The
Berlin SPARQL Benchmark

Christian Bizer
Andreas Schultz

Freie Universität Berlin

Overview

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

1. The SPARQL Query Language for RDF

- **Flexible query language for the RDF data model**
- **W3C Recommendation since 15 January 2008**
- **Example Query**

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

- **Query Result**

?name	?mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

SPARQL Implementations

- **There are currently 22 SPARQL implementations**
  - http://esw.w3.org/topic/SparqlImplementations
- **Example RDF stores**
  - Sesame
  - Jena SDB, TDB
  - OpenLink Virtuoso
  - SemWeb .Net Library
- **Relational database to RDF wrappers**
  - D2R Server
  - OpenLink Virtuoso
  - See also: W3C RDB 2 RDF Incubator Group
- **Federated SPARQL Query Engines**
  - DARQ
  - The Semantic Discovery System

The SPARQL Protocol for RDF

- **for sending SPARQL queries to an SPARQL endpoint**
  - HTTP Binding
  - SOAP Binding
- **W3C Recommendation since 15 January 2008**
- **Example Request**

```
http://dbpedia.org/sparql?default-graph-
uri=http%3A%2F%2Fdbpedia.org&query=SELECT+%3Ffilm%0D
%0AWHERE+%7B+%3Ffilm+%3Chttp%3A%2F%2Fwww.w3.org%2F20
04%2F02%2Fskos%2Fcore%23subject%3E+%3Chttp%3A%2F%2Fd
bpedia.org%2Fresource%2FCategory%3AFrench_films%3E+%
7D&format=text%2Fhtml
```

Public SPARQL Endpoints on the Web

- **The ESW Wiki lists 24 public SPARQL endpoints**
  - http://esw.w3.org/topic/SparqlEndpoints
- **Examples**
  - BBC Backstage
  - DBpedia
  - DBtune
  - DOAP Space
  - Linked Movie Data Base
  - DBLP Bibliography
  - Revyu
  - GovTrack.us
  - Project Gutenberg Metadata
  - Gene Ontology Database

2. Design of the Berlin SPARQL Benchmark (BSBM)

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, …

DBpedia Benchmark

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

SP²Bench

- from the databases group at Freiburg University, Germany
- uses an synthetic, scaleable version of the DBLP bibliography dataset
- queries are designed for the comparison of different RDF store layouts
- queries are not designed towards realistic workloads

Design Goals of the BSBM

1. **test storage systems with realistic workloads of use case motivated queries**
2. **allow the comparison of storage systems across internal data models**
   - native RDF stores
   - relational database to RDF wrappers
   - other data sources (OODBs or XML repositories)
3. **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.**
- **Products**
  - have varying number of textual and numeric properties
  - have varying number of product features
  - words for product names and product descriptions are randomly chosen from a dictionary
- **Relations: Products – Producers, Products – Offers, Offers – Vendors, Products – Reviews**
  - normal distributions with different parameters
- **Products are in a product hierarchy**
  - the hierarchy grows with dataset size

Data Generator

- **supports the creation of arbitrarily large datasets using the number of products as scale factor.**
- **Output formats**
  - Turtle
  - N-Triples
  - XML
  - MySQL dump
- **The XML and the relational representation can be used to compare stores across internal data models.**
- **The data generations is deterministic.**
- **The data generator is published under GPL license.**

Number of Instances in BSBM Datasets of different Sizes

	50K	250K	1M	5M	25M	100M
Number of Products	91	477	1,915	9,609	48,172	194,207
Number of Producers	2	10	41	199	974	3,844
Number of Reviews	2,275	11,925	47,875	240,225	1,204,300	4,855,175
Number of Offers	1,820	9,540	38,300	192,180	963,440	3,884,140
Number of Vendors	2	10	39	196	961	3,912
Number of Reviewers	116	622	2,452	12,351	61,862	248,730
Number of Product Features	580	580	1,390	3,307	3,307	10,931
Number of Product Types	13	13	31	73	73	411
Total Number of Instances	*4,899*	*23,177*	*92,043*	*458,140*	*2,283,089*	9,201,350
Exact Total Number of Triples	50,116	250,492	1,000,226	5,000,453	25,000,557	100,001,402
File Size N-Triple (unzipped)	*14 MB*	*70,7 MB*	*284,4 MB*	*1,4 GB*	*7 GB*	28,2 GB

The Benchmark Query Mix

- Emulates the search and navigation pattern on a consumer looking for a product.
- Queries form a sequence in order to simulate realistic workloads.

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 2: Retrieve basic information about a specific product for display purposes.
8. Query 4: Find products matching two different sets of features.
9. Query 2: Retrieve basic information about a specific product for display purposes.
10. Query 2: Retrieve basic information about a specific product for display purposes.
11. Query 5: Find products that are similar to a given product.
12. Query 7: Retrieve in-depth information about a product including offers and reviews.
13. Query 7: Retrieve in-depth information about a product including offers and reviews.
14. Query 6: Find products having a label that contains specific words.
15. Query 7: Retrieve in-depth information about a product including offers and reviews.
16. Query 7: Retrieve in-depth information about a product including offers and reviews.
17. Query 8: Give me recent English language reviews for a specific product.
18. Query 9: Get information about a reviewer.
19. Query 9: Get information about a reviewer.
20. Query 8: Give me recent English language reviews for a specific product.
21. Query 9: Get information about a reviewer.
22. Query 9: Get information about a reviewer.
23. Query 10: Get cheap offers which fulfill the consumer's delivery requirements.
24. Query 10: Get cheap offers which fulfill the consumer's delivery requirements.
25. Query 10: Get cheap offers which fulfill the consumer's delivery requirements.

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 patters
- 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 patters
- 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 specific words

```
SELECT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product rdf:type bsbm:Product .
  FILTER regex(?label, "%word1%|%word2%|%word3%")}
```

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

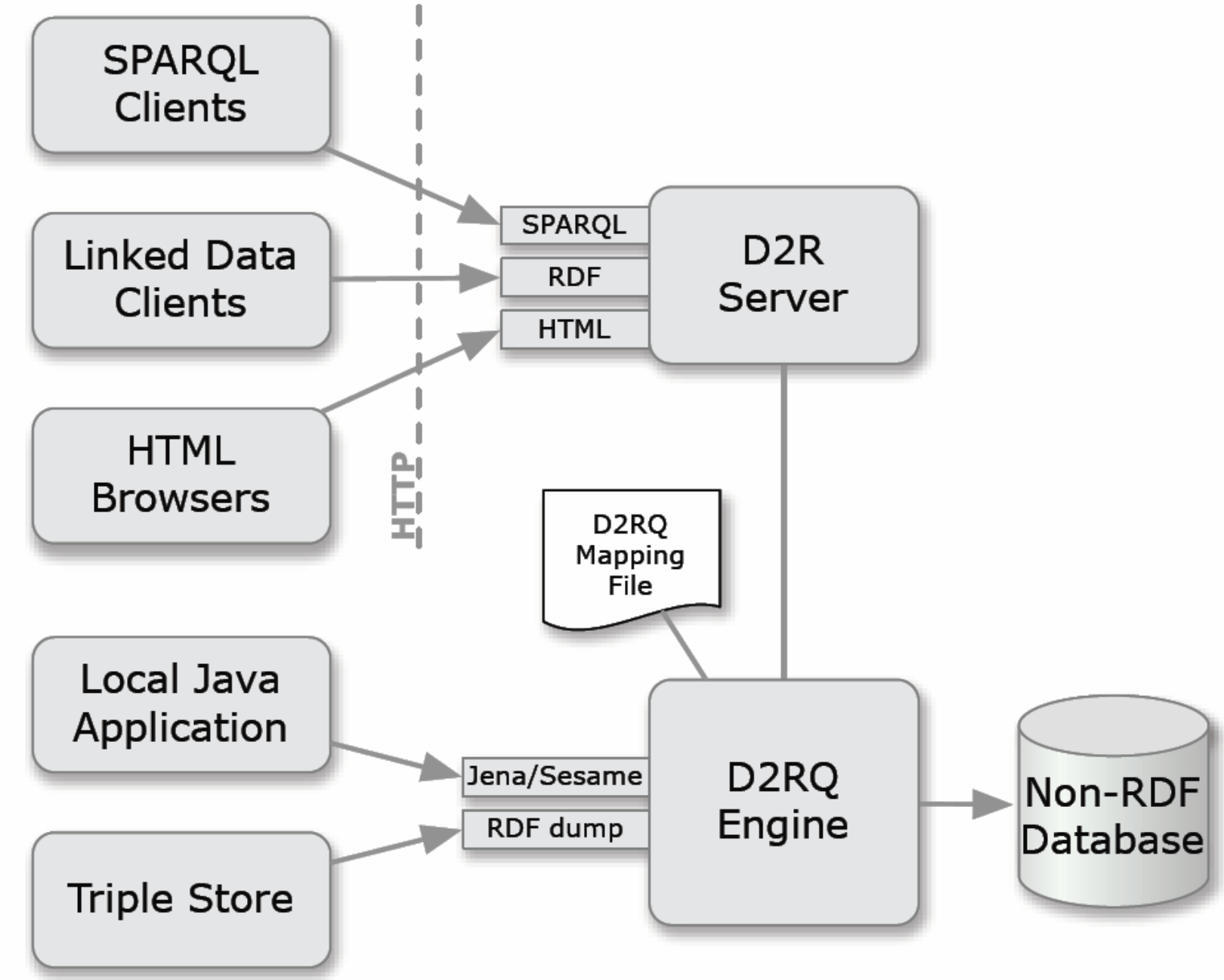
The Test Driver

- **executes sequences of parameterized SPARQL queries over the SPARQL protocol against the system under test.**
- **Test driver configuration**
  - SPARQL endpoint URL
  - number of warmup query mixes
  - Number of benchmark query mixes
- **The query sequences are deterministic.**
- **The test driver outputs detailed reports.**

3. Benchmark Experiment and Results

- **We were interested in comparing the performance of relational database to RDF wrappers with the performance of native RDF stores.**
- **We ran the benchmark against:**
- **3 RDF stores**
  - **Jena SDB** Version 1.1 (Hash Storage Layout) with MySQL Version 5.0.51a-3 as underlying RDMS and Joseki Version 3.2 (CVS) as HTTP interface
  - **Sesame** Version 2.2-Beta2 (Native Storage) with Tomcat Version 5.5.25.5 as HTTP interface
  - **Virtuoso** Open-Source Edition v5.0.6 (Native Storage)
- **1 relational database to RDF wrapper**
  - **D2R Server** Version 0.4 (Standalone Setup) with MySQL Version 5.0.51a-3 as underlying RDMS.

D2R Server Architecture



Setup of the Benchmark Experiment

- **Dataset sizes: 50K triples to 25M triples**
- **Test run**
  - 50 BSBM query mixes (altogether 1250 queries)
  - Warm-up 10 BSBM query mixes (altogether 250 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

	50K	250K	1M	5M	25M
Jena SDB	5	24	116	1053	13306
Sesame	3	18	132	1988	27674
Virtuoso	2	33	87	609	49096
D2R Server	0.4	2	6	38.0	202

- values in seconds
- for loading the N-Triples and relational representation
- 30,000 seconds are 8.3 hours

Overall Query Execution Time

	Sesame	Virtuoso	Jena SDB	D2R Server
50 K	9.4	162.0	23.4	66.4
250 K	28.5	162.8	72.9	153.4
1 M	115.2	201.4	268.0	484.4
5 M	569.1	476.8	1406.6	2188.1
25 M	3038.205	2089.118	7623.958	not applicable

- **50 Query mixes = 1250 queries**
- **Individual query times vary widely**

Results by Query

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

	50K	250K	1M	5M	25M
Sesame	**0.002106**	**0.002913**	**0.003904**	**0.005051**	**0.035575**
Virtuoso	0.017650	0.017195	0.015160	0.020225	0.082935
SDB	0.002559	0.004955	0.012592	0.057347	0.328271
D2R	0.004810	0.011643	0.038524	0.195950	0.968688

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

	50K	250K	1M	5M	25M
Sesame	**0.005274**	**0.004904**	**0.005523**	**0.006115**	**0.017859**
Virtuoso	0.071556	0.069347	0.069178	0.077868	0.085244
SDB	0.023058	0.024708	0.042374	0.071731	0.446813
D2R	0.038768	0.041448	0.047844	0.056018	0.041420

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

	50K	250K	1M	5M	25M
Sesame	**0.002364**	**0.003793**	**0.004681**	**0.010618**	**0.049690**
Virtuoso	0.009790	0.008394	0.008904	0.011443	0.055770
SDB	0.005324	0.006671	0.014379	0.059678	0.317215
D2R	0.007125	0.021041	0.054744	0.226272	1.080856

Results by Query

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

	50K	250K	1M	5M	25M
Sesame	**0.002595**	**0.004023**	**0.005491**	**0.009803**	**0.071493**
Virtuoso	0.016189	0.012962	0.013824	0.020450	0.100784
SDB	0.005694	0.008284	0.023286	0.112666	0.679248
D2R	0.008209	0.021871	0.075406	0.390003	1.935566

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

	50K	250K	1M	5M	25M
Sesame	**0.057673**	**0.310220**	1.313046	6.688869	32.895117
Virtuoso	0.221498	0.380958	**0.846915**	**3.344963**	**1.623720**
SDB	0.129683	0.509083	1.741927	8.199694	43.878071
D2R	0.718670	2.373025	8.639514	41.823326	time out

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

	50K	250K	1M	5M	25M
Sesame	0.004993	0.014792	0.064516	0.290293	1.483277
Virtuoso	0.003639	0.007368	0.020584	**0.109199**	**0.536659**
SDB	**0.003332**	**0.005291**	**0.014744**	0.254799	1.197798
D2R	0.009314	0.013181	0.038695	0.144180	0.383284

Results by Query

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

	50K	250K	1M	5M	25M
Sesame	**0.005265**	**0.005510**	**0.010948**	**0.007205**	0.505330
Virtuoso	0.125126	0.107109	0.125771	0.310843	2.290610
SDB	0.020116	0.078490	0.402448	2.193056	13.129511
D2R	0.028352	0.030421	0.068561	0.069431	**0.055678**

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

	50K	250K	1M	5M	25M
Sesame	**0.005111**	**0.005286**	**0.005770**	**0.006150**	0.315689
Virtuoso	0.026703	0.023340	0.024814	0.028546	4.535478
SDB	0.046099	0.159153	0.660130	3.535463	20.466382
D2R	0.058440	0.069917	0.078460	0.094175	**0.066114**

Query 9: Get information about a reviewer.

	50K	250K	1M	5M	25M
Sesame	0.010526	0.040371	0.202310	1.071040	5.728123
Virtuoso	0.039994	0.039972	0.039744	0.042302	0.106834
SDB	**0.006450**	**0.004545**	**0.004461**	**0.004560**	0.018882
D2R	0.016164	0.015216	0.015215	0.026702	**0.017857**

Results by Query

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

	50K	250K	1M	5M	25M
Sesame	**0.002722**	**0.002589**	**0.003027**	**0.002814**	0.179580
Virtuoso	0.585728	0.569584	0.642199	1.338361	6.708351
SDB	0.009257	0.033867	0.102852	1.028677	3.141508
D2R	0.005088	0.005209	0.004926	0.005565	**0.017070**

Conclusions from the experiment

- **Sesame is good for small and medium dataset sizes**
- **Virtuoso is getting better with increasing dataset size**
- **D2R Server**
  - has problems with query 5 and needs improvements here
  - is getting competitive with increasing dataset size

Results from OpenLink for Virtuoso

- **Benchmark setup**
  - 100M triples dataset
  - 2GHz dual-Xeon 5130 (8 cores) with 8G RAM, 64-bit Linux
- **Result**
  - *Physical Triples:* 1297 qmph
  - *Mapped Triples:* **3144 qmph**
- **OpenLink currently improves Virtuoso for the BSBM benchmark**
  - “If Chris Bizer et al launched the mapping ship, we will be the ones to pilot it to harbor!”
  - “We expect in the end mapping will lead RDF warehousing by a factor of 4”
- **Source**
  - http://www.openlinksw.com/weblog/oerling/?id=1409

Results from Andy Seaborne for Jena TDB

- **The TDB storage engine can be used as an alternative to SDB within Jena.**
- **Benchmark Setup**
  - 4 CPU: AMD Opteron(tm) Processor 280 @1.8GHz, memory: 10Gbytes, shared disk array in a data center
- **Benchmark Results**
- **Conclusion**
  - TDB is faster than SDB
  - But still slower than Virtuoso
- **Source**
  - http://lists.w3.org/Archives/Public/public-sparql-dev/2008JulSep/0029.html

	25M	100M
Query 1	0.442776	1.463148
Query 2	0.046882	0.056338
Query 3	0.410103	1.429356
Query 4	0.654449	2.571348
Query 5	21.729141	84.927111
Query 6	0.524726	11.309403
Query 7	0.711938	0.771399
Query 8	0.798117	0.834888
Query 9	0.032256	0.041381
Query 10	0.187552	0.197450

Results from Josh Tauber for the SemWeb .NET Library

- **Benchmark Setup**
  - Intel Core2 Duo at 3.00GHz, 2 GB RAM, 32bit Ubuntu 8.04
- **Benchmark Results**

	1M	25M
Query 1	0.019184	0.049200
Query 2	0.051187	0.048590
Query 3	0.030508	0.079187
Query 4	0.032693	0.075603
Query 5	0.172283	0.342828
Query 6	0.102105	3.277656
Query 7	0.256491	1.108414
Query 8	0.175357	0.572258
Query 9	0.059674	0.088451
Query 10	0.089215	0.322246

- **Conclusion**
  - roughly comparable to Jena SDB and Virtuoso 5.0.6
- **Source**

  http://lists.w3.org/Archives/Public/public-lod/2008Aug/0073.html

100M Dataset Experiment on our Machine

1. New Version of Virtuoso 5.0.7 (not BSBM-optimized 5.0.8 yet)
2. D2R Server with better MySQL configuration (more indexes)
3. Equivalent SQL queries against relational representation

	mysql	Virtuoso	D2R opt
Query 1	0.021092	0.033174	0.021249
Query 2	0.000412	0.069636	0.031386
Query 3	0.016938	0.028237	0.113710
Query 4	0.022807	0.045437	0.026987
Query 5	3.712385	6.312297	0.000000
Query 6	0.491348	2.033258	1.611730
Query 7	0.240195	1.116620	0.561890
Query 8	0.113520	0.586523	0.409793
Query 9	0.029887	0.096961	0.031321
Query 10	0.073800	1.055452	0.206961

- Overall time 50 mixes: MySQL 289s D2R 290s* Virtuoso 906 s

*without query 5.

Initial Interpretation (My Current Guess)

For bigger RDF datasets (100M+)

1. **RDF stores are three times slower than relational databases**
2. **SPARQL against RDF mapped databases is**
   - **slower than SQL**
   - **but outperforms all triple stores**

Next step: Validate this guess by rerunning the benchmark for all stores on our machine.

Thanks!

Berlin SPARQL Benchmark (BSBM) Website

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