Graph databases

Iztok Savnik, Famnit, UP

Joint project with:
Kiyoshi Nitta, Yahoo Japan Research

Maj, 2016.

Outline

1) Graph data model (RDF)
2) Popular graph databases on Web
3) Big3store

Graph data model (RDF + RDFS)

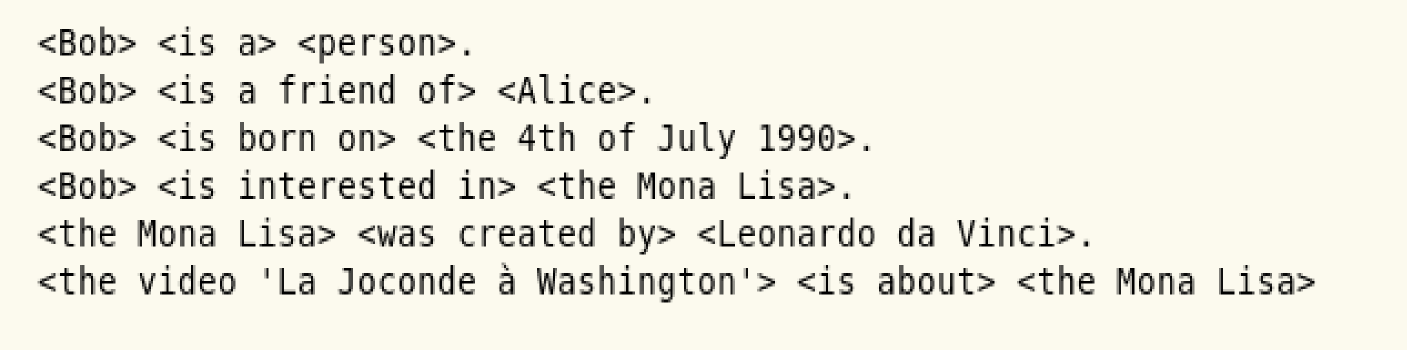
Graph data model

- Graph database
  - Database that uses graphs for the representation of data and queries
- Vertexes
  - Represent things, persons, concepts, classes, ...
- Arcs
  - Represent properties, relationships, associations, ...
  - Arcs have labels !

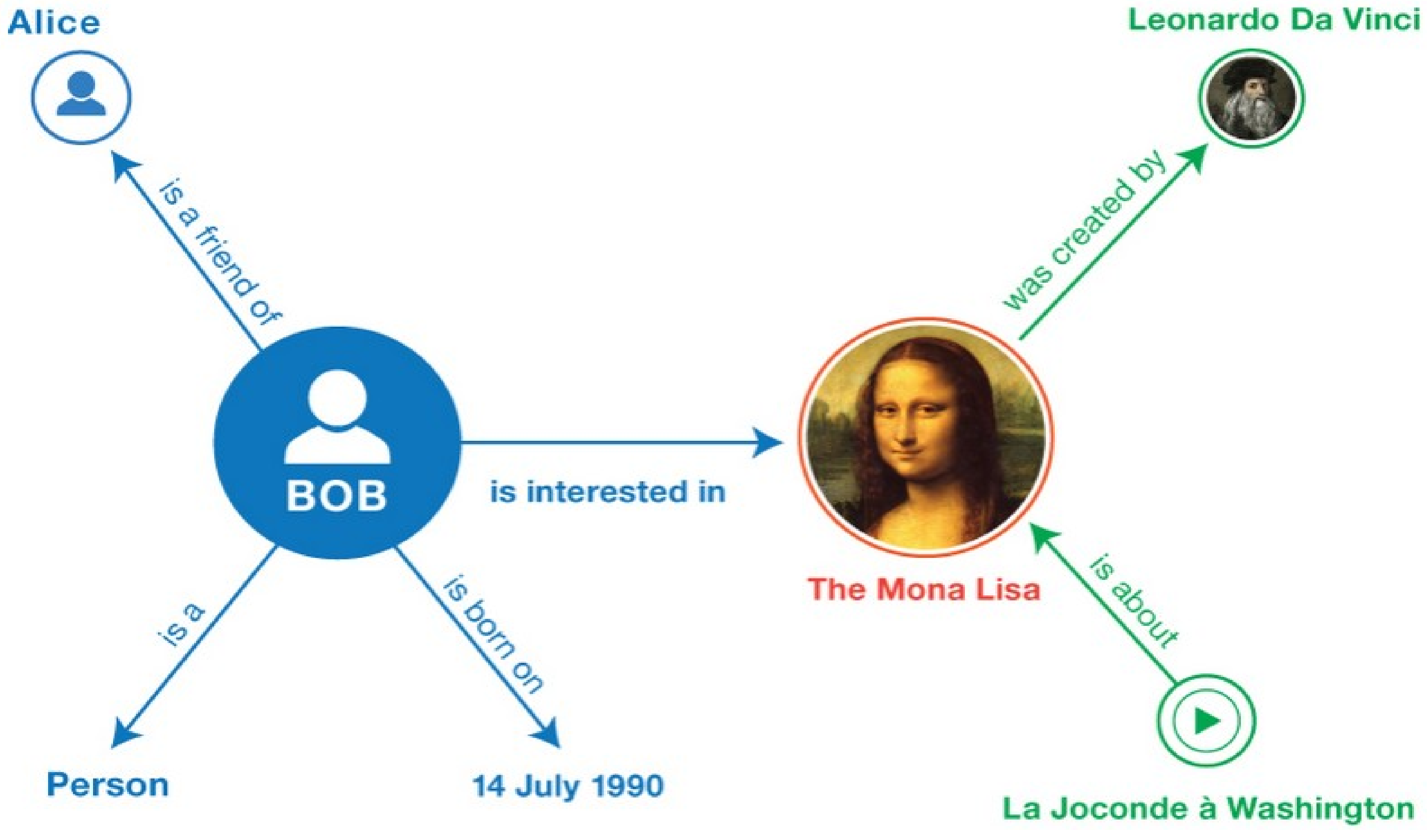
RDF

- Resource Description Framework
  - Tim Berners Lee, 1998-2009
  - This is movement !
- What is behind ?
  - Graphs
  - Triples (3)
  - Semantic data models
  - Human associative memory (psychology)
  - Associative neural networks
  - Hopfield Network

RDF



```
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa>
```


Alice
is a friend of
BOB
is interested in
is a
is born on
Person
14 July 1990
Leonardo Da Vinci
was created by
The Mona Lisa
is about
La Joconde à Washington

RDF syntax

- N3, TVS
- Turtle
- TriG
- N-Triples
- RDF/XML
- RDF/JSON

Name spaces

- Using short names for URL-s
  - Long names are tedious
- Simple but strong concept
- Defining name space:

prefix rdf:, namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#

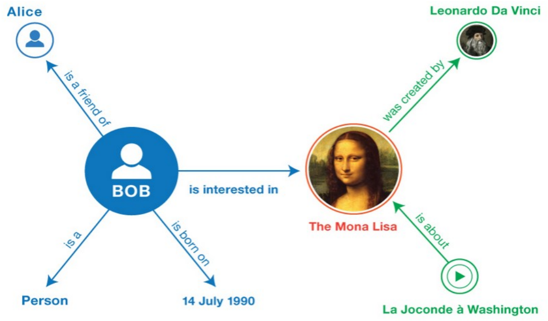
prefix rdfs:, namespace URI: http://www.w3.org/2000/01/rdf-schema#

prefix dc:, namespace URI: http://purl.org/dc/elements/1.1/

prefix owl:, namespace URI: http://www.w3.org/2002/07/owl#

prefix ex:, namespace URI: http://www.example.org/ (or http://www.example.com/)

prefix xsd:, namespace URI: http://www.w3.org/2001/XMLSchema#



N-Triples

```
<http://example.org/bob#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/knows> <http://example.org/alice#me> .
<http://example.org/bob#me> <http://schema.org/birthDate> "1990-07-04"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/topic_interest> <http://www.wikidata.org/entity/Q12418> .
<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/title> "Mona Lisa" .
<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/creator> <http://dbpedia.org/resource/Leonardo_da_Vinci> .
<http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619> <http://purl.org/dc/terms/subject> <
```

Turtle

```
BASE   <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX schema: <http://schema.org/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX wd: <http://www.wikidata.org/entity/>

<bob#me>
    a foaf:Person ;
    foaf:knows <alice#me> ;
    schema:birthDate "1990-07-04"^^xsd:date ;
    foaf:topic_interest wd:Q12418 .

wd:Q12418
    dcterms:title "Mona Lisa" ;
    dcterms:creator <http://dbpedia.org/resource/Leonardo_da_Vinci> .

<http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
    dcterms:subject wd:Q12418 .
```

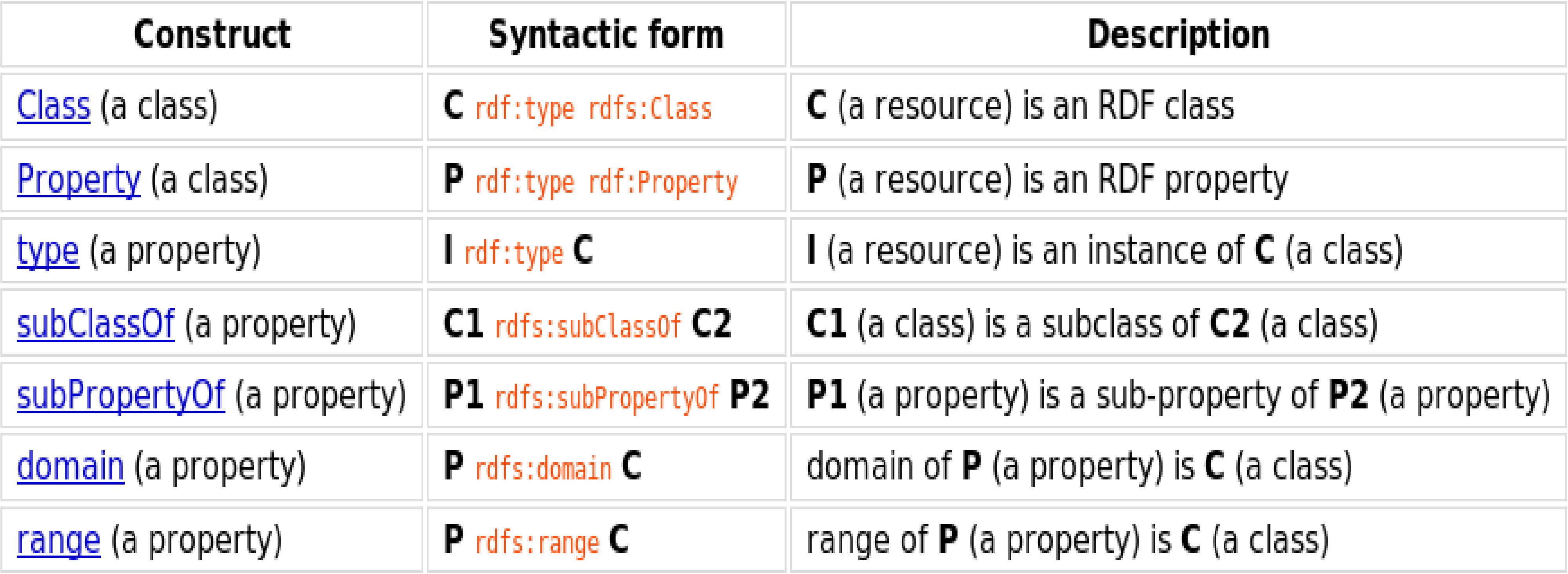
Additional RDF Constructs

- Complex values
  - Bags, lists, trees, graphs
- Empty nodes
- Types of atomic values
- Types of nodes
- Reification

RDF Schema

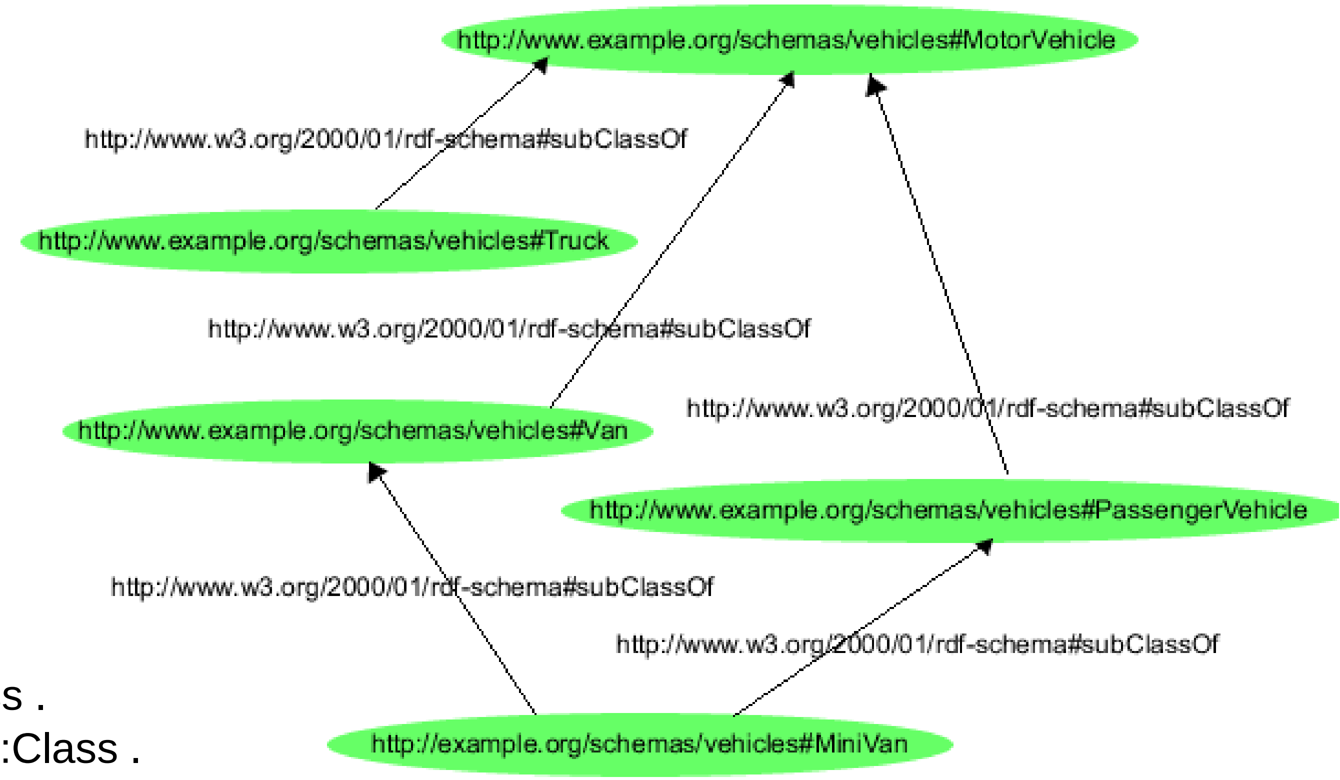
- RDFS
- Knowledge representation language
  - Not just graph any more !
  - AI Frames, Object Model
- Small dictionary for RDFS
  - rdfs:class, rdfs:subClassOf, rdfs:type
  - rdfs:property, rdfs:subPropertyOf
  - rdfs:domain, rdfs:range

RDFS Concepts



Construct	Syntactic form	Description
Class (a class)	**C** rdf:type rdfs:Class	**C** (a resource) is an RDF class
Property (a class)	**P** rdf:type rdf:Property	**P** (a resource) is an RDF property
type (a property)	**I** rdf:type **C**	**I** (a resource) is an instance of **C** (a class)
subClassOf (a property)	**C1** rdfs:subClassOf **C2**	**C1** (a class) is a subclass of **C2** (a class)
subPropertyOf (a property)	**P1** rdfs:subPropertyOf **P2**	**P1** (a property) is a sub-property of **P2** (a property)
domain (a property)	**P** rdfs:domain **C**	domain of **P** (a property) is **C** (a class)
range (a property)	**P** rdfs:range **C**	range of **P** (a property) is **C** (a class)

Classes


http://www.example.org/schemas/vehicles#MotorVehicle
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://www.example.org/schemas/vehicles#Truck
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://www.example.org/schemas/vehicles#Van
http://www.example.org/schemas/vehicles#PassengerVehicle
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://example.org/schemas/vehicles#MiniVan

ex:MotorVehicle rdf:type rdfs:Class .
ex:PassengerVehicle rdf:type rdfs:Class .
ex:Van rdf:type rdfs:Class .
ex:Truck rdf:type rdfs:Class .
ex:MiniVan rdf:type rdfs:Class .

ex:PassengerVehicle rdfs:subClassOf ex:MotorVehicle .
ex:Van rdfs:subClassOf ex:MotorVehicle .
ex:Truck rdfs:subClassOf ex:MotorVehicle .

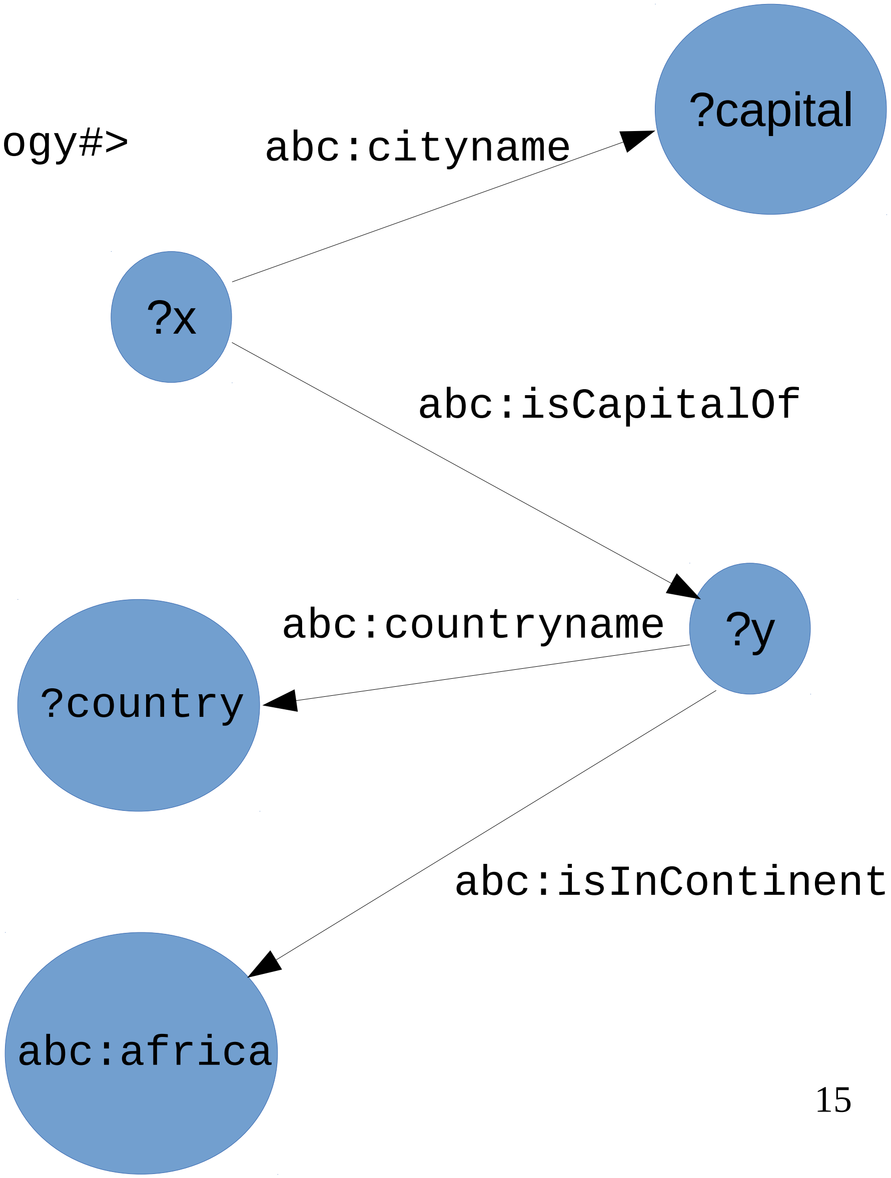
ex:MiniVan rdfs:subClassOf ex:Van .
ex:MiniVan rdfs:subClassOf ex:PassengerVehicle .

SPARQL

- SPARQL Protocol and RDF Query Language
- SPARQL query
  - Graph can include variables in place of constants
- Operations
  - JOIN (natural, left-join)
  - AND, FILTER, UNION, OPTIONAL
- Commercial DBMS-s
  - Implement RDF and SPARQL

Example SPARQL query

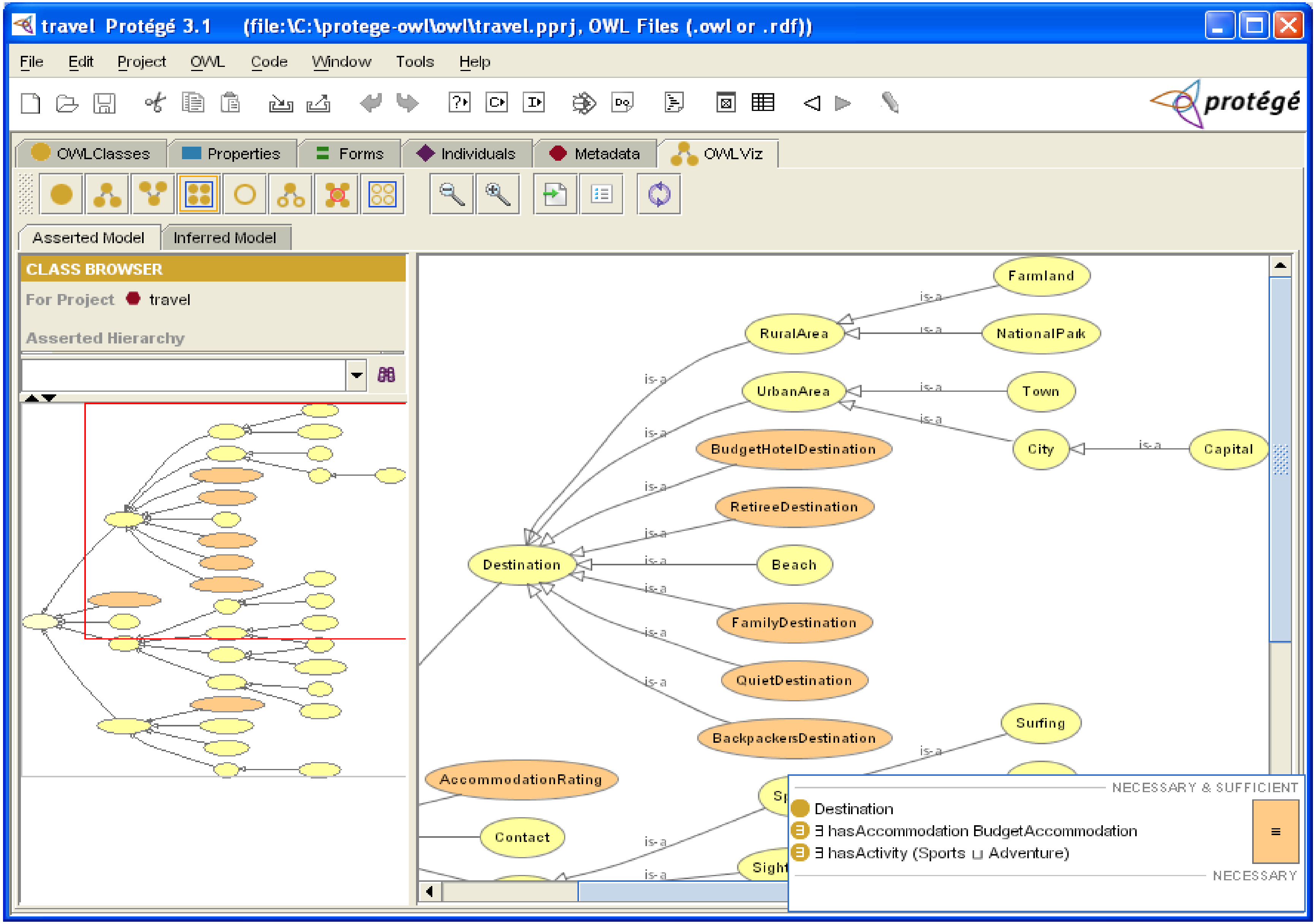
```
PREFIX
   abc: <http://mynamespace.com/exampleOntology#>
SELECT ?capital ?country
WHERE { ?x abc:cityname ?capital.
        ?y abc:countryname ?country.
        ?x abc:isCapitalOf ?y.
        ?y abc:isInContinent abc:africa. }
```



Logic - OWL

- Ontology language
  - Knowledge representation + Logic
- Based on description logic
  - Fragments of predicate calculus
  - Hierarchy of DL languages
- OWL reasoners
  - FaCT++, HermiT, RacerPro, Pellet, ...

Protégé



travel Protégé 3.1 (file:\C:\protege-owl\owl\travel.pprj, OWL Files (.owl or .rdf))

NECESSARY & SUFFICIENT

Destination

∃ hasAccommodation BudgetAccommodation

∃ hasActivity (Sports ⊔ Adventure)

NECESSARY

Popular graph databases on Web

Terminology

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

Wordnet

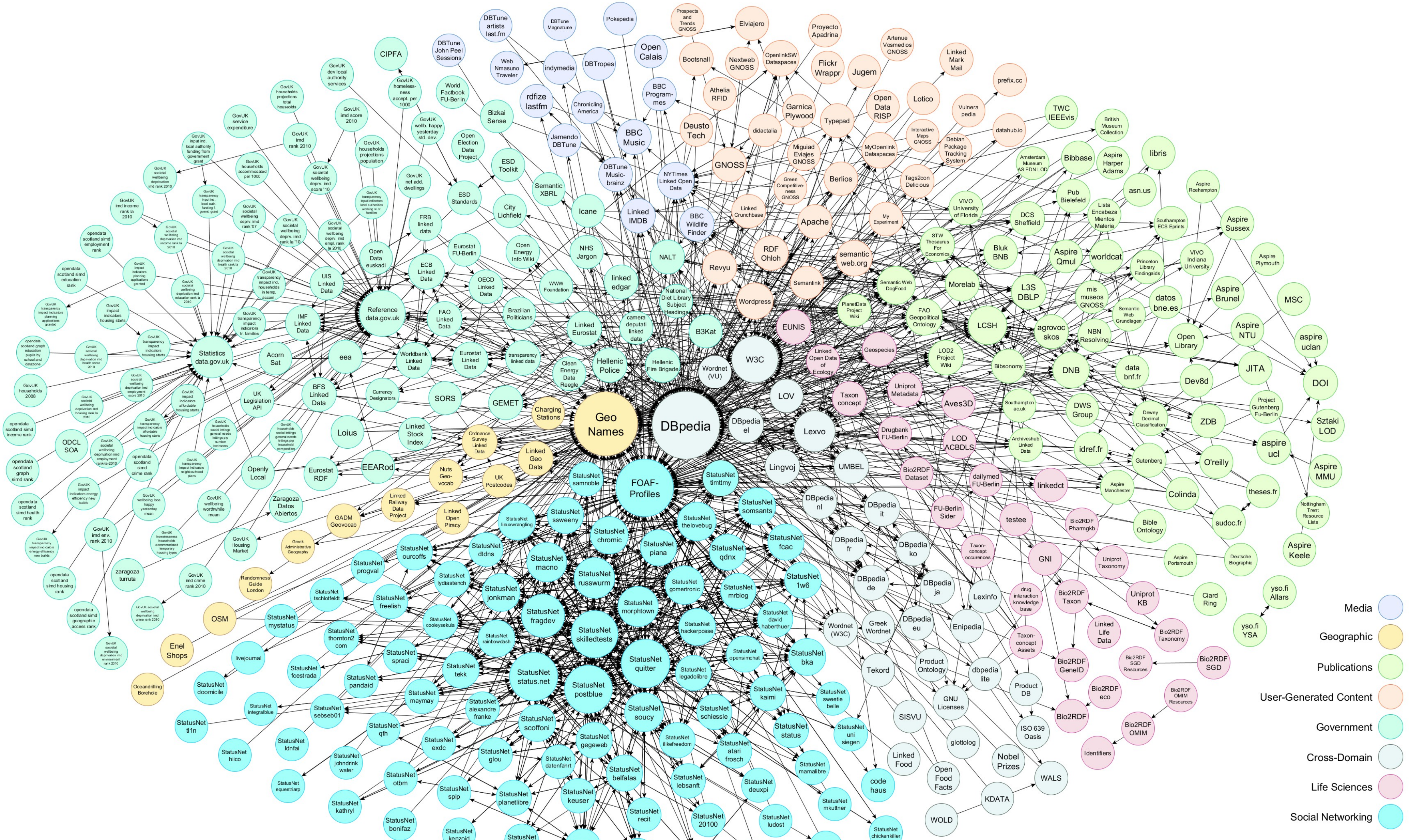
- Princeton's large lexical database of English.
  - Cognitve synonims: synsets
    - 117,000 synsets
  - Synsets are linked by:
    - conceptual-semantic relationships, and
    - lexical relationships.
    - Include definitions of synsets.
  - Main relationships:
    - Synonymy, hyponymy (ISA), meronymy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

Linked Open Data

LINKINGOPENDATA
W3C SWEO Community Project

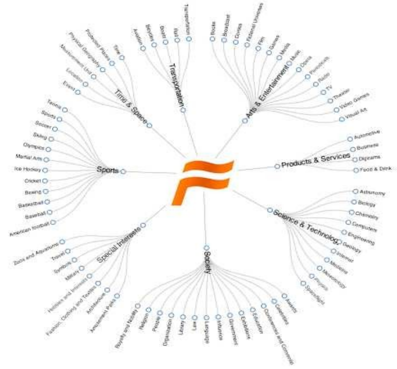
- Datasets are represented in RDF
  - Wikipedia, Wikibooks, Geonames, MusicBrainz, WordNet, DBLP bibliography
- Number of triples: 33 Giga (10^9) (2011)
- Governments:
  - USA, UK, Japan, Austria, Belgium, France, Germany, ...
- Active community
  - http://en.wikipedia.org/wiki/Open_Data
  - http://www.w3.org/LOD

LOD Cloud, 2014

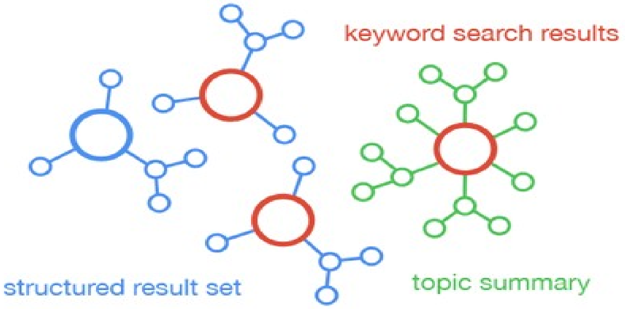


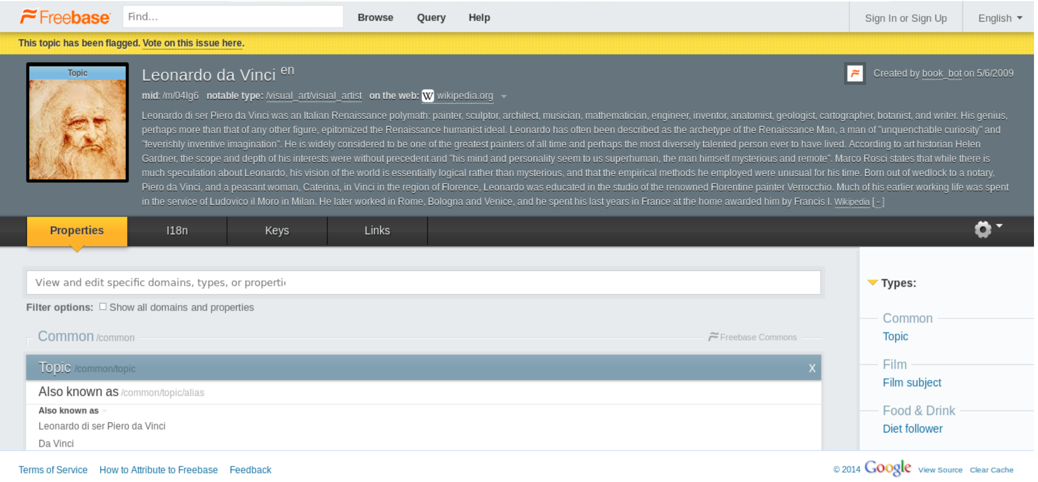
Crawlable Linked Datasets as of April 2014

Freebase



- Free, knowledge graph:
  - people, places and things,
  - 3,041,722,635 facts, 49,947,845 topics
- Semantic search engines are here !





This topic has been flagged. Vote on this issue here.

Leonardo da Vinci en

mid: /m/04lg6 notable type: /visual_art/visual_artist on the web: wikipedia.org

Created by book_bot on 5/6/2009

Leonardo di ser Piero da Vinci was an Italian Renaissance polymath: painter, sculptor, architect, musician, mathematician, engineer, inventor, anatomist, geologist, cartographer, botanist, and writer. His genius, perhaps more than that of any other figure, epitomized the Renaissance humanist ideal. Leonardo has often been described as the archetype of the Renaissance Man, a man of "unquenchable curiosity" and "feverishly inventive imagination". He is widely considered to be one of the greatest painters of all time and perhaps the most diversely talented person ever to have lived. According to art historian Helen Gardner, the scope and depth of his interests were without precedent and "his mind and personality seem to us superhuman, the man himself mysterious and remote". Marco Rosci states that while there is much speculation about Leonardo, his vision of the world is essentially logical rather than mysterious, and that the empirical methods he employed were unusual for his time. Born out of wedlock to a notary, Piero da Vinci, and a peasant woman, Caterina, in Vinci in the region of Florence, Leonardo was educated in the studio of the renowned Florentine painter Verrocchio. Much of his earlier working life was spent in the service of Ludovico il Moro in Milan. He later worked in Rome, Bologna and Venice, and he spent his last years in France at the home awarded him by Francis I. Wikipedia [-]

View and edit specific domains, types, or properti

Filter options: ☐ Show all domains and properties

Common /common — Freebase Commons

Topic /common/topic x

Also known as /common/topic/alias

Also known as

Leonardo di ser Piero da Vinci

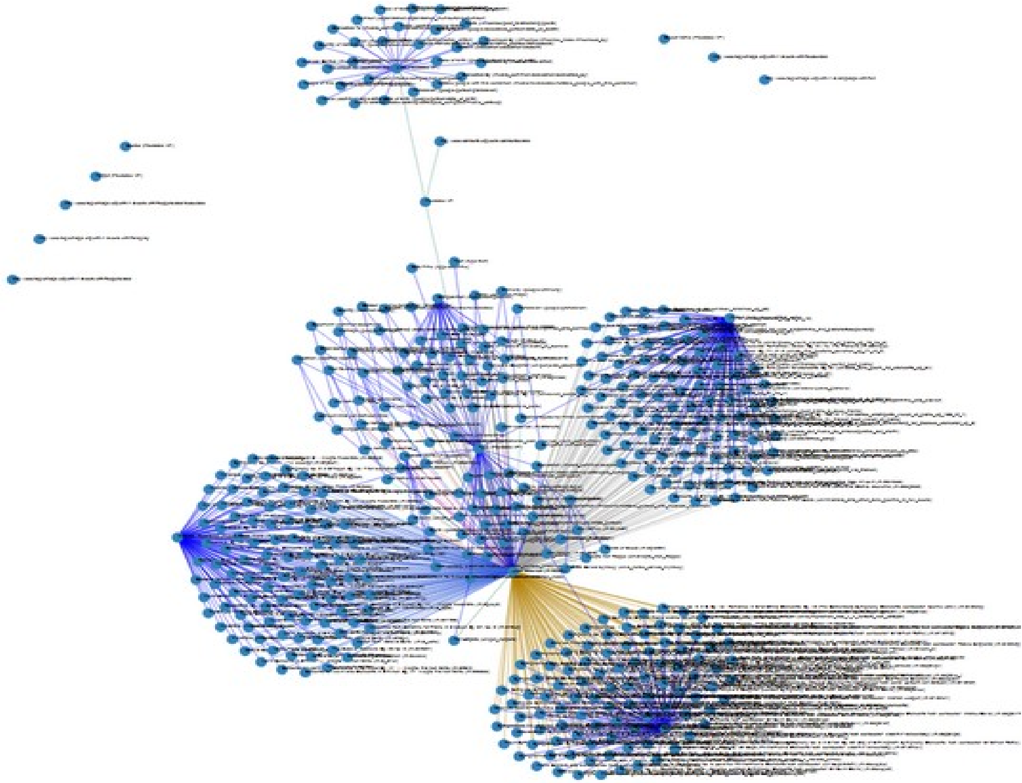
Da Vinci

Types:

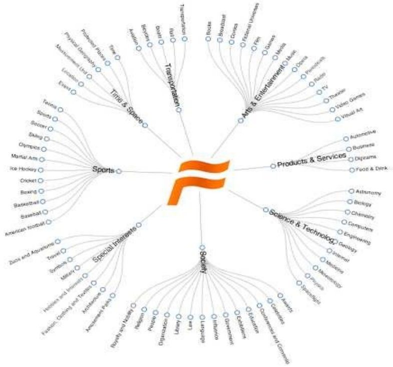
Common
Topic

Film
Film subject

Food & Drink
Diet follower



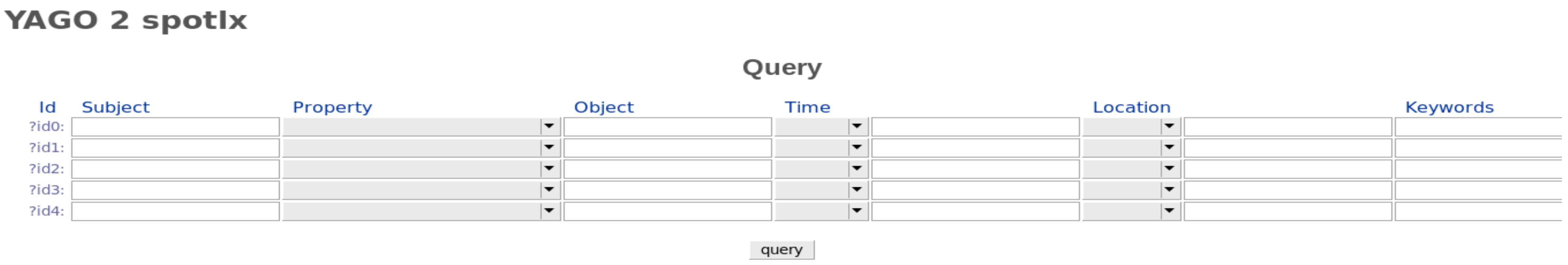
Freebase



- 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



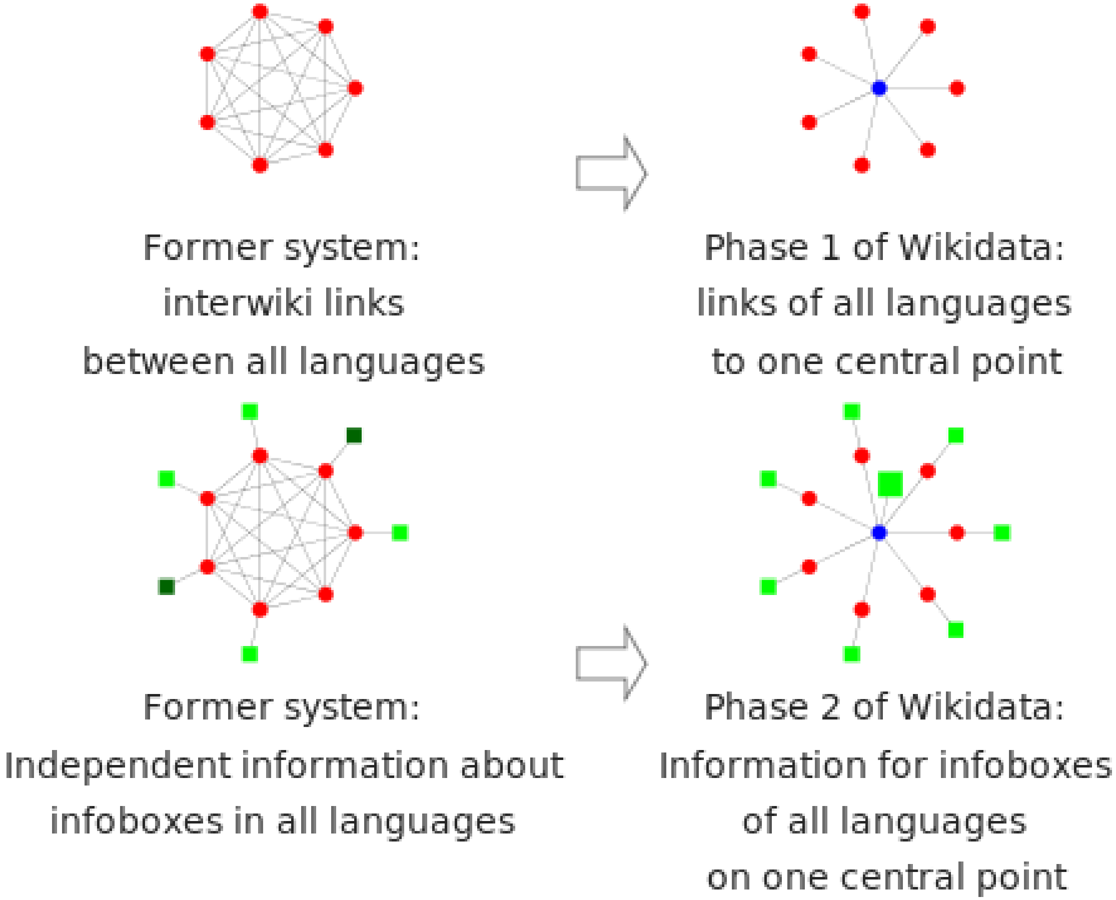
YAGO 2 spotlx

Query

Id	Subject	Property	Object	Time	Location	Keywords
?id0:						
?id1:						
?id2:						
?id3:						
?id4:						

Wikidata

- Free knowledge base with 14,913,910 items
- Collecting structured data
- Properties of
  - person, organization, works, events, etc.

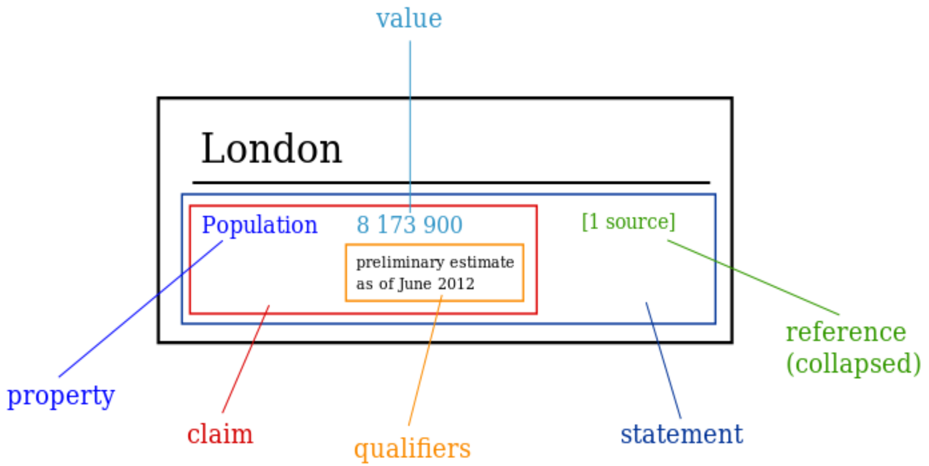


Former system: interwiki links between all languages

Phase 1 of Wikidata: links of all languages to one central point

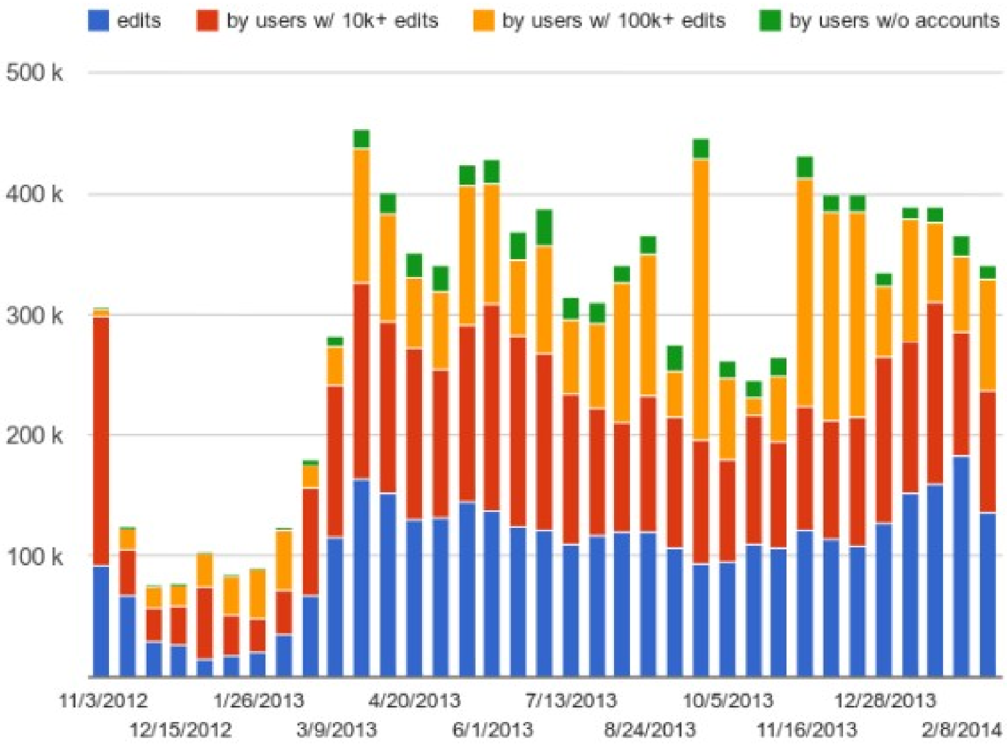
Former system: Independent information about infoboxes in all languages

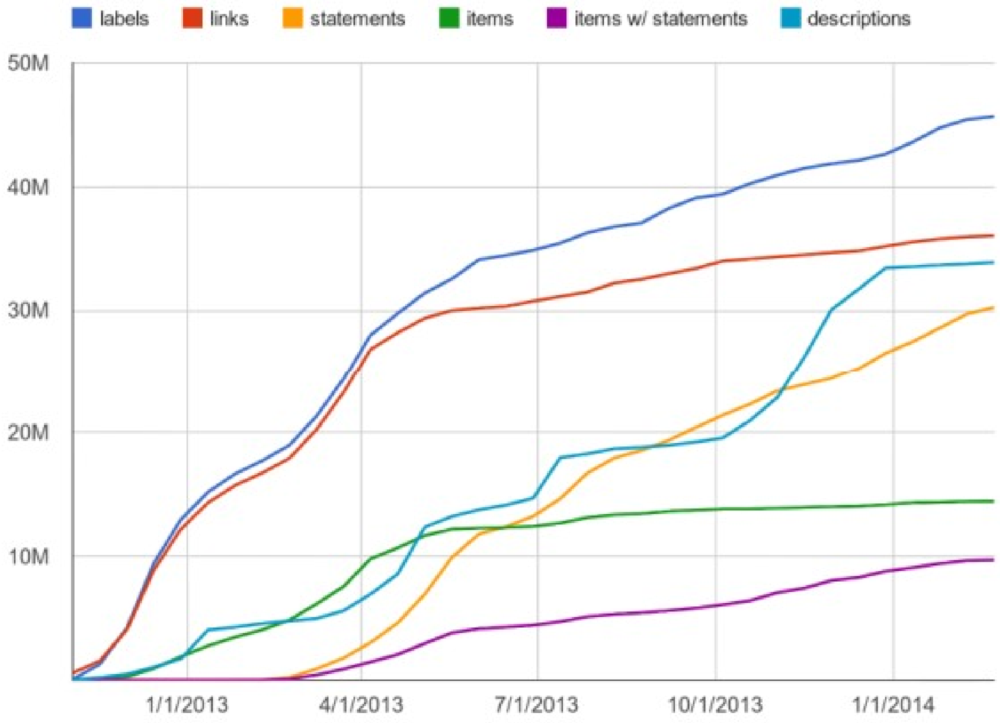
Phase 2 of Wikidata: Information for infoboxes of all languages on one central point



Wikidata

- Free knowledge base with 14,550,852 items



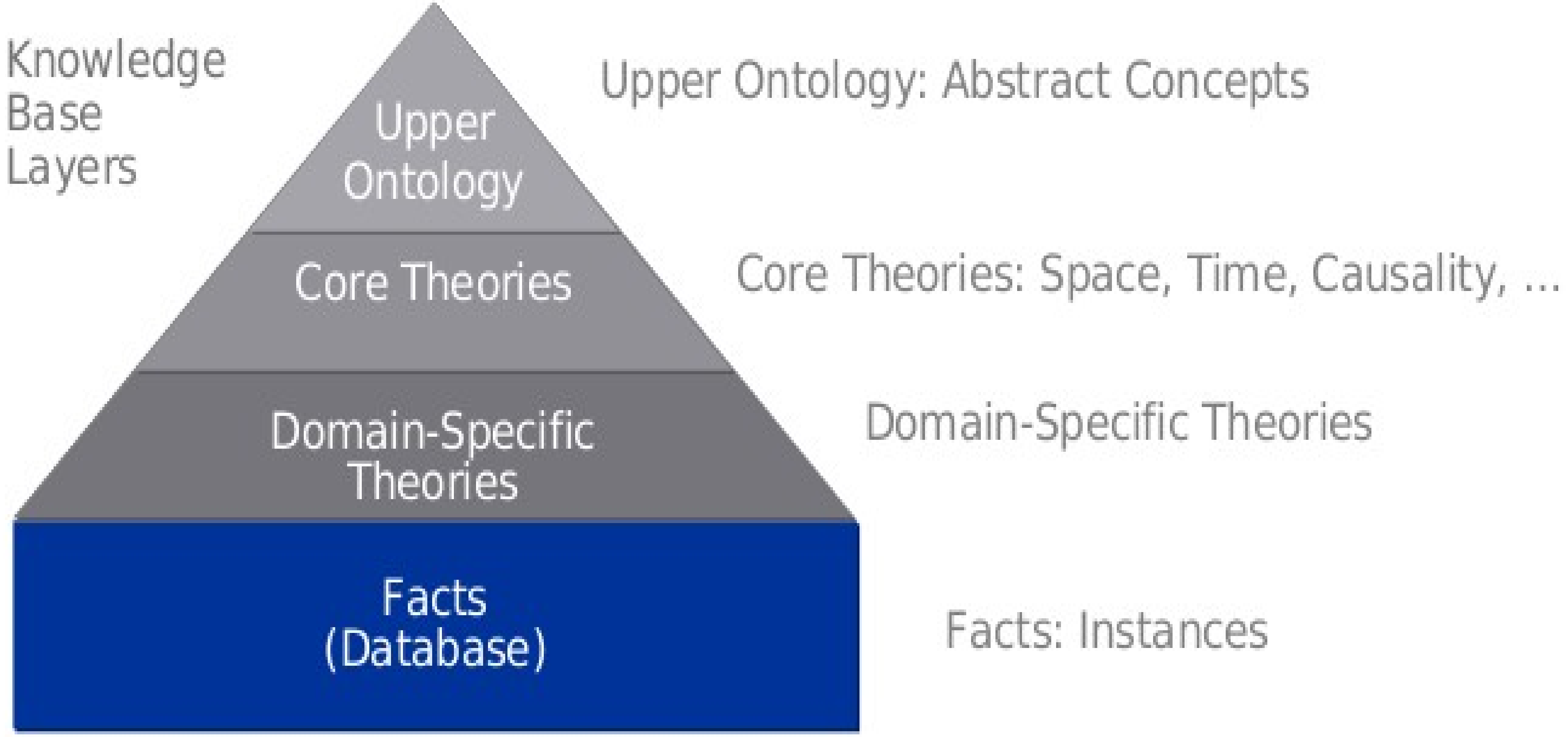


Cycorp®
Home of smarter solutions

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

Design of big3store

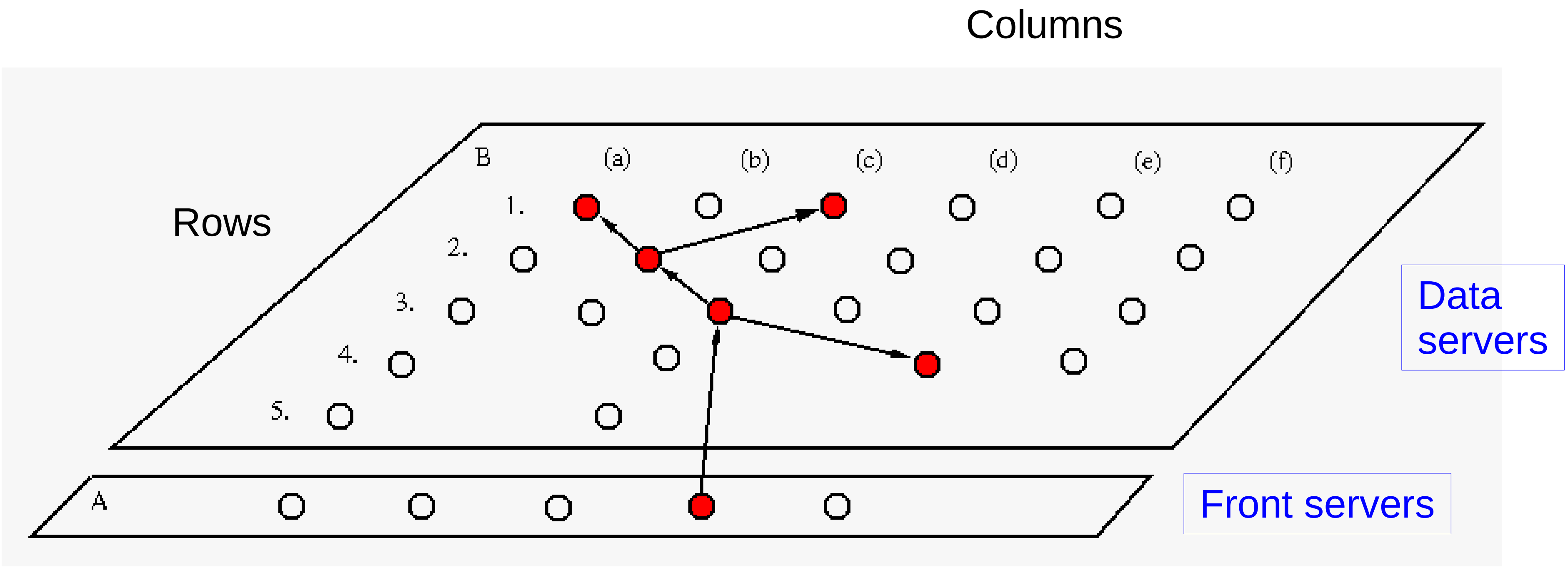
Basic decisions

- Use of inexpensive commodity hardware in shared-nothing cluster
- Concurrent programming language Erlang
- Use relational database system as local triple-store
- Exploit dataflow nature of RDF algebra for parallelisation of query execution

Erlang

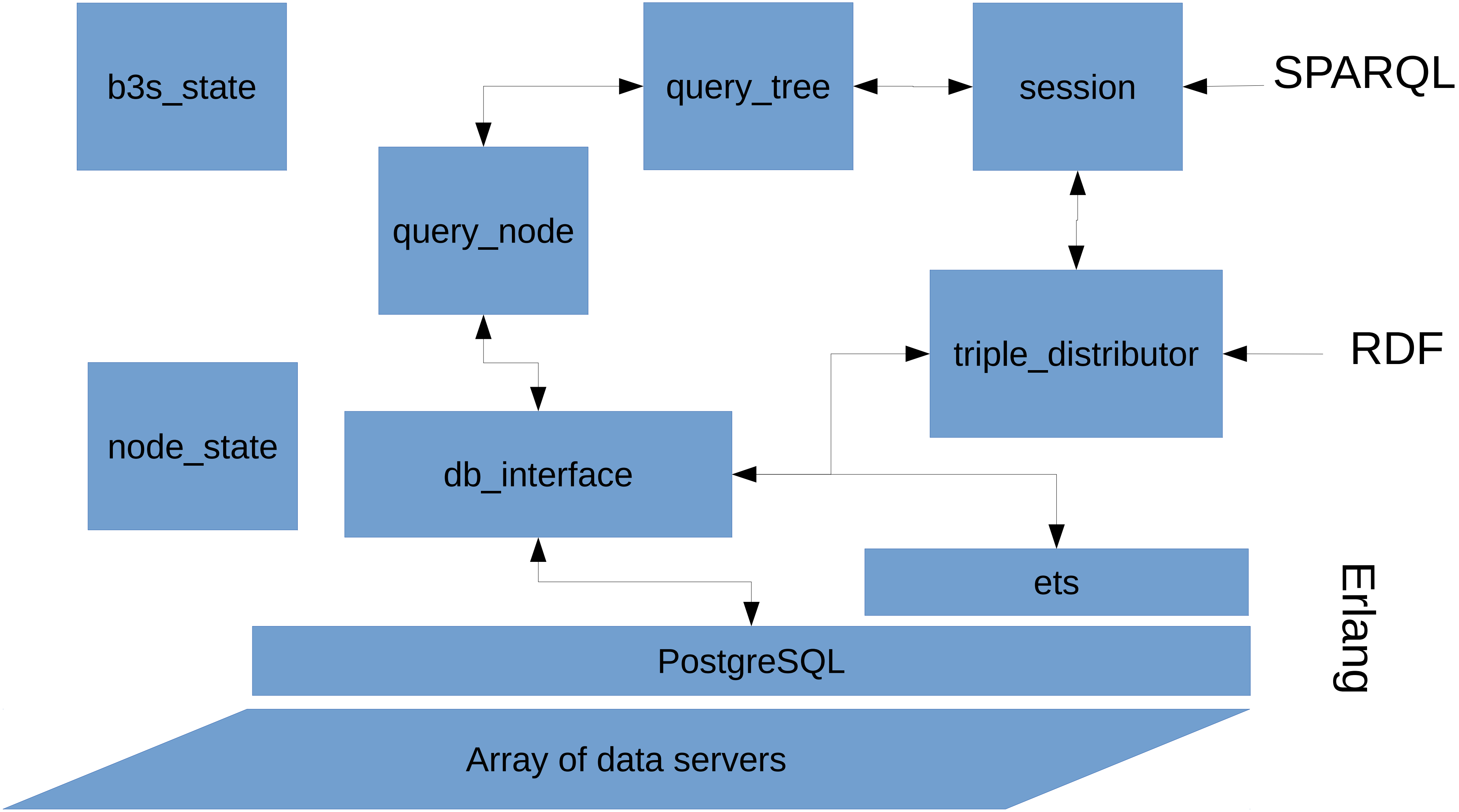
- Build massively scalable soft real-time systems
- Language features
  - Tends to be pure functional language
  - Prolog unification and clauses
  - Many build-in data structures
  - Relational dbms Mnesia
- Light-weight processes
  - Ingenious computing model
  - Processes are true objects !
  - Distributed programming

Architecture

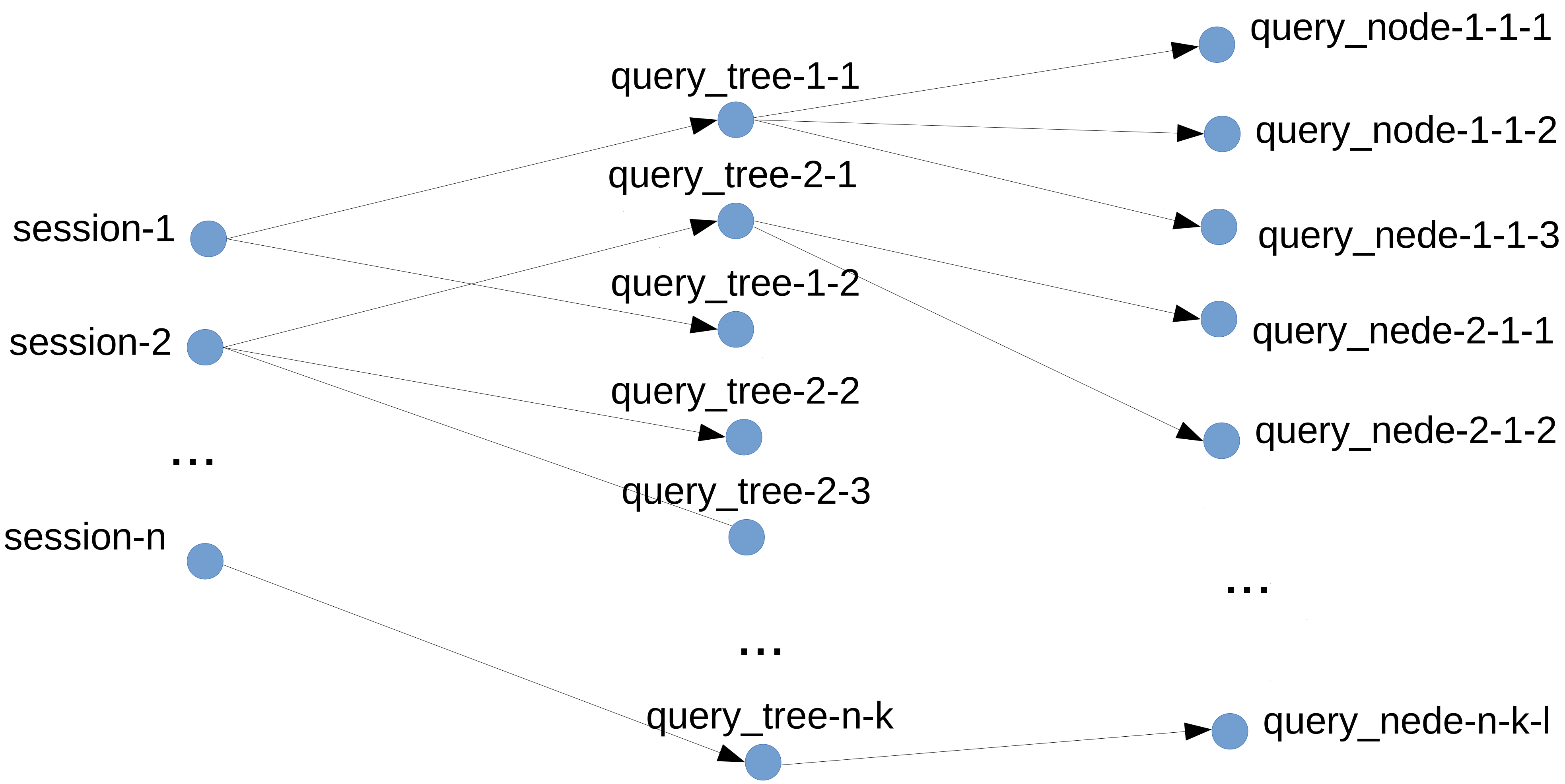


- Triple-base distributed to columns
- Triple-base parts replicated to rows

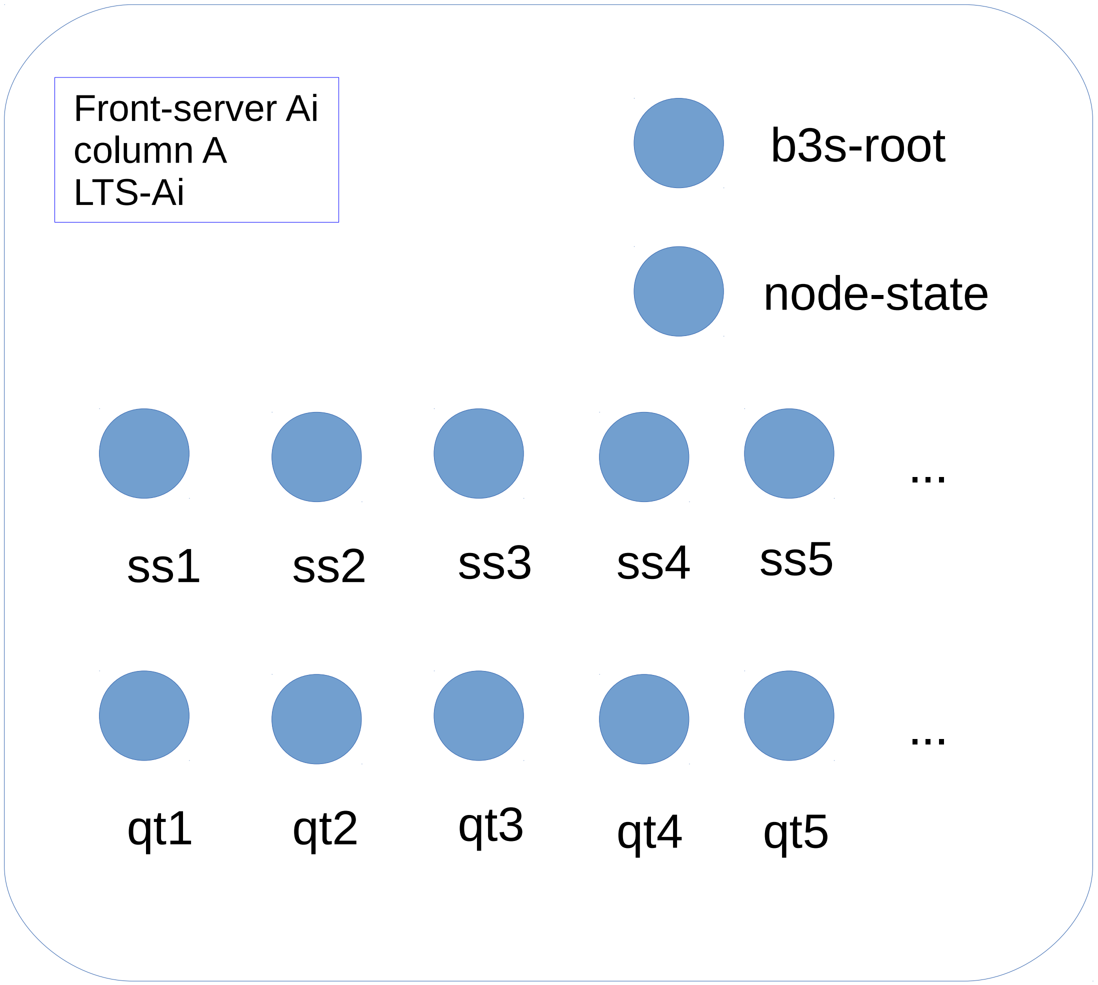
b3s modules – static view



b3s processes – logical view



b3s processes – front server



Front-server Ai
column A
LTS-Ai

b3s-root

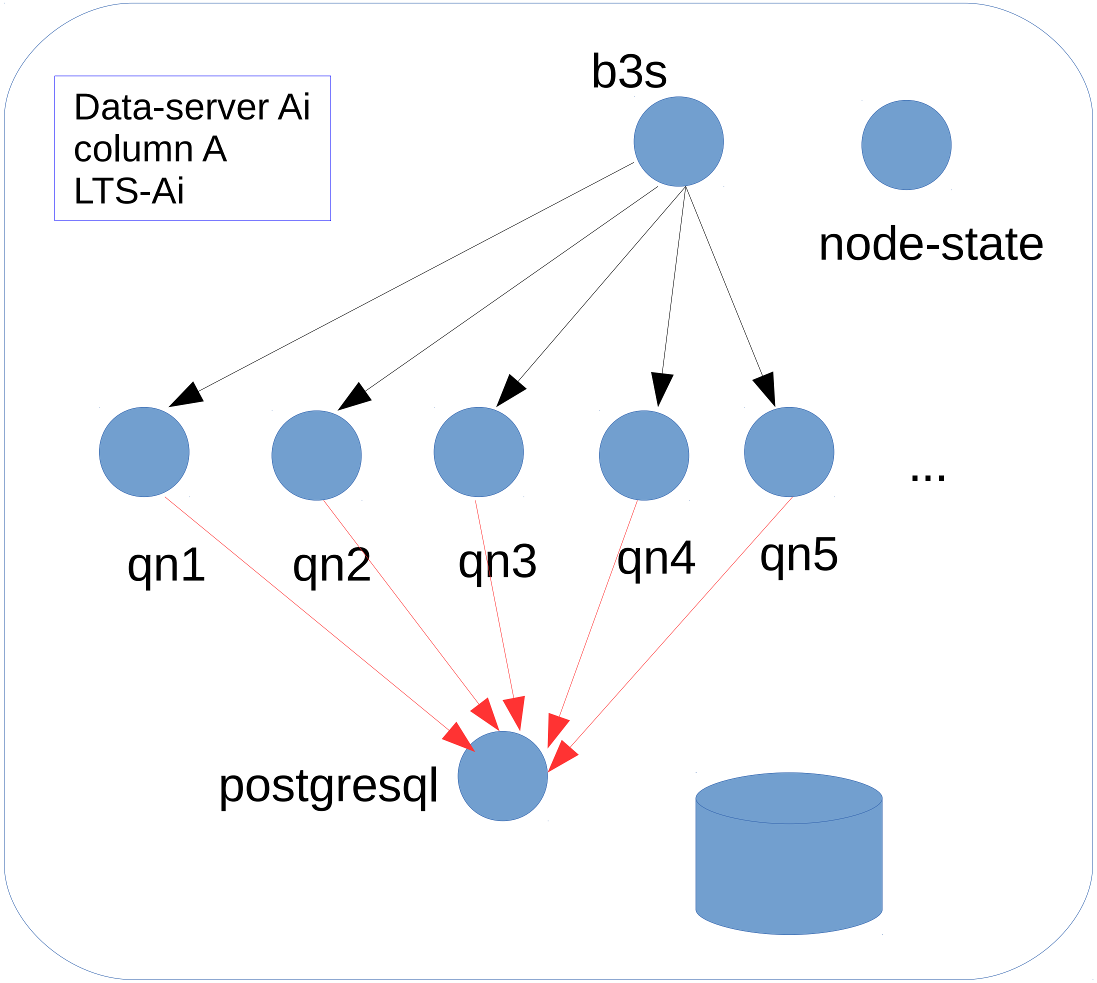
node-state

ss1 ss2 ss3 ss4 ss5 ...

qt1 qt2 qt3 qt4 qt5 ...

- ss = session
- qt = query-tree

b3s processes – data server



- one node-state process
- triple-db split to columns!
- qn = query-node
- one supervisor b3s per site

access triple-store

Research topics

- Graph algebra
- Graph partitioning
- Local storage manager
- Query scheduling
- Computation of database statistics
- Query optimisation
- Multi-threaded architecture of query executor

Research topics

- Design of algebra of graphs
- RDF algebra based on relational algebra
  - Graph pattern = SQL block
- Denotational semantics
- Implementation in parallel comp env

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

RDF algebra

Triple-patterns

Graph-patterns

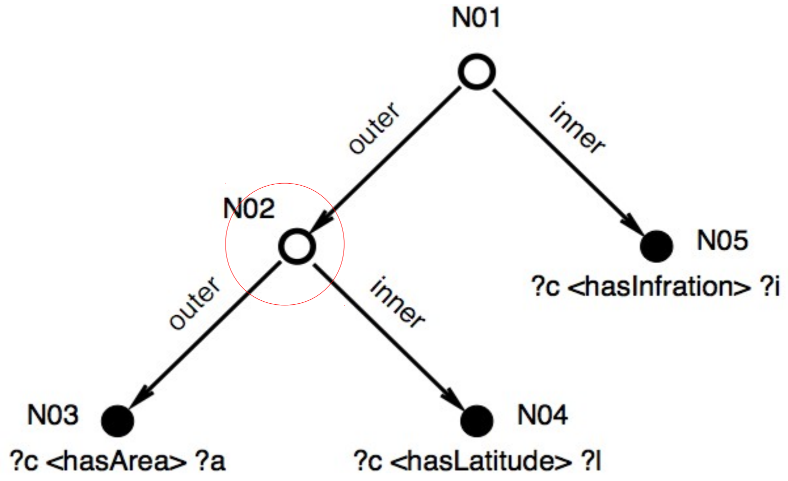
$$
\begin{aligned}
GP &::= TP \mid select(GP, C) \mid join(GP, GP) \mid union(GP, GP) \mid \\
&\quad\; intsc(GP, GP) \mid diff(GP, GP) \mid leftjoin(GP, GP) \\
TP &::= (S \mid V, P \mid V, O \mid V) \\
C &::= V\ OP\ V \mid V\ OP\ O \mid C \wedge C \mid C \vee C \mid \neg\, C \\
OP &::= \,=\, \mid \,\neq\,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

Conditions

Variables

RDF algebra

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



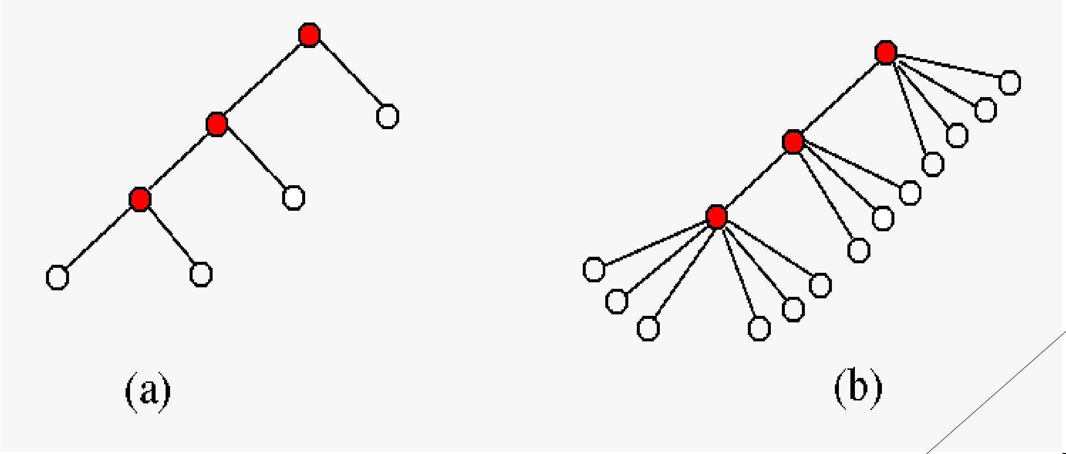
$$\begin{aligned}[\![join(gp_1, gp_2)]\!]_{db} = \{\, g_1 \cup g_2 \mid g_1 \in [\![gp_1]\!]_{db} \wedge g_2 \in [\![gp_2]\!]_{db} \wedge \\ \forall v \in vs : val(v, gp_1, g_1) = val(v, gp_2, g_2) \,\}\end{aligned}$$

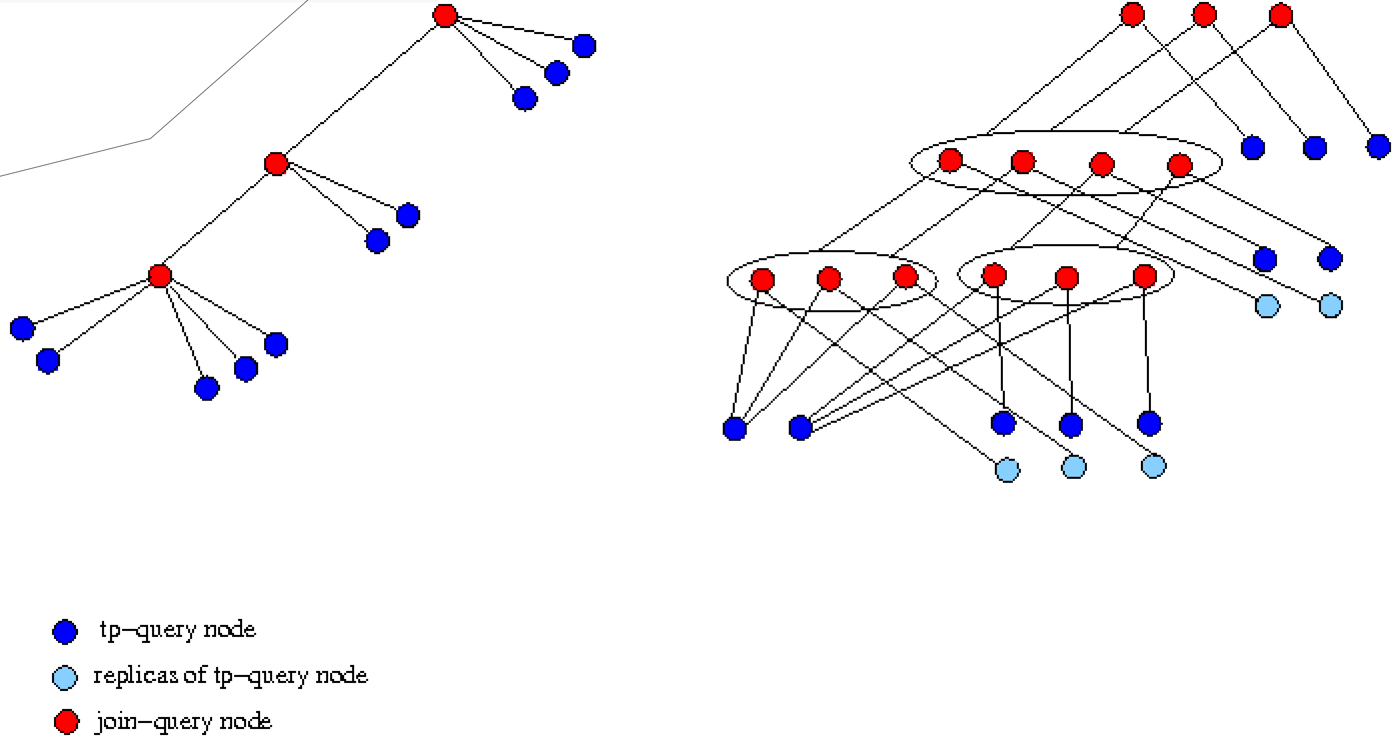
- Index nested-loop join
  - Exploiting DB indexes on subsets of { S, P, O }

RDF algebra implementation

- Algebra operations implemented as processes on data-servers
- Query trees are left-deep trees (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
- Experiments with bushy trees

Query tree implementation





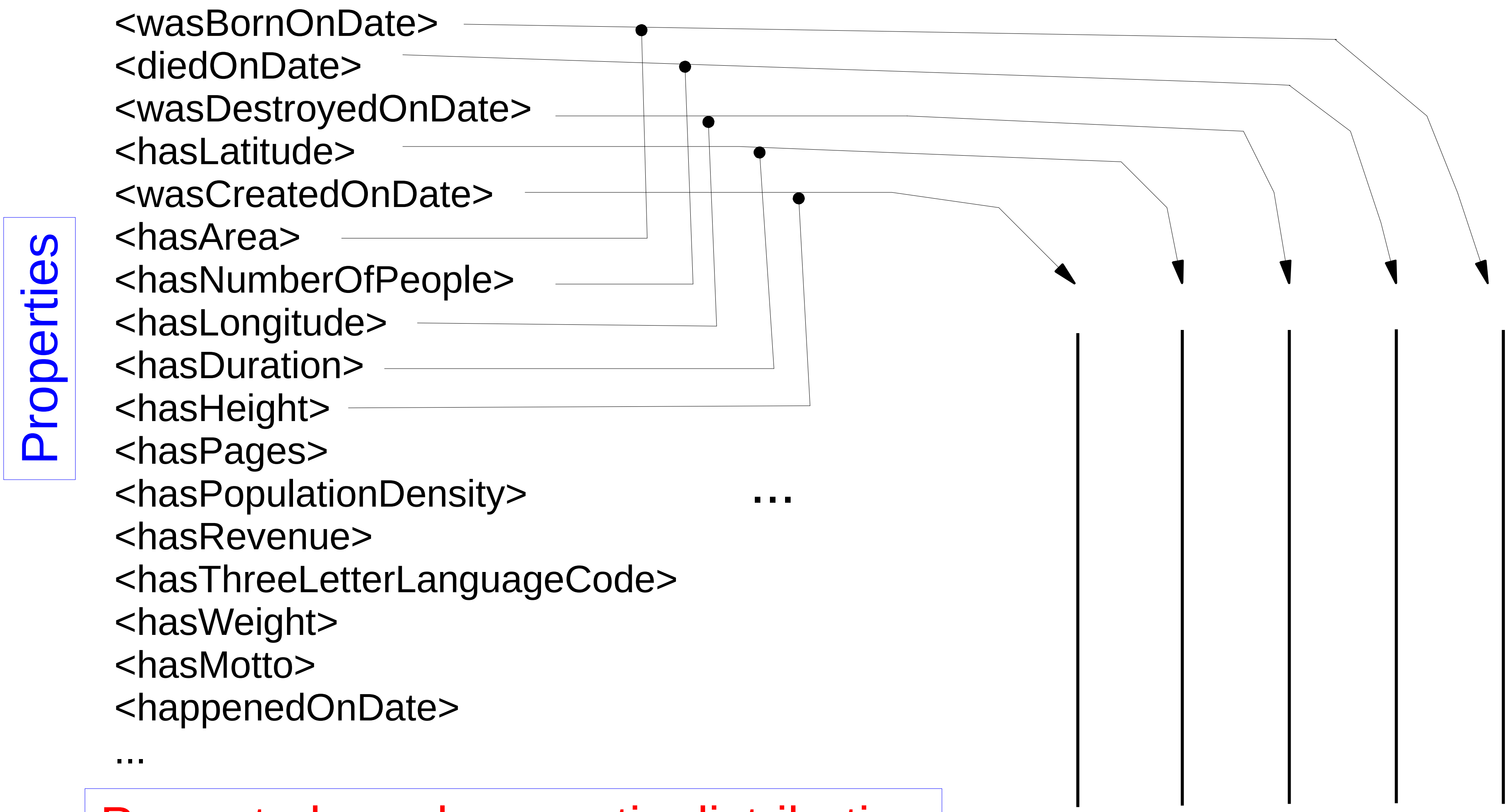
Research topic

- Graph partitioning
- How to partition large graph among the multiple servers to speed-up graph processing?
- Graph-theoretic approaches
  - Identify strongly connected components
- DB approaches
  - Hash-based partitioning
  - Semantic partitioning methods

Graph partitioning

- Query that addresses large part of database should be distributed to as many data servers as possible
- Query that addresses small part of database needs few data servers
- Semantic distribution
  - Distribution based on triple-base schema
  - Property-based distribution
  - Class-based distribution
  - Based on {S, P, O} subset lattice

Semantic distribution



Properties

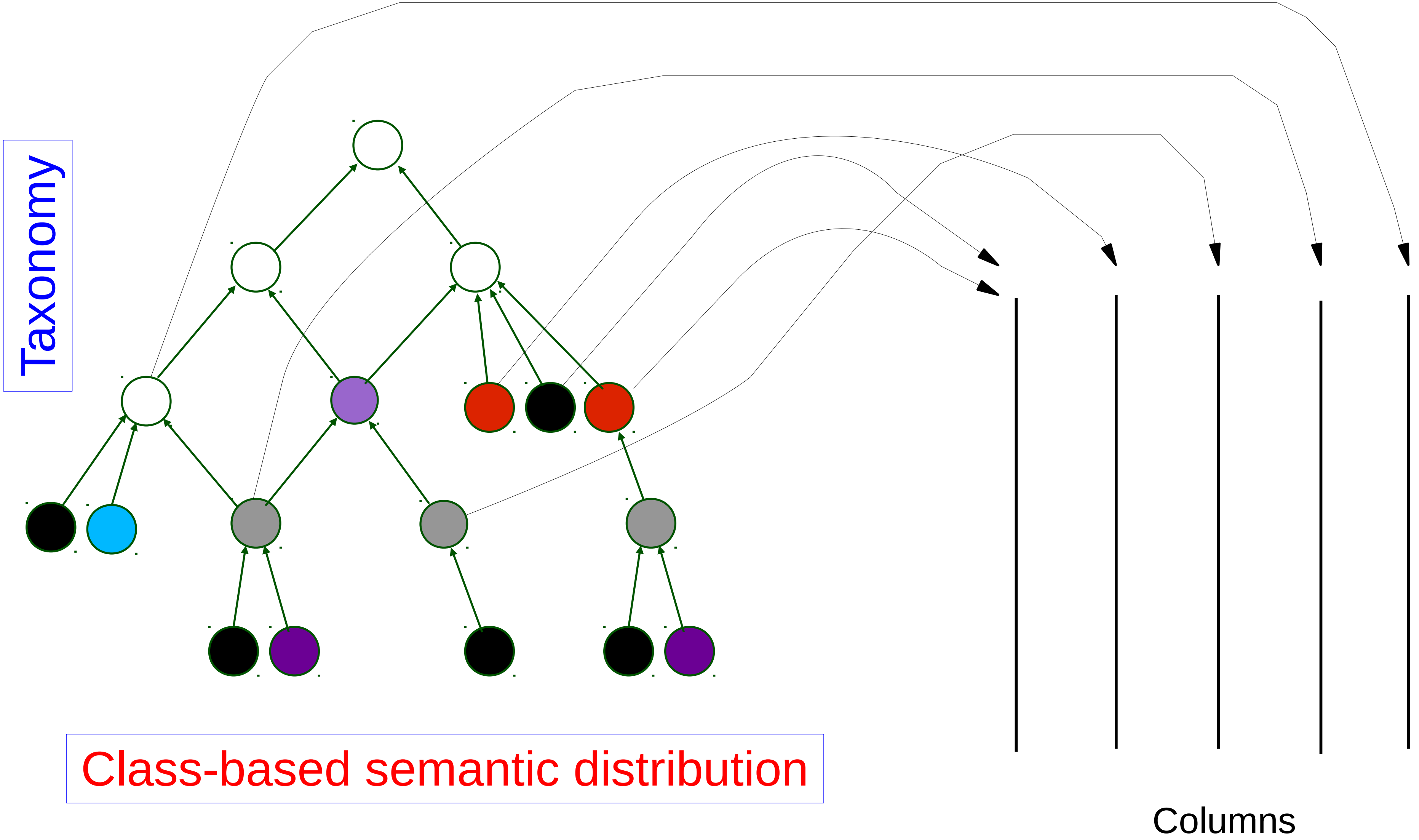
<wasBornOnDate>
<diedOnDate>
<wasDestroyedOnDate>
<hasLatitude>
<wasCreatedOnDate>
<hasArea>
<hasNumberOfPeople>
<hasLongitude>
<hasDuration>
<hasHeight>
<hasPages>
<hasPopulationDensity>
<hasRevenue>
<hasThreeLetterLanguageCode>
<hasWeight>
<hasMotto>
<happenedOnDate>
...

...

Columns

Property-based semantic distribution

Semantic distribution



Research topic

- Local storage manager
  - Relational approach
    - Triple-table with 7 indexes
  - Special new storage system
    - New indexes and storage structures
  - Graph-theoretic approach
    - Graph represented as nodes and links
    - New paradigm (neo4j), no joins
  - Our approach
    - Postgers triple-table + 7 indexes + Large cache in RAM

Main-memory usage

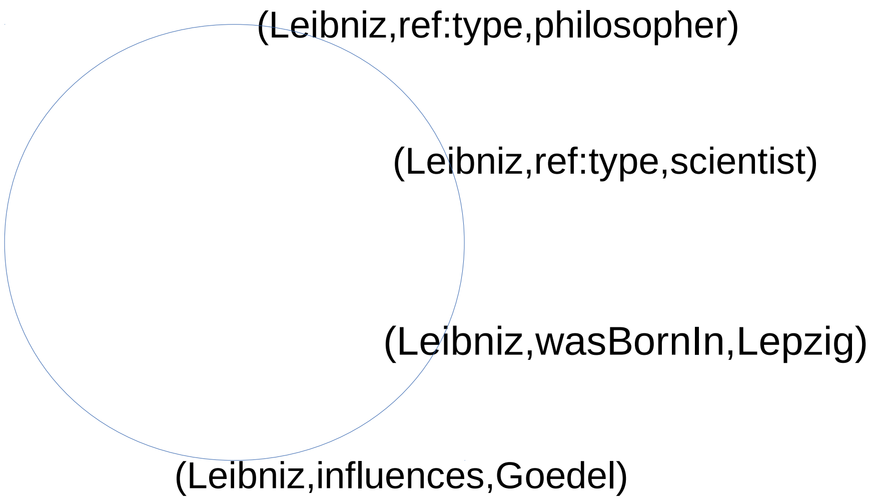
- Main-memory databases
  - Trinity DBMS
- Hybrids using large RAM and disk
  - Caching data into RAM
  - Storage manager cache

Distributed cache

- Cost of RAM allows moving significant part of triple-store in RAM
- Problem similar to using cache in multi-processor systems
  - We will use affinity scheduling
  - Queries of one session tend to allocate the same servers to utilise DB cache

Epsilon cache

```
philosopher rdfs:subClassOf   person .
scientist     rdfs:subClassOf   person .
person        influences   person .
person        wasBornIn   location .
Plato  rdf:type       philosopher .
Leibniz       rdf:type       philosopher .
Leibniz       rdf:type       scientist .
Goedel        rdf:type       scientist .
Athens        rdf:type       location .
Leipzig       rdf:type       location .
Brno  rdf:type       location .
Plato  wasBornIn   Athens .
Plato  influences   Leibniz .
Leibniz       wasBornIn   Leipzig .
Leibniz       influences   Goedel .
Goedel        wasBornIn   Brno .
```


(Leibniz,ref:type,philosopher)
(Leibniz,ref:type,scientist)
(Leibniz,wasBornIn,Lepzig)
(Leibniz,influences,Goedel)

```
epsilon# load ../simple.tsv
Loading...
done load
epsilon# print store
3store
rid=0 S=philosopher P=rdfs:subClassOf O=person iS=0 iP=1 iO=2 iSP=0 iSO=0 iPO=1
rid=1 S=scientist P=rdfs:subClassOf O=person iS=1 iP=0 iO=0 iSP=1 iSO=1 iPO=0
rid=2 S=person P=influences O=person iS=3 iP=14 iO=1 iSP=2 iSO=2 iPO=2
rid=3 S=person P=wasBornIn O=location iS=2 iP=15 iO=10 iSP=3 iSO=3 iPO=3
rid=4 S=Plato P=rdf:type O=philosopher iS=12 iP=10 iO=5 iSP=4 iSO=4 iPO=5
rid=5 S=Leibniz P=rdf:type O=philosopher iS=14 iP=4 iO=4 iSP=6 iSO=5 iPO=4
rid=6 S=Leibniz P=rdf:type O=scientist iS=5 iP=5 iO=7 iSP=5 iSO=6 iPO=7
rid=7 S=Goedel P=rdf:type O=scientist iS=15 iP=6 iO=6 iSP=7 iSO=7 iPO=6
rid=8 S=Athens P=rdf:type O=location iS=8 iP=7 iO=3 iSP=8 iSO=8 iPO=10
rid=9 S=Leipzig P=rdf:type O=location iS=9 iP=8 iO=8 iSP=9 iSO=9 iPO=8
rid=10 S=Brno P=rdf:type O=location iS=10 iP=9 iO=9 iSP=10 iSO=10 iPO=9
rid=11 S=Plato P=wasBornIn O=Athens iS=4 iP=3 iO=11 iSP=11 iSO=11 iPO=11
rid=12 S=Plato P=influences O=Leibniz iS=11 iP=2 iO=12 iSP=12 iSO=12 iPO=12
rid=13 S=Leibniz P=wasBornIn O=Leipzig iS=6 iP=11 iO=13 iSP=13 iSO=13 iPO=13
rid=14 S=Leibniz P=influences O=Goedel iS=13 iP=12 iO=14 iSP=14 iSO=14 iPO=14
rid=15 S=Goedel P=wasBornIn O=Brno iS=7 iP=13 iO=15 iSP=15 iSO=15 iPO=15
done print store
epsilon#
```

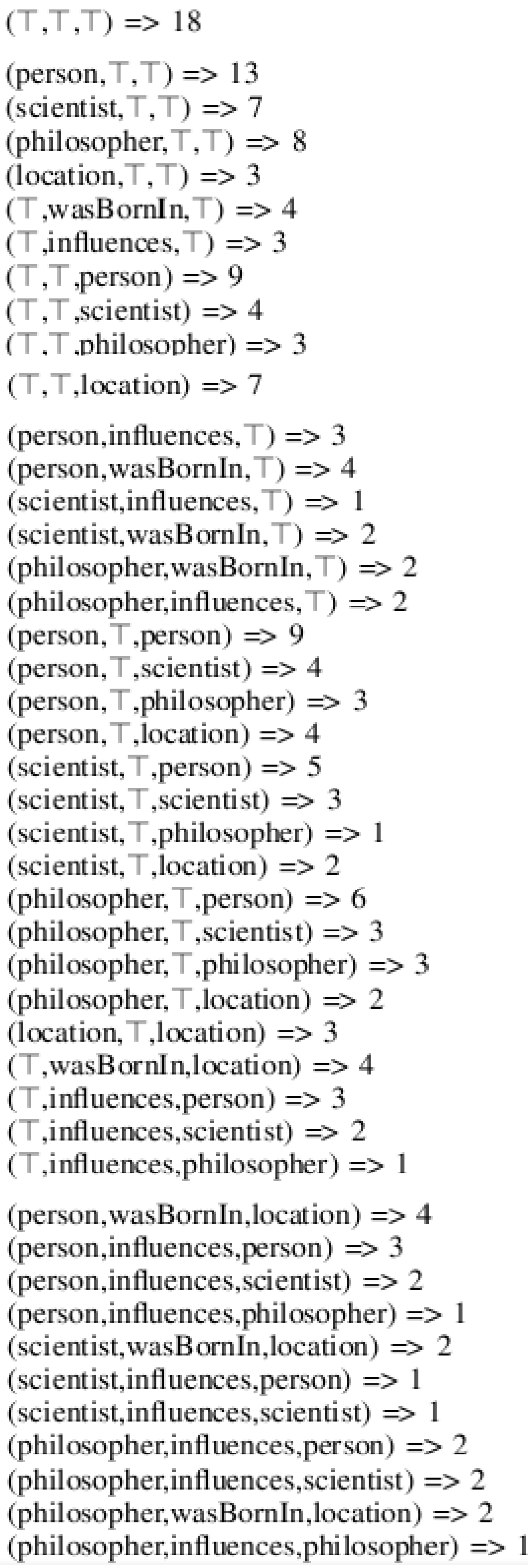
Research topic

- Computation of database statistics
- Estimation of the size of triple-pattern result
- The use of statistics
  - Query optimization
  - Data distribution
- Some solutions
  - Statistics of indexes S,P,O,SP,SO,PO,SPO
  - Gathering histograms for all triple-patterns
  - Gathering statistics for frequent paths

```
get_types_of(i: identifier) -> set_of_identifiers
begin
   if (i is individual identifier) then
      return { c | (i,rdf:type,c) IN  g };
   if (i is class identifier)
      return { i };
end;



transitive_closure(a: set_of_identifiers)
      -> set_of_identifiers
begin
   repeat
      b = a;
      for each i IN b do
        extend a with c: (i,rdfs:subClassOf,c) IN g;
        extend a with c: (i,rdfs:subPropertyOf,c) IN g;
   until a == b;
end;



compute_statistics((s,p,o): triple)
begin
   gs = get_types_of(s);
   gs = transitive_closure(gs);

   gp = get_types_of(p);
   gp = transitive_closure(gp);

   go = get_types_of(o);
   go = transitive_closure(go);

   for each cs IN gs do
     for each cp IN gp do
       for each co IN go do
         increment couter of (cs,cp,co) by 1;
end;
```

Complete schema

ε



(⊤,⊤,⊤) => 18

(person,⊤,⊤) => 13
(scientist,⊤,⊤) => 7
(philosopher,⊤,⊤) => 8
(location,⊤,⊤) => 3
(⊤,wasBornIn,⊤) => 4
(⊤,influences,⊤) => 3
(⊤,⊤,person) => 9
(⊤,⊤,scientist) => 4
(⊤,⊤,philosopher) => 3
(⊤,⊤,location) => 7

(person,influences,⊤) => 3
(person,wasBornIn,⊤) => 4
(scientist,influences,⊤) => 1
(scientist,wasBornIn,⊤) => 2
(philosopher,wasBornIn,⊤) => 2
(philosopher,influences,⊤) => 2
(person,⊤,person) => 9
(person,⊤,scientist) => 4
(person,⊤,philosopher) => 3
(person,⊤,location) => 4
(scientist,⊤,person) => 5
(scientist,⊤,scientist) => 3
(scientist,⊤,philosopher) => 1
(scientist,⊤,location) => 2
(philosopher,⊤,person) => 6
(philosopher,⊤,scientist) => 3
(philosopher,⊤,philosopher) => 3
(philosopher,⊤,location) => 2
(location,⊤,location) => 3
(⊤,wasBornIn,location) => 4
(⊤,influences,person) => 3
(⊤,influences,scientist) => 2
(⊤,influences,philosopher) => 1

(person,wasBornIn,location) => 4
(person,influences,person) => 3
(person,influences,scientist) => 2
(person,influences,philosopher) => 1
(scientist,wasBornIn,location) => 2
(scientist,influences,person) => 1
(scientist,influences,scientist) => 1
(philosopher,influences,person) => 2
(philosopher,influences,scientist) => 2
(philosopher,wasBornIn,location) => 2
(philosopher,influences,philosopher) => 1

Research topic

- Distributed query optimization
  - The hardest problem in database systems
  - Exploiting relational query optimization
  - Simplicity of triple-store model gives hope...
  - Regular path queries
    - New paradigm for optimization
    - Andreas T. Schmidt, KIT

Regular path queries

- SPARQL 1.1
  - Includes regular path expressions
- Examples

```
?x foaf:mbox <mailto:alice@example> .
?x foaf:knows/foaf:knows/foaf:name ?name .
```

=

```
?x  foaf:mbox <mailto:alice@example> .
?x  foaf:knows ?a1 .
?a1 foaf:knows ?a2 .
?a2 foaf:name ?name .
```

```
?x foaf:mbox <mailto:alice@example> .
?x foaf:knows+/foaf:name ?name .
```

Find the names of all the people that
can be reached from Alice by foaf:knows.

```
<http://example/thing> rdf:type/rdfs:subClassOf* ?type .
```

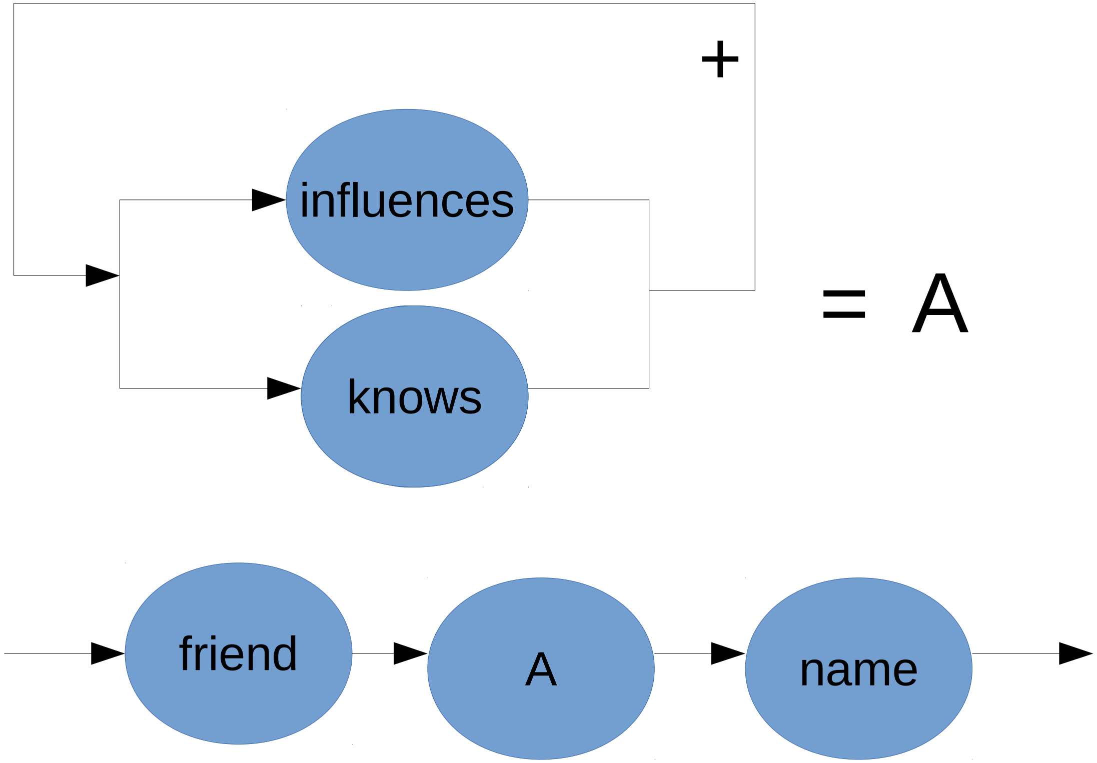
Limited inference:
all types and supertypes
of a resource.

```
?x rdf:type/rdfs:subClassOf* ?type .
```

All resources and all their inferred types.

Regular path queries

- Hierarchical optimization based on dynamic programming
  - RPQ: friend /(knows|influences)+/name



```
# Algorithm: optimize regular path query
optimize-rpq q

if q is simple path then
   qt <- construct-query-tree q;
   ot <- optimize-path-block qt;
   return ot;

ql <- decompose q into outermost components;

tl <- empty list;
for each qi in ql do
   ti <- optimize_rpq qi;
   tl <- ti tl;

qt <- construct-query-tree tl;
ot <- optimize-path-block qt;
```

Research topic

- Efficient scheduling of queries on cluster of servers
  - Task: map nodes of query tree to processes on data servers
  - Input: query tree as data structure
  - Output: tree of processes running on cluster
  - Front server function
- Distribution of queries into cluster columns depends entirely on data distribution
  - Should work so that queries adressing large part of DB should allocate more columns

Scheduling

- Many query trees can be executed in parallel
- Triple-pattern query node must be evaluated on server where data is stored
- Join query node can be evaluated either on inner or outer query node of join
- Load-ballancing among replicas (data servers) of columns
  - Each query node can be started on one of rows (data servers) of a given column

Scheduling

- Load balancing algorithms:
  - Random
  - Dynamic load-balancing
  - Affinity scheduling
- Dynamic load ballancing and affinity sheduling are not easy to implement fast
  - The rows (replica server) of columns must be decided fast
  - Global data structure or data synchronisation

Research topic

- Multi-threaded architecture of query executor
  - We have multiple cores that could be utilized
  - Exploit programming languages paradigm
    - Erlang
  - Parallel algorithm design
    - Boris Motik, Oxford