Design of Distributed Storage Manager for Large-Scale RDF Graphs

Iztok Savnik
University of Primorska & Jožef Stefan Institute

Kiyoshi Nitta
Yahoo Japan Research

GraphSM, 2014

Aims

- Storage manager for large-scale RDF graphs
  - Storing and querying peta (10^{15}) triples
- Using graph data model
  - RDF and Linked Data
  - Other models: JSON, XML, ...
- Momentum:
  - From hyper-text Web to data Web
  - From HTML to RDF and graphs

Outline

1) Current state of graph DBs
2) Challanges in designing big3store
3) Design of big3store
4) Algebra of graphs
5) Implementation of big3store
6) Conclusions

Current state of graph DBs

Terminology

- Linked data
  - Linked Open Data
- Open data
- Graph databases
- Knowledge bases
- Knowledge graphs

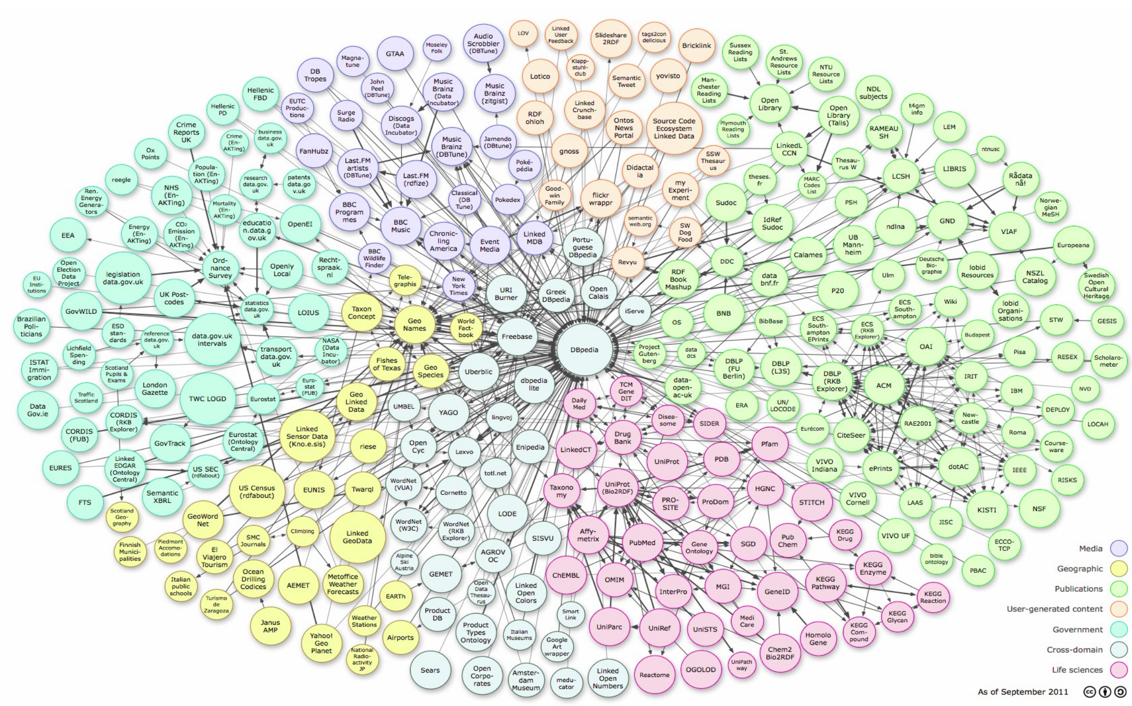
Wordnet

- Princeton's large lexical database of English.
  - Cognitve synonims: synsets ≡ concepts
    - 117,000 synsets
  - Synsets are linked by:
    - conceptual-semantic relationships, and
    - lexical relationships.
    - Include definitions of synsets.
  - Main relationships:
    - Synonymy, hyponymy (ISA), meronymy (part-whole), antonymy

Linked Open Data

LINKINGOPENDATA
W3C SWEO Community Project

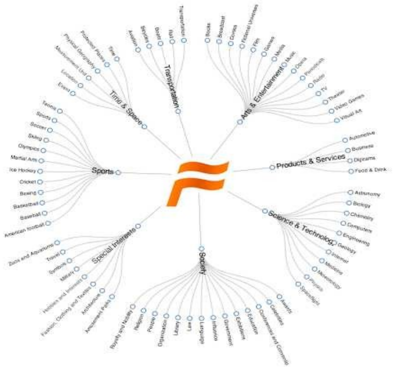
- Datasets are represented in RDF
  - Wikipedia, Wikibooks, Geonames, MusicBrainz, WordNet, DBLP bibliography
- Number of triples: 33 Giga (10^9) (2011)
- Governments:
  - USA, UK, Japan, Austria, Belgium, France, Germany, ...
- Active community



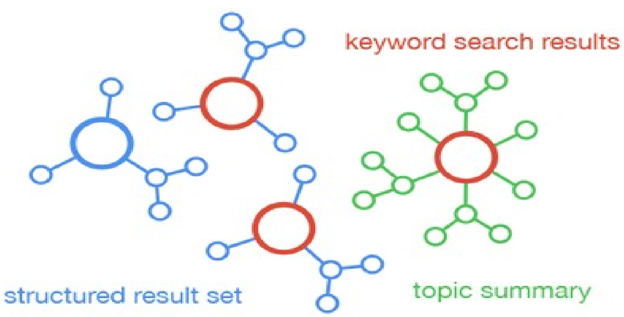
http://en.wikipedia.org/wiki/Open_Data

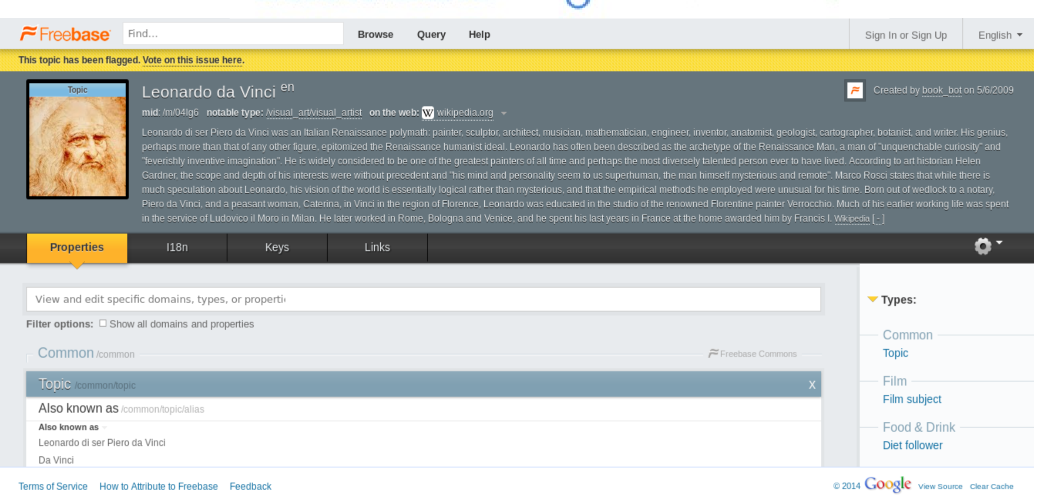
http://www.w3.org/LOD

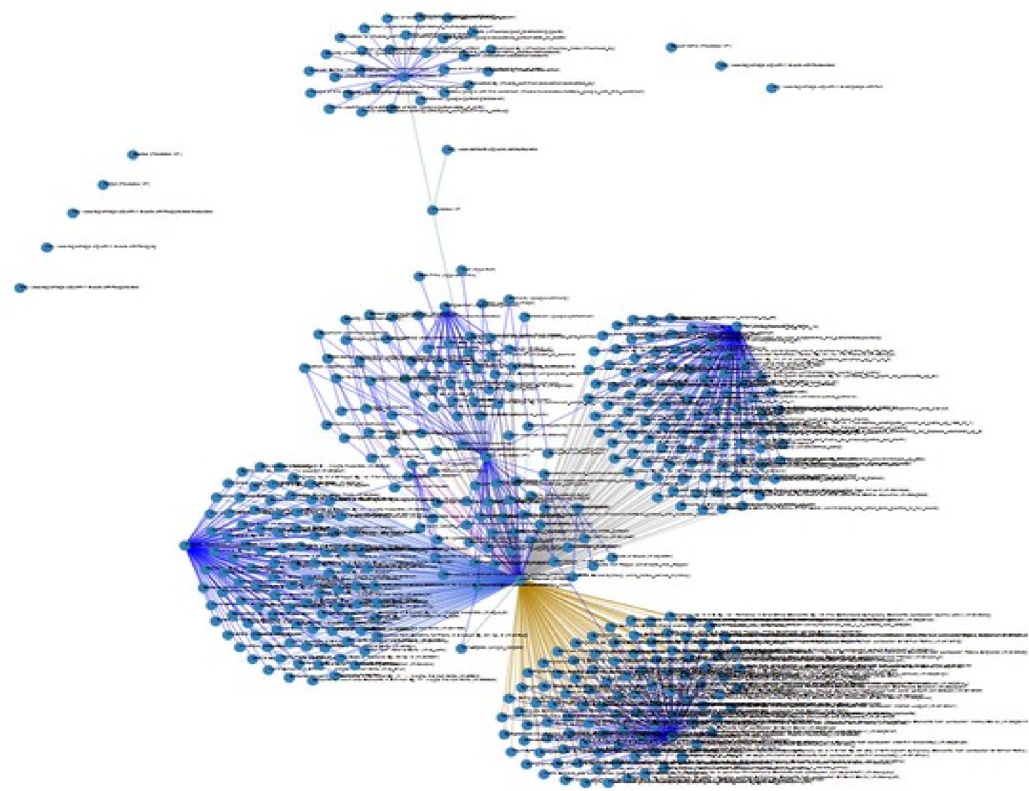
Freebase



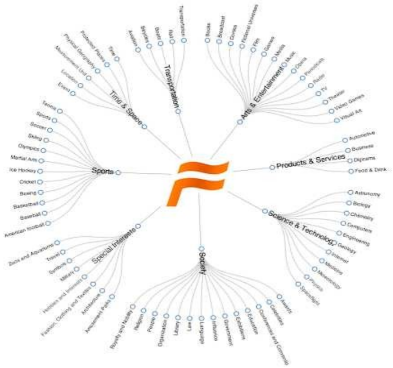
- Free, knowledge graph:
  - people, places and things,
  - 2,478,168,612 facts, 43,459,442 topics
- Semantic search engines are here !







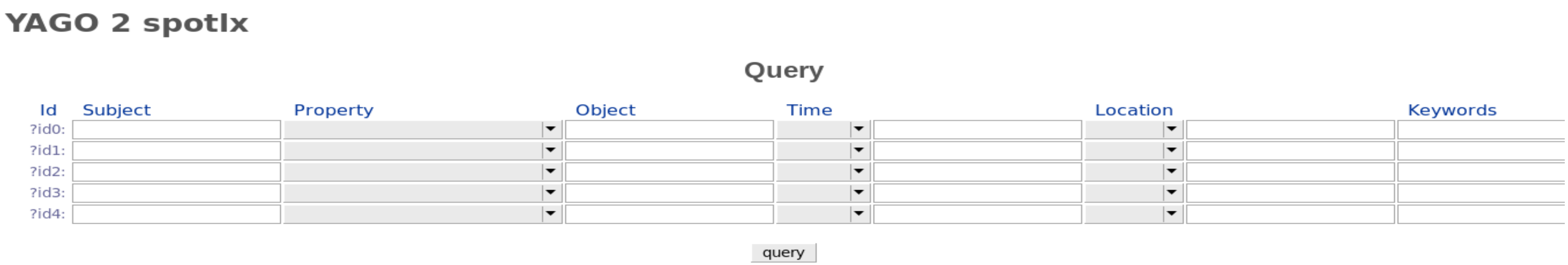
Freebase



- Based on graphs:
  - nodes, links, types, properties, namespaces
- Google use of Freebase
  - Knowledge graph
  - Words become concepts
  - Semantic questions
  - Semantic associations
  - Browsing knowledge
  - Knowledge engine
- Available in RDF

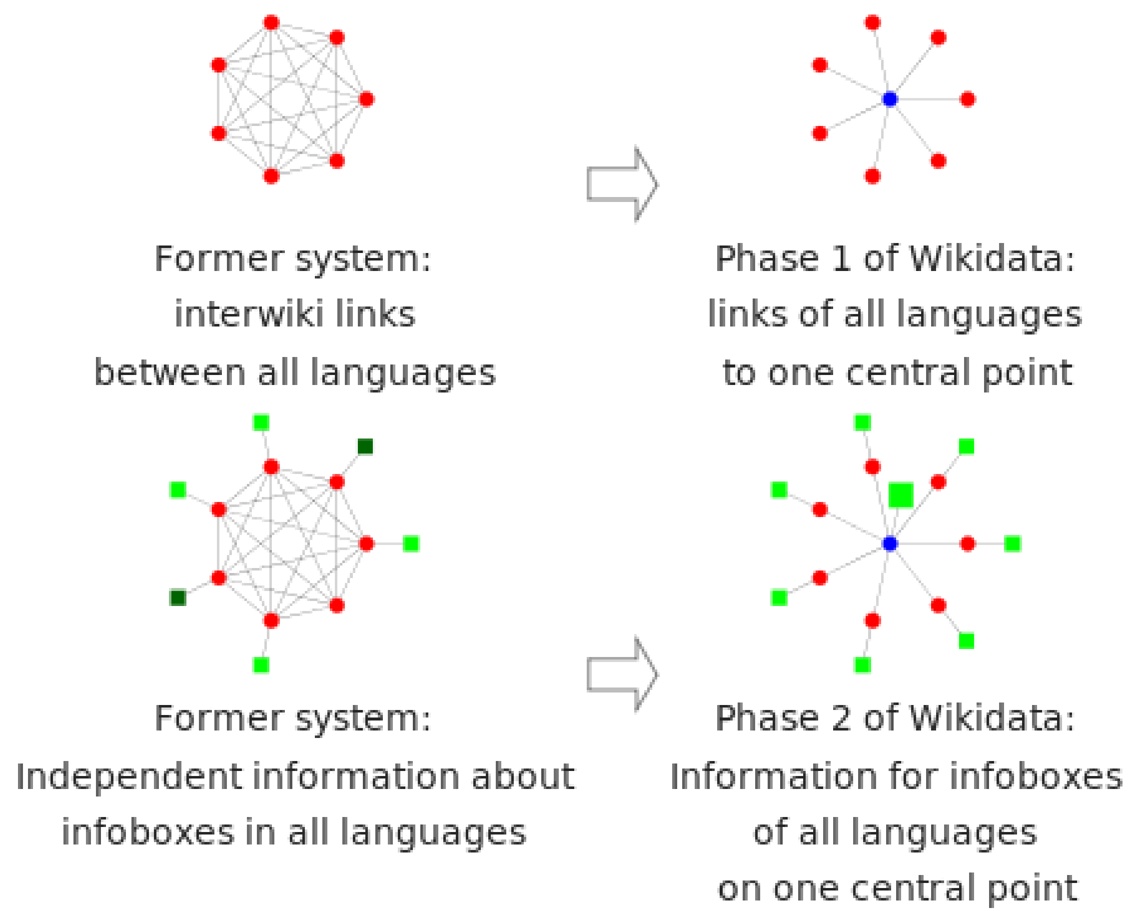
YAGO

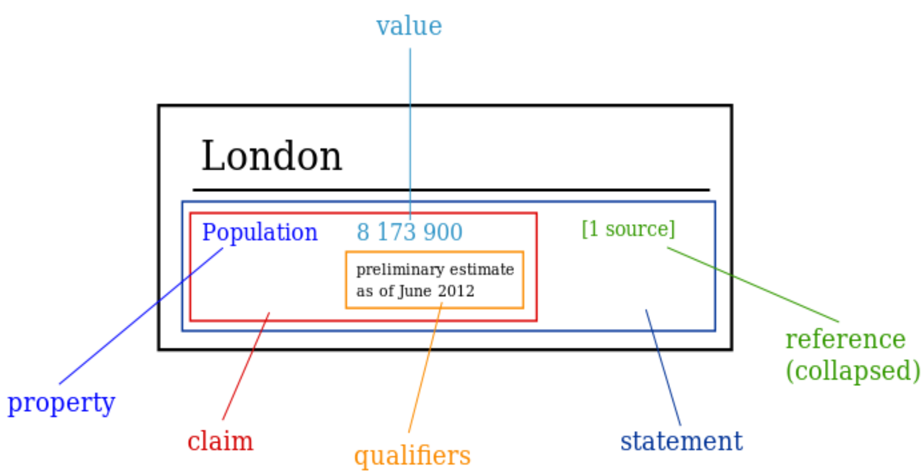
- 10 Mega (10^6) concepts
  - Max Planc Institute, Informatik
  - Accuracy of 95%
- Includes:
  - Wikipedia, WordNet, GeoNames
  - Links Wordnet to Wikipedia taxonomy (350K concepts)
  - Anchored in time and space



Wikidata

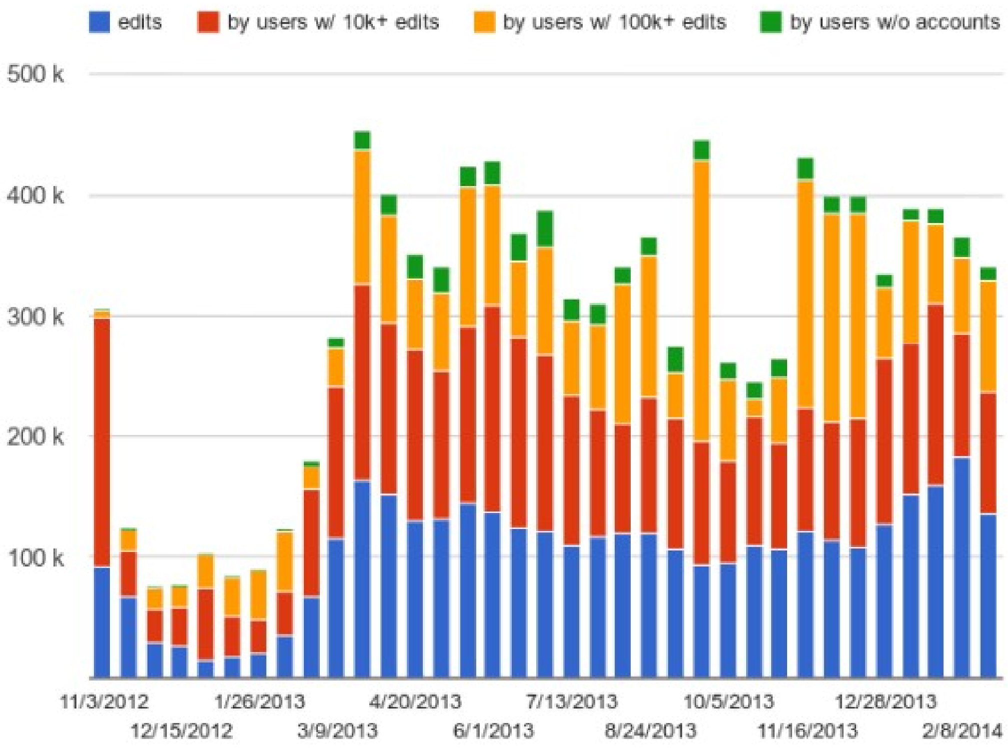
- Free knowledge base with 14,550,852 items
- Collecting structured data
- Properties of
  - person, organization, works, events, etc.

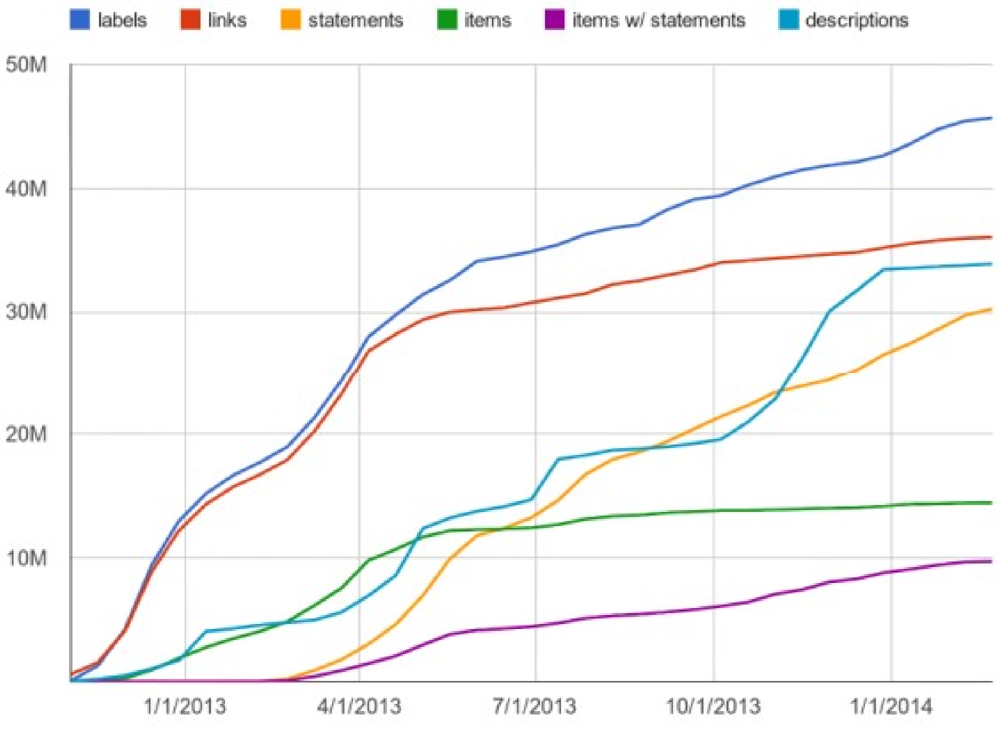




Wikidata

- Free knowledge base with 14,550,852 items

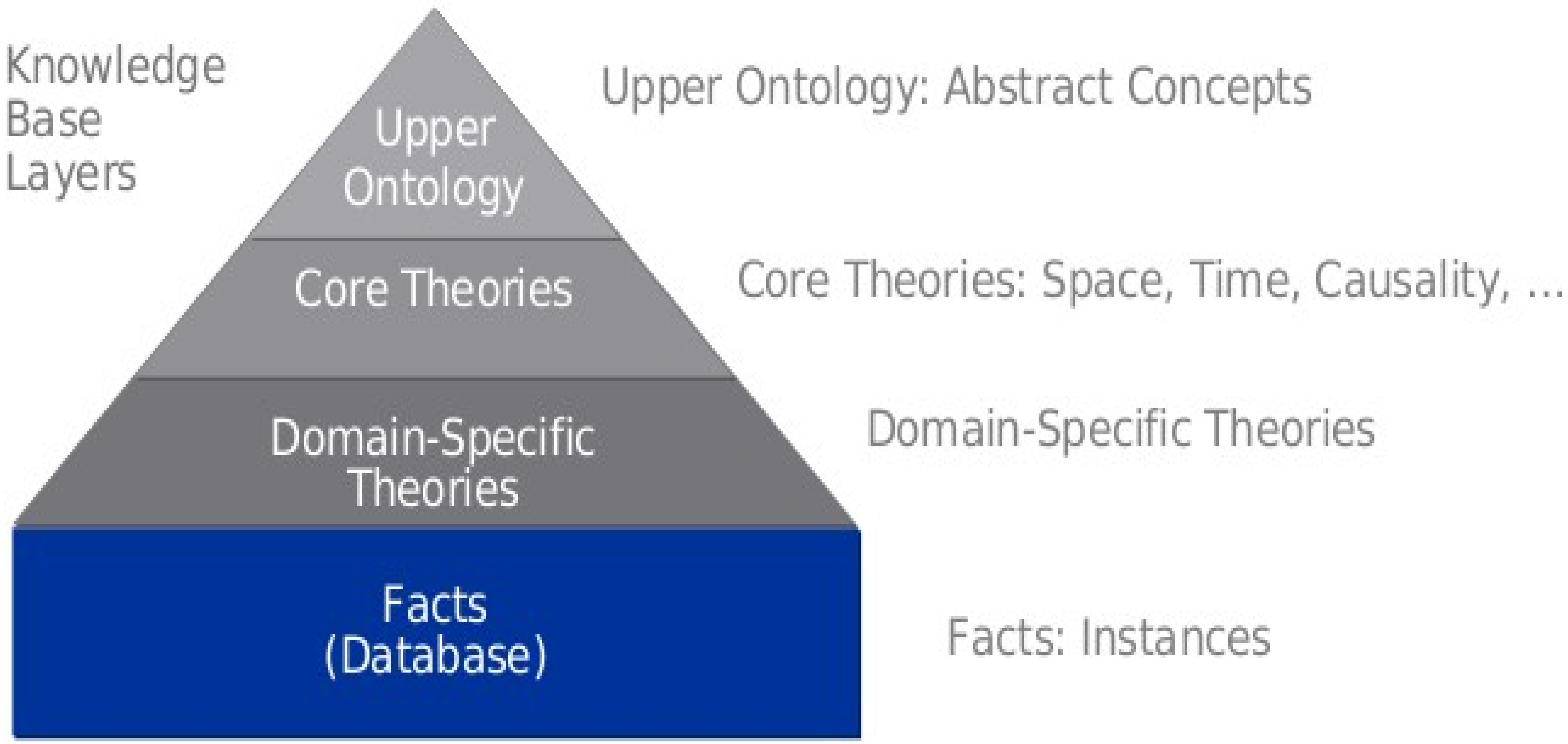




Cyc - knowledge base

- Knowledge base
  - Doug Lenat
  - Conceptual networks (ontologies)
  - Higher ontology, basic theories, specific theories
  - Predefined semantic relationships
- Common sense reasoner
  - Based on predicate calculus
  - Rule-based reasoning

Cyc


Knowledge
Base
Layers
Upper
Ontology
Core Theories
Domain-Specific
Theories
Facts
(Database)
Upper Ontology: Abstract Concepts
Core Theories: Space, Time, Causality, ...
Domain-Specific Theories
Facts: Instances

Some conclusions

- There exist a variety of different dictionaries, properties, concepts, ...
  - Common definitions are not frequent
- There exist a variety of formats and models for knowledge and data representation
  - RDF is common data/knowledge model
- Senses of words are not represented

Challanges in designing big3store

Challenges (1)

- Definition of namespace of RDF triple-store
  - Uniform access to RDF datasets regardless of distribution, replication, etc.
- Automatic distribution and replication of RDF data
  - Triples are distributed, not files
  - Would not like to dispers triples using hash function
- Intelligent distribution of query processing
  - Distribution of query processing follows distribution of triples
  - Dataflow architecture following novel supercomputer design
-

Challenges (2)

- Dynamic updates in RDF storage manager
  - RDF datasets are periodically updated and new are added
- Multi-threaded architecture of query executor
  - Commodity hardware is equipped with many CPUs and cores
- Distributed cache for query executor
  - Cost of RAM allows moving significant part triple-store in RAM
  - Problem similar to using cache in multi-processor system

Design of big3store

Basic decisions (1)

- Use of inexpensive commodity hardware
- Concurrent programming language Erlang

Basic decisions (2)

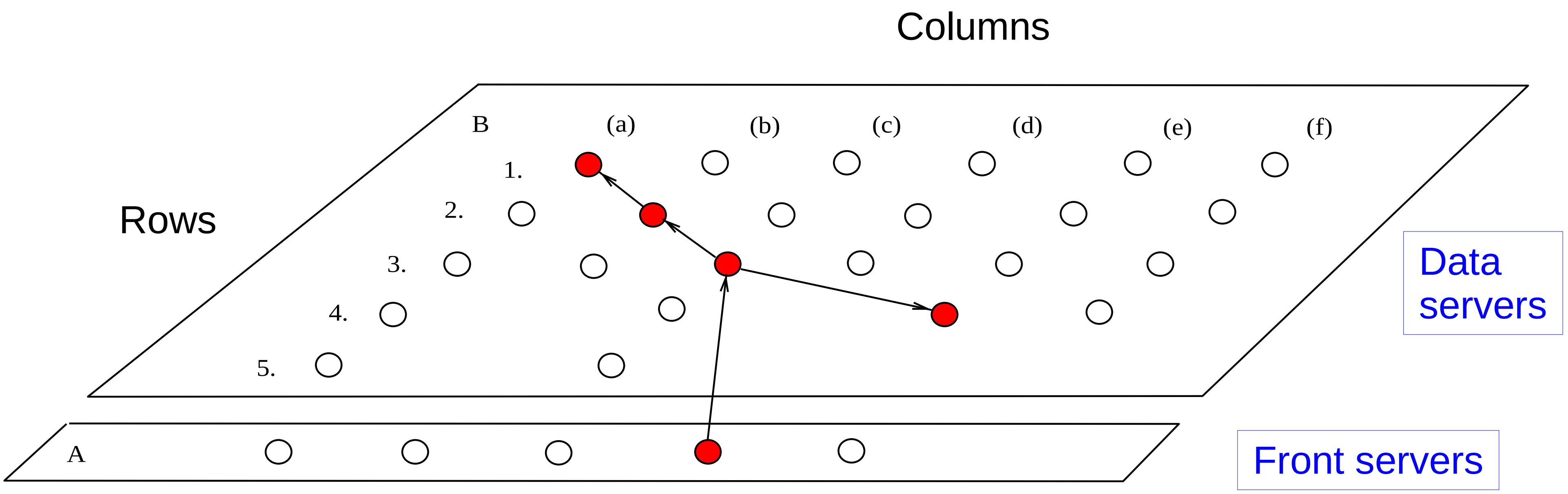
- Adapt relational technology for the query optimization and execution
- Consider relational view of Hadoop data processing principles
- Use relational database system as local triple-store

Basic decisions (3)

- Exploit dataflow nature of RDF algebra for parallelisation of query execution
  - Query tree is dataflow program
  - Assign query trees to arrays of servers
  - Communications of ACM, May 2013:

    “Moving from petaflops to petadata”

Architecture



- Triple-base distributed to columns
- Triple-base parts replicated to rows

Semantic distribution

- Distribution based on triple-base schema
  - Property-based distribution
  - Class-based distribution
- More general distribution schema possible
  - Based on {S, P, O} subset lattice

Triple-base distribution

Properties

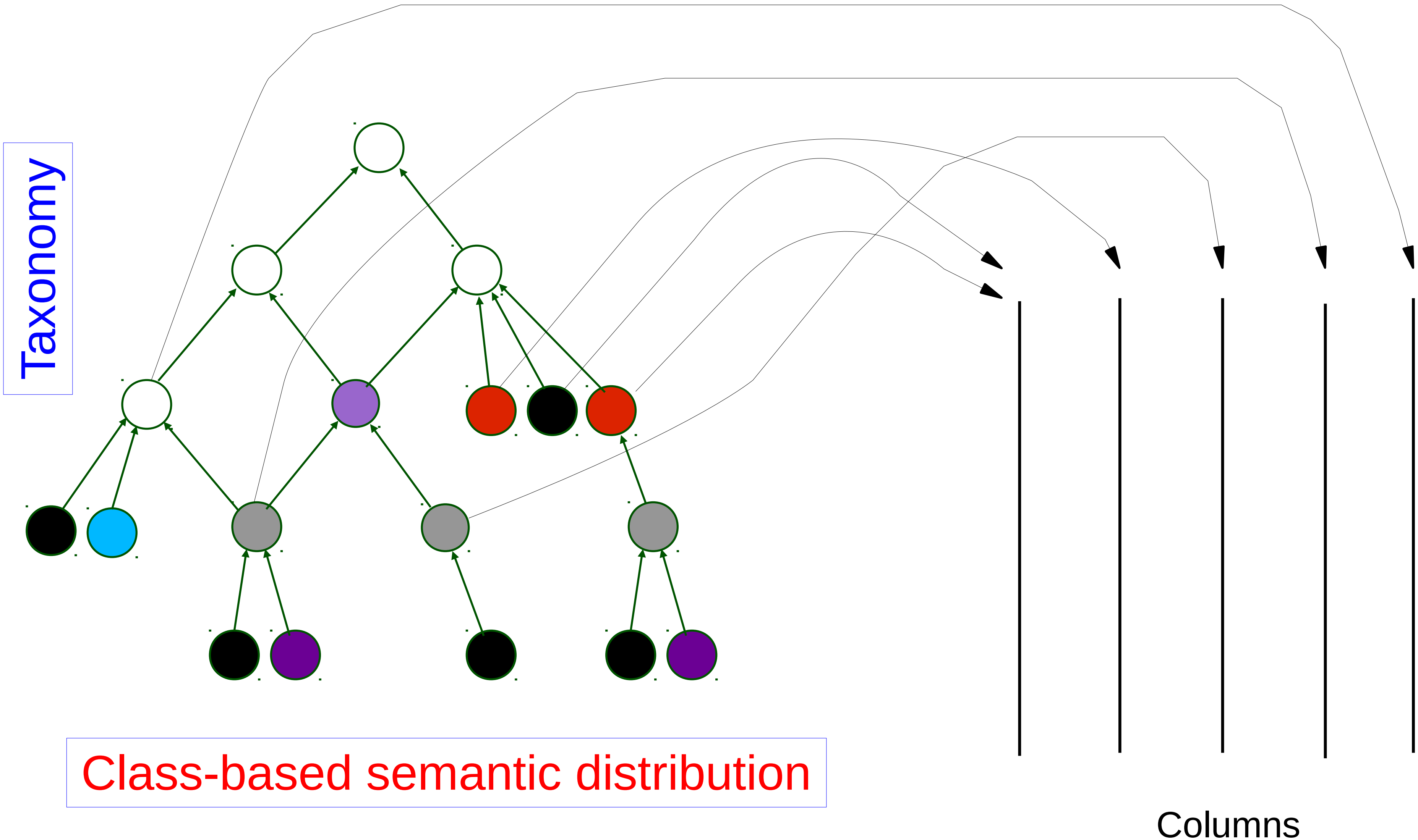
<wasBornOnDate>
<diedOnDate>
<wasDestroyedOnDate>
<hasLatitude>
<wasCreatedOnDate>
<hasArea>
<hasNumberOfPeople>
<hasLongitude>
<hasDuration>
<hasHeight>
<hasPages>
<hasPopulationDensity>
<hasRevenue>
<hasThreeLetterLanguageCode>
<hasWeight>
<hasMotto>
<happenedOnDate>
...

...

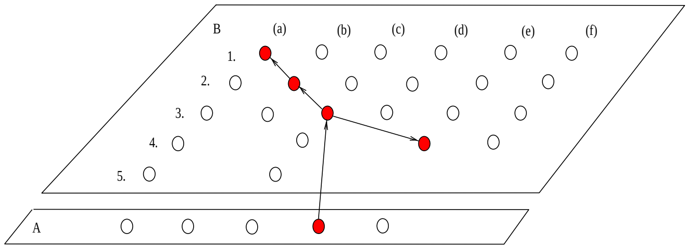
Property-based semantic distribution

Columns

Triple-base distribution

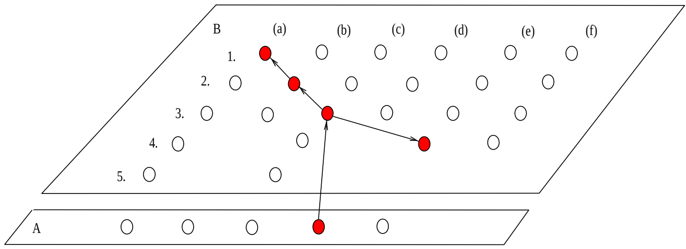


b3s query processing



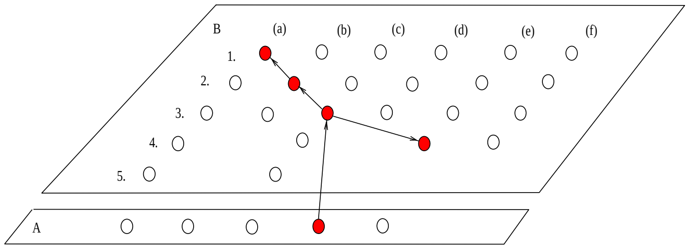
- b3s queries are trees of RDF algebra operations
  - Operations assigned to process on data-server machines
  - Many b3s queries can be mapped to array of data-servers
  - Query trees are optimized to read and process minimal number of triples

b3s query processing



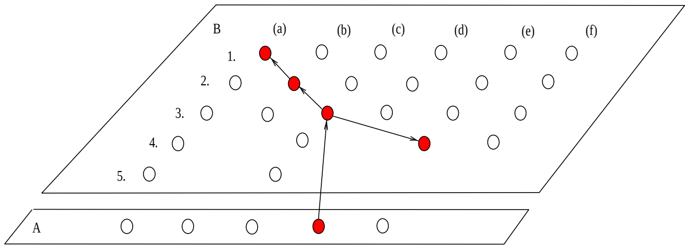
- Front-servers functions
  - Optimization of b3s queries
    - Minimization of disk access
    - Minimization of triple-flow
  - Mapping optimized query trees to array of data-servers
    - Load-ballancing among replicas in columns

b3s query processing



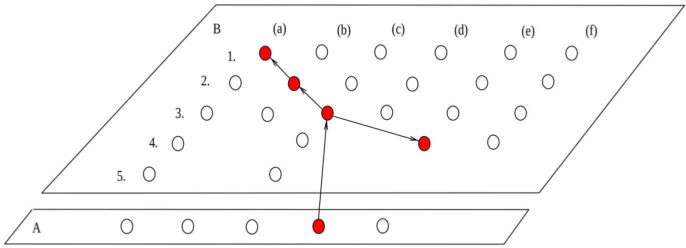
- Algebra operations implemented as processes on data-servers
  - Operations are organized in pipelines
  - Flows (streams) of triples among physical machines
  - Speed of reading output triples ≅ speed of processing one algebra operation
  - Other operations of query work concurrently

b3s query processing

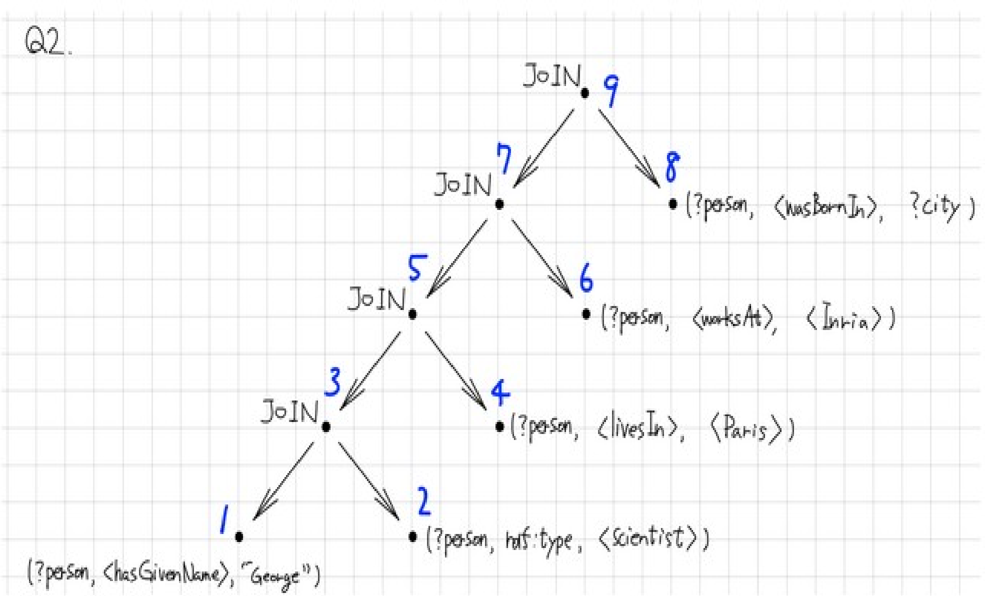


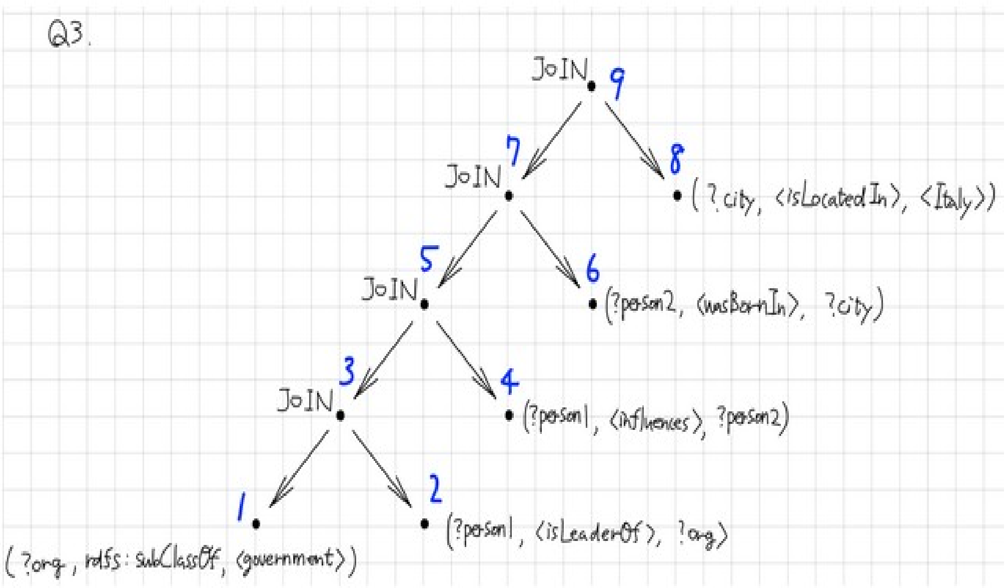
- Algebra operations defined on streams (bags) of triples
  - Flow programming (functional query lang on streams) [John Backus: “Can programming be liberated from the von Neumann style?”,CACM, 1978]
  - Flow ≡ Bag of triples
    - Flow of columns ? (see Abadi's work)
  - Similar to Hadoop indexes (maps)
    - Algebra ops instead of map-reduce

b3s query processing



- Many query trees can be executed in parallel
  - Load-ballance using replicas (data servers) of columns
  - Load-ballance using distributed query nodes





Algebra of graphs

RDF algebra

- `select`
- `project`
- `join`
- `union, intersect, difference`
- `leftjoin`

- Algebra of sets of graphs
- Sets of graphs are input and output of operations
  - Triple is a very simple graph
  - Graph is a set of triples

Syntax

Triple-patterns

Graph-patterns

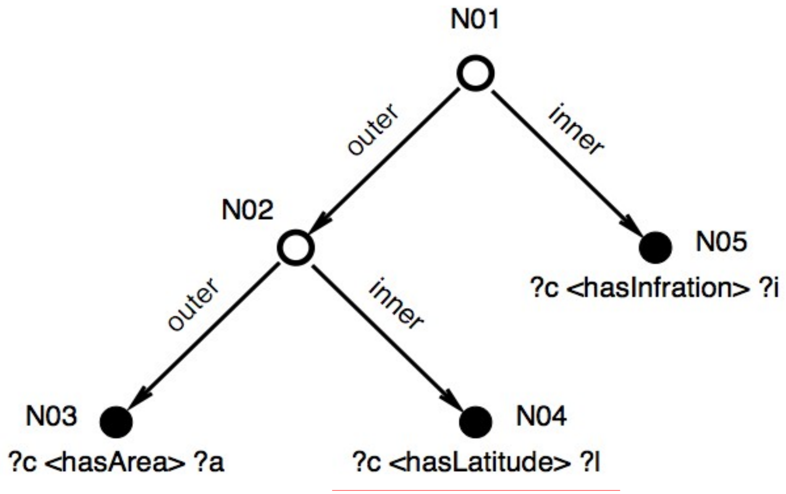
$$
\begin{aligned}
GP &::= TP \mid select(GP, C) \mid join(GP, GP) \mid union(GP, GP) \mid \\
&\quad\;\; intsc(GP, GP) \mid diff(GP, GP) \mid leftjoin(GP, GP) \\
TP &::= (S \mid V, P \mid V, O \mid V) \\
C &::= V\ OP\ V \mid V\ OP\ O \mid C \wedge C \mid C \vee C \mid \neg\, C \\
OP &::= \; = \mid \; \neq \; \mid \; > \; \mid \; \geq \; \mid \; < \; \mid \leq \\
S &::= \text{URI} \mid \text{Blank-Node} \\
P &::= \text{URI} \\
O &::= \text{URI} \mid \text{Blank-Node} \mid \text{Literal} \\
V &::= ?a \;..\; ?z
\end{aligned}
$$

Conditions

Variables

Triple-patterns

```
SELECT * WHERE {
  ?c <hasArea>       ?a .
  ?c <hasLatitude>   ?l .
  ?c <hasInfration> ?i
}
```



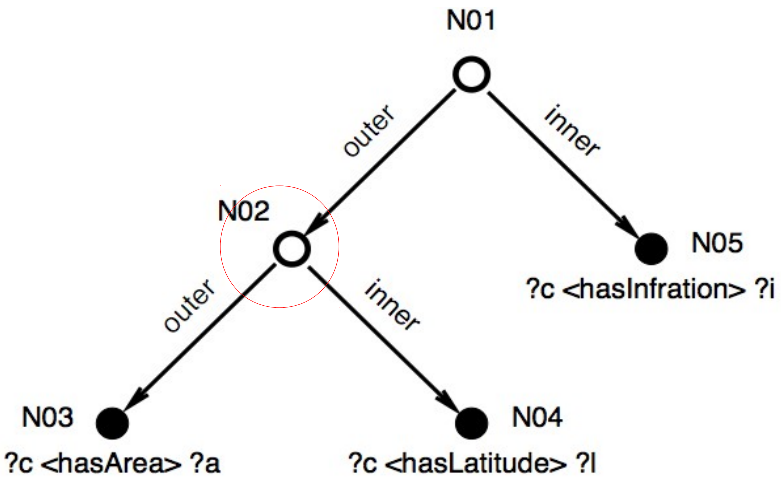
$$TP ::= (S \mid V, P \mid V, O \mid V)$$

$$[\![(t_1, t_2, t_3)]\!]_{db} = \{\, (s,p,o) \mid (s,p,o) \preceq db \wedge ground((s,p,o)) \wedge (s,p,o) \sim (t_1, t_2, t_3) \,\}$$

- Triple-patterns correspond to DB access methods
  - Iterator returning triples
  - Using indexes to acces TP

Join

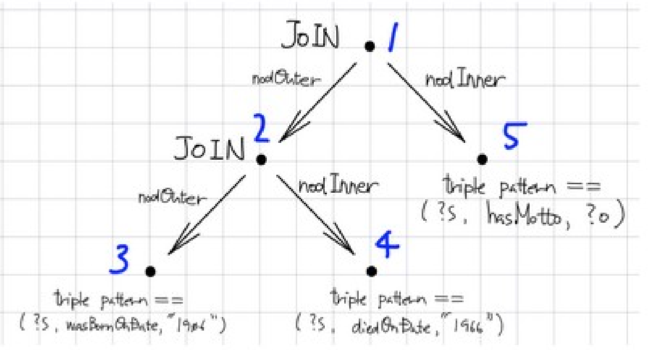
```
SELECT * WHERE {
  ?c <hasArea>        ?a .
  ?c <hasLatitude>    ?l .
  ?c <hasInfration>   ?i
}
```



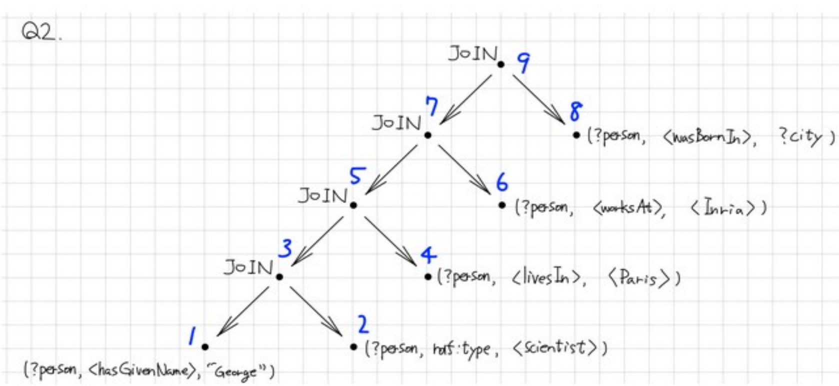
$$[\![join(gp_1, gp_2)]\!]_{db} = \{\, g_1 \cup g_2 \mid g_1 \in [\![gp_1]\!]_{db} \wedge g_2 \in [\![gp_2]\!]_{db} \wedge \forall v \in vs : val(v, gp_1, g_1) = val(v, gp_2, g_2) \,\}$$

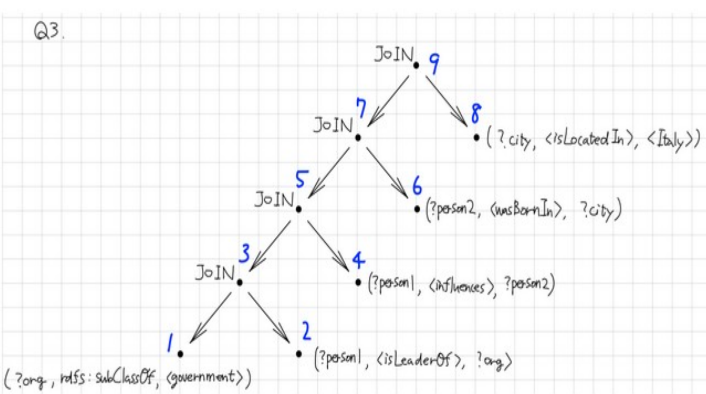
- Index nested-loop join
  - Exploiting DB indexes on subsets of { S, P, O }

Graph-patterns



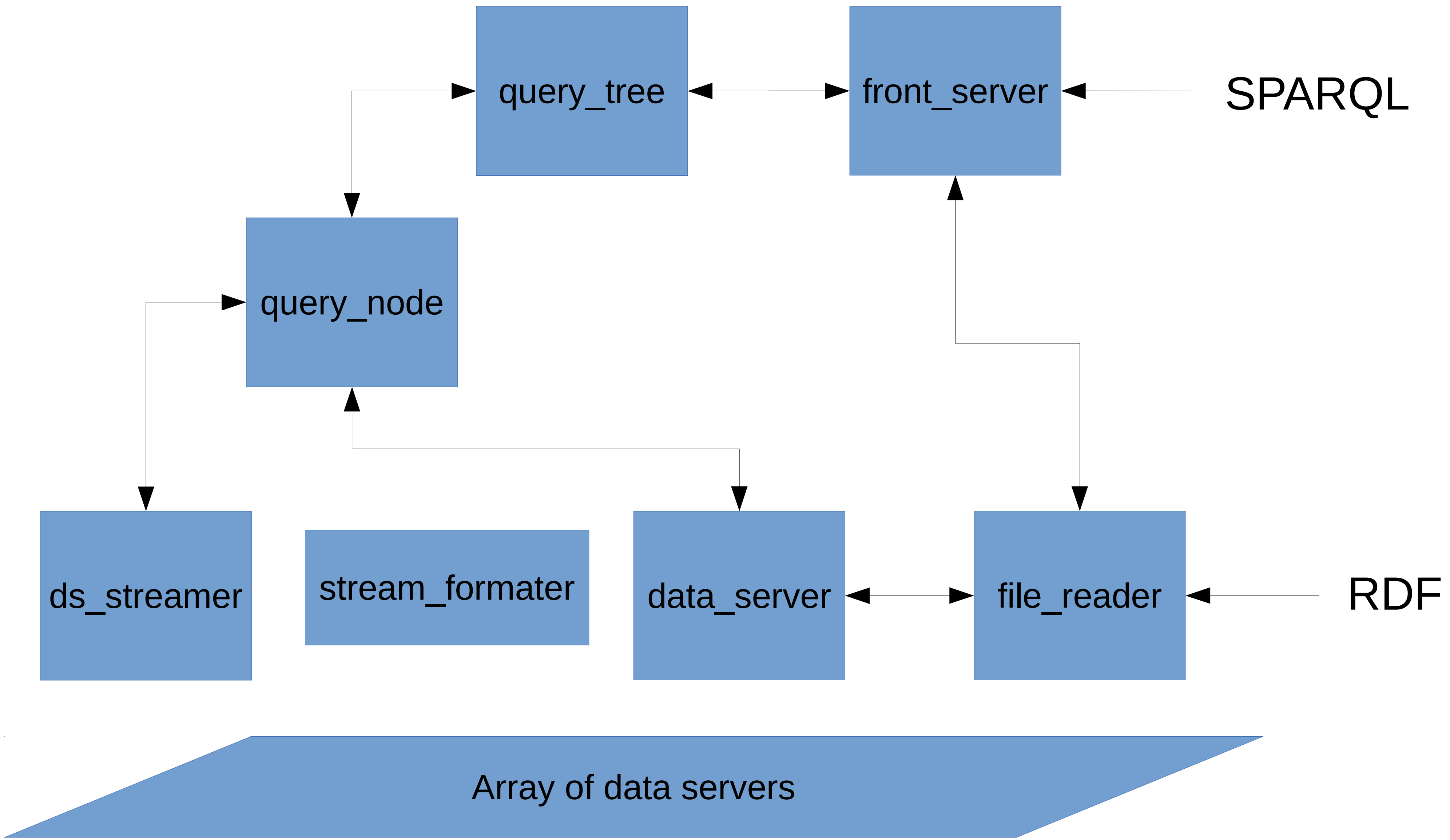
- Graph-patterns similar to SQL blocks
  - Includes only joins and TPs
  - select and project packed into join and TP
  - Evaluated after host is evaluated
- Graph-patterns are units of optimization
  - Optimization based on dynamic programming
  - Relatively simple and clean implementation



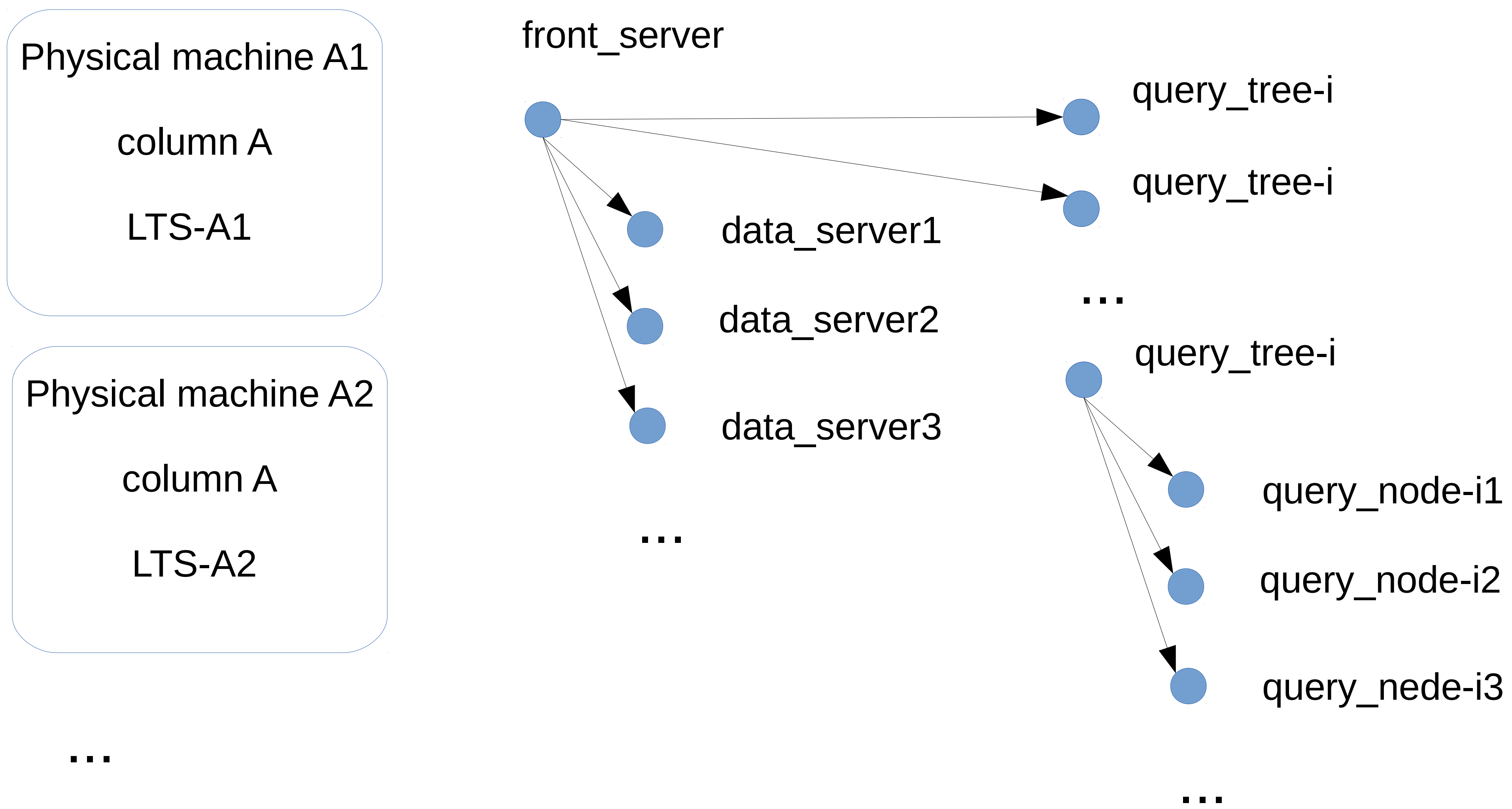


Implementation of big3store

b3s modules – static view



b3s modules – dynamic view



Conclusions

Conclusions

- big3store design was presented
- First prototype of b3s was implemented
  - Data distribution, query evaluation
- Second prototype will be available in few months
  - Improved distribution, extending query evaluation, load ballancing with replicas, experiments with data and query distribution, query optimization
- Problems:
  - Efficient data distribution
  - Efficient query distribution

Further work

- Dynamic updates
- Use of main memory cache for data servers
- Experiments with query and data distribution
- Searching for distributed query tree patterns for fast execution

Thank you !