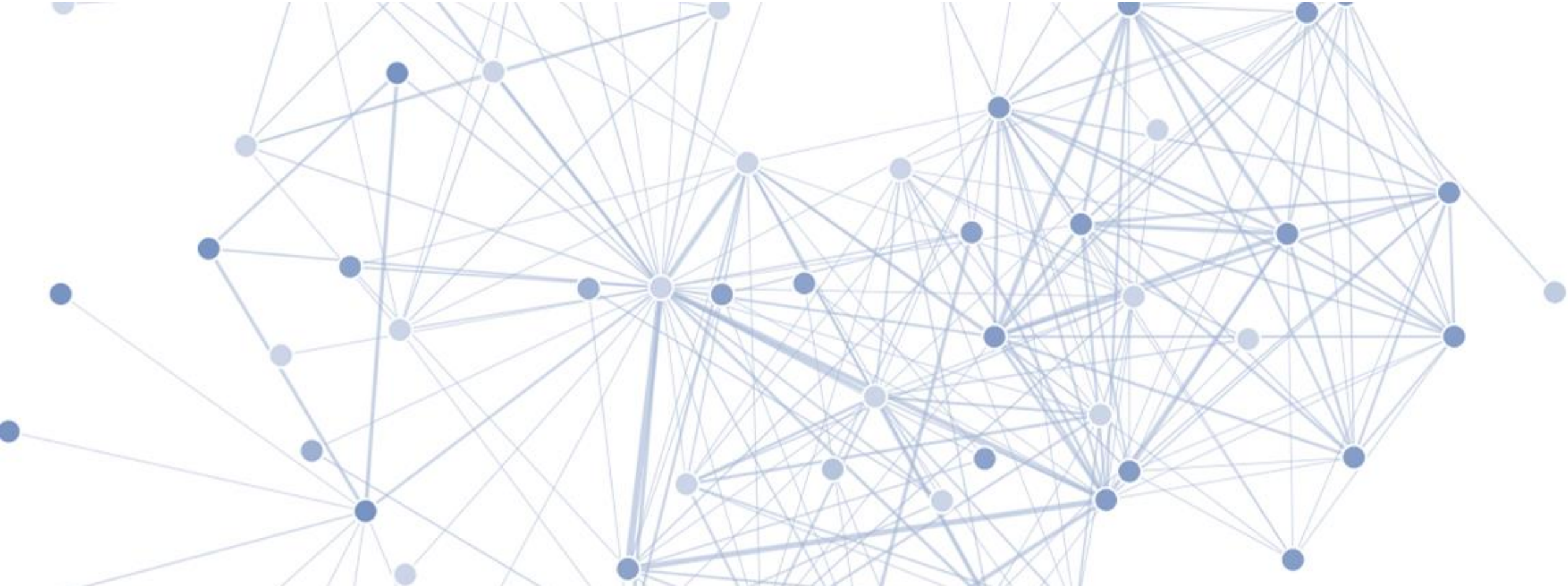


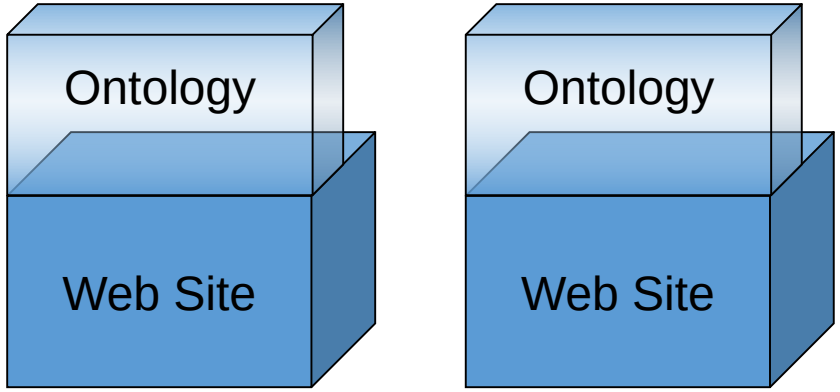
A MapReduce Approach to NoSQL RDF Databases

Albert Haque
Research in Bioinformatics and Semantic Web Lab
University of Texas at Austin
November 25, 2013

What is the Semantic Web?

- Semantic Web – Structured graphical representation of anything on the Internet
- Vision: We want (artificially) intelligent access to the contents of the Internet

- How? Every website associates with an ontology that:
  - Structures and describes the website contents
  - Provides links and tags the content



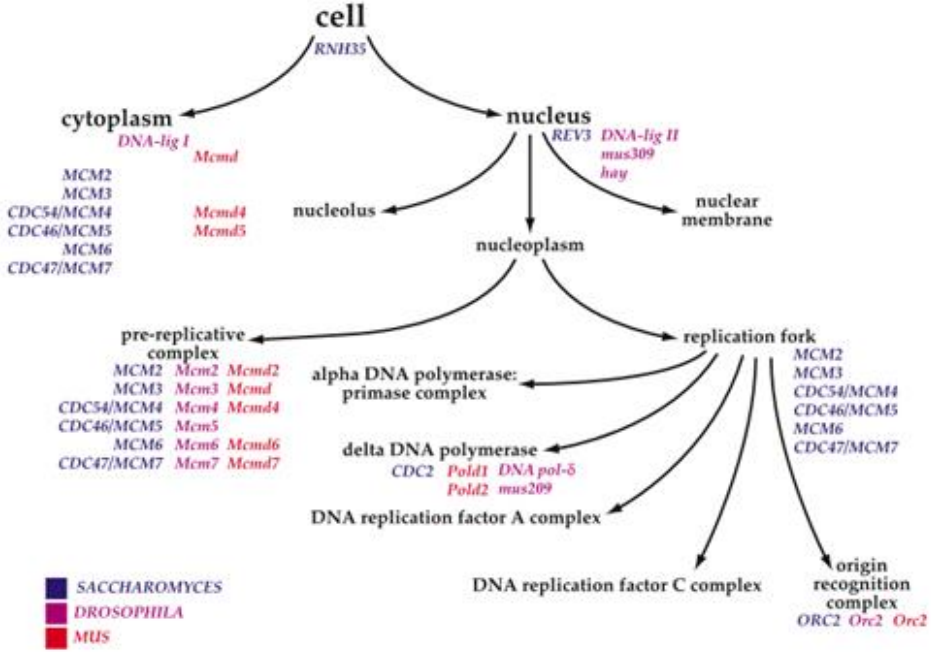
What is an Ontology?

- Human vocabulary (particularly synonyms) are problematic for machines
  - University of Texas at Austin := {“UT”, “UT Austin”, “University of Texas”, “Texas”}
- **Ontology – hierarchical representation of a vocabulary**

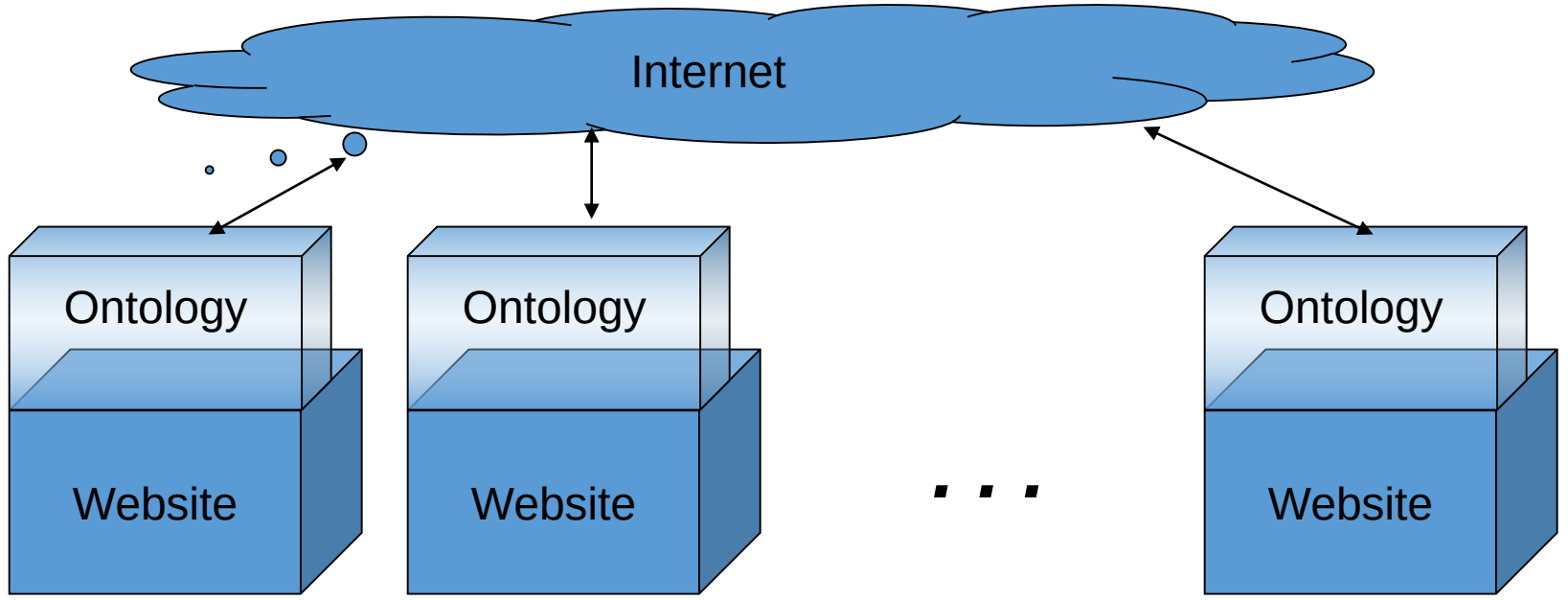
Ontology in Tabular Form

- GO:0003673 : Gene_Ontology (59650)
  - GO:0008150 : biological_process (41074)
    - GO:0007610 : behavior (344)
    - GO:0000004 : biological_process unknown (7292)
    - GO:0007154 : cell communication (7458)
      - GO:0007155 : cell adhesion (526)
        - GO:0016337 : cell-cell adhesion (163)
          - GO:0016339 : calcium-dependent cell-cell adhesion (30)
            - GO:0000501 : flocculation (sensu Saccharomyces) (0)
            - GO:0046586 : regulation of calcium-dependent cell-cell adhesion (1)
              - GO:0046588 : negative regulation of calcium-dependent cell-cell adhesion (1)
                - GO:0046587 : positive regulation of calcium-dependent cell-cell adhesion (0)
          - GO:0016338 : calcium-independent cell-cell adhesion (1)
          - GO:0007157 : heterophilic cell adhesion (66)
          - GO:0007156 : homophilic cell adhesion (37)
        - GO:0007160 : cell-matrix adhesion (64)
        - GO:0000128 : flocculation (6)
        - GO:0030155 : regulation of cell adhesion (28)
          - GO:0007162 : negative regulation of cell adhesion (18)
            - GO:0046588 : negative regulation of calcium-dependent cell-cell adhesion (1)
          - GO:0046586 : regulation of calcium-dependent cell-cell adhesion (1)
            - GO:0046588 : negative regulation of calcium-dependent cell-cell adhesion (1)
              - GO:0046587 : positive regulation of calcium-dependent cell-cell adhesion (0)

Ontology in Graphical Form



What is the Semantic Web?



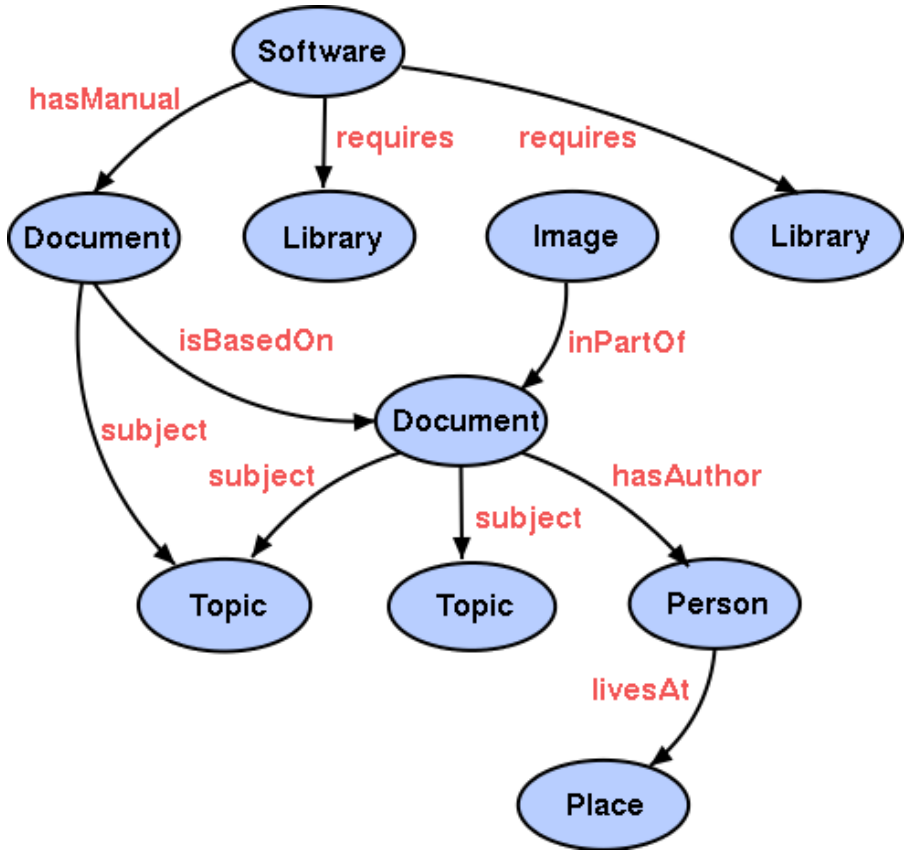
- How is this different from Google?
  - Current Internet search is done by exploring the link graph and keyword frequency
  - In 2012, Google released “Knowledge Graph” – Semantic Web

What is Linked Data and RDF?

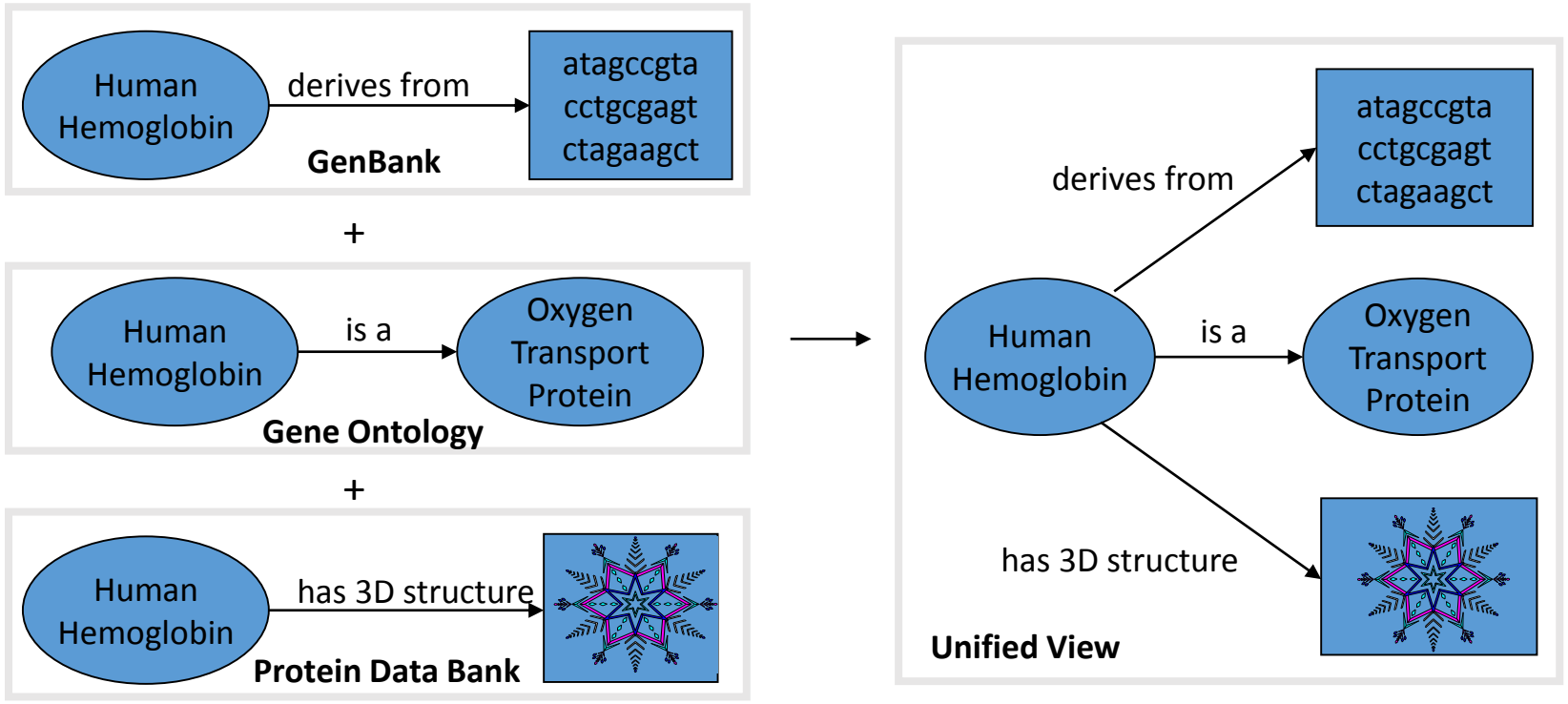
- Linked Data
  - Data on the Internet expressed as graphs
  - Focus is on data integration
  - In the end, everything becomes RDF

- Resource Description Framework
  - Directed labeled graph
  - Edges are called “triples”
  - (*Subject, Predicate, Object)*
  - *(Albert, livesIn, Austin)*
  - *(Austin, hasPopulation, 850000)*

Sample Graph of Linked Data



Linked Data + Data Integration Is Powerful



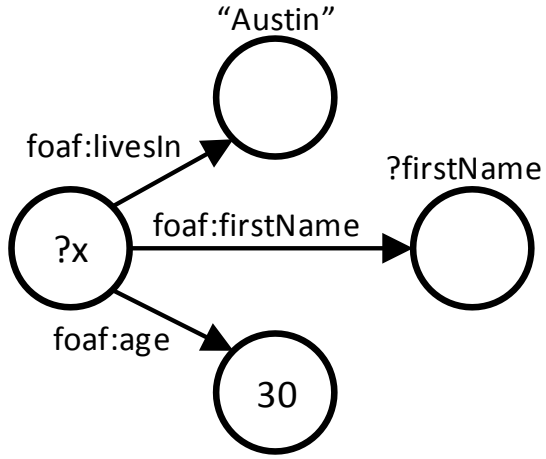
Triples contain URIs (uniform resource identifier)

<http://data.bgs.ac.uk/id/EarthMaterialClass/RockName/AKSED> <http://www.w3.org/2004/02/skos/core#definition> "ANKERITE SEDIMENTS"@en .

How Do We Query Linked Data?

- Answer: **SPARQL** – SPARQL Protocol and RDF Query Language
- User describes a graph using triple patterns
- SPARQL will match it with RDF data on the Semantic Web and return subgraphs
- SPARQL has expressive power as powerful as relational algebra

Sample SPARQL Query Graph



Sample SPARQL Query

```
SELECT ?firstName
WHERE {
          ?x foaf:firstName ?firstName .
          ?x foaf:age 30 .
          ?x foaf:livesIn "Austin"
}
```

Angles, Renzo, and Claudio Gutierrez. "The expressive power of SPARQL." *The Semantic Web-ISWC 2008*. Springer Berlin Heidelberg, 2008. 114-129.

How Much Data?

Dbpedia (Wikipedia)
- 4 million “things”
- 2.46 billion triples

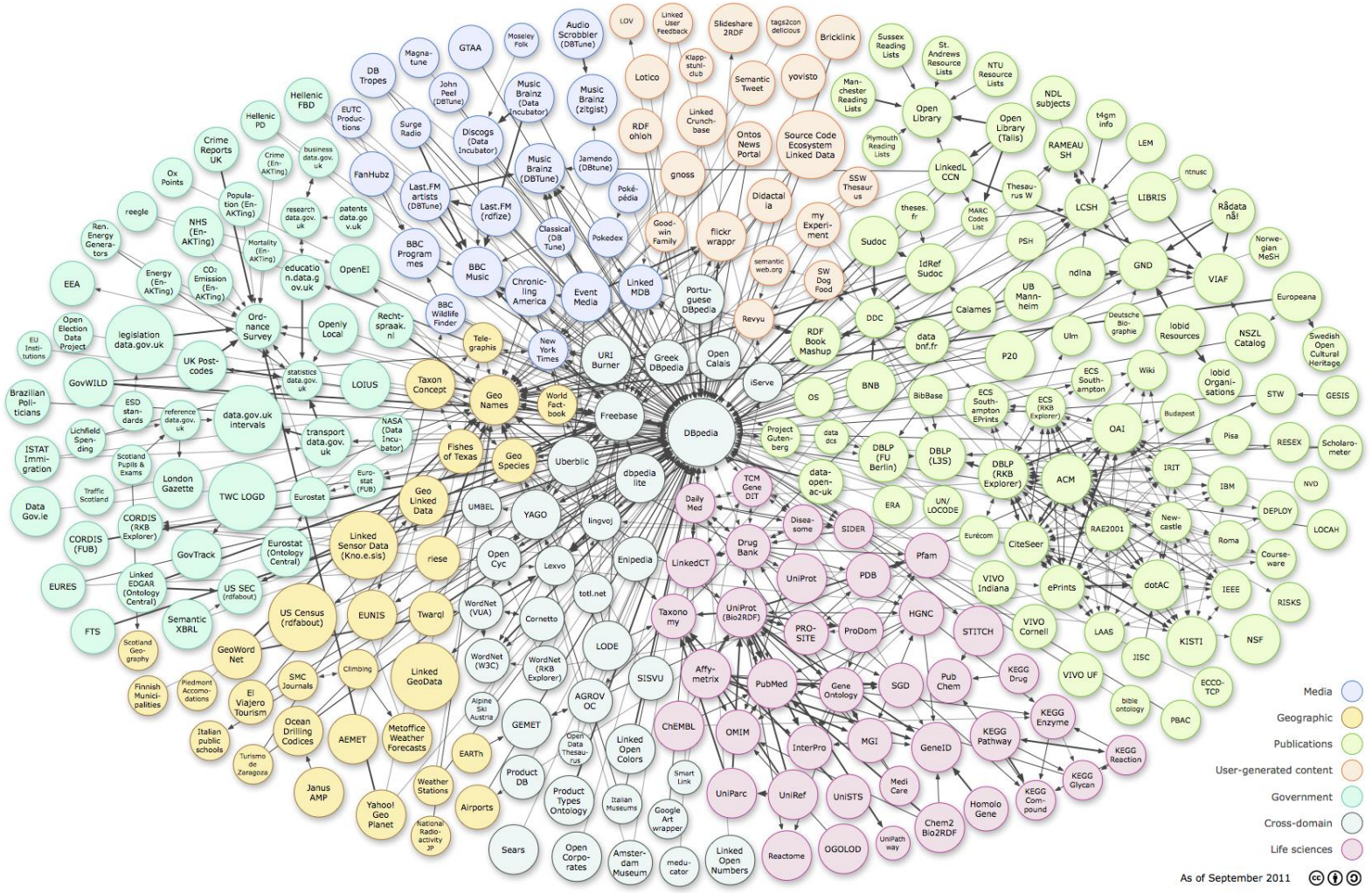
Bio2RDF
- 2.7 billion triples

Crawled Web Data
- 3.2 billion triples (2010)

U.S. Government Data
- 5+ billion triples

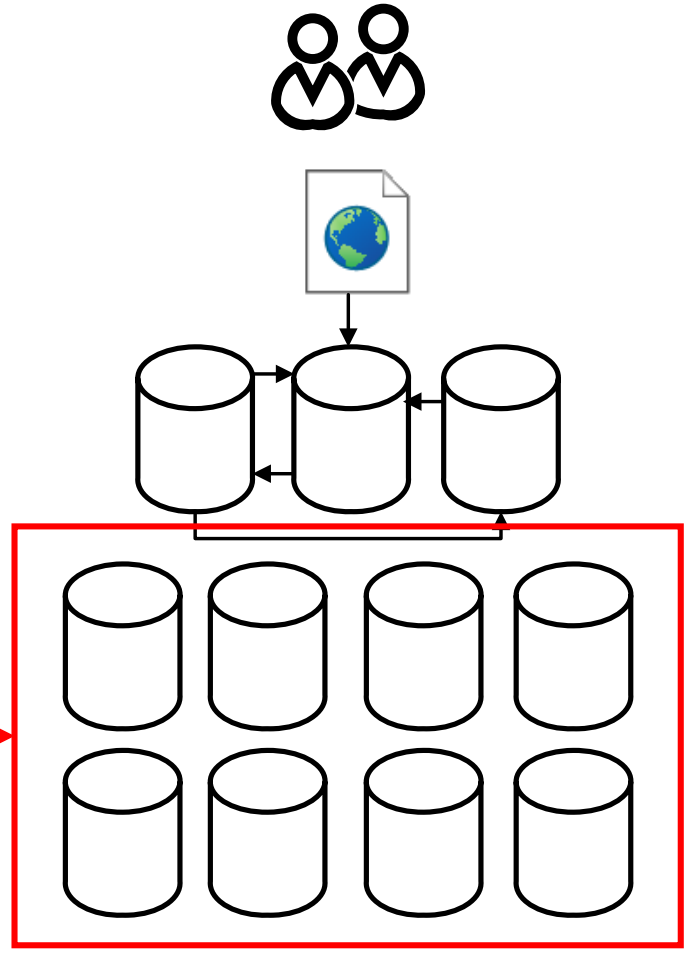
Weather Sensor Readings
- 1.7 billion triples

The Web of Linked Data



More data is stored in the deep web

- Deep Web – unindexed and unsearchable part of the Internet (e.g. dynamic webpages)
- Deep web contains ~**500 times** more data than the static web
- Imagine the data size if someone was to index the deep web!

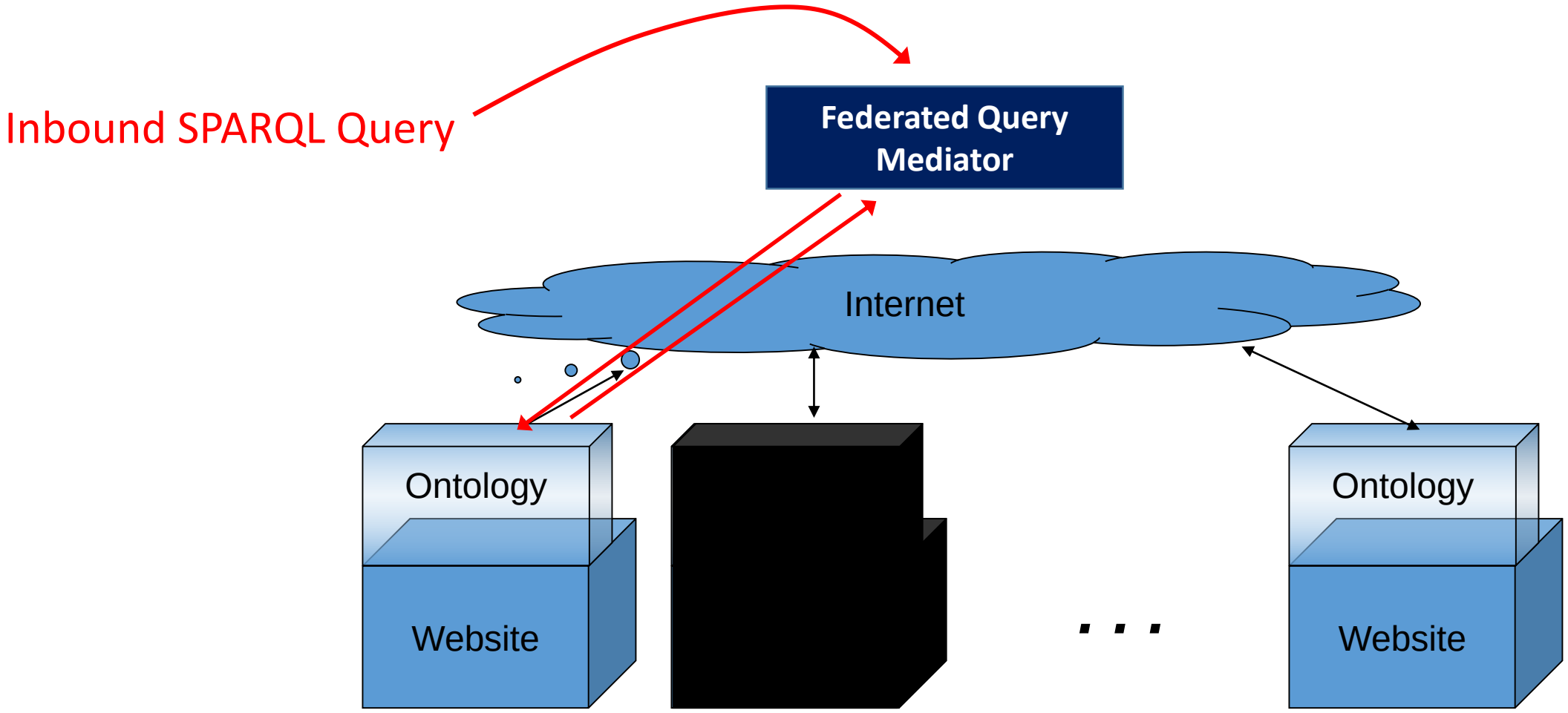


Bergman, Michael K. "The deep web: Surfacing hidden value." *journal of electronic publishing* 7.1 (2001): 07-01.

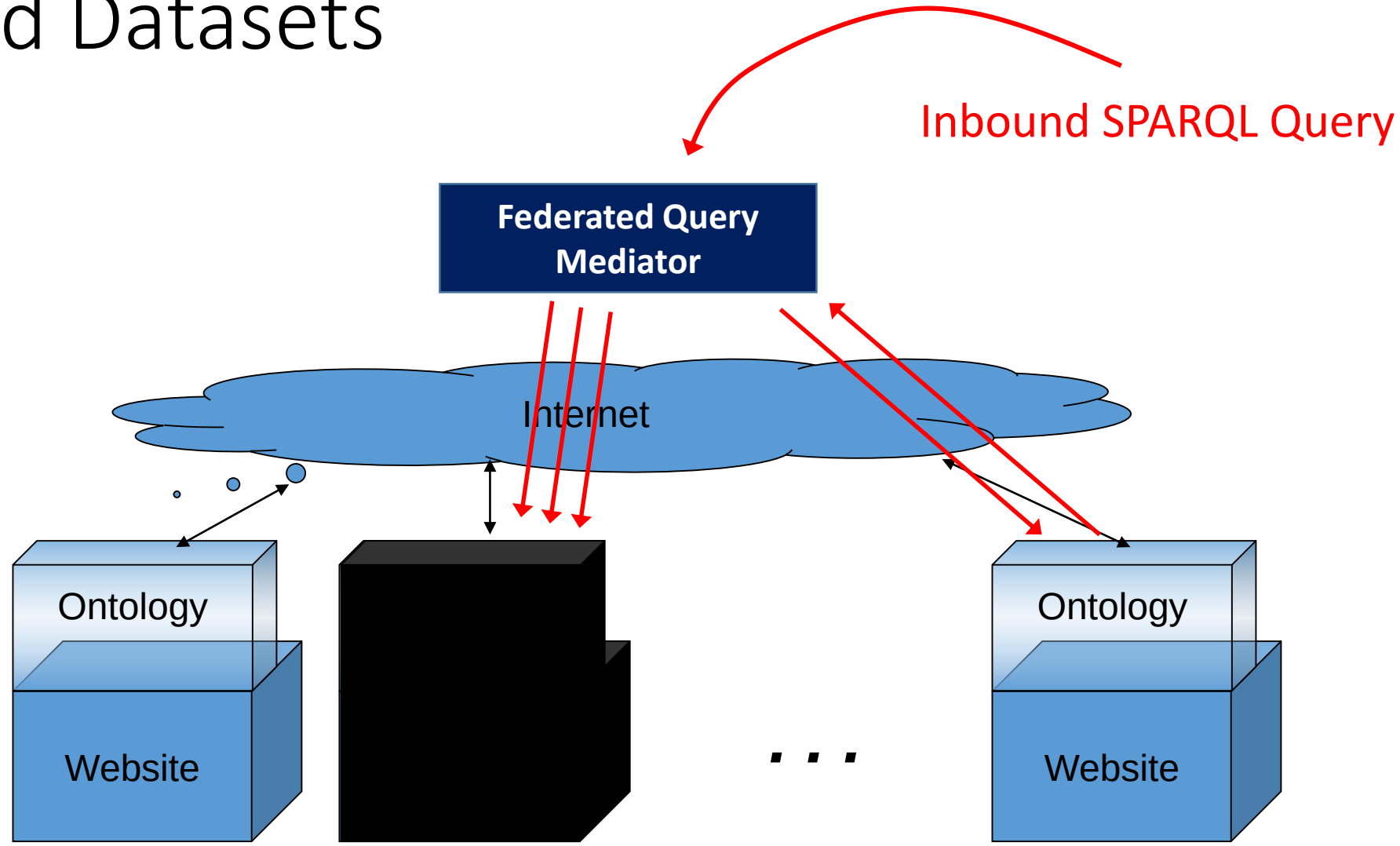
How is RDF Data Stored?

1. Distributed Datasets (Span multiple databases/endpoints)
2. Single Point-of-Access (Centralized Cache)

Distributed Datasets



Distributed Datasets

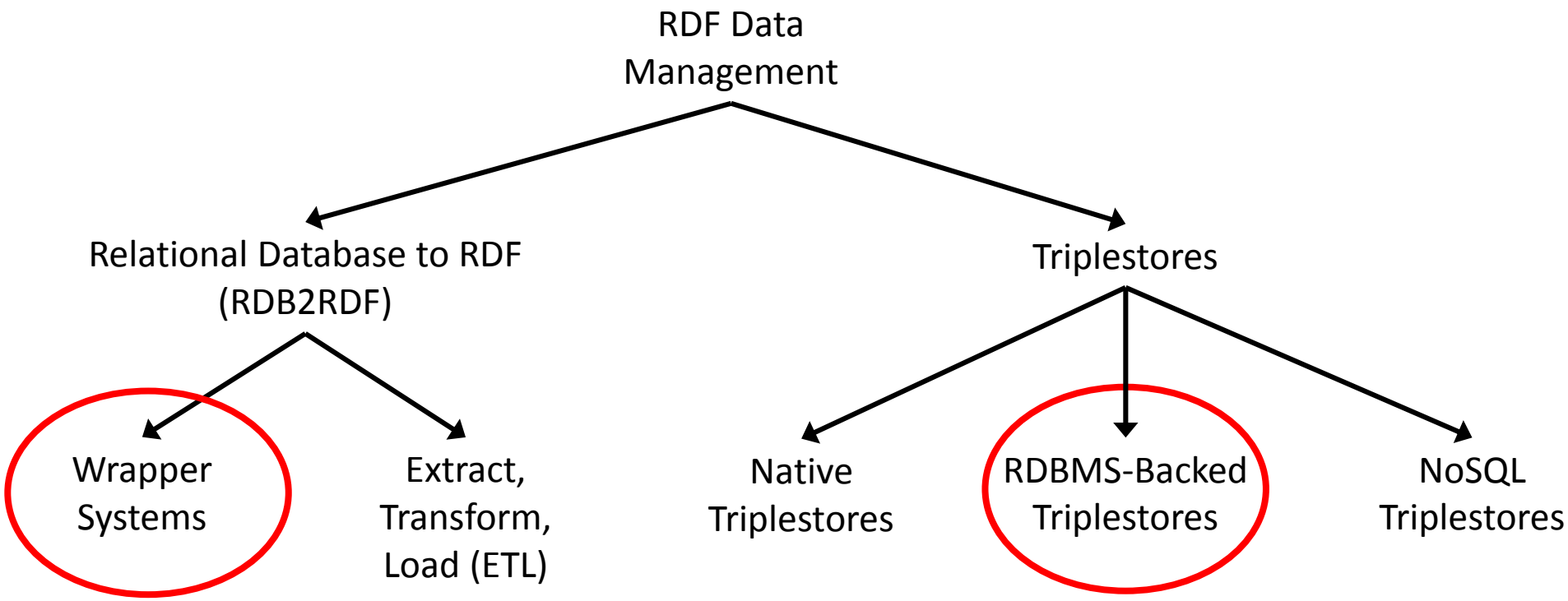


How is RDF Data Stored?

1. **Distributed Datasets (Span multiple databases/endpoints)**
   - Queries may visit several endpoints/databases before finding result
   - Longer query runtimes

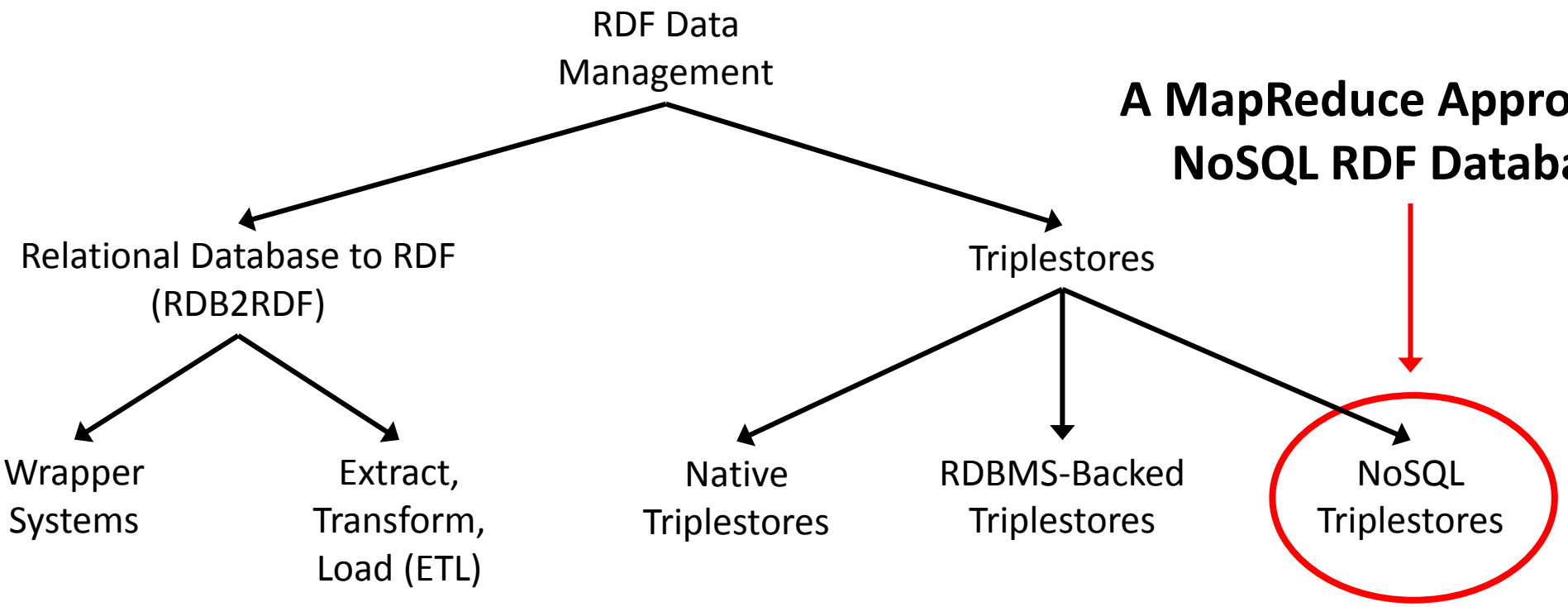
2. **Single Point-of-Access (Centralized Cache)**
   - Not dependent on other data sources
   - Must be highly scalable since storing all RDF data
   - Open research problems: cache coherence and transactions
   - Example: DBpedia

Taxonomy of RDF Data Management



J. F. Sequeda and D. P. Miranker. Ultrawrap: Sparql execution on relational data. Technical Report TR-12-10, University of Texas at Austin, 2012.

Taxonomy of RDF Data Management



J. F. Sequeda and D. P. Miranker. Ultrawrap: Sparql execution on relational data. Technical Report TR-12-10, University of Texas at Austin, 2012.

Triple Table Approach

RDF Data as a Triple Table

Subject	Predicate	Object
Albert123	foaf:firstName	Albert
Albert123	foaf:age	21
Albert123	foaf:livesIn	Austin
Albert123	foaf:collegeMajor	CS
…	…	…
…	…	…
…	…	…
…	…	…
…	…	…
…	…	…
…	…	…

SPARQL Query

```
SELECT ?firstName
WHERE {
        ?x foaf:firstName ?firstName .
        ?x foaf:age 21 .
        ?x foaf:livesIn "Austin"
}
```

- **Join – two triples that share a variable**

Triple Table Self-Join

1. To compute the result, we must join the table with itself (self-join)
2. Need to check if a specific subject, ***x***, has a non-null first name, is 21 years old, and lives in Austin

Subject	Predicate	Object
Albert123	foaf:firstName	Albert
Albert123	foaf:age	21
Albert123	foaf:livesIn	Austin
Albert123	foaf:collegeMajor	CS

Subject	Predicate	Object	Predicate	Object	Predicate	Object
Albert123	foaf:firstName	Albert	foaf:age	21	foaf:livesIn	Austin
Albert123	foaf:firstName	Albert	foaf:livesIn	Austin	foaf:collegeMajor	CS
Albert123	foaf:firstName	Albert	foaf:collegeMajor	CS	foaf:age	21
…	…	…	…	…	…	…

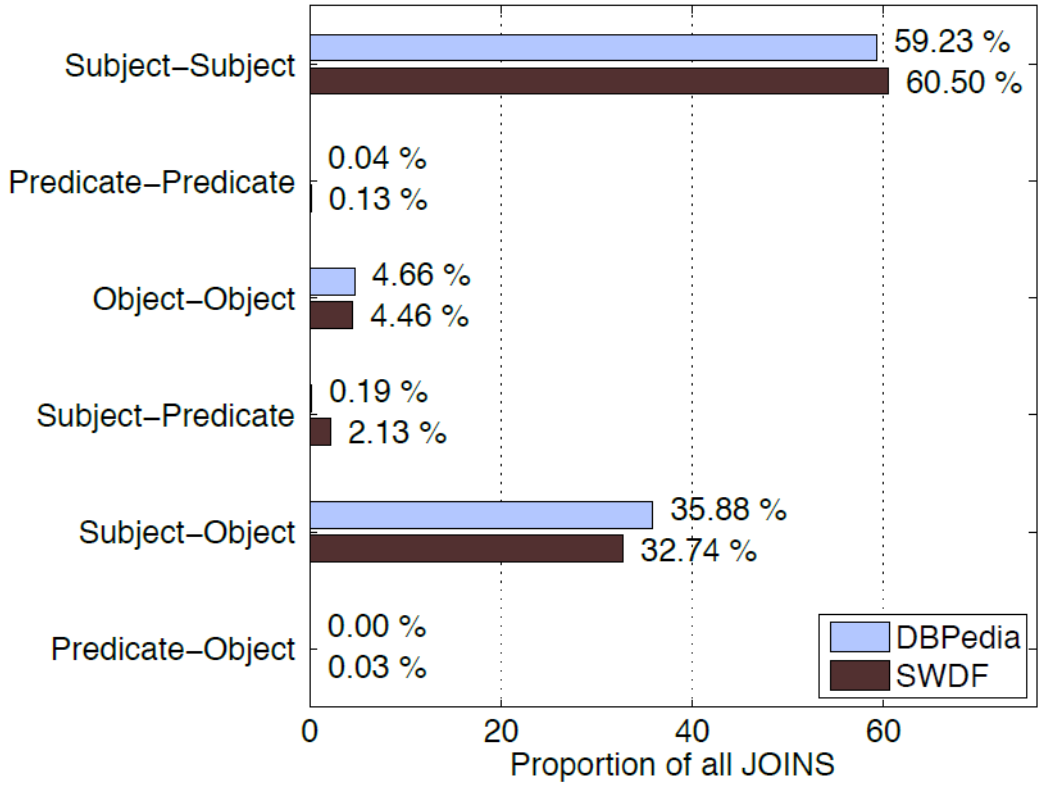
Subject	Predicate1	Object1	Predicate2	Object2	Predicate3	Object3
Albert123	foaf:firstName	Albert	foaf:age	21	foaf:livesIn	Austin

Disadvantages of a Triple Table

- Single table does not give us flexibility
- Over half of DBpedia SPARQL queries are subject-subject self-joins

```
SELECT ?firstName
WHERE {
     ?x foaf:firstName ?firstName .
     ?x foaf:age 21 .
     ?x foaf:livesIn "Austin"
}
```

- Relational joins are expensive



M. A. Gallego, J. D. Fernandez, M. A. Martinez-Prieto, and P. de la Fuente. An empirical study of real-world sparql queries. In 1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011) at the 20th International World Wide Web Conference (WWW 2011), Hydebarabad, India, 2011.

Are Joins That Important?

- Answer: Yes.
- Queries with 1 triple or zero joins can be answered efficiently by triple tables
- However:

  For large datasets and complex queries:

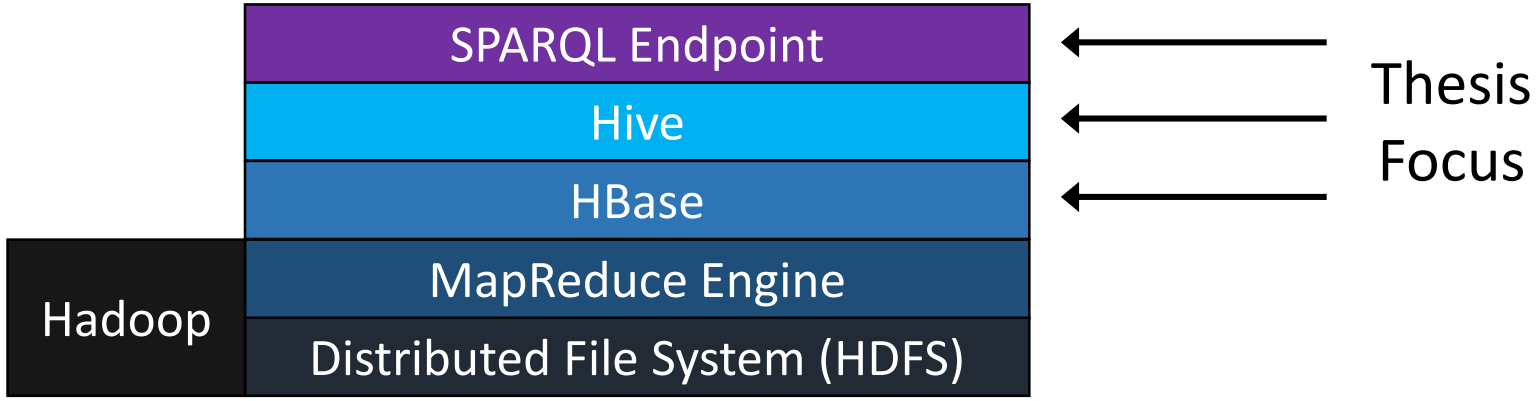
  1. Triple table is very tall; joins will require many equality comparisons and long table scans
  2. Most native-triple stores/triple tables store data and/or index in memory

- My research contributes to solving these problems:
- Developed and evaluated a NoSQL triplestore
  - Maximal reuse of existing “big data” stack
  - Optimize design to enable fast joins
  - Evaluated scalability

NoSQL Triplestore

I implemented a NoSQL triplestore over an open-source software stack:

1. Linked data is very big ⇒ MapReduce/Hadoop is an ideal candidate
2. HBase has appropriate characteristics and features for storing RDF
3. Hive, a leading NoSQL query engine
4. I built custom SPARQL endpoint transforms SPARQL to HiveQL and handles views



HDFS and HBase

Hadoop Distributed File System

- Key-value store
- Handles distributed operations:
  - Fault tolerance
  - Load balancing
  - Cluster management
  - File replication

HBase

- Data store over Hadoop
- Relational table with dynamic schema
- Adds columns and column families to Hadoop/HDFS
- Key-Value Pair:

  (<row key, column family, column>,value)
- HBase bloom filters allow us to optimize SPARQL selection operator

Hive Runs SQL on HDFS

- HiveQL query engine over HBase
- Hive tables don’t replicate HBase data
- Converts HiveQL (SQL) into a DAG of MapReduce jobs
- Implements a naïve join algorithm not tailored for RDF and SPARQL

My Implementation:

- Modify the physical representation of RDF data
- Apply RDBMS techniques to NoSQL: views and projection push-down
- Perform table and query optimizations before Hive is executed

Thusoo, Ashish, et al. "Hive: a warehousing solution over a map-reduce framework." *Proceedings of the VLDB Endowment* 2.2 (2009): 1626-1629.

Property Table Approach

- Subjects as the row key
- Predicates as table columns
- Objects as values

RDF Data as a Property Table

Row Key (Subject)	foaf: firstName	foaf: lastName	foaf: age	foaf: livesIn	foaf: hasChild	foaf: hasPopulation	rdfs: isType
Albert123	Albert		21	Austin			
John1	John	Doe	30	Austin	{Sally, Billy}		
Austin						850000	
Facebook							Website
English							Language

Self-Joins Are No Longer Required

Main Advantage of a Property Table

1. Pre-computes and therefore eliminates self-joins in the triple table approach

```
SELECT ?firstName
WHERE {
        ?x foaf:firstName ?firstName .
        ?x foaf:age 21 .
        ?x foaf:livesIn "Austin"
}
```

3-Way Join on Triple Table

Subject	Predicate1	Object1	Predicate2	Object2	Predicate3	Object3
Albert123	foaf:firstName	Albert	foaf:age	21	foaf:livesIn	Austin

Property Table

Row Key	foaf						rdfs
(Subject)	firstName	lastName	age	livesIn	hasChild	hasPopulation	isType
Albert123	Albert		21	Austin			

Self-Joins Are No Longer Required

Disadvantages

- Sparse data – lots of empty table cells
- Is there a way to resolve this issue?

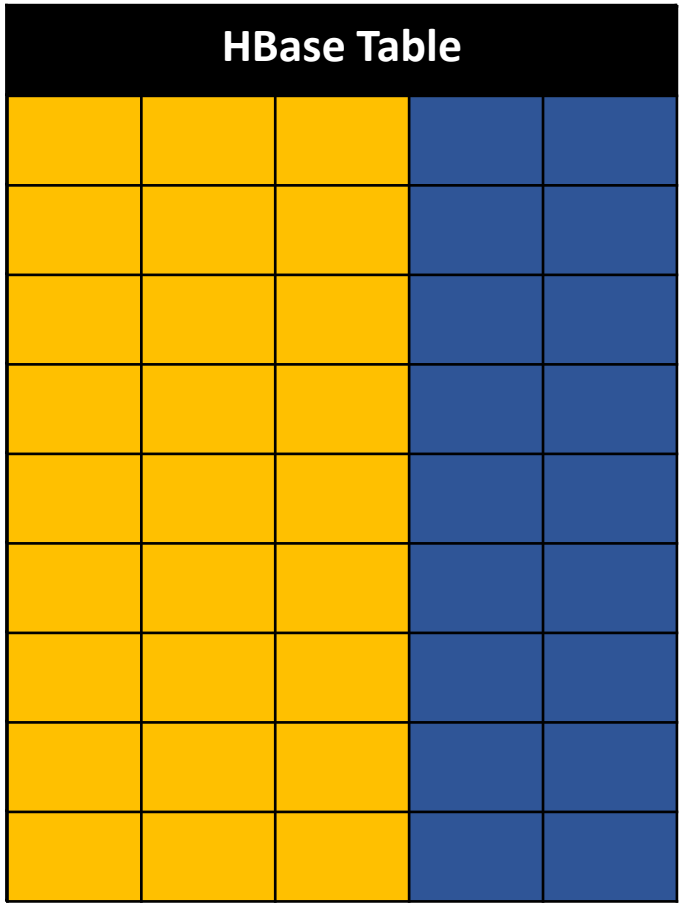
3-Way Join on Triple Table

Subject	Predicate1	Object1	Predicate2	Object2	Predicate3	Object3
Albert123	foaf:firstName	Albert	foaf:age	21	foaf:livesIn	Austin

Property Table

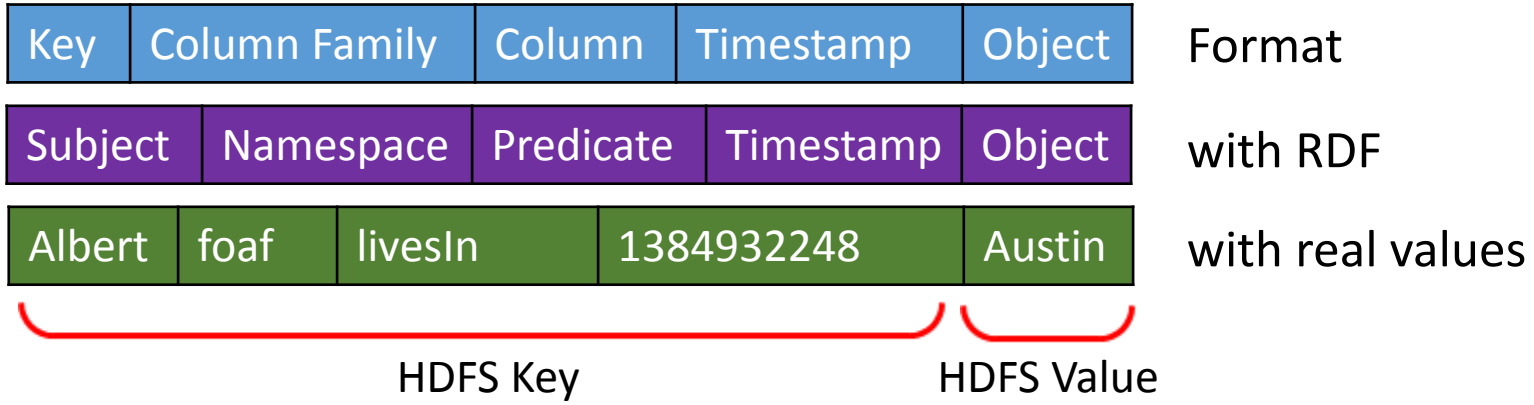
Row Key	foaf						rdfs
(Subject)	firstName	lastName	age	livesIn	hasChild	hasPopulation	isType
Albert123	Albert		21	Austin			

Property Table: Implemented on HBase



- Each row is uniquely identified by row key
- Columns are grouped into column families
- A column family is a collection of columns
- Values are tagged with a timestamp upon insertion

How are triples physically represented on disk?



Property Table: Physical Representation on Disk

RDF Data As A Property Table (Colors Denote Column Families)

Row Key	foaf						rdfs
(Subject)	firstName	lastName	age	livesIn	hasChild	hasPopulation	isType
Albert123	Albert		21	Austin			

Triples As Stored On HDFS

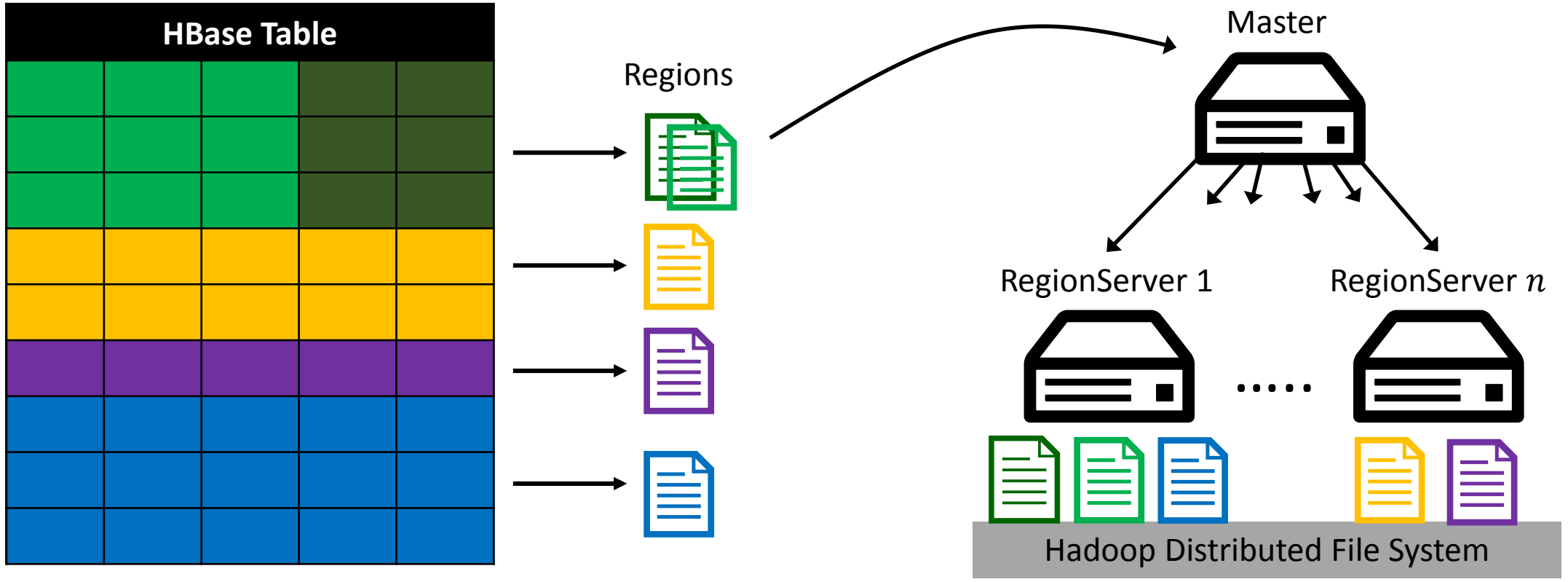
Albert123	foaf	firstName	1384932248	Albert
Albert123	foaf	age	1384239026	21
Albert123	foaf	livesin	1384239027	Austin

Nothing Needed

- **Sparse data is no longer an issue** since HBase does not store nulls on disk

HBase Architecture

- Table is split into groups of rows, called regions, and are stored on RegionServers



We Employ Bloom Filters

- A bloom filter is a bit vector used to tell us if an element is in a set
  - Hash functions set the bits in the bit vector
  - Tells us element is “possibly in the set” or “definitely not in the set” (i.e. no false negatives)
- Each HBase block is equipped with a bloom filter
- As a result, bloom filters significantly reduce number of blocks scanned from disk

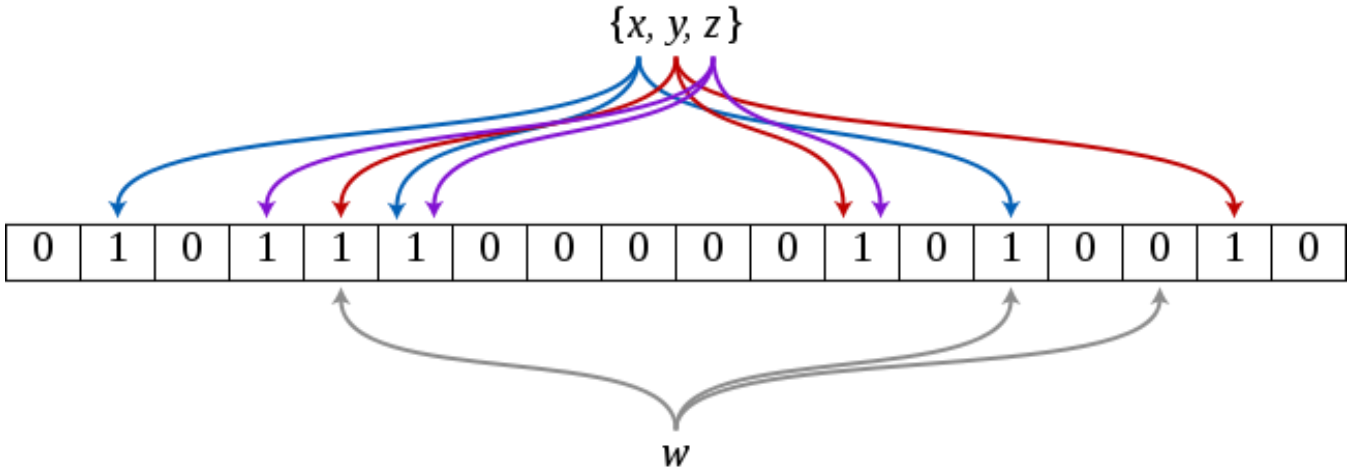
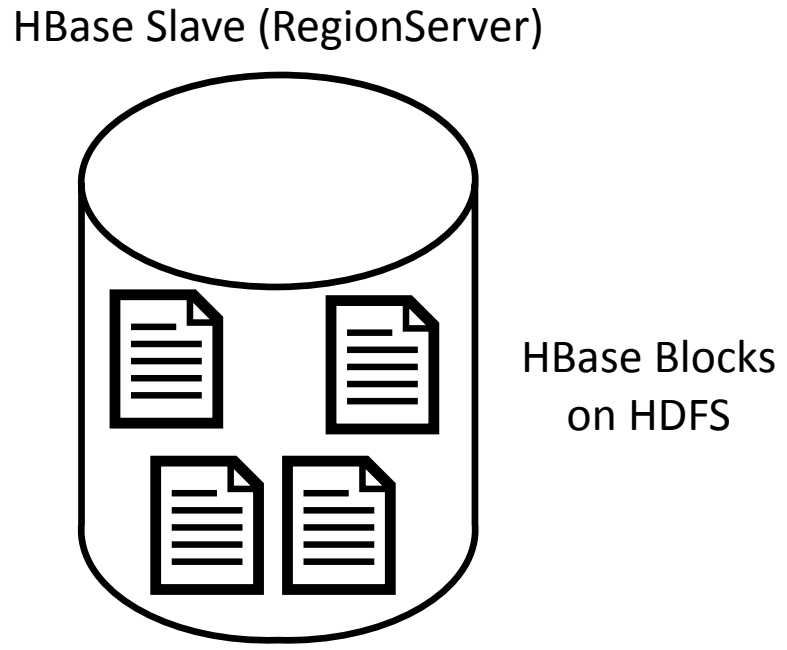
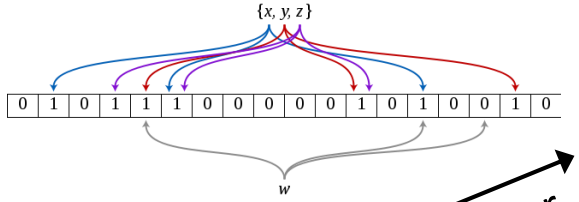


Image: http://en.wikipedia.org/wiki/Bloom_filter

We Employ Bloom Filters



We Employ Bloom Filters

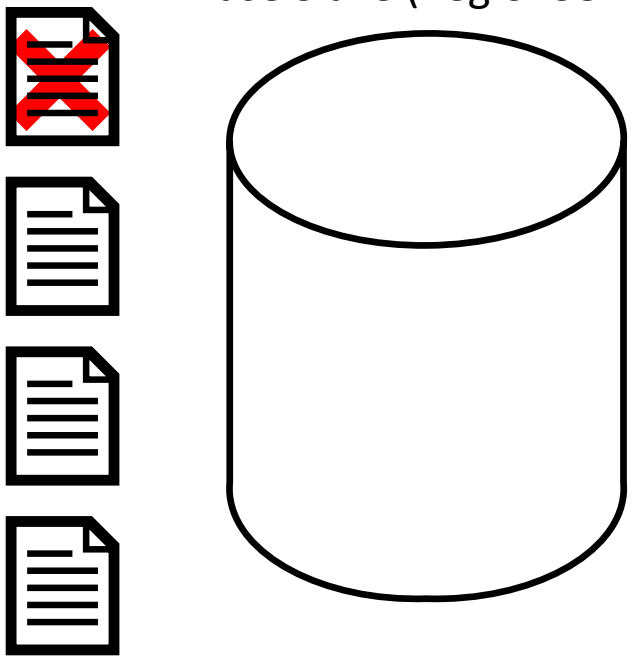


(**albert, livesIn**, ?city)

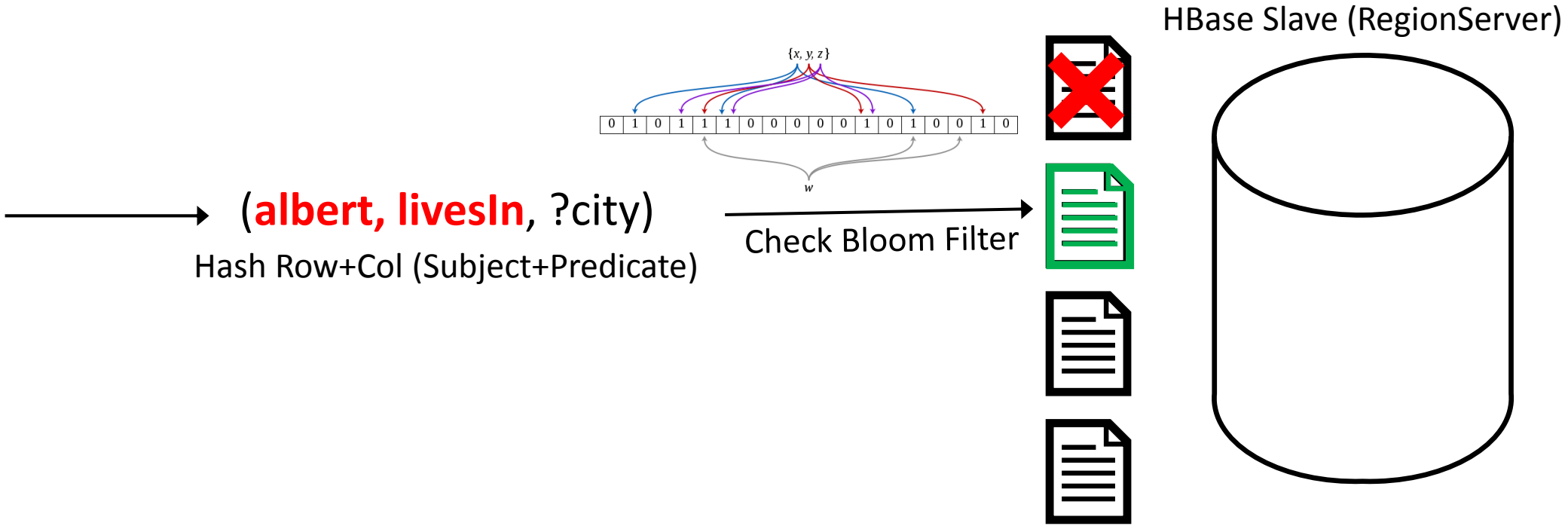
Hash Row+Col (Subject+Predicate)

Check Bloom Filter

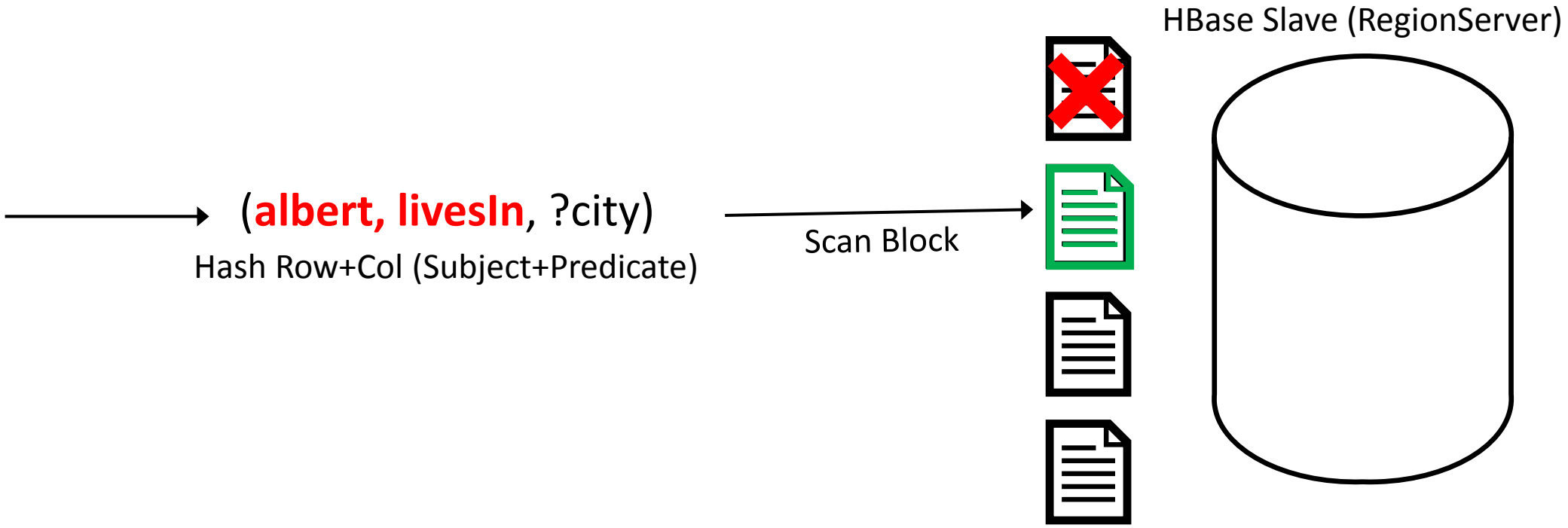
HBase Slave (RegionServer)



We Employ Bloom Filters



We Employ Bloom Filters

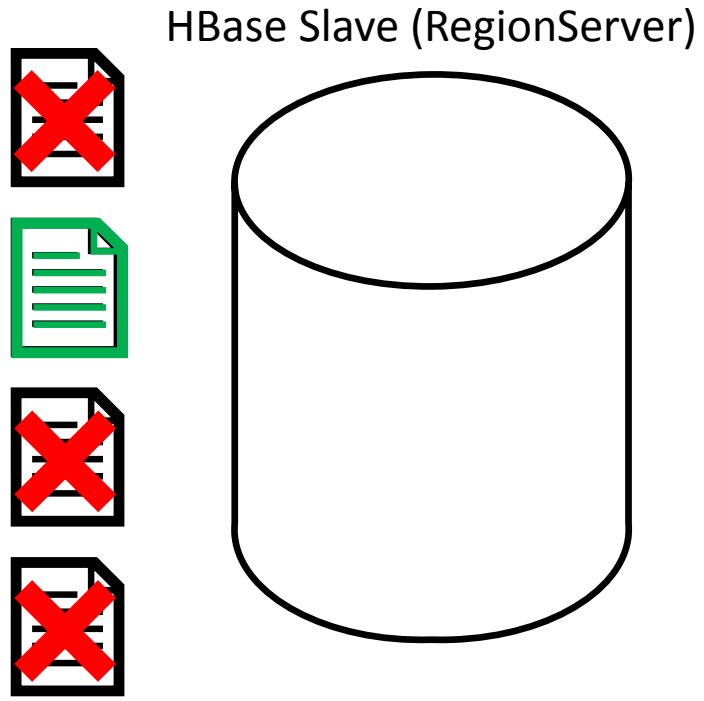


We Employ Bloom Filters

(**albert, livesIn**, ?city)

Hash Row+Col (Subject+Predicate)

Only scanned one block instead of four



Bloom Filters Continued

Additional design decisions we make to exploit bloom filters:

- Given the query: (**albert, livesIn**, ?city)

1. **MapReduce bulk loading → all triples are grouped within Region files**
   - Reduce phase sorts keys → all “albert” triples are placed in 1-2 regions
   - If data is randomly inserted, “albert” triples are spread over many regions

2. **ROW+COL (subject+predicate) bloom filters further reduce blocks read**
   - Two types of bloom filters: (i) ROW ONLY and (ii) ROW+COL bloom filter
   - If “albert” triples lie in two blocks, ROW+COL lets us skip one block
   - ROW ONLY forces us to scan both blocks

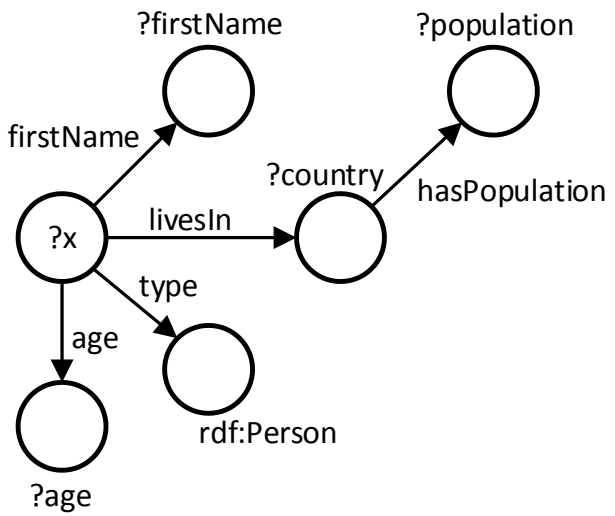
Property Table Doesn't Solve Subject-Object Joins

SPARQL Query

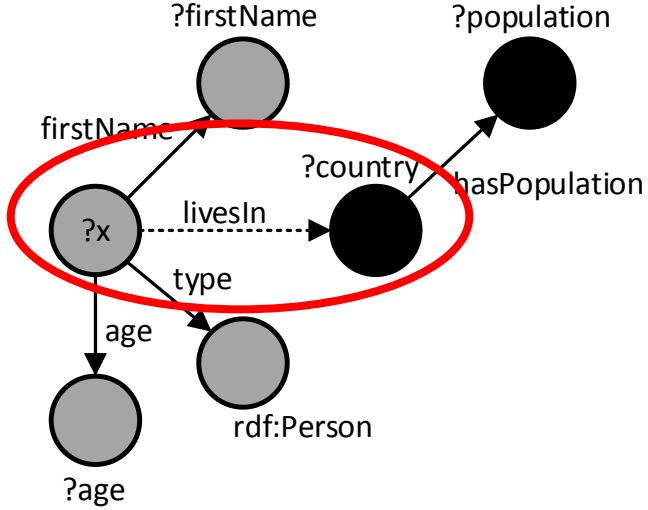
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf#>

SELECT ?firstName, ?age, ?population
WHERE {
    ?x foaf:firstName ?firstName .
    ?x foaf:age ?age .
    ?x rdf:type rdf:Person .
    ?x foaf:livesIn ?country .
    ?country foaf:population ?population
}
```

SPARQL Query as Graph Form



Subject-Object Join



- How do we solve this problem? Answer: perform a self-join (subject-object)
- Create two separate Hive tables – this is done by the SPARQL endpoint

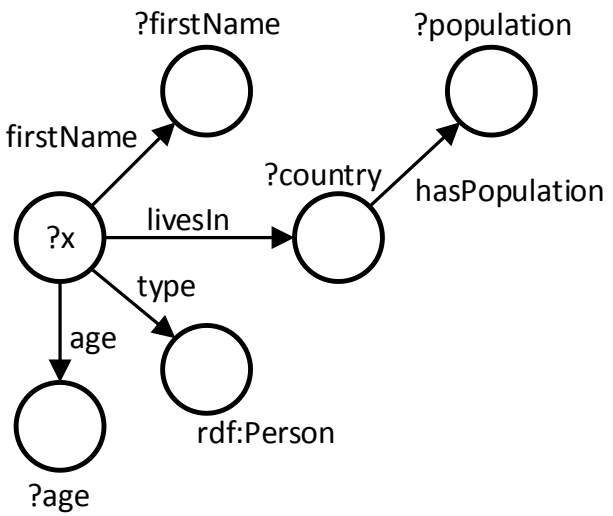
Property Table Doesn't Solve Subject-Object Joins

SPARQL Query

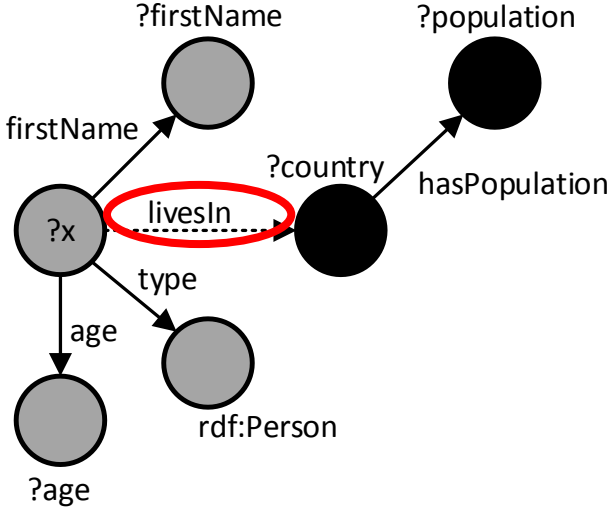
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf#>

SELECT ?firstName, ?age, ?population
WHERE {
    ?x foaf:firstName ?firstName .
    ?x foaf:age ?age .
    ?x rdf:type rdf:Person .
    ?x foaf:livesIn ?country .
    ?country foaf:population ?population
}
```

SPARQL Query as Graph Form



Subject-Object Join

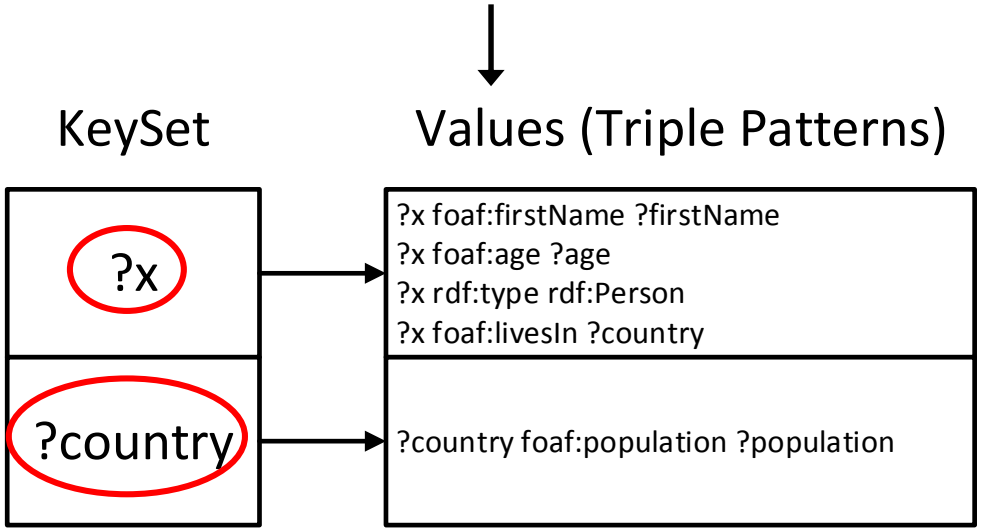


- How do we solve this problem? Answer: perform a self-join (subject-object)
- Create two separate Hive tables – this is done by the SPARQL endpoint

How does the SPARQL Endpoint Create Hive Tables?

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf#>

SELECT ?firstName, ?age, ?population
WHERE {
    ?x foaf:firstName ?firstName .
    ?x foaf:age ?age .
    ?x rdf:type rdf:Person .
    ?x foaf:livesIn ?country .
    ?country foaf:population ?population
}
```



Master Table on HBase

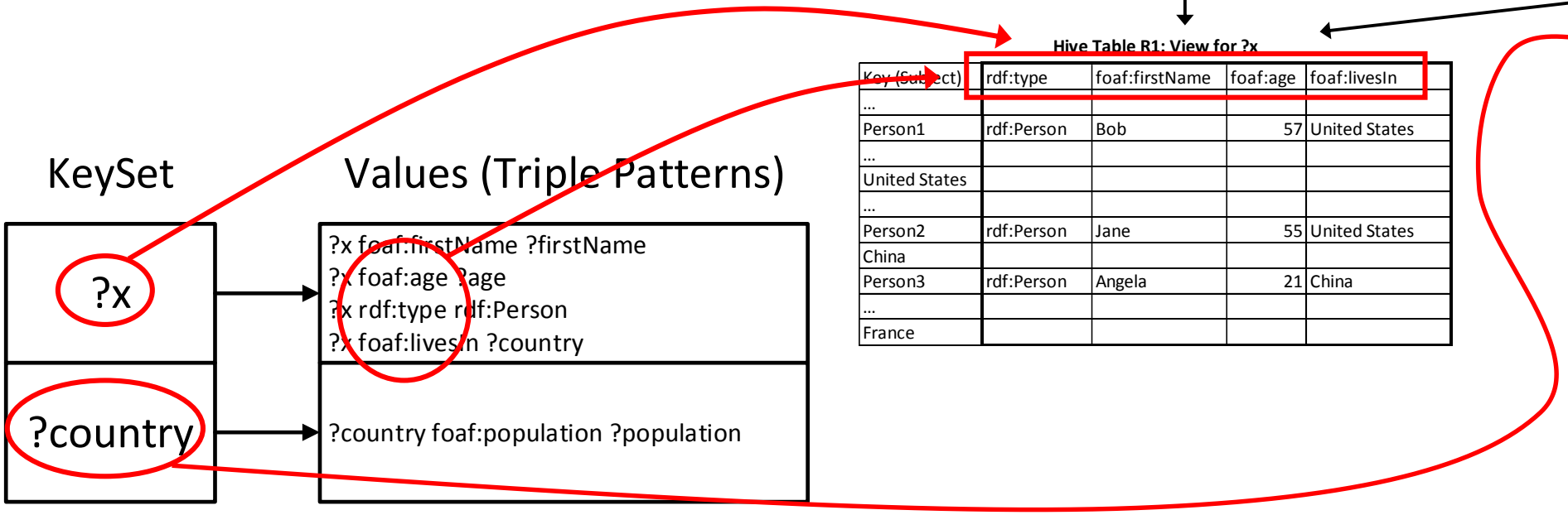
Key (Subject)	rdf:type	foaf:firstName	foaf:age	rdfs:description	foaf:gender	foaf:population	foaf:livesIn
…							
Person1	rdf:Person	Bob	57		Male		United States
…							
United States						314,000,000	
…							
Person2	rdf:Person	Jane	55		Female		United States
China						1,351,000,000	
Person3	rdf:Person	Angela	21		Female		China
…							
France						65,700,000	

How does the SPARQL Endpoint Create Hive Tables?

- A Hive table is a ***view*** over HBase
- Can be created on the fly
- **Keep only relevant information specified by the SPARQL query**

Master Table on HBase

Key (Subject)	rdf:type	foaf:firstName	foaf:age	rdfs:description	foaf:gender	foaf:population	foaf:livesIn
…							
Person1	rdf:Person	Bob	57		Male		United States
…							
United States						314,000,000	
…							
Person2	rdf:Person	Jane	55		Female		United States
China						1,351,000,000	
Person3	rdf:Person	Angela	21		Female		China
…							
France						65,700,000	



Hive Table R1: View for ?x

Key (Subject)	rdf:type	foaf:firstName	foaf:age	foaf:livesIn
…				
Person1	rdf:Person	Bob	57	United States
…				
United States				
…				
Person2	rdf:Person	Jane	55	United States
China				
Person3	rdf:Person	Angela	21	China
…				
France				

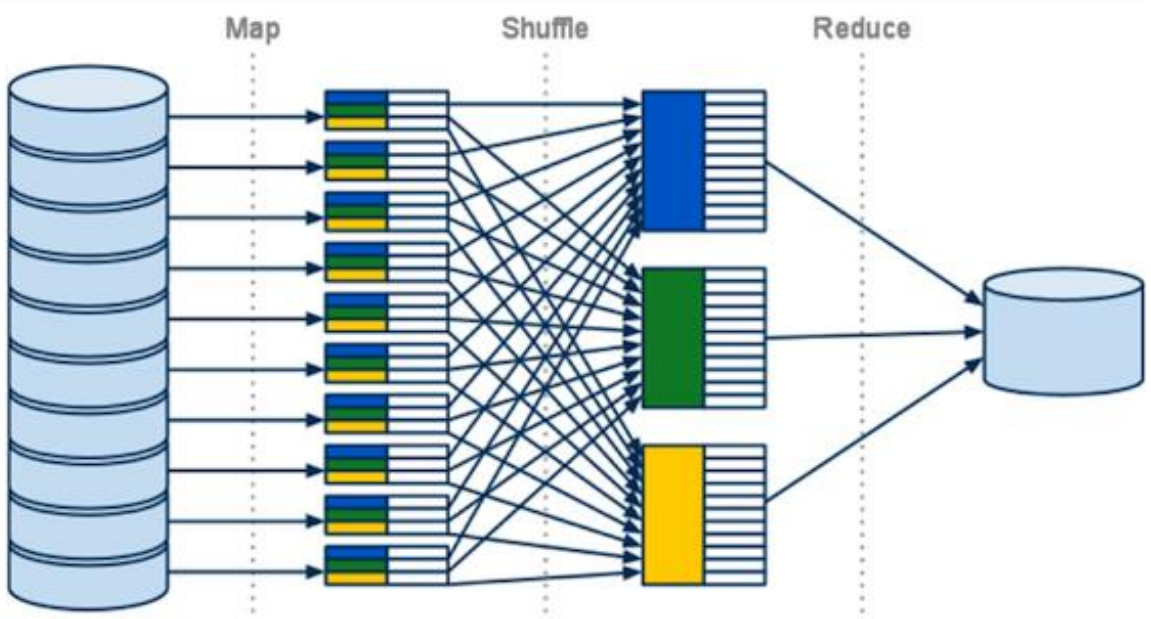
Hive Table R2: View for ?country

Key (Subject)	foaf:population
…	
Person1	
…	
United States	314,000,000
…	
Person2	
China	1,351,000,000
Person3	
…	
France	65,700,000

Why Create Separate Hive Tables?

MapReduce shuffle stage is a bottleneck

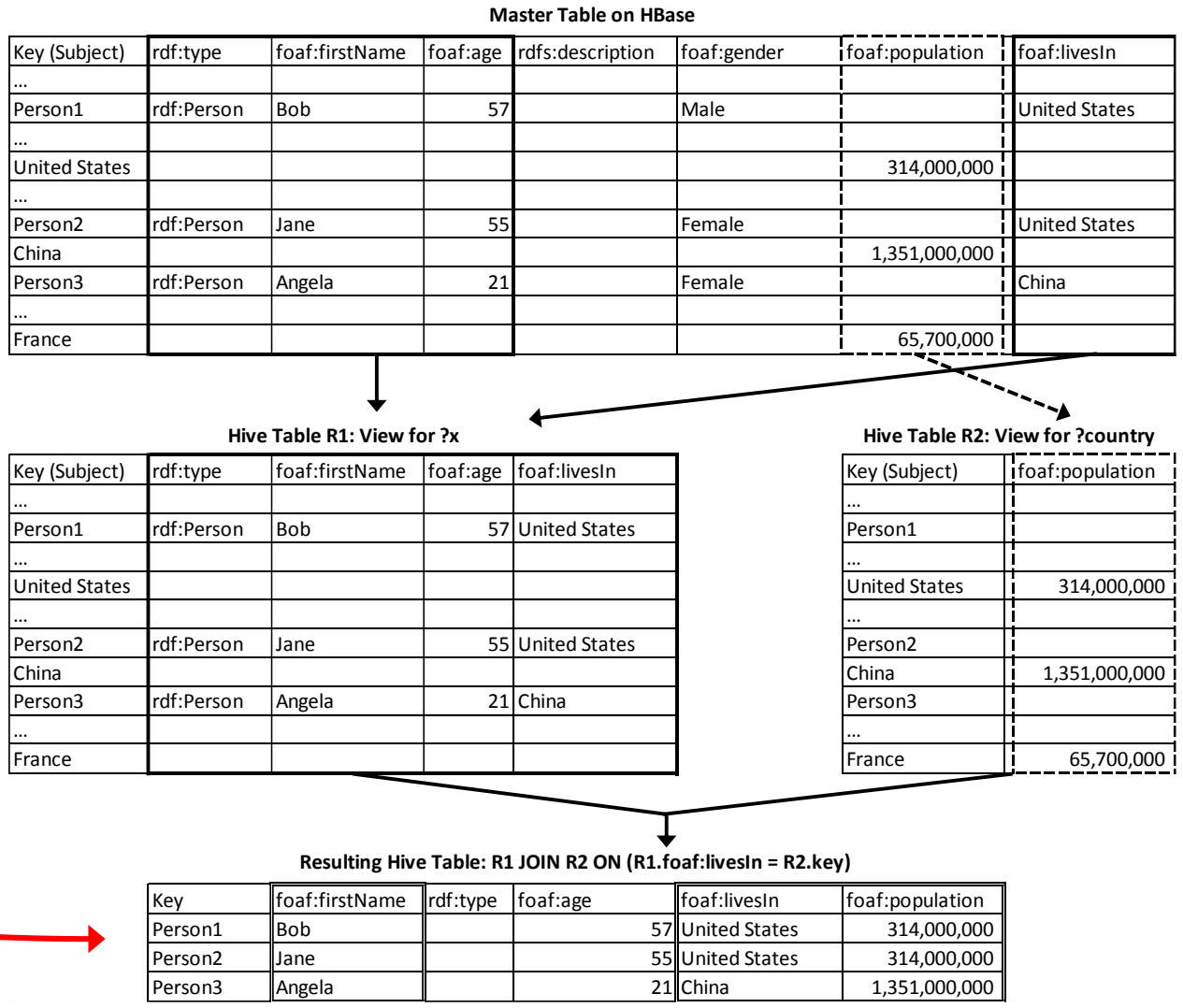
- Shuffle Stage - key/value pair moves from map node to reducer node
- Reducer node cannot start until all blocks have arrived
- Optimize by performing a **“smart self-join”**
- Perform a projection over HBase and only include RDF data relevant to the query



The SPARQL Endpoint Generates HiveQL

Final HiveQL Query

```
SELECT
    R1.foaf:firstName,
    R1.foaf:age,
    R2.foaf:population
FROM
    R1 JOIN R2
    ON ( R1.foaf:livesIn = R2.key
        AND R1.rdf:type = rdf:Person)
```



Master Table on HBase

Key (Subject)	rdf:type	foaf:firstName	foaf:age	rdfs:description	foaf:gender	foaf:population	foaf:livesIn
…							
Person1	rdf:Person	Bob	57		Male		United States
…							
United States						314,000,000	
…							
Person2	rdf:Person	Jane	55		Female		United States
China						1,351,000,000	
Person3	rdf:Person	Angela	21		Female		China
…							
France						65,700,000	

Hive Table R1: View for ?x

Key (Subject)	rdf:type	foaf:firstName	foaf:age	foaf:livesIn
…				
Person1	rdf:Person	Bob	57	United States
…				
United States				
…				
Person2	rdf:Person	Jane	55	United States
China				
Person3	rdf:Person	Angela	21	China
…				
France				

Hive Table R2: View for ?country

Key (Subject)	foaf:population
…	
Person1	
…	
United States	314,000,000
…	
Person2	
China	1,351,000,000
Person3	
…	
France	65,700,000

Resulting Hive Table: R1 JOIN R2 ON (R1.foaf:livesIn = R2.key)

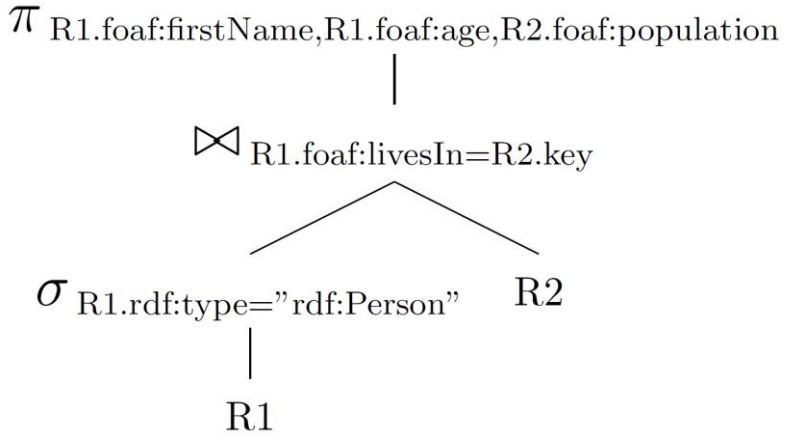
Key	foaf:firstName	rdf:type	foaf:age	foaf:livesIn	foaf:population
Person1	Bob		57	United States	314,000,000
Person2	Jane		55	United States	314,000,000
Person3	Angela		21	China	1,351,000,000

SPARQL to HiveQL Conversion

Step 1: SPARQL Query

```
SELECT ?firstName, ?age, ?population
WHERE {
    ?x foaf:firstName ?firstName .
    ?x foaf:age ?age .
    ?x rdf:type rdf:Person .
    ?x foaf:livesIn ?country .
    ?country foaf:population ?population
}
```

Step 2: Abstract Syntax Tree



Step 3: HiveQL Query

```
SELECT
    R1.foaf:firstName,
    R1.foaf:age,
    R2.foaf:population
FROM
    R1 JOIN R2
    ON ( R1.foaf:livesIn = R2.key
         AND R1.rdf:type = rdf:Person)
```

Summary

1. **Property table eliminates subject-subject self-joins**
   - Property table model pre-computes these joins
2. **Bloom filters allow us to accelerate table scans**
   - MapReduce loading sorts triples inside Regions
   - Row+column hash provide additional dimension of filtering for selection operations
3. **Hive tables contain only relevant columns**
   - Views enable us to perform relational projections over HBase
   - Reduces data being joined and sent across network → shuffle stage will be faster
4. **Joins are performed via MapReduce**
   - Leverages parallelization and horizontal scalability on commodity hardware

- We've applied various relational database techniques to our system
- How will it perform in a cloud environment?

Datasets and Benchmarks

1. Berlin SPARQL Benchmark (BSBM)

- Synthetic dataset
- 10 million triples – 2.5 GB
- 100 million triples – 25 GB
- 1 billion triples – 250 GB

2. DBpedia SPARQL Benchmark

- Real data based on Wikipedia
- 150 million triples – 27 GB

Bizer, Christian, and Andreas Schultz. "The berlin sparql benchmark." *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.2 (2009): 1-24.

Compute Environment

Cluster Configuration

- Amazon EC2 North Virginia Region
- 1, 2, 4, 8, and 16 worker nodes
- Plus 1 master node

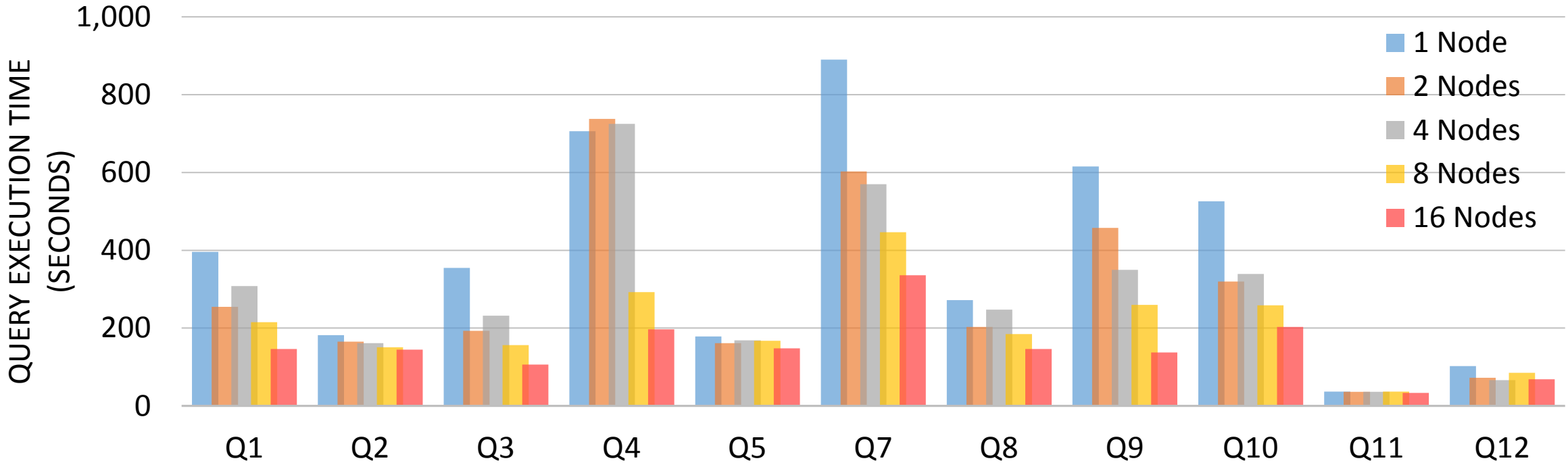
Software

- Hadoop 1.0.3
- HBase 0.92
- Hive 0.8.1

Hardware

- Amazon EC2 m1.large instances
- 8 GB main memory
- 840 GB (2x420 GB) local disk @ 7200 RPM
- Intel Core 2 Duo T6400 @ 2.00GHz
- 4 Elastic Compute Units
  - 1 ECU ≈ 1.0-1.2 GHz 2007 Xeon processor
- 2 Virtual CPUs (2 Physical Cores)

Results: BSBM 10 Million Triples

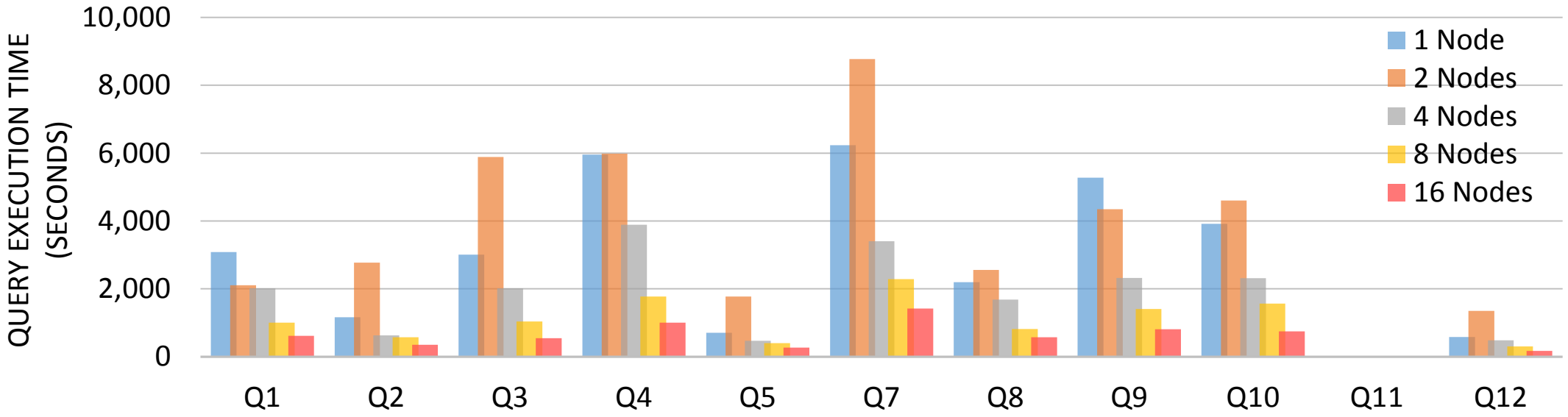


	Inner Join	Left-Outer Join	Var. Predicate
Low Selectivity	1, 3 10		9, 11
High Selectivity	4, 5, 6, 12	2, 7, 8	

Query	1	2	3	4	5	7	8	9	10	11	12
# Joins	0	2	0	0	1	4	1	1	1	0	0

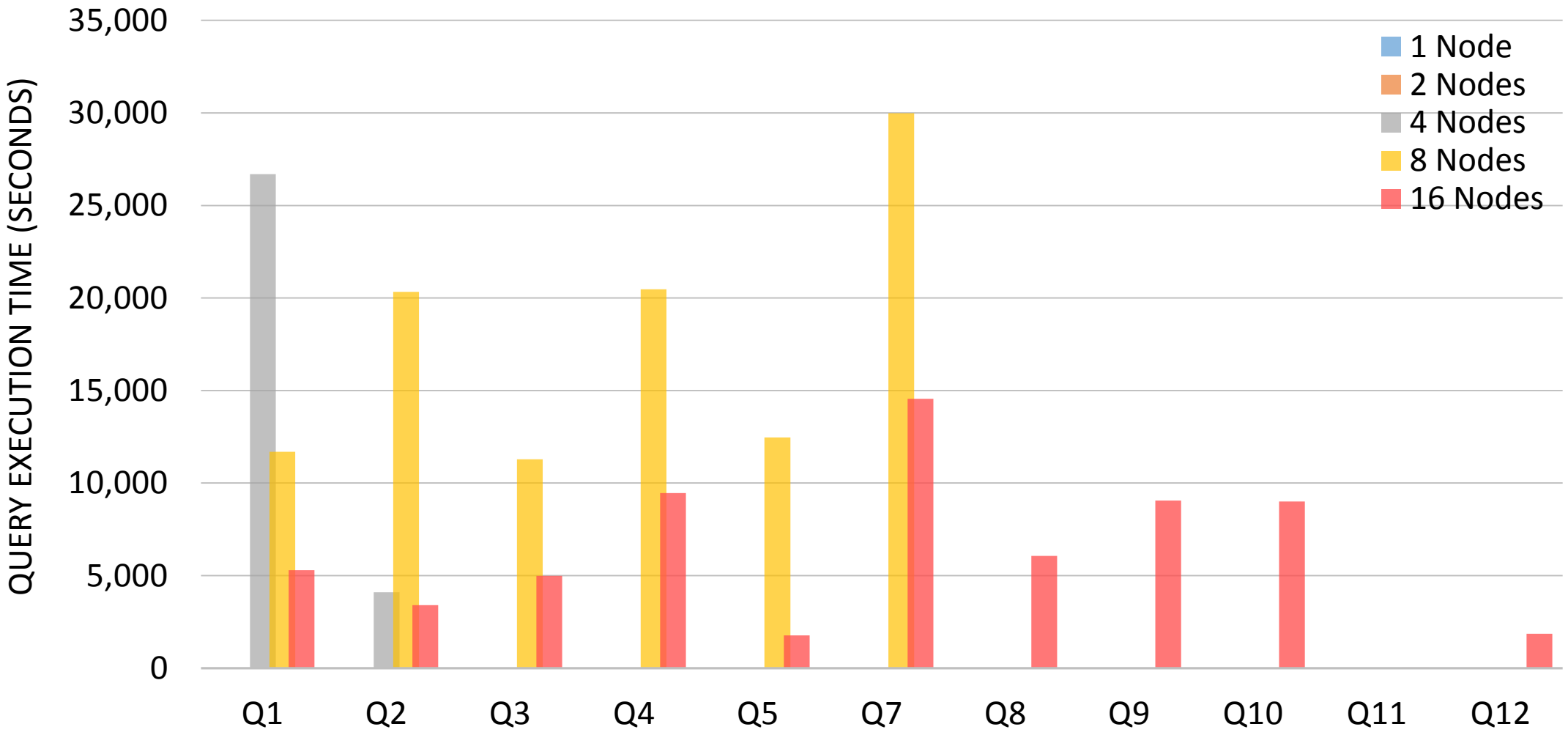
J. F. Sequeda and D. P. Miranker. Ultrawrap: Sparql execution on relational data. Technical Report TR-12-10, University of Texas at Austin, 2012.

Results: BSBM 100 Million Triples



Characteristic	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Simple filters	√		√	√			√	√	√	√		
Complex filters					√	√						
More than 9 patterns		√		√			√	√				
Unbound predicates											√	
Negation			√									
OPTIONAL operator		√	√				√	√				
LIMIT modifier	√		√	√	√			√		√		
ORDER BY modifier	√		√	√	√			√		√		
DISTINCT modifier	√				√					√		

Results: BSBM 1 Billion Triples

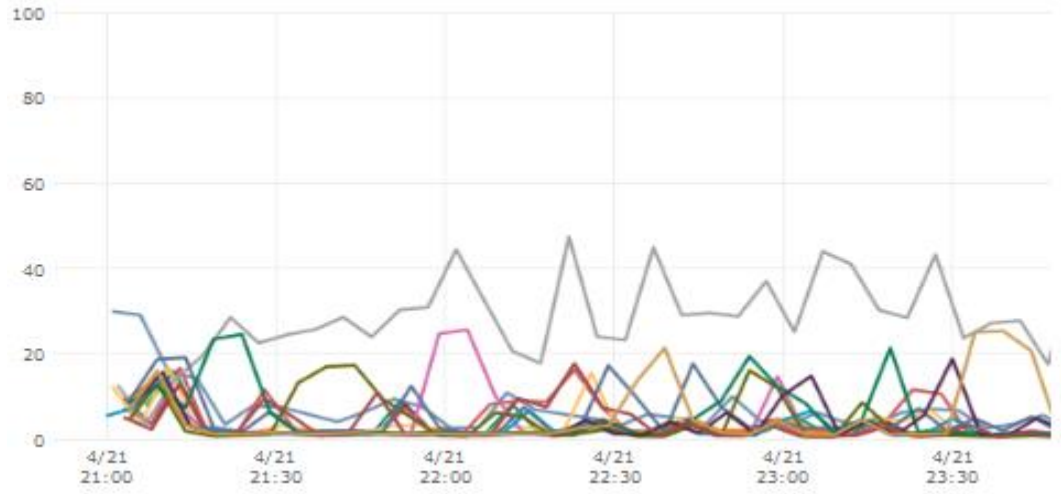


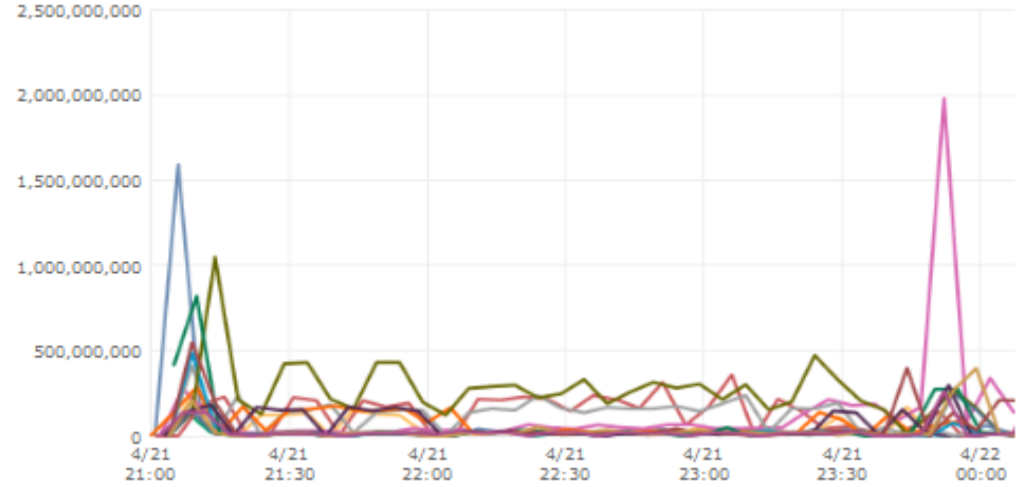
Comparing 10 Million to 1 Billion (16 Nodes)

CPU Utilization of Cluster Nodes

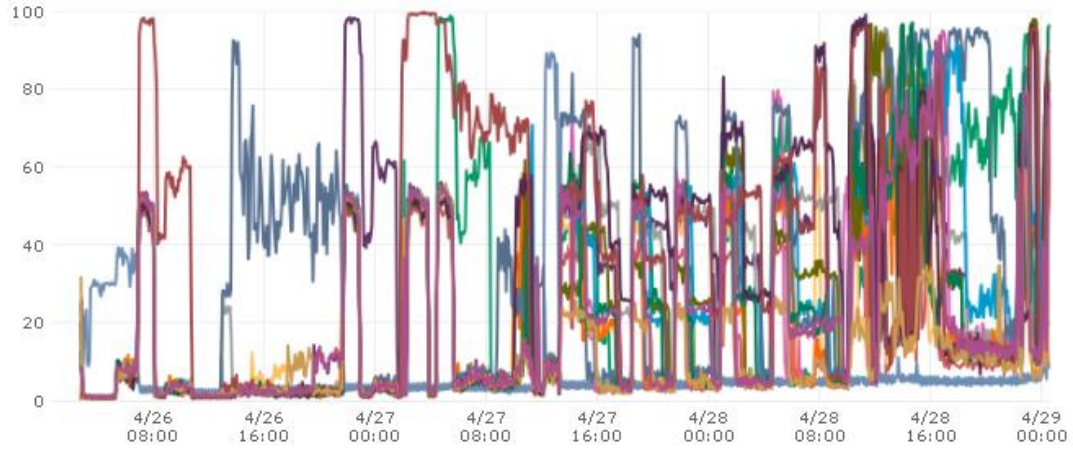
Network Out of Cluster Nodes

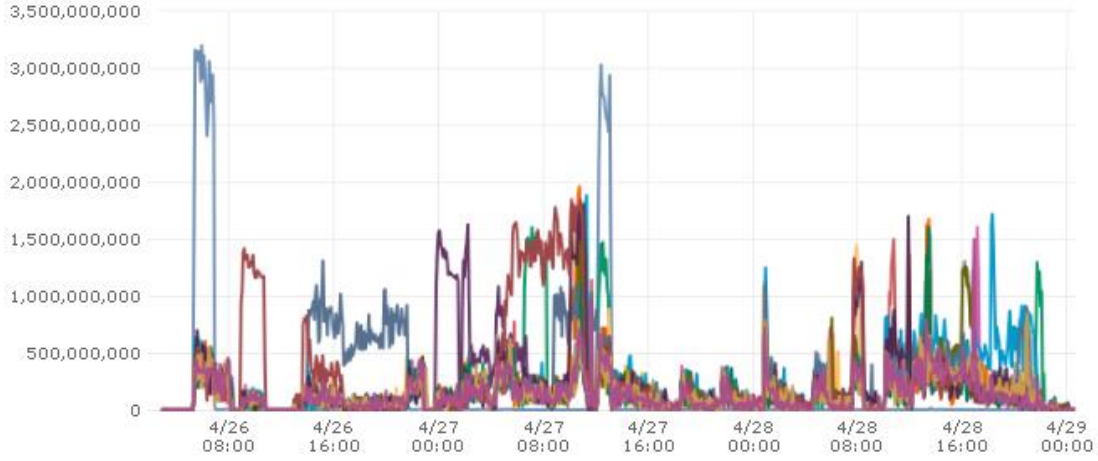
10 Million Triples



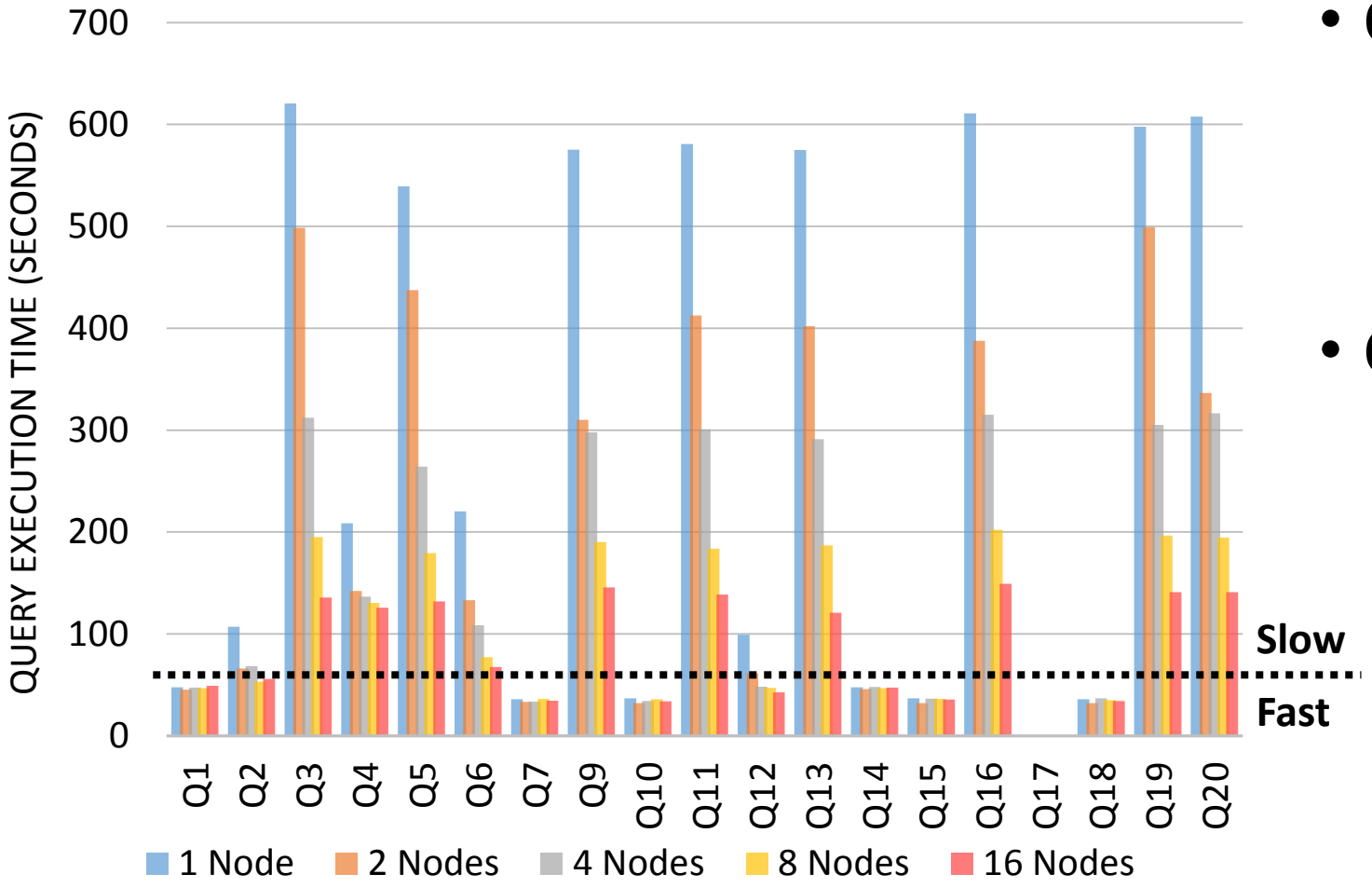


1 Billion Triples



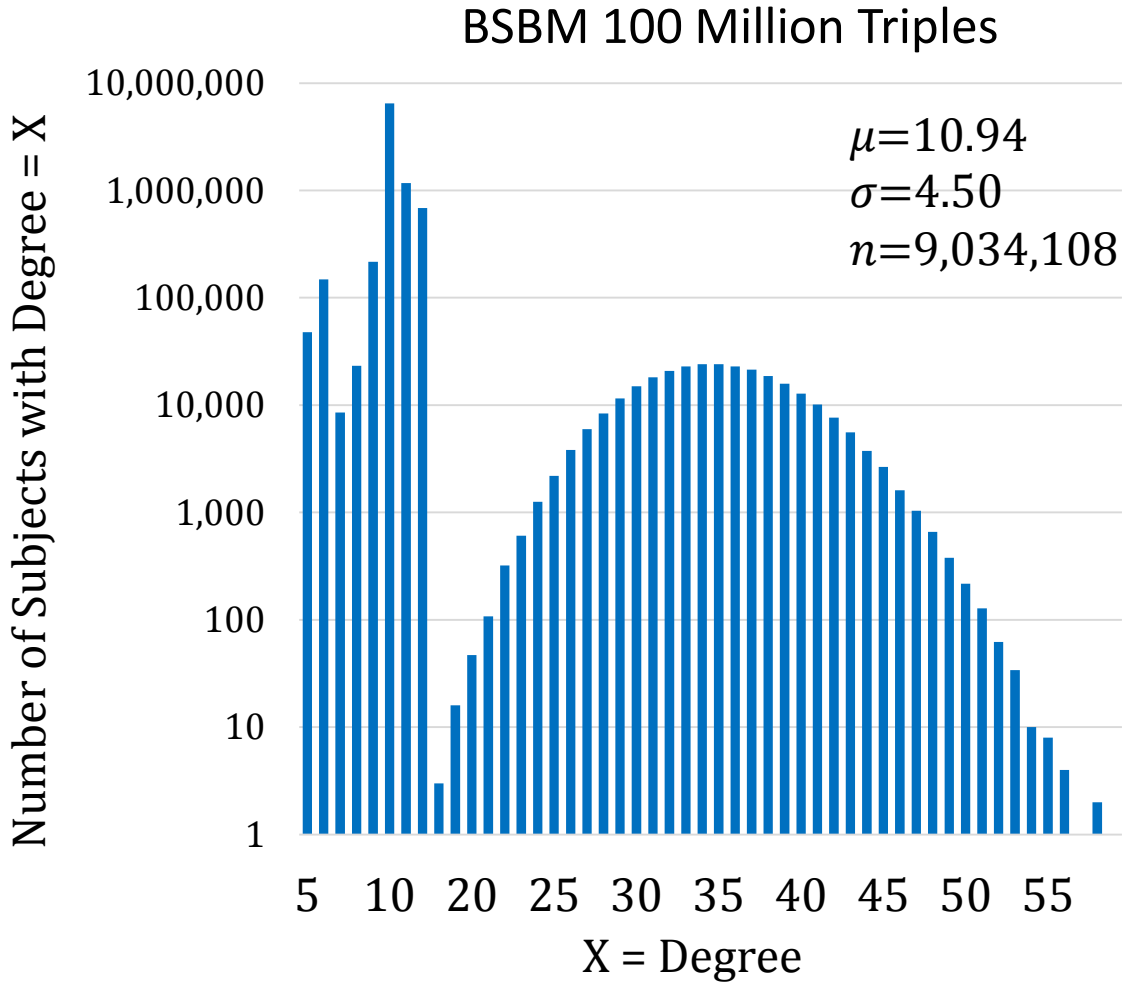


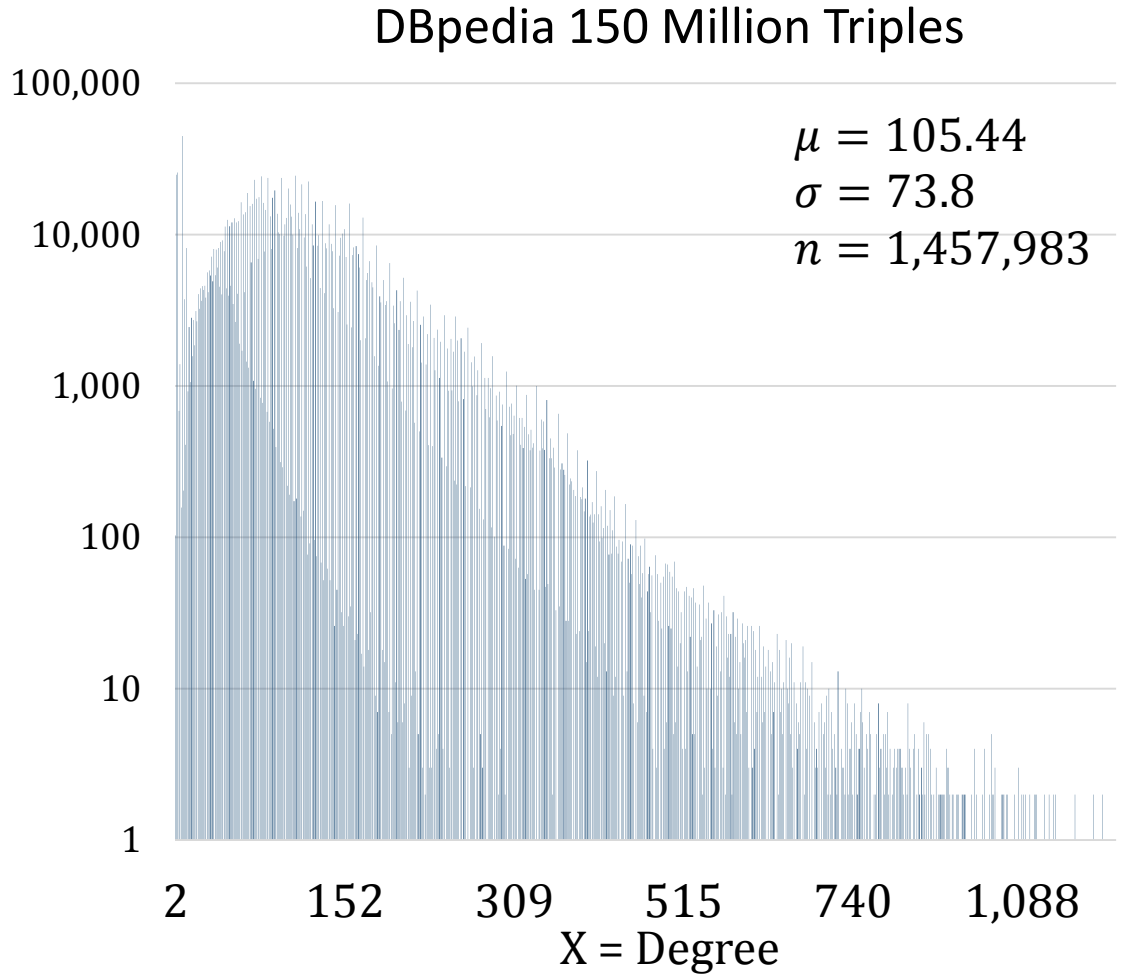
Results: DBpedia 150 Million Triples



- Characteristics of fast queries:
  - Fixed subject – bloom filters
  - NOT NULL
- Characteristics of slow queries:
  - Substring search operations
  - LIKE keyword
  - Regular expressions
  - Variable subjects

Dataset Analysis



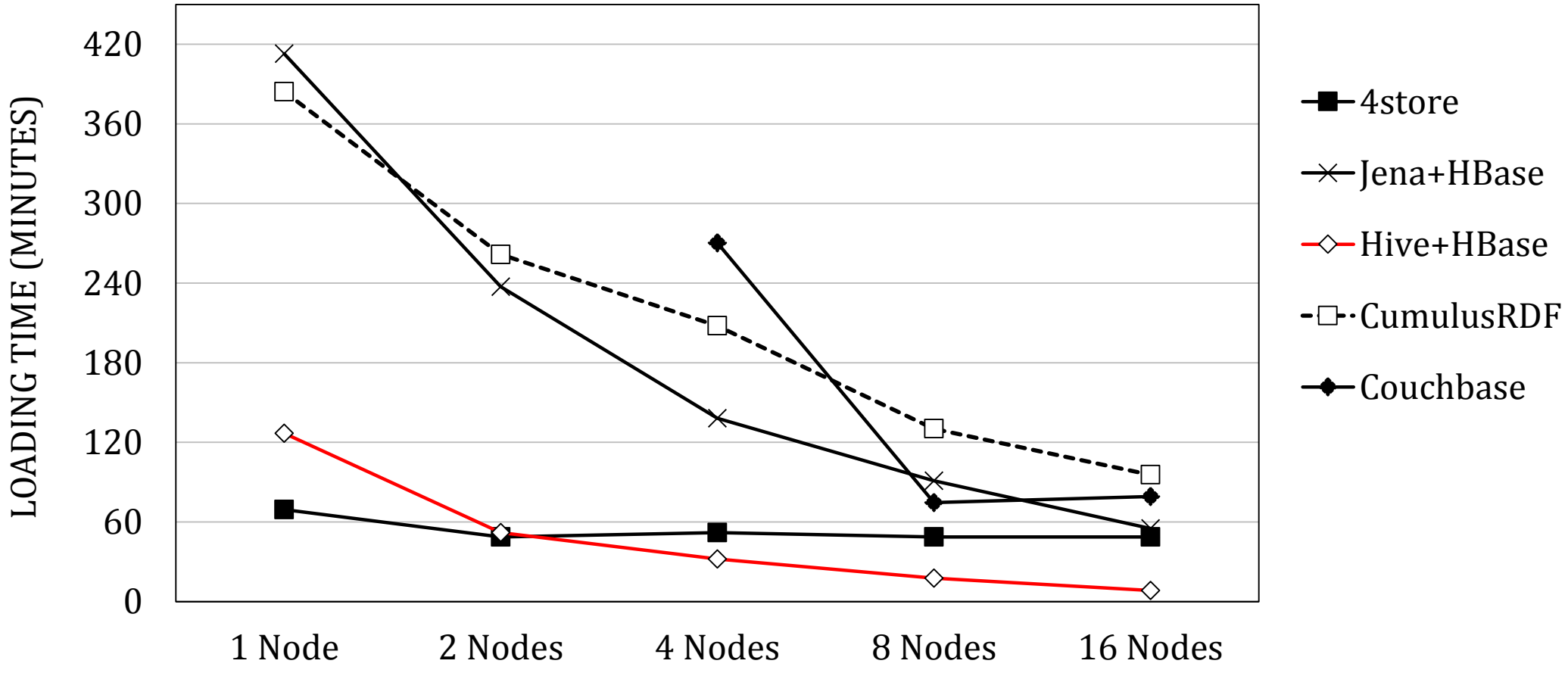


Other NoSQL RDF Databases

1. 4store (native triplestore)
2. HBase+Jena (MapReduce)
3. CumulusRDF & Cassandra (NoSQL)
4. Couchbase (in-memory JSON NoSQL)

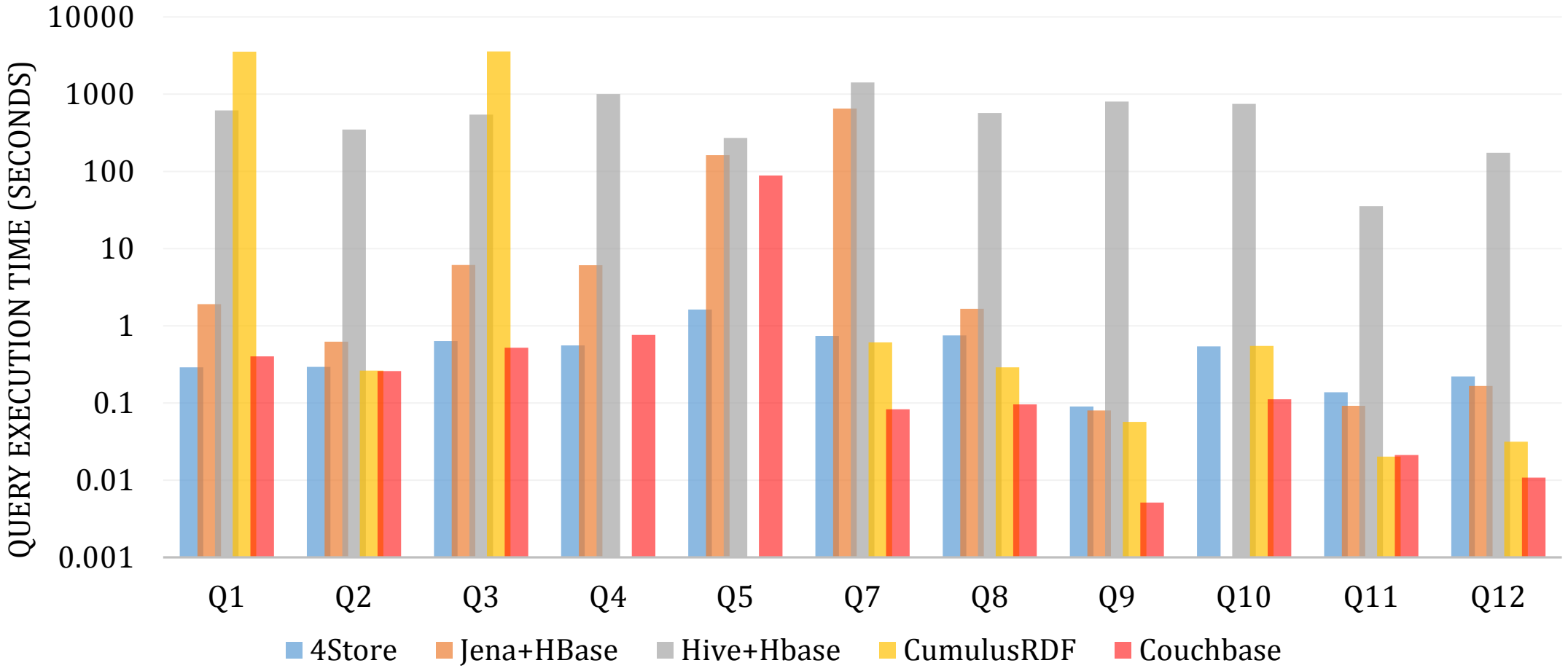
P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. Keppmann, D. Miranker, J. Sequeda, and M. Wylot. "NoSQL Databases for RDF: An Empirical Evaluation." Proceedings of the 12th International Semantic Web Conference (ISWC). LNCS, vol. 8219, pp. 310-325. Springer, 2013.

Hive+HBase Has Fastest Loading Time



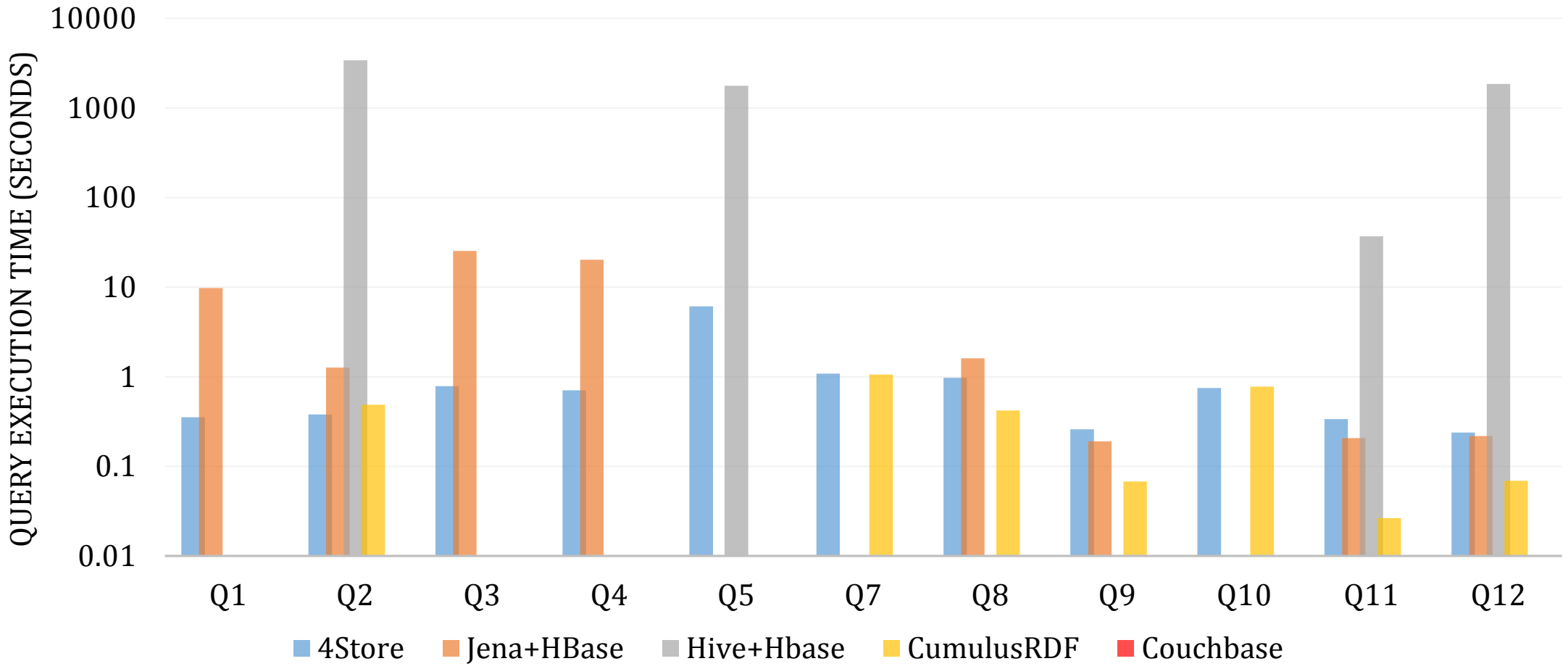
P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. Keppmann, D. Miranker, J. Sequeda, and M. Wylot. "NoSQL Databases for RDF: An Empirical Evaluation." Proceedings of the 12th International Semantic Web Conference (ISWC). LNCS, vol. 8219, pp. 310-325. Springer, 2013.

BSBM Results – 100 Million, 16 Nodes



P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. Keppmann, D. Miranker, J. Sequeda, and M. Wylot. "NoSQL Databases for RDF: An Empirical Evaluation." Proceedings of the 12th International Semantic Web Conference (ISWC). LNCS, vol. 8219, pp. 310-325. Springer, 2013.

BSBM Results – 1 Billion, 16 Nodes



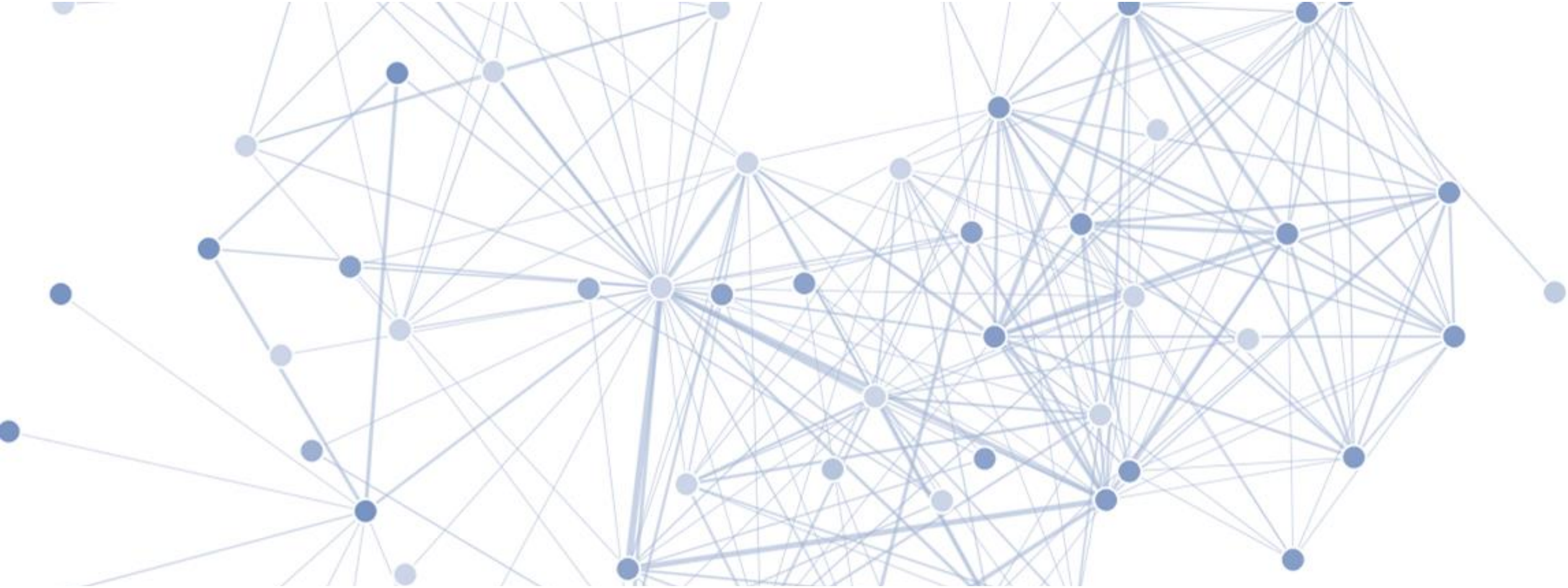
P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. Keppmann, D. Miranker, J. Sequeda, and M. Wylot. "NoSQL Databases for RDF: An Empirical Evaluation." Proceedings of the 12th International Semantic Web Conference (ISWC). LNCS, vol. 8219, pp. 310-325. Springer, 2013.

Conclusion & Future Work

- Linked data is growing ⇒ need to use distributed system (MapReduce)
- NoSQL triplestores are still in their infancy
- This work was a first attempt at assessing current technologies (Hive)
- Hive implements a naïve join algorithm, other NoSQL RDF databases have custom join algorithms specific to particular use cases

Next Steps:

1. Explore Apache Spark + Shark
   - Distributed, shared memory warehouse, uses MapReduce/Hive, and 30x faster
2. Improve default join algorithm and incorporate query cost optimizer



A MapReduce Approach to NoSQL RDF Databases

Albert Haque
Research in Bioinformatics and Semantic Web Lab
University of Texas at Austin
November 25, 2013

BSBM Query 1

Appendix

```
SELECT DISTINCT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product a <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductType66> .
    ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature3> .
    ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature1967> .
    ?product bsbm:productPropertyNumeric1 ?value1 .
           FILTER (?value1 > 136)
           }
ORDER BY ?label
LIMIT 10
```

BSBM Query 2

Appendix

```
SELECT ?label ?comment ?producer ?productFeature ?propertyTextual1 ?propertyTextual2 ?propertyTextual3
 ?propertyNumeric1 ?propertyNumeric2 ?propertyTextual4 ?propertyTextual5 ?propertyNumeric4
WHERE {
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> rdfs:label ?label .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> rdfs:comment ?comment .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:producer ?p .
    ?p rdfs:label ?producer .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> dc:publisher ?p .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productFeature ?f .
    ?f rdfs:label ?productFeature .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyTextual1 ?propertyTextual1 .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyTextual2 ?propertyTextual2 .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyTextual3 ?propertyTextual3 .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyNumeric1 ?propertyNumeric1 .
    <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyNumeric2 ?propertyNumeric2 .
    OPTIONAL { <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyTextual4 ?propertyTextual4 }
    OPTIONAL { <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyTextual5 ?propertyTextual5 }
    OPTIONAL { <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer2/Product72> bsbm:productPropertyNumeric4 ?propertyNumeric4 }
}
```

BSBM Query 3

Appendix

```
SELECT ?product ?label
WHERE {
    ?product rdfs:label ?label .
    ?product a <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductType87> .
           ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature541> .
           ?product bsbm:productPropertyNumeric1 ?p1 .
           FILTER ( ?p1 > 156 )
           ?product bsbm:productPropertyNumeric3 ?p3 .
           FILTER (?p3 < 152 )
    OPTIONAL {
        ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature553> .
        ?product rdfs:label ?testVar }
    FILTER (!bound(?testVar))
}
ORDER BY ?label
LIMIT 10
```

BSBM Query 4

Appendix

```
SELECT DISTINCT ?product ?label ?propertyTextual
WHERE {
    {
       ?product rdfs:label ?label .
       ?product rdf:type <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductType138> .
       ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature4305> .
       ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature1427> .
       ?product bsbm:productPropertyTextual1 ?propertyTextual .
       ?product bsbm:productPropertyNumeric1 ?p1 .
       FILTER ( ?p1 > 457 )
    } UNION {
       ?product rdfs:label ?label .
       ?product rdf:type <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductType138> .
       ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature4305> .
       ?product bsbm:productFeature <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/ProductFeature1444> .
       ?product bsbm:productPropertyTextual1 ?propertyTextual .
       ?product bsbm:productPropertyNumeric2 ?p2 .
       FILTER ( ?p2> 488 )
    }
}
ORDER BY ?label
OFFSET 5
LIMIT 10
```

BSBM Query 5

Appendix

```
SELECT DISTINCT ?product ?productLabel
WHERE {
            ?product rdfs:label ?productLabel .
    FILTER (<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer31/Product1390> != ?product)
            <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer31/Product1390> bsbm:productFeature ?prodFeature .
            ?product bsbm:productFeature ?prodFeature .
            <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer31/Product1390> bsbm:productPropertyNumeric1 ?origProperty1 .
            ?product bsbm:productPropertyNumeric1 ?simProperty1 .
            FILTER (?simProperty1 < (?origProperty1   120) && ?simProperty1 > (?origProperty1 - 120))
            <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer31/Product1390> bsbm:productPropertyNumeric2 ?origProperty2 .
            ?product bsbm:productPropertyNumeric2 ?simProperty2 .
            FILTER (?simProperty2 < (?origProperty2   170) && ?simProperty2 > (?origProperty2 - 170))
}
ORDER BY ?productLabel
LIMIT 5
```

BSBM Query 7

Appendix

```
SELECT ?productLabel ?offer ?price ?vendor ?vendorTitle ?review ?revTitle
       ?reviewer ?revName ?rating1 ?rating2
WHERE {
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer22/Product1001> rdfs:label ?productLabel .
   OPTIONAL {
           ?offer bsbm:product <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer22/Product1001> .
           ?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 > "2008-06-20T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> )
    }
    OPTIONAL {
           ?review bsbm:reviewFor <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer22/Product1001> .
           ?review rev:reviewer ?reviewer .
           ?reviewer foaf:name ?revName .
           ?review dc:title ?revTitle .
    OPTIONAL { ?review bsbm:rating1 ?rating1 . }
    OPTIONAL { ?review bsbm:rating2 ?rating2 . }
    }
}
```

BSBM Query 8

Appendix

```
SELECT ?title ?text ?reviewDate ?reviewer ?reviewerName ?rating1 ?rating2 ?rating3 ?rating4
WHERE {
      ?review bsbm:reviewFor <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer21/Product978> .
      ?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
```

BSBM Query 9

Appendix

```
DESCRIBE ?x
WHERE { <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromRatingSite2/Review15194> rev:reviewer ?x }
```

BSBM Query 10

Appendix

```
SELECT DISTINCT ?offer ?price
WHERE {
     ?offer bsbm:product <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromProducer9/Product396> .
     ?offer bsbm:vendor ?vendor .
     ?offer dc:publisher ?vendor .
     ?vendor bsbm:country <http://downlode.org/rdf/iso-3166/countries#US> .
     ?offer bsbm:deliveryDays ?deliveryDays .
     FILTER (?deliveryDays <= 3)
     ?offer bsbm:price ?price .
     ?offer bsbm:validTo ?date .
     FILTER (?date > "2008-06-20T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> )
}
ORDER BY xsd:double(str(?price))
LIMIT 10
```

BSBM Query 11

Appendix

```
SELECT ?property ?hasValue ?isValueOf
WHERE {
  { <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor12/Offer21250> ?property ?hasValue }
  UNION
  { ?isValueOf ?property <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor12/Offer21250> }
}
```

BSBM Query 12

Appendix

```
CONSTRUCT {
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:product ?productURI .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:productlabel ?productlabel .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:vendor ?vendorname .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:vendorhomepage ?vendorhomepage .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:offerURL ?offerURL .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:price ?price .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:deliveryDays ?deliveryDays .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm-export:validuntil ?validTo
}
WHERE {
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm:product ?productURI .
?productURI rdfs:label ?productlabel .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm:vendor ?vendorURI .
?vendorURI rdfs:label ?vendorname .
?vendorURI foaf:homepage ?vendorhomepage .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm:offerWebpage ?offerURL .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm:price ?price .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm:deliveryDays ?deliveryDays .
<http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/dataFromVendor7/Offer13035> bsbm:validTo ?validTo
}
```