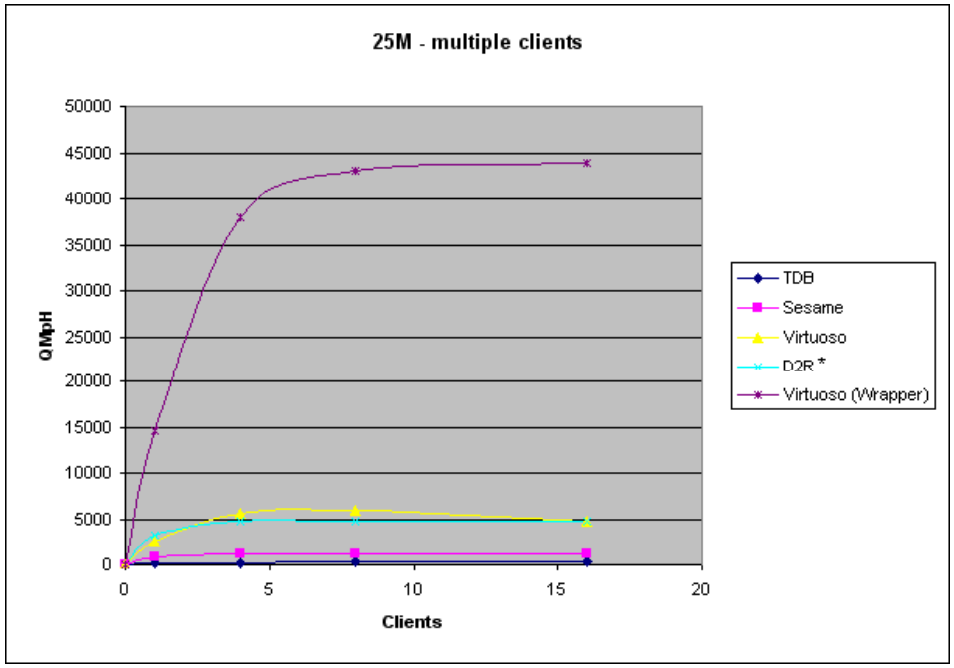
- in query mixes per hour.

Results for Multiple Client (25M dataset)

Dataset size 25M	Number of clients			
	1	4	8	16
Jena TDB	160	230	357	255
Sesame Native	899	1,108	1,190	1,192
Virtuoso TS	2,563	5,595	6,004	4,759
D2R Server	n/a*	n/a*	n/a*	n/a*
Virtuoso RV	14,543	38,044	42,980	43,826
MySQL SQL	18,986	38,208	38,654	39,006
Virtuoso SQL	19,375	38,515	49,618	49,836

* as D2R Server timed out on query 5.

- in query mixes per hour



Conclusions from the Experiment

- **Concerning Triple Stores**
  - Sesame is good for small and medium data sets.
  - Virtuoso TS is faster for large datasets.
- **Concerning Triple Stores vs. SPARQL-to-SQL rewriters**
  - SPARQL-to-SQL rewriters clearly outperform triple stores.
- **Concering Systems for Mutiple Clients**
  - Virtuoso TS and Virtuoso RV clearly outperform the other systems.
- **Concerning Triple Stores vs. Relational Databases**
  - Relational databases outperform triple stores by at least factor 9.

Thanks!

Berlin SPARQL Benchmark (BSBM) Website

http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/