Approaches to RDF Data Management and SPARQL Query Processing

M. Tamer Özsu

University of Waterloo
David R. Cheriton School of Computer Science
tamer.ozsu@uwaterloo.ca

- M. T. Özsu, A Survey of RDF Data Management Systems, *Front. Comp. Sci.*, 10(3): 418-432. DOI: 10.1007/s11704-016-5554-y
- L. Zou and M. T. Özsu. Graph-based RDF Data Management, *Data Science and Engineering*, 2017. DOI: 10.1007/s41019-016-0029-6

Acknowledgements

This presentation draws upon collaborative research and discussions with the following colleagues (in alphabetical order)


Güneş Aluç, University of Waterloo; now at SAP


Khuzaima Daudjee, University of Waterloo


Olaf Hartig, University of Waterloo; now at Linköping Univ.


Lei Chen, Hong Kong University of Science & Technology


Lei Zou, Peking University

RDF and Semantic Web

- RDF is a language for the conceptual modeling of information about resources (web resources in our context)
- A building block of semantic web
  - Facilitates exchange of information
  - Search engine results can be more focused and structured
  - Facilitates data integration (mashes)
- Machine understandable
  - Understand the information on the web and the interrelationships among them

RDF Uses

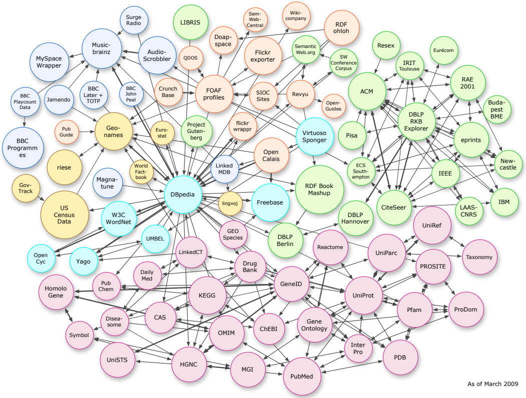
- Yago and DBpedia extract facts from Wikipedia & represent as RDF → structural queries
- Communities build RDF data
  - E.g., biologists: Bio2RDF and Uniprot RDF
- Web data integration
  - Linked Open Data Cloud
- ...

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 3000 datasets with $>$84B triples
  - Size almost doubling every year

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 3000 datasets with >84B triples
  - Size almost doubling every year

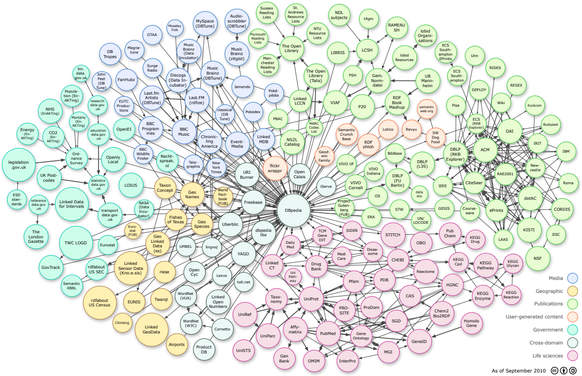


March '09:
89 datasets

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http://lod-cloud.net/

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 3000 datasets with >84B triples
  - Size almost doubling every year

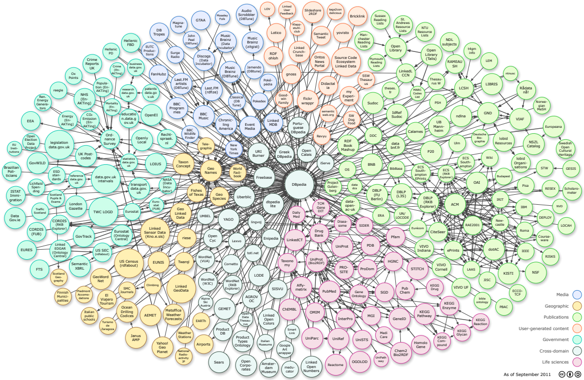


September '10:
203 datasets

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

RDF Data Volumes . . .

- . . . are growing – and fast
  - Linked data cloud currently consists of 3000 datasets with >84B triples
  - Size almost doubling every year

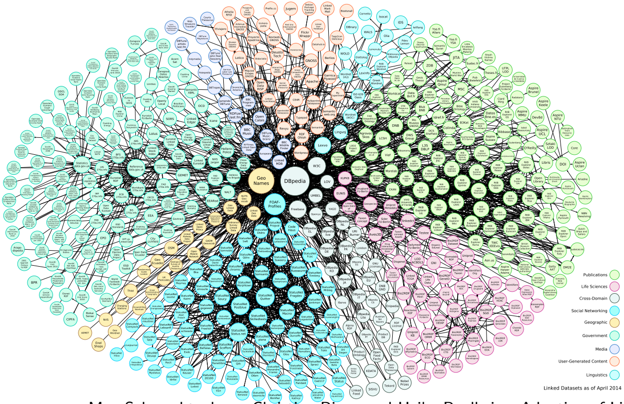


September ’11:
295 datasets, 25B
triples

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http://lod-cloud.net/

RDF Data Volumes . . .

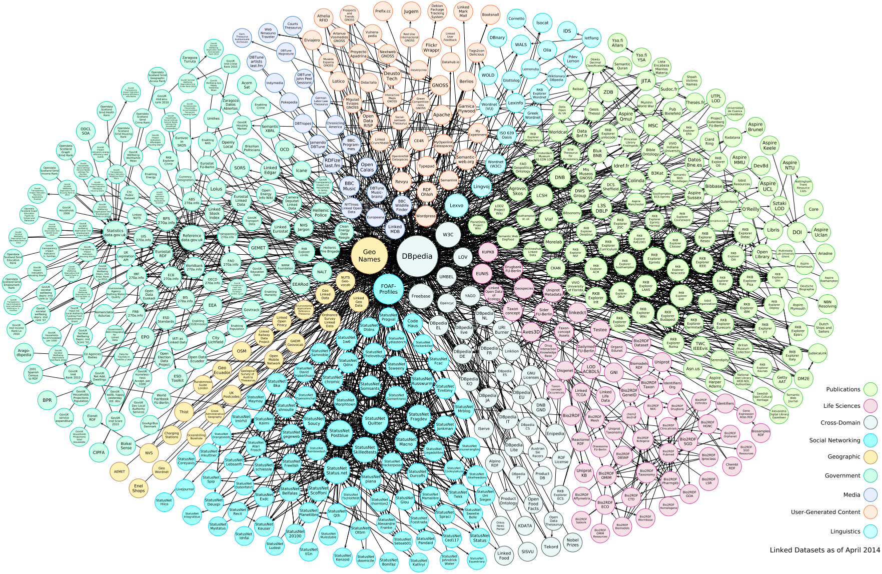
- . . . are growing – and fast
  - Linked data cloud currently consists of 3000 datasets with >84B triples
  - Size almost doubling every year



April ’14:
570 datasets, ???
triples

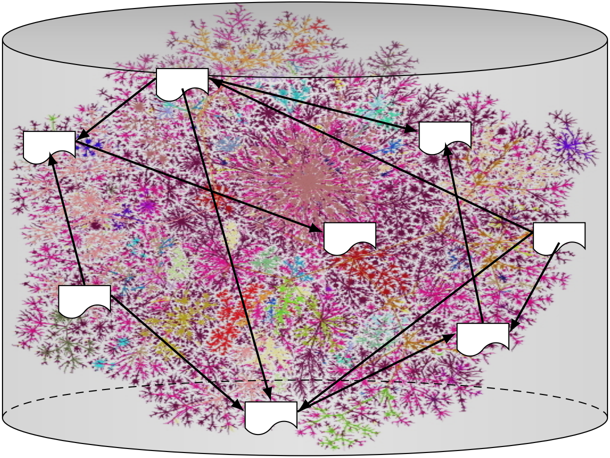
Max Schmachtenberg, Christian Bizer, and Heiko Paulheim: Adoption of Linked Data Best Practices in Different Topical Domains. In *Proc. ISWC*, 2014.

Linked Object Data – Closer Look



Linked Datasets as of April 2014

Globally Distributed Network of Data



Three Approaches

[Özsu, 2016]

- Data warehousing
  - Consolidate data in a centralized repository and query it
- Distributed SPARQL execution
  - Typical distribution
    - Take a warehouse and distribute
    - For scalability and performance
  - SPARQL federation
    - Leverage query services provided by data publishers
- Live Linked Data querying
  - Navigate through LOD by looking up URIs at query execution time

Outline

Outline

RDF Introduction

http://data.linkedmdb.org/resource/actor/JN29704

- Everything is an uniquely named resource

RDF Introduction

xmlns:y=http://data.linkedmdb.org/resource/actor/

y:JN29704

- Everything is an uniquely named resource
- Prefixes can be used to shorten the names

RDF Introduction

- Everything is an uniquely named resource
- Prefixes can be used to shorten the names
- Properties of resources can be defined

xmlns:y=http://data.linkedmdb.org/resource/actor/

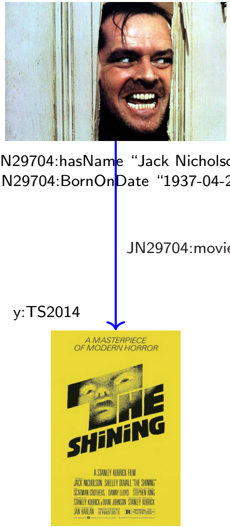
y:JN29704

y:JN29704:hasName "Jack Nicholson"

y:JN29704:BornOnDate "1937-04-22"

RDF Introduction

- Everything is an uniquely named resource
- Prefixes can be used to shorten the names
- Properties of resources can be defined
- Relationships with other resources can be defined



xmlns:y=http://data.linkedmdb.org/resource/actor/

y:JN29704

y:JN29704:hasName “Jack Nicholson”
y:JN29704:BornOnDate “1937-04-22”

JN29704:movieActor

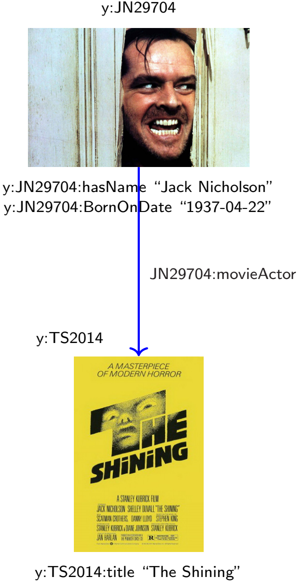
y:TS2014

y:TS2014:title “The Shining”
y:TS2014:releaseDate “1980-05-23”

RDF Introduction

- Everything is an uniquely named resource
- Prefixes can be used to shorten the names
- Properties of resources can be defined
- Relationships with other resources can be defined
- Resource descriptions can be contributed by different people/groups and can be located anywhere in the web
    - Integrated web "database"

xmlns:y=http://data.linkedmdb.org/resource/actor/



y:JN29704

y:JN29704:hasName "Jack Nicholson"
y:JN29704:BornOnDate "1937-04-22"

JN29704:movieActor

y:TS2014

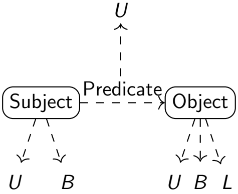
y:TS2014:title "The Shining"
y:TS2014:releaseDate "1980-05-23"

RDF Data Model

- Triple: Subject, Predicate (Property), Object (s, p, o)

Subject:	the entity that is described (URI or blank node)
Predicate:	a feature of the entity (URI)
Object:	value of the feature (URI, blank node or literal)

- $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$
- Set of RDF triples is called an RDF graph



U: set of URIs
B: set of blank nodes
L: set of literals

Subject	**Predicate**	**Object**
http://...imdb.../film/2014	rdfs:label	"The Shining"
http://...imdb.../film/2014	movie:releaseDate	"1980-05-23"
http://...imdb.../29704	movie:actor_name	"Jack Nicholson"
...	...	...

RDF Example Instance

Prefixes: mdb=http://data.linkedmdb.org/resource/; geo=http://sws.geonames.org/
bm=http://wifo5-03.informatik.uni-mannheim.de/bookmashup/
lexvo=http://lexvo.org/id/;wp=http://en.wikipedia.org/wiki/

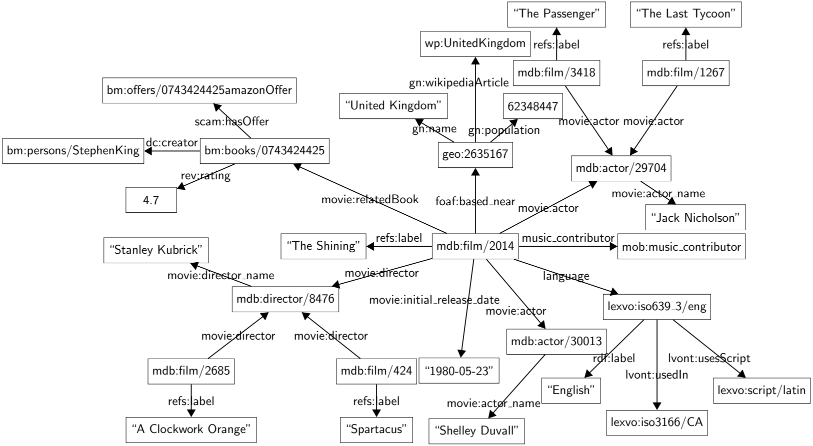
Subject	Predicate	Object
mdb: film/2014	rdfs:label	“The Shining”
mdb:film/2014	movie:initial_release_date	“1980-05-23”’
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	“Stanley Kubrick”
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	“A Clockwork Orange”
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	“Spartacus”
mdb:actor/29704	movie:actor_name	“Jack Nicholson”
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	“The Last Tycoon”
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	“The Passenger”
geo:2635167	gn:name	“United Kingdom”
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	“English”
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

URI

Literal

URI

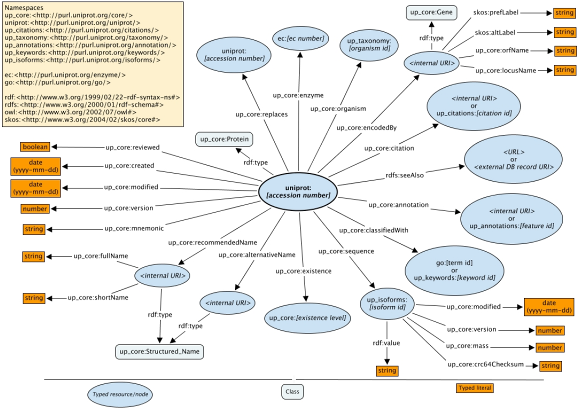
RDF Graph


"The Passenger"
"The Last Tycoon"
wp:UnitedKingdom
refs:label
mdb:film/3418
refs:label
mdb:film/1267
bm:offers/0743424425amazonOffer
gn:wikipediaArticle
"United Kingdom"
62348447
scam:hasOffer
gn:name
gn:population
movie:actor
movie:actor
bm:persons/StephenKing
dc:creator
bm:books/0743424425
geo:2635167
mdb:actor/29704
rev:rating
4.7
movie:relatedBook
foaf:based_near
movie:actor
movie:actor_name
"Jack Nicholson"
"Stanley Kubrick"
"The Shining"
refs:label
mdb:film/2014
music_contributor
mob:music_contributor
movie:director_name
movie:director
language
mdb:director/8476
movie:initial_release_date
movie:actor
lexvo:iso639_3/eng
movie:director
movie:director
mdb:actor/30013
mdb:film/2685
mdb:film/424
"1980-05-23"
rdf:label
lvont:usesScript
lvont:usedIn
"English"
lexvo:script/latin
refs:label
refs:label
movie:actor_name
"A Clockwork Orange"
"Spartacus"
"Shelley Duvall"
lexvo:iso3166/CA

UniProt in RDF

http://www.uniprot.org

- UniProt collects data from >150 biological resources
- Claim: “lack of a common standard to represent and link information makes data integration an expensive business” ⇒ RDF can help

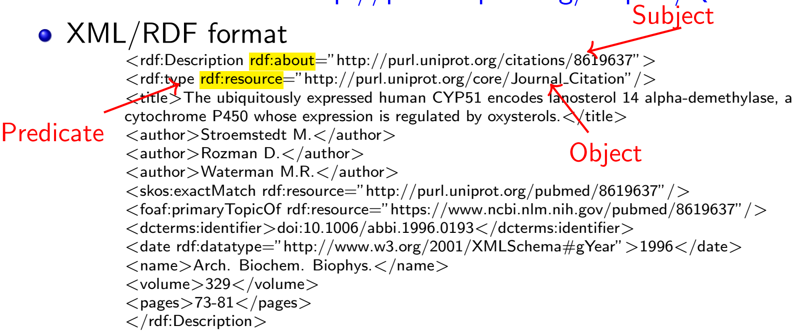


UniProt in RDF – What does the data look like?

- UniProt accession for the human CYP51 protein – Q16850
- Encode it as RDF: http://purl.uniprot.org/uniprot/Q16850.rdf

UniProt in RDF – What does the data look like?

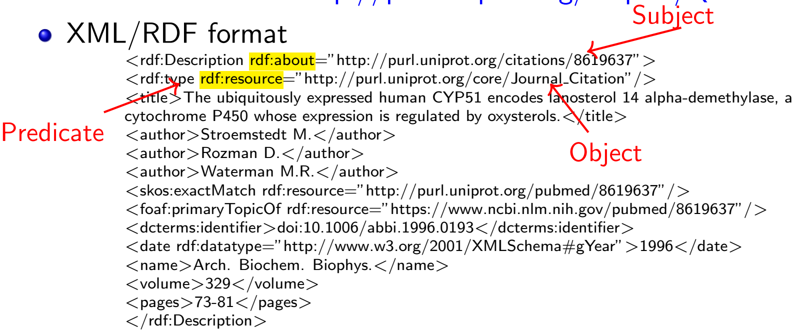
- UniProt accession for the human CYP51 protein – Q16850
- Encode it as RDF: http://purl.uniprot.org/uniprot/Q16850.rdf
- XML/RDF format



```
<rdf:Description rdf:about="http://purl.uniprot.org/citations/8619637">
<rdf:type rdf:resource="http://purl.uniprot.org/core/Journal_Citation"/>
<title>The ubiquitously expressed human CYP51 encodes lanosterol 14 alpha-demethylase, a
cytochrome P450 whose expression is regulated by oxysterols.</title>
<author>Stroemstedt M.</author>
<author>Rozman D.</author>
<author>Waterman M.R.</author>
<skos:exactMatch rdf:resource="http://purl.uniprot.org/pubmed/8619637"/>
<foaf:primaryTopicOf rdf:resource="https://www.ncbi.nlm.nih.gov/pubmed/8619637"/>
<dcterms:identifier>doi:10.1006/abbi.1996.0193</dcterms:identifier>
<date rdf:datatype="http://www.w3.org/2001/XMLSchema#gYear">1996</date>
<name>Arch. Biochem. Biophys.</name>
<volume>329</volume>
<pages>73-81</pages>
</rdf:Description>
```

UniProt in RDF – What does the data look like?

- UniProt accession for the human CYP51 protein – Q16850
- Encode it as RDF: http://purl.uniprot.org/uniprot/Q16850.rdf
- XML/RDF format



Subject

Predicate

Object

```
<rdf:Description rdf:about="http://purl.uniprot.org/citations/8619637">
<rdf:type rdf:resource="http://purl.uniprot.org/core/Journal_Citation"/>
<title>The ubiquitously expressed human CYP51 encodes lanosterol 14 alpha-demethylase, a
cytochrome P450 whose expression is regulated by oxysterols.</title>
<author>Stroemstedt M.</author>
<author>Rozman D.</author>
<author>Waterman M.R.</author>
<skos:exactMatch rdf:resource="http://purl.uniprot.org/pubmed/8619637"/>
<foaf:primaryTopicOf rdf:resource="https://www.ncbi.nlm.nih.gov/pubmed/8619637"/>
<dcterms:identifier>doi:10.1006/abbi.1996.0193</dcterms:identifier>
<date rdf:datatype="http://www.w3.org/2001/XMLSchema#gYear">1996</date>
<name>Arch. Biochem. Biophys.</name>
<volume>329</volume>
<pages>73-81</pages>
</rdf:Description>
```

- This can be shown as a table <Subject, Predicate, Object >

RDF Query Model – SPARQL

- Query Model - **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- Given U (set of URIs), L (set of literals), and V (set of variables), a SPARQL expression is defined recursively:
  - an atomic triple pattern, which is an element of

    $$(U \cup V) \times (U \cup V) \times (U \cup V \cup L)$$

    - ?x rdfs:label "The Shining"
  - P FILTER R, where P is a graph pattern expression and R is a built-in SPARQL condition (i.e., analogous to a SQL predicate)
    - ?x rev:rating ?p FILTER(?p $>$ 3.0)
  - $P1$ AND/OPT/UNION $P2$, where $P1$ and $P2$ are graph pattern expressions

RDF Query Model – SPARQL

- Query Model - **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- Given U (set of URIs), L (set of literals), and V (set of variables), a SPARQL expression is defined recursively:
  - an atomic triple pattern, which is an element of

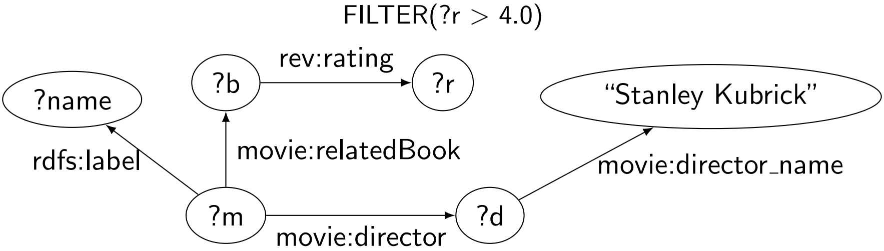
    $$(U \cup V) \times (U \cup V) \times (U \cup V \cup L)$$

    - ?x rdfs:label “The Shining”
  - P FILTER R, where P is a graph pattern expression and R is a built-in SPARQL condition (i.e., analogous to a SQL predicate)
    - ?x rev:rating ?p FILTER(?p $>$ 3.0)
  - $P1$ AND/OPT/UNION $P2$, where $P1$ and $P2$ are graph pattern expressions
- Example:

```
SELECT ?name
WHERE {
  ?m rdfs:label ?name. ?m movie:director ?d.
  ?d movie:director_name "Stanley Kubrick".
  ?m movie:relatedBook ?b. ?b rev:rating ?r.
  FILTER(?r > 4.0)
}
```

SPARQL Queries

```
SELECT ?name
WHERE {
  ?m rdfs:label ?name. ?m movie:director ?d.
  ?d movie:director_name "Stanley Kubrick".
  ?m movie:relatedBook ?b. ?b rev:rating ?r.
  FILTER(?r > 4.0)
}
```



UniProt in RDF – The Data Can Be Queried

- RDF encoded UniProt data can be queried using SPARQL: `http://sparql.uniprot.org/sparql`

UniProt in RDF – The Data Can Be Queried

- RDF encoded UniProt data can be queried using SPARQL:
  `http://sparql.uniprot.org/sparql`
- Get the GO function for Q16850 (from UniProt SPARQL endpoint)

```
PREFIX upc:<http://purl.uniprot.org/core/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?goid ?golabel
WHERE {
  <http://purl.uniprot.org/uniprot/Q16850> a upc:Protein ;
  upc:classifiedWith ?keyword .
  ?keyword rdfs:seeAlso ?goid .
  ?goid rdfs:label ?golabel .
}
```

UniProt in RDF – The Data Can Be Queried

- RDF encoded UniProt data can be queried using SPARQL: http://sparql.uniprot.org/sparql
- Get the GO function for Q16850 (from UniProt SPARQL endpoint)

```
PREFIX upc:<http://purl.uniprot.org/core/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?goid ?golabel
WHERE {
  <http://purl.uniprot.org/uniprot/Q16850> a upc:Protein ;
  upc:classifiedWith ?keyword .
  ?keyword rdfs:seeAlso ?goid .
  ?goid rdfs:label ?golabel .
}
```

- Find the differential expression of probes and the p value that map to Q16850 (from Expression Atlas SPARQL endpoint)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX atlasterms: <http://rdf.ebi.ac.uk/terms/atlas/>
SELECT distinct ?valueLabel ?pvalue
WHERE {
  ?value rdfs:label ?valueLabel .
  ?value atlasterms:pValue ?pvalue .
  ?value atlasterms:isMeasurementOf ?probe .
  ?probe atlasterms:dbXref <http://purl.uniprot.org/uniprot/Q16850> .
}
ORDER BY ASC(?pvalue)
```

Outline

1. RDF Introduction
2. Data Warehousing Approach
   - Relational Approaches
   - Graph-Based Approaches
3. Distributed RDF Processing
   - Cloud-based Solutions
   - Partition-based Approaches
   - Partial Execution Approach
   - SPARQL Endpoint Federation
4. Live Querying Approach
   - Traversal-based approach
   - Index-based approach
   - Hybrid approach
5. Conclusions

Naïve Triple Store Design

```
SELECT ?name
WHERE {
  ?m rdfs:label ?name. ?m movie:director ?d.
  ?d movie:director_name "Stanley Kubrick".
  ?m movie:relatedBook ?b. ?b rev:rating ?r.
  FILTER(?r > 4.0)
}
```

Subject	Property	Object
mdb:film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:initial_release_date	"1980-05-23"
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	"Stanley Kubrick"
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	"Spartacus"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	"The Last Tycoon"
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	"The Passenger"
geo:2635167	gn:name	"United Kingdom"
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	"English"
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

Naïve Triple Store Design

```
SELECT ?name
WHERE {
  ?m rdfs:label ?name. ?m movie:director ?d.
  ?d movie:director_name "Stanley Kubrick".
  ?m movie:relatedBook ?b. ?b rev:rating ?r.
  FILTER(?r > 4.0)
}
```

Subject	Property	Object
mdb:film/2014	rdfs:label	“The Shining”
mdb:film/2014	movie:initial_release_date	“1980-05-23”
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	“Stanley Kubrick”
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	“A Clockwork Orange”
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	“Spartacus”
mdb:actor/29704	movie:actor_name	“Jack Nicholson”
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	“The Last Tycoon”
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	“The Passenger”
geo:2635167	gn:name	“United Kingdom”
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	“English”
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

```
SELECT T1.object
FROM
T as T1, T as T2, T as T3,
      T as T4, T as T5
WHERE T1.p="rdfs:label"
AND T2.p="movie:relatedBook"
AND T3.p="movie:director"
AND T4.p="rev:rating"
AND T5.p="movie:director_name"
AND T1.s=T2.s
AND T1.s=T3.s
AND T2.o=T4.s
AND T3.o=T5.s
AND T4.o > 4.0
AND T5.o="Stanley Kubrick"
```

Naïve Triple Store Design

```
SELECT ?name
WHERE {
  ?m rdfs:label ?name. ?m movie:director ?
  ?d movie:director_name "Stanley K
  ?m movie:relatedBook ?b. ?b rev:r
  FILTER(?r > 4.0)
}
```

Easy to implement
but
too many self-joins!

Subject	Property	Object
mdb:film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:initial_release_date	"1980-05-23"
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	"Stanley Kubrick"
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	"Spartacus"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	"The Last Tycoon"
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	"The Passenger"
geo:2635167	gn:name	"United Kingdom"
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	"English"
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

```
         ,        T5
WHERE T1.p="rdfs:
AND T2.p="movie:rel     Book"
AND T3.p="movie:director"
AND T4.p="rev:rating"
AND T5.p="movie:director_name"
AND T1.s=T2.s
AND T1.s=T3.s
AND T2.o=T4.s
AND T3.o=T5.s
AND T4.o > 4.0
AND T5.o="Stanley Kubrick"
```

Exhaustive Indexing

- RDF-3X [Neumann and Weikum, 2008, 2009], Hexastore [Weiss et al., 2008]
- Strings are mapped to ids using a mapping table

Original triple table

Subject	**Property**	**Object**
mdb: film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:initial_release_date	"1980-05-23"
mdb:director/8476	movie:director_name	"Stanley Kubrick"
mdb:film/2685	movie:director	mdb:director/8476

Mapping table

ID	**Value**
0	mdb: film/2014
1	rdfs:label
2	"The Shining"
3	movie:initial_release_date
4	"1980-05-23"
5	mdb:director/8476
6	movie:director_name
7	"Stanley Kubrick"
8	mdb:film/2685
9	movie:director

Encoded triple table

Subject	**Property**	**Object**
0	1	2
0	3	4
5	6	7
8	9	5

Exhaustive Indexing

- RDF-3X [Neumann and Weikum, 2008, 2009], Hexastore [Weiss et al., 2008]
- Strings are mapped to ids using a mapping table
- Triples are indexed in a clustered B+ tree in lexicographic order

Subject	**Property**	**Object**
0	1	2
0	3	4
5	6	7
8	9	5
⋮	⋮	⋮

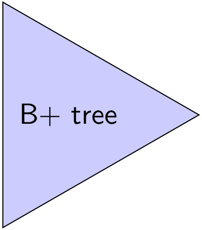
B+ tree

Easy querying through mapping table

Exhaustive Indexing

- RDF-3X [Neumann and Weikum, 2008, 2009], Hexastore [Weiss et al., 2008]
- Strings are mapped to ids using a mapping table
- Triples are indexed in a clustered B+ tree in lexicographic order
- Create indexes for permutations of the three columns: SPO, SOP, PSO, POS, OPS, OSP

Subject	**Property**	**Object**
0	1	2
0	3	4
5	6	7
8	9	5
⋮	⋮	⋮



Easy querying through mapping table

Exhaustive Indexing–Query Execution

- Each triple pattern can be answered by a range query
- Joins between triple patterns computed using merge join
- Join order is easy due to extensive indexing

Subject	**Property**	**Object**
0	1	2
0	3	4
5	6	7
8	9	5
⋮	⋮	⋮

ID	**Value**
0	mdb: film/2014
1	rdfs:label
2	"The Shining"
3	movie:initial_release_date
4	"1980-05-23"
5	mdb:director/8476
6	movie:director_name
7	"Stanley Kubrick"
8	mdb:film/2685
9	movie:director

Exhaustive Indexing–Query Execution

- Each triple pattern can be answered by a range query
- Joins between triple patterns computed using merge join
- Join order is easy due to extensive indexing

Advantages

- Eliminates some of the joins – they become range queries
- Merge join is easy and fast

8	9	5
⋮	⋮	⋮

3	movie:initial_release_date
4	“1980-05-23”
5	mdb:director/8476
6	movie:director_name
7	“Stanley Kubrick”
8	mdb:film/2685
9	movie:director

Exhaustive Indexing–Query Execution

- Each triple pattern can be answered by a range query
- Joins between triple patterns computed using merge join
- Join order is easy due to extensive indexing

Advantages

- Eliminates some of the joins – they become range queries
- Merge join is easy and fast

8	9	5

3	movie:initial_release_date
9	movie:director

Disadvantages

- Space usage
- Expensive updates

Property Tables

- Grouping by entities; Jena [Wilkinson, 2006], DB2-RDF [Bornea et al., 2013]
- *Clustered property table*: group together the properties that tend to occur in the same (or similar) subjects
- *Property-class table*: cluster the subjects with the same type of property into one property table

Subject	**Property**	**Object**
mdb:film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
...	...	...

Subject	**refs:label**	**movie:director**
mob:film/2014	"The Shining"	mob:director/8476
mob:film/2685	"The Clockwork Orange"	mob:director/8476

Subject	**movie:actor_name**
mdb:actor	"Jack Nicholson"

Property Tables

- Grouping by entities; Jena [Wilkinson, 2006], DB2-RDF [Bornea et al., 2013]
- *Clustered property table*: group together the properties that

Advantages

- Fewer joins
- If the data is structured, we have a relational system – similar to normalized relations

mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
...	...	...

Subject	**refs:label**	**movie:director**
mob:film/2014	"The Shining"	mob:director/8476
mob:film/2685	"The Clockwork Orange"	mob:director/8476

Subject	**movie:actor_name**
mdb:actor	"Jack Nicholson"

Property Tables

- Grouping by entities; Jena [Wilkinson, 2006], DB2-RDF [Bornea et al., 2013]
- *Clustered property table*: group together the properties that

Advantages

- Fewer joins
- If the data is structured, we have a relational system – similar to normalized relations

mdb:film/2685	movie:director	mdb:director/8476

Disadvantages

- Potentially a lot of NULLs
- Clustering is not trivial
- Multi-valued properties are complicated

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects [Abadi et al., 2007, 2009]
- Also called vertical partitioned tables
- n two column tables (n is the number of unique properties in the data)

Subject	**Property**	**Object**
mdb:film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
...	...	...

refs:label

Subject	**Object**
mob:film/2014	"The Shining"
mob:film/2685	"The Clockwork Orange"

movie:director

Subject	**Object**
mdb:film/2014	mdb:director/8476
mdb:film/2685	mdb:director/8476

movie:actor_name

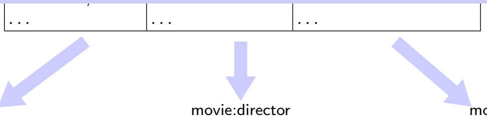
Subject	**Object**
mdb:actor/29704	"Jack Nicholso

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects [Abadi et al., 2007, 2009]

Advantages

- Supports multi-valued properties
- No NULLs
- No clustering
- Read only needed attributes (i.e. less I/O)
- Good performance for subject-subject joins



refs:label

Subject	Object
mob:film/2014	“The Shining”
mob:film/2685	“The Clockwork Orange”

movie:director

Subject	Object
mdb:film/2014	mdb:director/8476
mdb:film/2685	mdb:director/8476

movie:actor_name

Subject	Object
mdb:actor/29704	“Jack Nicholso

Binary Tables

- Grouping by properties: For each property, build a two-column table, containing both subject and object, ordered by subjects

Advantages

- Supports multi-valued properties
- No NULLs
- No clustering
- Read only needed attributes (i.e. less I/O)
- Good performance for subject-subject joins

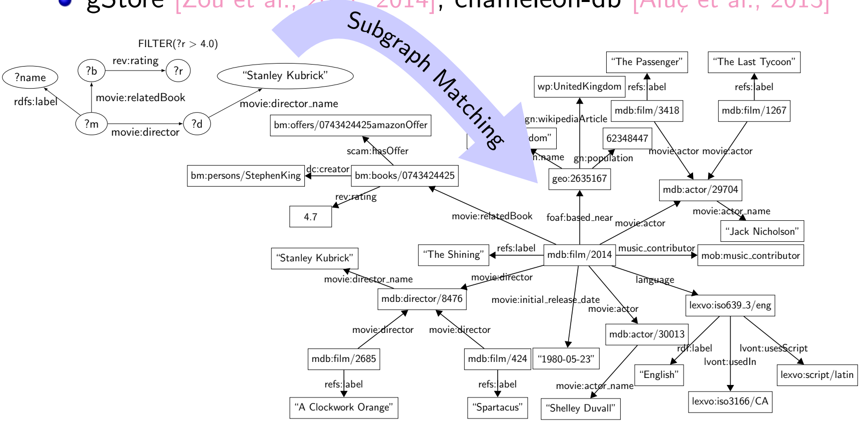
Disadvantages

- Not useful for subject-object joins
- Expensive inserts

Graph-based Approach

[Zou and Özsu, 2017]

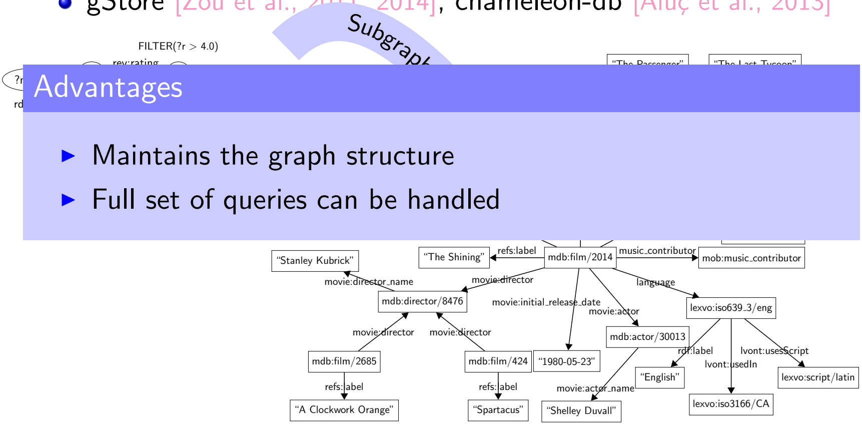
- Answering SPARQL query ≡ subgraph matching using homomorphism
- gStore [Zou et al., 2011, 2014], chameleon-db [Aluç et al., 2013]



Graph-based Approach

[Zou and Özsu, 2017]

- Answering SPARQL query ≡ subgraph matching using homomorphism
- gStore [Zou et al., 2011, 2014], chameleon-db [Aluç et al., 2013]



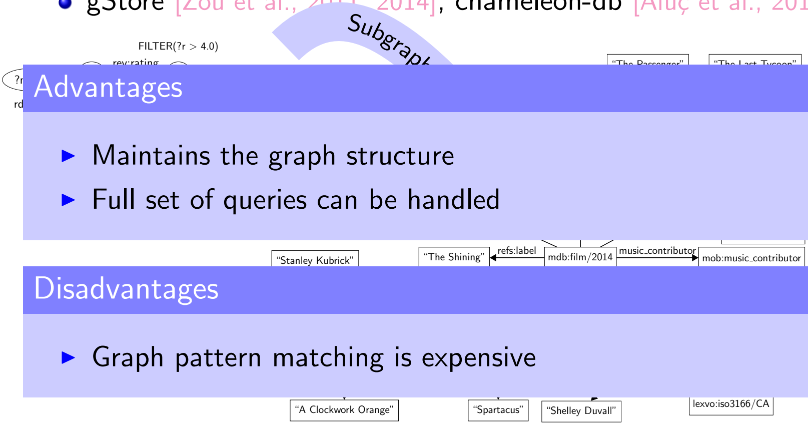
Advantages

- Maintains the graph structure
- Full set of queries can be handled

Graph-based Approach

[Zou and Özsu, 2017]

- Answering SPARQL query ≡ subgraph matching using homomorphism
- gStore [Zou et al., 2011, 2014], chameleon-db [Aluç et al., 2013]



Advantages

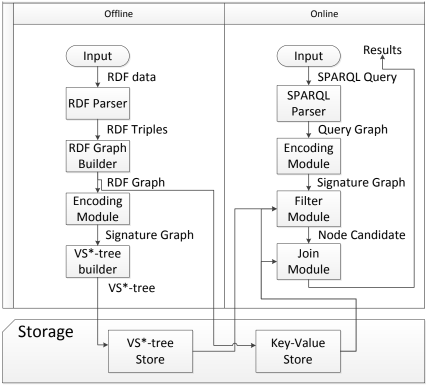
- Maintains the graph structure
- Full set of queries can be handled

Disadvantages

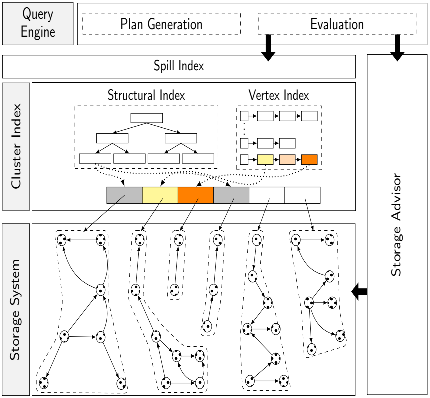
- Graph pattern matching is expensive

Two Systems

gStore

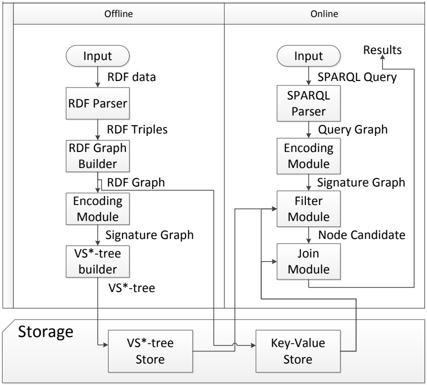


chameleon-db

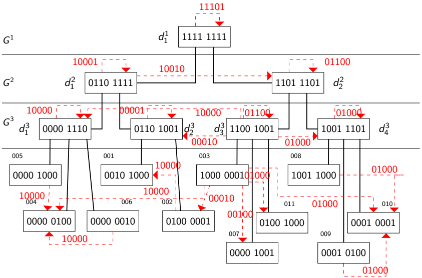


Two Systems

gStore

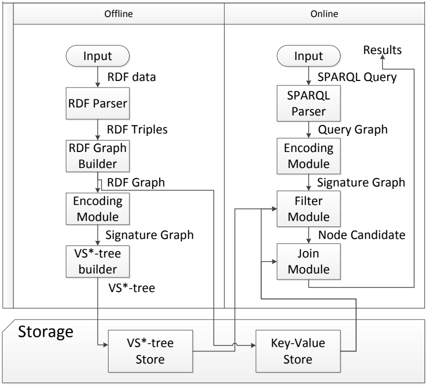


- 12,000 lines of C++ code under Linux (plus code for SPARQL parser)
- Encode each vertex of RDF graph as a bit array capturing the neighbourhood relationship (G^*)
- Build a multilevel summary tree index (VS*-tree) to capture "connections"



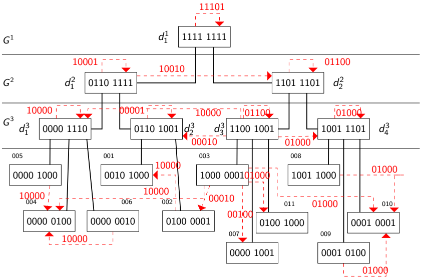
Two Systems

gStore



- Encode the query graph similarly (Q^*)
- Find candidate matching nodes of Q^* in G^* using VS*-tree
- Multiway join of the candidates

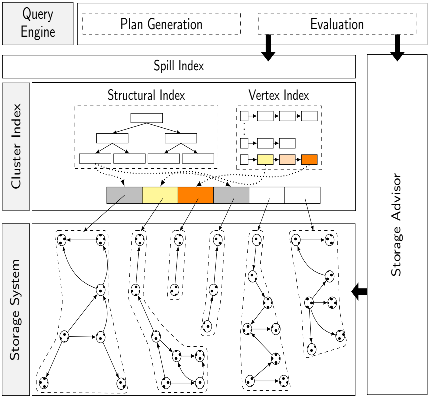
- 12,000 lines of C++ code under Linux (plus code for SPARQL parser)
- Encode each vertex of RDF graph as a bit array capturing the neighbourhood relationship (G^*)
- Build a multilevel summary tree index (VS*-tree) to capture “connections”



Two Systems

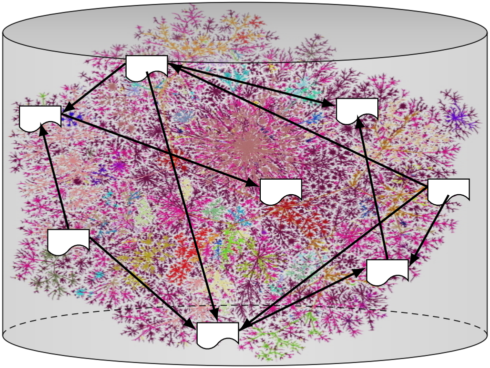
- 35,000 lines of C++ code under Linux (plus code for SPARQL 1.0 parser)
- Adaptivity to workload due to variability of Web workloads and the variability of composition of SPARQL triple patterns
- An experiment [Aluç et al., 2014a]
    - No single system is a sole winner across all queries
    - No single system is the sole loser across all queries, either
    - 2–5 orders of magnitude difference in the performance between the best and the worst system for a given query
    - The winner in one query may timeout in another
    - Performance difference widens as dataset size increases
- Group-by-query approach [Aluç et al., 2014b]

chameleon-db



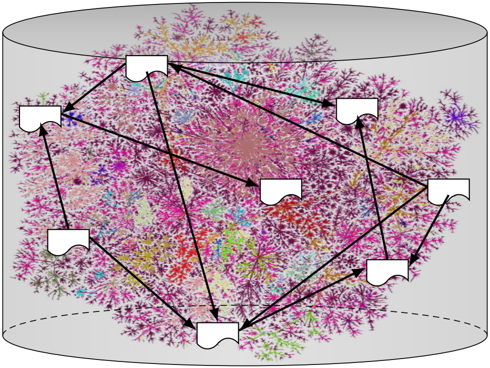
Outline

Remember the Environment



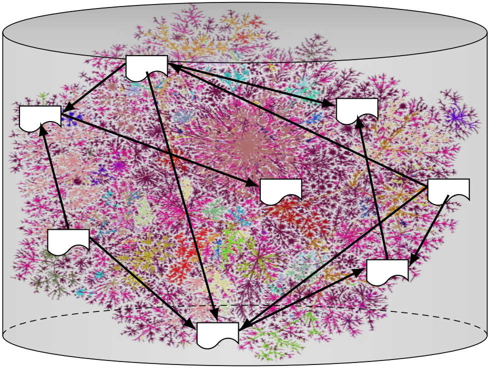
- Distributed environment
- Some of the data sites can process SPARQL queries – SPARQL endpoints
- Not all data sites can process queries

Remember the Environment



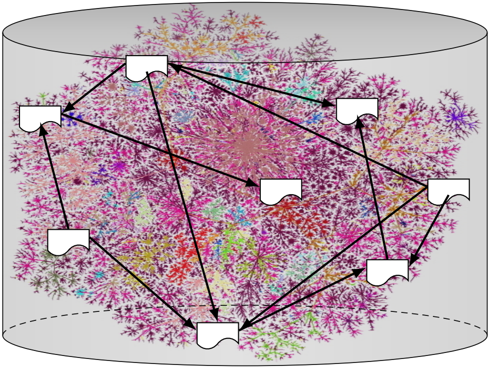
- Distributed environment
- Some of the data sites can process SPARQL queries – SPARQL endpoints
- Not all data sites can process queries
- Alternatives

Remember the Environment



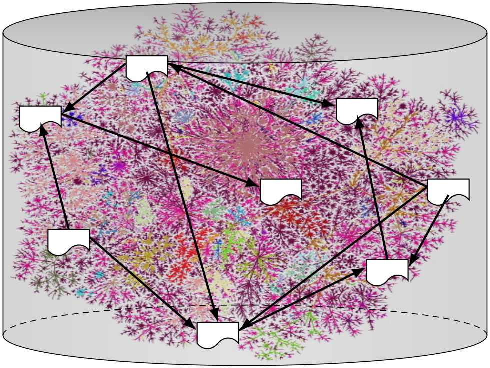
- Distributed environment
- Some of the data sites can process SPARQL queries – SPARQL endpoints
- Not all data sites can process queries
- Alternatives
  - Cloud-based approaches

Remember the Environment



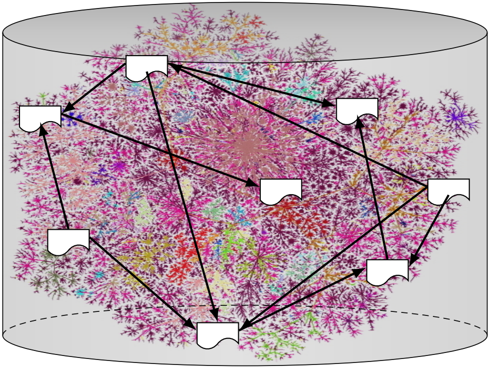
- Distributed environment
- Some of the data sites can process SPARQL queries – SPARQL endpoints
- Not all data sites can process queries
- Alternatives
  - Cloud-based approaches
  - Data re-distribution + query decomposition

Remember the Environment



- Distributed environment
- Some of the data sites can process SPARQL queries – SPARQL endpoints
- Not all data sites can process queries
- Alternatives
    - Cloud-based approaches
    - Data re-distribution + query decomposition
    - Data re-distribution + partial evaluation

Remember the Environment



- Distributed environment
- Some of the data sites can process SPARQL queries – SPARQL endpoints
- Not all data sites can process queries
- Alternatives
    - Cloud-based approaches
    - Data re-distribution + query decomposition
    - Data re-distribution + partial evaluation
    - SPARQL federation: just process at SPARQL endpoints

Cloud-based Solutions

[Kaoudi and Manolescu, 2015]

- RDF data warehouse D is partitioned ($\{D_1, \ldots, D_n\}$) and placed on cloud platforms (such as HDFS, HBase)

Cloud-based Solutions

[Kaoudi and Manolescu, 2015]

- RDF data warehouse D is partitioned ($\{D_1, \ldots, D_n\}$) and placed on cloud platforms (such as HDFS, HBase)
- SPARQL query is run through MapReduce jobs
- Data parallel execution

Cloud-based Solutions

[Kaoudi and Manolescu, 2015]

- RDF data warehouse D is partitioned ($\{D_1, \ldots, D_n\}$) and placed on cloud platforms (such as HDFS, HBase)
- SPARQL query is run through MapReduce jobs
- Data parallel execution
- Examples: HARD [Rohloff and Schantz, 2010] , HadoopRDF [Husain et al., 2011] , EAGRE [Zhang et al., 2013] and JenaHBase [Khadilkar et al., 2012]

Cloud-based Solutions

[Kaoudi and Manolescu, 2015]

- RDF data warehouse D is partitioned ($\{D_1, \ldots, D_n\}$) and placed on cloud platforms (such as HDFS, HBase)
- SPARQL query is run through MapReduce jobs
- Data parallel execution
- Examples: HARD [Rohloff and Schantz, 2010] , HadoopRDF [Husain et al., 2011] , EAGRE [Zhang et al., 2013] and JenaHBase [Khadilkar et al., 2012]

- High scalability and fault-tolerance
- Possibly low performance since MapReduce is not suitable for graph processing

Partition-based Approaches

- (Offline) Partition an RDF data warehouse (graph) into several fragments that are distributed to sites
  - RDF data $D = \{D_1, \ldots, D_n\}$
  - Allocate each D_i to a site

Partition-based Approaches

- (Offline) Partition an RDF data warehouse (graph) into several fragments that are distributed to sites
    - RDF data $D = \{D_1, \ldots, D_n\}$
    - Allocate each D_i to a site
- Partitioning alternatives
    - Table-based (e.g., [Husain et al., 2011])
    - Graph-based (e.g., [Huang et al., 2011; Zhang et al., 2013])
    - Unit-based (e.g., [Gurajada et al., 2014; Lee and Liu, 2013])

Partition-based Approaches

- (Offline) Partition an RDF data warehouse (graph) into several fragments that are distributed to sites
  - RDF data $D = \{D_1, \ldots, D_n\}$
  - Allocate each D_i to a site
- Partitioning alternatives
  - Table-based (e.g., [Husain et al., 2011])
  - Graph-based (e.g., [Huang et al., 2011; Zhang et al., 2013])
  - Unit-based (e.g., [Gurajada et al., 2014; Lee and Liu, 2013])
- (Online) SPARQL query decomposed $Q = \{Q_1, \ldots, Q_k\} \Rightarrow$ query graph is decomposed
- Distributed execution of $\{Q_1, \ldots, Q_k\}$ over $\{D_1, \ldots, D_n\}$

Partition-based Approaches

- (Offline) Partition an RDF data warehouse (graph) into several fragments that are distributed to sites
  - RDF data $D = \{D_1, \ldots, D_n\}$
  - Allocate each D_i to a site
- Partitioning alternatives
  - Table-based (e.g., [Husain et al., 2011])
  - Graph-based (e.g., [Huang et al., 2011; Zhang et al., 2013])
  - Unit-based (e.g., [Gurajada et al., 2014; Lee and Liu, 2013])
- (Online) SPARQL query decomposed $Q = \{Q_1, \ldots, Q_k\} \Rightarrow$ query graph is decomposed
- Distributed execution of $\{Q_1, \ldots, Q_k\}$ over $\{D_1, \ldots, D_n\}$
- Examples: GraphPartition [Huang et al., 2011], WARP [Hose and Schenkel, 2013] , Partout [Galarraga et al., 2014] , Vertex-block [Lee and Liu, 2013]

Partition-based Approaches

- (Offline) Partition an RDF data warehouse (graph) into several fragments that are distributed to sites
  - RDF data $D = \{D_1, \ldots, D_n\}$
  - Allocate each D_i to a site
- Partitioning alternatives
  - Table-based (e.g., [Husain et al., 2011])

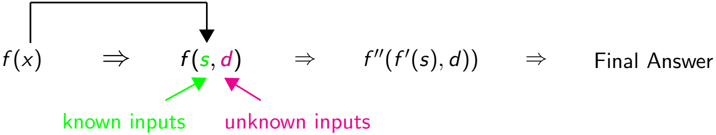
- High performance
- Great for parallelizing centralized RDF data
- May not be possible to re-partition and re-allocate Web data (i.e., LOD)
- Each approach requires a specific partitioning strategy – no generic partitioning
- Query decomposition may not be easy

Partial Query Evaluation (PQE)

- RDF data warehouse is partitioned and distributed as before
  - RDF data $D = \{D_1, \ldots, D_n\}$
  - Allocate each D_i to a site
- SPARQL query is not decomposed
- Partial query evaluation – Distributed gStore [Peng et al., 2016]

Partial Query Evaluation (PQE)

- RDF data warehouse is partitioned and distributed as before
    - RDF data $D = \{D_1, \ldots, D_n\}$
    - Allocate each D_i to a site
- SPARQL query is not decomposed
- Partial query evaluation – Distributed gStore [Peng et al., 2016]



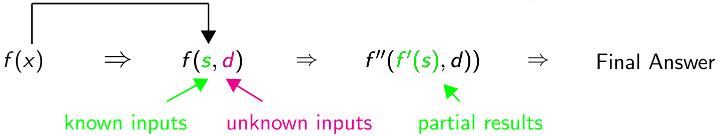
$f(x) \Rightarrow f(s, d) \Rightarrow f''(f'(s), d)) \Rightarrow$ Final Answer

known inputs

unknown inputs

Partial Query Evaluation (PQE)

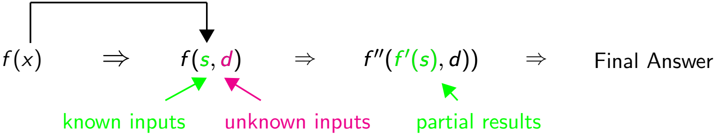
- RDF data warehouse is partitioned and distributed as before
  - RDF data $D = \{D_1, \ldots, D_n\}$
  - Allocate each D_i to a site
- SPARQL query is not decomposed
- Partial query evaluation – Distributed gStore [Peng et al., 2016]



$$f(x) \quad \Rightarrow \quad f(s, d) \quad \Rightarrow \quad f''(f'(s), d) \quad \Rightarrow \quad \text{Final Answer}$$

Partial Query Evaluation (PQE)

- RDF data warehouse is partitioned and distributed as before
    - RDF data $D = \{D_1, \ldots, D_n\}$
    - Allocate each D_i to a site
- SPARQL query is not decomposed
- Partial query evaluation – Distributed gStore [Peng et al., 2016]



$f(x) \quad \Rightarrow \quad f(s, d) \quad \Rightarrow \quad f''(f'(s), d) \quad \Rightarrow$ Final Answer

known inputs unknown inputs partial results

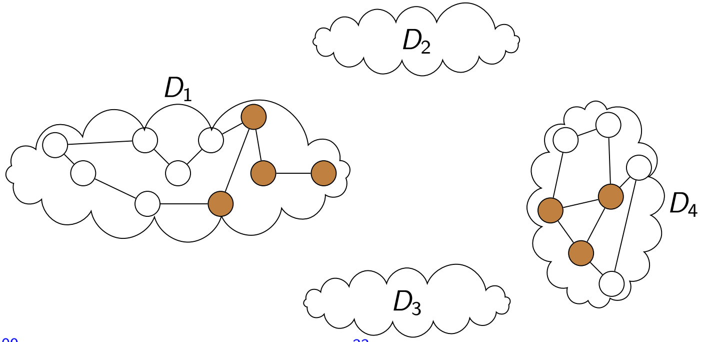
- Query is the function and each D_i is the known input

Distributed SPARQL Using PQE

[Peng et al., 2016]

Two steps:

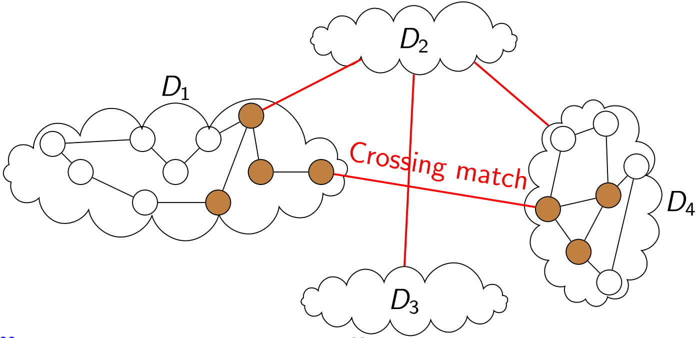
1. Evaluate a query at each site to find local matches
   - These are local partial matches



Distributed SPARQL Using PQE [Peng et al., 2016]

Two steps:

1. Evaluate a query at each site to find local matches
   - These are local partial matches
2. Assemble the partial matches to get final result
   - Crossing match
   - Centralized assembly
   - Distributed assembly



Distributed SPARQL Using PQE

[Peng et al., 2016]

Two steps:

1. Evaluate a query at each site to find local matches
    - These are local partial matches
2. Assemble the partial matches to get final result
    - Crossing match
    - Centralized assembly
    - Distributed assembly

- High performance due to parallelization
- Do not have to deal with query decomposition
- May not be possible to re-partition and re-allocate Web data (i.e., LOD)
- RDF storage sites need to be modified to handle partial query processing

SPARQL Endpoint Federation

- Consider only the SPARQL endpoints for query execution
- No data re-partitioning/re-distribution
- Consider $D = D_1 \cup D_2 \cup \ldots \cup D_n$; D_i: SPARQL endpoint

SPARQL Endpoint Federation

- Consider only the SPARQL endpoints for query execution
- No data re-partitioning/re-distribution
- Consider $D = D_1 \cup D_2 \cup \ldots \cup D_n$; D_i: SPARQL endpoint
- SPARQL query decomposed $Q = \{Q_1, \ldots, Q_k\}$
- Distributed execution of $\{Q_1, \ldots, Q_k\}$ over $\{D_1, \ldots, D_n\}$

SPARQL Endpoint Federation

- Consider only the SPARQL endpoints for query execution
- No data re-partitioning/re-distribution
- Consider $D = D_1 \cup D_2 \cup \ldots \cup D_n$; D_i: SPARQL endpoint
- SPARQL query decomposed $Q = \{Q_1, \ldots, Q_k\}$
- Distributed execution of $\{Q_1, \ldots, Q_k\}$ over $\{D_1, \ldots, D_n\}$
- Systems
  - DARQ [Quilitz and Leser, 2008], FedX [Schwarte et al., 2011], SPLENDID [Görlitz and Staab, 2011], ANAPSID [Acosta et al., 2011]

SPARQL Endpoint Federation

- Consider only the SPARQL endpoints for query execution
- No data re-partitioning/re-distribution
- Consider $D = D_1 \cup D_2 \cup \ldots \cup D_n$; D_i: SPARQL endpoint
- SPARQL query decomposed $Q = \{Q_1, \ldots, Q_k\}$
- Distributed execution of $\{Q_1, \ldots, Q_k\}$ over $\{D_1, \ldots, D_n\}$
- Systems
  - DARQ [Quilitz and Leser, 2008], FedX [Schwarte et al., 2011], SPLENDID [Görlitz and Staab, 2011], ANAPSID [Acosta et al., 2011]

- Data integration approach
- May be the only way to proceed if data is distributed
- Not all RDF data storage points are SPARQL endpoints

UniProt Federation – EBI RDF Platform

BioModels	Curated computational models of biological processes
BioSamples	Sample information for reference samples and samples for which data exist in one of the EBI's assay databases
ChEMBL	Curated chemical database of bioactive molecules with drug-like properties
Ensembl	Genome databases for vertebrates and other eukaryotic species
Expression Atlas	Gene expression data from the Gene Expression Atlas
Reactome	Curated and peer-reviewed pathways
UniProt	

Federated Access to UniProt Collection

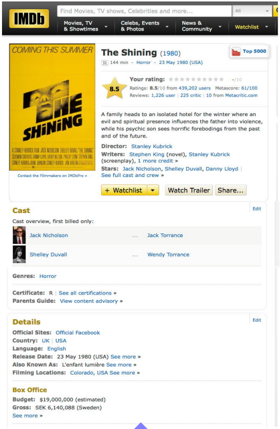
Get the Reactome pathways where Q16850 is associated, then get all the other proteins in that pathway and pull out their expression from the atlas, along with the GO annotations from UniProt

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX biopax3: <http://www.biopax.org/release/biopax-level3.owl#>
PREFIX atlasterms: <http://rdf.ebi.ac.uk/terms/atlas/>
PREFIX upc:<http://purl.uniprot.org/core/>
SELECT DISTINCT ?pathwayname ?expressionValue ?golabel
WHERE {
 # Get the pathways that reference Q16850
 ?pathway rdf:type biopax3:Pathway .
 ?pathway biopax3:displayName ?pathwayname .
 ?pathway biopax3:pathwayComponent
 [?rel [biopax3:entityReference ?dbXref]] .
 ?pathway biopax3:pathwayComponent
 [?rel [biopax3:entityReference <http://purl.uniprot.org/uniprot/Q16850>]] .

 # Get the expression for those proteins
 SERVICE <http://www.ebi.ac.uk/rdf/services/atlas/sparql> {
  ?value rdfs:label ?expressionValue .
  ?value atlasterms:pValue ?pvalue .
  ?value atlasterms:isMeasurementOf ?probe .
  ?probe atlasterms:dbXref ?dbXref .
 }

 # get the GO functions from Uniprot
 SERVICE <http://uniprot.org/sparql> {
  ?dbXref a upc:Protein ;
  upc:classifiedWith ?keyword .
  ?keyword rdfs:seeAlso ?goid .
  ?goid rdfs:label ?golabel .
 }
}
```

Outline

1. RDF Introduction
2. Data Warehousing Approach
   - Relational Approaches
   - Graph-Based Approaches
3. Distributed RDF Processing
   - Cloud-based Solutions
   - Partition-based Approaches
   - Partial Execution Approach
   - SPARQL Endpoint Federation
4. **Live Querying Approach**
   - **Traversal-based approach**
   - Index-based approach
   - Hybrid approach
5. Conclusions

Traditional Hypertext-based Web Access



The Shining (1980)

R 144 min - Horror - 23 May 1980 (USA)

Ratings: 8.5/10 from 439,202 users Metascore: 61/100
Reviews: 1,226 user | 225 critic | 10 from Metacritic.com

A family heads to an isolated hotel for the winter where an evil and spiritual presence influences the father into violence, while his psychic son sees horrific forebodings from the past and of the future.

Director: Stanley Kubrick
Writers: Stephen King (novel), Stanley Kubrick (screenplay), 1 more credit »
Stars: Jack Nicholson, Shelley Duvall, Danny Lloyd | See full cast and crew »

Contact the Filmmakers on IMDbPro »

Cast

Cast overview, first billed only:

Jack Nicholson	...	Jack Torrance
Shelley Duvall	...	Wendy Torrance

Genres: Horror

Certificate: R | See all certifications »
Parents Guide: View content advisory »

Details

Official Sites: Official Facebook
Country: UK | USA
Language: English
Release Date: 23 May 1980 (USA) See more »
Also Known As: L'enfant lumière See more »
Filming Locations: Colorado, USA See more »

Box Office

Budget: $19,000,000 (estimated)
Gross: SEK 6,140,088 (Sweden)
See more »

PAGE LAST UPDATED ON JUNE 20, 2014

Introduction :: UNITED KINGDOM

Background:

The United Kingdom has historically played a leading role in developing parliamentary democracy and in advancing literature and science. At its zenith in the 19th century, the British Empire stretched over one-fourth of the earth's surface. The first half of the 20th century saw the UK's strength seriously depleted in two world wars and the Irish Republic's withdrawal from the union. The second half witnessed the dismantling of the Empire and the UK rebuilding itself into a modern and prosperous European nation. As one of five permanent members of the UN Security Council and a founding member of NATO and the Commonwealth, the UK pursues a global approach to foreign policy. The UK is also an active member of the EU, although it chose to remain outside the Economic and Monetary Union. The Scottish Parliament, the National Assembly for Wales, and the Northern Ireland Assembly were established in 1999. The latter was suspended until May 2007 due to wrangling over the peace process, but devolution was fully completed in March 2010.

Geography :: UNITED KINGDOM

Location:

Western Europe, islands - including the northern one-sixth of the island of Ireland - between the North Atlantic Ocean and the North Sea; northwest of France

Geographic coordinates:

54 00 N, 2 00 W

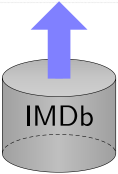
Map references:

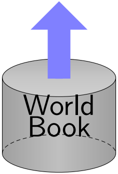
Europe

Area:

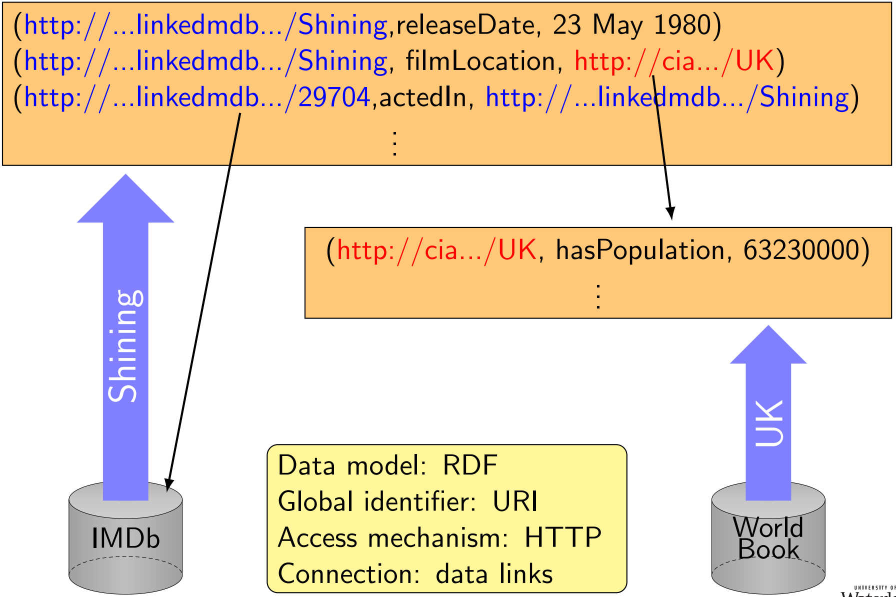
total: 243,610 sq km
country comparison to the world: 80

Data exposed to the Web via HTML





Linked Data Publishing Principles

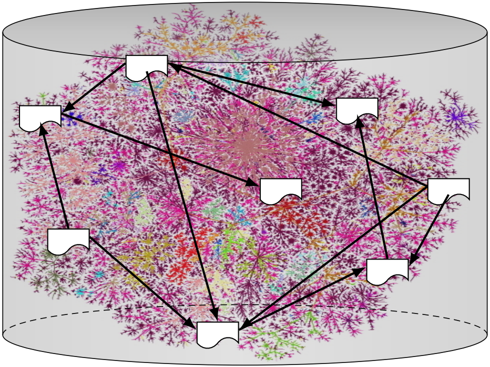


(http://...linkedmdb.../Shining,releaseDate, 23 May 1980)
(http://...linkedmdb.../Shining, filmLocation, http://cia.../UK)
(http://...linkedmdb.../29704,actedIn, http://...linkedmdb.../Shining)
⋮

(http://cia.../UK, hasPopulation, 63230000)
⋮

Data model: RDF
Global identifier: URI
Access mechanism: HTTP
Connection: data links

Live Query Processing



- Not all data resides at SPARQL endpoints
- Freshness of access to data important
- Potentially countably infinite data sources
- Live querying
  - On-line execution
  - Only rely on linked data principles
- Alternatives
  - Traversal-based approaches
  - Index-based approaches
  - Hybrid approaches

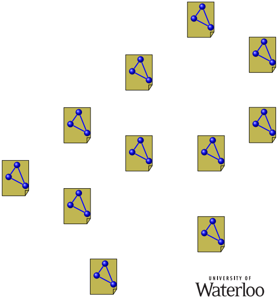
Linked Data Model

[Hartig, 2012]

Web of Linked Data

Given a finite or countably infinite set $\mathcal{D}$ of Linked Documents, a Web of Linked Data is a tuple $W = (D, adoc, data)$ where:

- $D \subseteq \mathcal{D}$,
- *adoc* is a partial mapping from URIs to D, and
- *data* is a total mapping from D to finite sets of RDF triples.



Linked Data Model

[Hartig, 2012]

Web of Linked Data

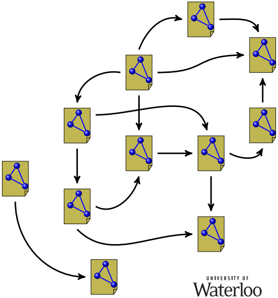
Given a finite or countably infinite set $\mathcal{D}$ of Linked Documents, a Web of Linked Data is a tuple $W = (D, adoc, data)$ where:

- $D \subseteq \mathcal{D}$,
- *adoc* is a partial mapping from URIs to D, and
- *data* is a total mapping from D to finite sets of RDF triples.

Data Links

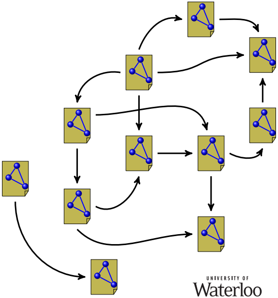
A Web of Linked Data $W = (D, adoc, data)$ contains a data link from document $d \in D$ to document $d' \in D$ if there exists a URI u such that:

- u is mentioned in an RDF triple $t \in data(d)$, and
- $d' = adoc(u)$.



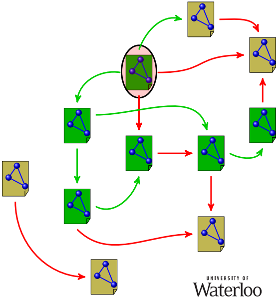
SPARQL Query Semantics in Live Querying

- Full-web semantics
  - Scope of evaluating a SPARQL expression is **all** Linked Data
  - Query result completeness cannot be guaranteed by any (terminating) execution



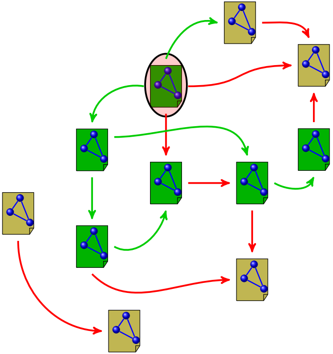
SPARQL Query Semantics in Live Querying

- Full-web semantics
  - Scope of evaluating a SPARQL expression is **all** Linked Data
  - Query result completeness cannot be guaranteed by any (terminating) execution
- Reachability-based query semantics
  - Query consists of a SPARQL expression, a set of seed URIs *S*, and a reachability condition *c*
  - Scope: all data along paths of data links that satisfy the condition
  - Computationally feasible



Traversal Approaches

- Discover relevant URIs recursively by traversing (specific) data links at query execution runtime [Hartig, 2013; Ladwig and Tran, 2011]
- Implements reachability-based query semantics
  - Start from a set of seed URIs
  - Recursively follow and discover new URIs
- Important issue is selection of seed URIs
- Retrieved data serves to discover new URIs and to construct result



Traversal Approaches

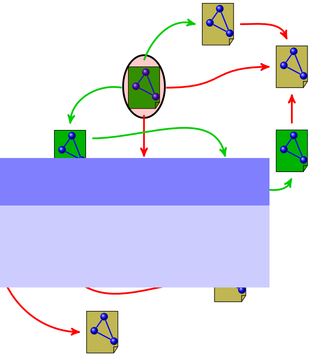
- Discover relevant URIs recursively by traversing (specific) data links at query execution runtime [Hartig, 2013; Ladwig and Tran, 2011]
- Implements reachability-based query semantics

Advantages

Easy to implement.

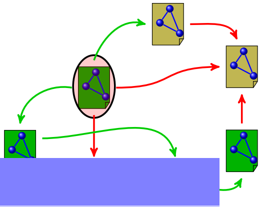
No data structure to maintain.

- Important issue is selection of seed URIs
- Retrieved data serves to discover new URIs and to construct result



Traversal Approaches

- Discover relevant URIs recursively by traversing (specific) data links at query execution runtime [Hartig, 2013; Ladwig and Tran, 2011]
- Implements reachability-based query semantics



Advantages

Easy to implement.

No data structure to maintain.

- Important issue is selection of seed

Disadvantages

Possibilities for parallelized data retrieval are limited

Repeated data retrieval introduces significant query latency.

Outline

Conclusions

- RDF and Linked Object Data seem to have considerable promise and have found use
- There are prototype systems that provide alternative solutions
- There are commercial systems too: e.g., MarkLogic, Virtuoso
- Listings: `http://db-engines.com/en/ranking/rdf+store` & `https://www.w3.org/wiki/SparqlImplementations`

Conclusions

- RDF and Linked Object Data seem to have considerable promise and have found use
- There are prototype systems that provide alternative solutions
- There are commercial systems too: e.g., MarkLogic, Virtuoso
- Listings: `http://db-engines.com/en/ranking/rdf+store` & `https://www.w3.org/wiki/SparqlImplementations`
- More work needs to be done (and I did not talk about these)
  - Query semantics
  - Adaptive system design
  - Optimizations – both in data warehousing and distributed environments
  - Live querying requires significant thought to reduce latency

Conclusions

- RDF and Linked Object Data seem to have considerable promise and have found use
- There are prototype systems that provide alternative solutions
- There are commercial systems too: e.g., MarkLogic, Virtuoso
- Listings: `http://db-engines.com/en/ranking/rdf+store` & `https://www.w3.org/wiki/SparqlImplementations`
- More work needs to be done (and I did not talk about these)
  - Query semantics
  - Adaptive system design
  - Optimizations – both in data warehousing and distributed environments
  - Live querying requires significant thought to reduce latency
- What I did not talk about:
  - Not much on general distributed/parallel processing
  - Not much on SPARQL semantics
  - Nothing about RDFS – no schema stuff
  - Nothing about entailment regimes $> 0 \Rightarrow$ no reasoning

Thank you!

Research supported by

NSERC
CRSNG

INNOVATION.CA
CANADA FOUNDATION FOR INNOVATION
FONDATION CANADIENNE POUR L'INNOVATION

Ontario
MINISTRY OF RESEARCH AND INNOVATION
MINISTÈRE DE LA RECHERCHE ET DE L'INNOVATION

Google

IBM Canada Lab
CAS

References I

Abadi, D. J., Marcus, A., Madden, S., and Hollenbach, K. (2009). SW-Store: a vertically partitioned DBMS for semantic web data management. *VLDB J.*, 18(2):385–406.

Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2007). Scalable semantic web data management using vertical partitioning. In *Proc. 33rd Int. Conf. on Very Large Data Bases*, pages 411–422.

Acosta, M., Vidal, M.-E., Lampo, T., Castillo, J., and Ruckhaus, E. (2011). ANAPSID: an adaptive query processing engine for SPARQL endpoints. In *Proc. 10th Int. Semantic Web Conf.*, pages 18–34.

Aluç, G., Hartig, O., Özsu, M. T., and Daudjee, K. (2014a). Diversified stress testing of RDF data management systems. In *Proc. 13th Int. Semantic Web Conf.*, pages 197–212.

Aluç, G., Özsu, M. T., and Daudjee, K. (2014b). Workload matters: Why RDF databases need a new design. *Proc. VLDB Endowment*, 7(10):837–840.

References II

Aluç, G., Özsu, M. T., Daudjee, K., and Hartig, O. (2013). chameleon-db: a workload-aware robust RDF data management system. Technical Report CS-2013-10, University of Waterloo. Available at `https://cs.uwaterloo.ca/sites/ca.computer-science/files/uploads/files/CS-2013-10.pdf`.

Bornea, M. A., Dolby, J., Kementsietsidis, A., Srinivas, K., Dantressangle, P., Udrea, O., and Bhattacharjee, B. (2013). Building an efficient RDF store over a relational database. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 121–132.

Galarraga, L., Hose, K., and Schenkel, R. (2014). Partout: a distributed engine for efficient RDF processing. In *Proc. 23rd Int. World Wide Web Conf. (Companion Volume)*, pages 267–268.

Görlitz, O. and Staab, S. (2011). SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In *Proc. 2nd Int. Workshop on Consuming Linked Data*.

References III

Gurajada, S., Seufert, S., Miliaraki, I., and Theobald, M. (2014). TriAD: A distributed shared-nothing RDF engine based on asynchronous message passing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 289–300.

Hartig, O. (2012). SPARQL for a web of linked data: Semantics and computability. In *Proc. 9th Extended Semantic Web Conf.*, pages 8–23.

Hartig, O. (2013). SQUIN: a traversal based query execution system for the web of linked data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1081–1084.

Hose, K. and Schenkel, R. (2013). WARP: Workload-aware replication and partitioning for RDF. In *Proc. Workshops of 29th Int. Conf. on Data Engineering*, pages 1–6.

Huang, J., Abadi, D. J., and Ren, K. (2011). Scalable SPARQL querying of large RDF graphs. *Proc. VLDB Endowment*, 4(11):1123–1134.

Husain, M. F., McGlothlin, J., Masud, M. M., Khan, L. R., and Thuraisingham, B. (2011). Heuristics-based query processing for large RDF graphs using cloud computing. *IEEE Trans. Knowl. and Data Eng.*, 23(9):1312–1327.

References IV

Kaoudi, Z. and Manolescu, I. (2015). RDF in the clouds: A survey. *VLDB J.*, 24:67–91.

Khadilkar, V., Kantarcioglu, M., Thuraisingham, B. M., and Castagna, P. (2012). Jena-HBase: A distributed, scalable and efficient RDF triple store. In *Proc. International Semantic Web Conference Posters & Demos Track*.

Ladwig, G. and Tran, T. (2011). SIHJoin: Querying remote and local linked data. In *Proc. 8th Extended Semantic Web Conf.*, pages 139–153.

Lee, K. and Liu, L. (2013). Scaling queries over big rdf graphs with semantic hash partitioning. *Proc. VLDB Endowment*, 6(14):1894–1905.

Neumann, T. and Weikum, G. (2008). RDF-3X: a RISC-style engine for RDF. *Proc. VLDB Endowment*, 1(1):647–659.

Neumann, T. and Weikum, G. (2009). The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113.

Özsu, M. T. (2016). A survey of RDF data management systems. *Front. Comput. Sci.*, 10(3):418–432.

Peng, P., Zou, L., Özsu, M. T., Chen, L., and Zhao, D. (2016). Processing SPARQL queries over distributed RDF graphs. *VLDB J.*, 25(2):243–268.

References V

Quilitz, B. and Leser, U. (2008). Querying distributed RDF data sources with SPARQL. In *Proc. 5th European Semantic Web Conf.*, pages 524–538.

Rohloff, K. and Schantz, R. E. (2010). High-performance, massively scalable distributed systems using the mapreduce software framework: the shard triple-store. In *Proc. Int. Workshop on Programming Support Innovations for Emerging Distributed Applications*. Article No. 4.

Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M. (2011). Fedx: A federation layer for distributed query processing on linked open data. In *Proc. 8th Extended Semantic Web Conf.*, pages 481–486.

Weiss, C., Karras, P., and Bernstein, A. (2008). Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endowment*, 1(1):1008–1019.

Wilkinson, K. (2006). Jena property table implementation. Technical Report HPL-2006-140, HP Laboratories Palo Alto.

Zhang, X., Chen, L., Tong, Y., and Wang, M. (2013). EAGRE: Towards scalable I/O efficient SPARQL query evaluation on the cloud. In *Proc. 29th Int. Conf. on Data Engineering*, pages 565–576.

References VI

Zou, L., Mo, J., Chen, L., Özsu, M. T., and Zhao, D. (2011). gStore: answering SPARQL queries via subgraph matching. *Proc. VLDB Endowment*, 4(8):482–493.

Zou, L. and Özsu, M. T. (2017). Graph-based rdf data management. *Data Science and Engineering*, pages 1–15. Available from: `http:https://dx.doi.org/10.1007/s41019-016-0029-6`.

Zou, L., Özsu, M. T., Chen, L., Shen, X., Huang, R., and Zhao, D. (2014). gStore: A graph-based SPARQL query engine. *VLDB J.*, 23(4):565–590.