Графы знаний

Лекция 3 - Хранение знаний в графах и обработка запросов

М. Галкин, Д. Муромцев

Сегодня

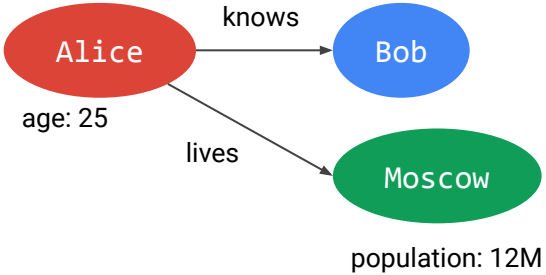
1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. **Хранение знаний в графах - SPARQL & Graph Databases**
4. Однородность знаний - Reification & RDF* & SHACL & ShEx
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Введение в теорию графов - Graph Theory Intro
7. Векторные представления графов - Knowledge Graph Embeddings
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding

Содержание

- Специфика графов знаний в аспекте хранения данных (RDF / LPG) (5 мин)
- Стандарт SPARQL (40 мин)
  - Базовый синтаксис
  - Простые запросы
  - Агрегация и фильтры
  - SPARQL 1.1
- Типы хранилищ для графов знаний (индексы, представление) (30 мин)
  - SQL-based СУБД, трансляция SPARQL -> SQL
  - RDF stores (+HDT)
  - Graph databases (neo4j, graphdb), трансляция SPARQL -> Cypher
  - Краткое интро Cypher
- Apache Tinkerpop (Gremlin), GraphQL (5 мин)

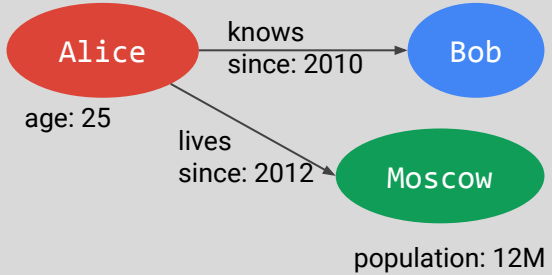
Графовые СУБД: RDF vs LPG

RDF



- Язык запросов: SPARQL
- Атрибуты предикатов ограничены RDFS/OWL
- Схема графа - семантическая
- Возможен логический вывод в процессе выполнения запроса

LPG (Labeled Property Graph)

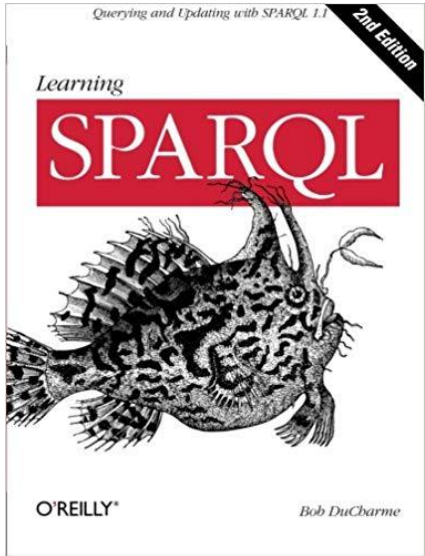


- Язык запросов: Cypher, Gremlin, GraphQL
- Атрибуты предикатов не ограничены
- Схема графа - не семантическая
- Не способны к логическому выводу
- Разные виды графов: (не) направленные, взвешенные, гиперграфы

SPARQL

SPARQL - **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage

- Подразумевает графовую модель организации
- Язык запросов и сетевой протокол взаимодействия СУБД
- 2008 - Стандарт W3C
- 2013 - SPARQL 1.1
- Синтаксис основан на Turtle
- Поддерживает логический вывод
- Позволяет конструировать графы во время запроса
- Позволяет производить федеративные запросы



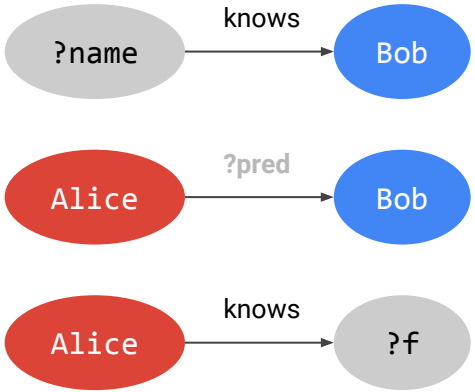
SPARQL - Basic Graph Pattern

```
?name ?friend ?city
```

- Переменные объявляются с помощью ? или $

SPARQL - Basic Graph Pattern

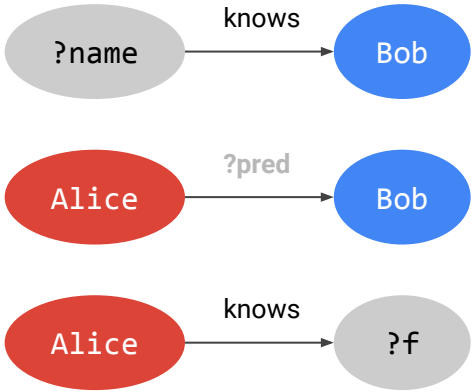
```
?name ?friend ?city
```



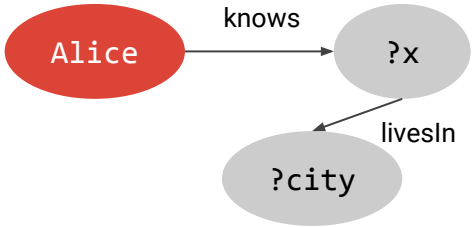
- Переменные объявляются с помощью ? или $
- Шаблон подграфа (graph pattern) - RDF триплет, содержащий переменную на месте s, p, o
  - `?name :knows :Bob`
  - `:Alice ?predicate :Bob`
  - `:Alice :knows ?name`
  - `?s ?p ?o` - наиболее общий

SPARQL - Basic Graph Pattern

```
?name ?friend ?city
```



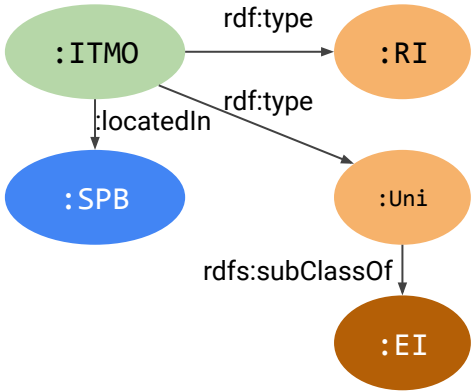
- Переменные объявляются с помощью ? или $
- Шаблон подграфа (graph pattern) - RDF триплет, содержащий переменную на месте s, p, o
  - `?name :knows :Bob`
  - `:Alice ?predicate :Bob`
  - `:Alice :knows ?name`
  - `?s ?p ?o` - наиболее общий



- Базовый шаблон графа (Basic Graph Pattern, BGP) - конъюнкция graph patterns:

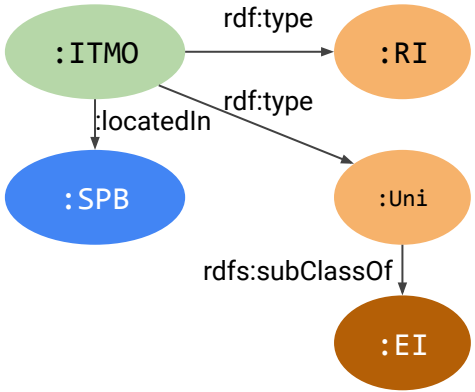
```
{ :Alice :knows   ?x .
  ?x     :livesIn ?city . }
```

SPARQL Graph Pattern Matching




```
:ITMO_University   rdf:type            :University .
:ITMO_University   rdf:type            :Research_Institution .
:ITMO_University   :locatedIn          :Saint_Petersburg .
:University        rdfs:subClassOf     :Educational_Institution .
```

SPARQL Graph Pattern Matching

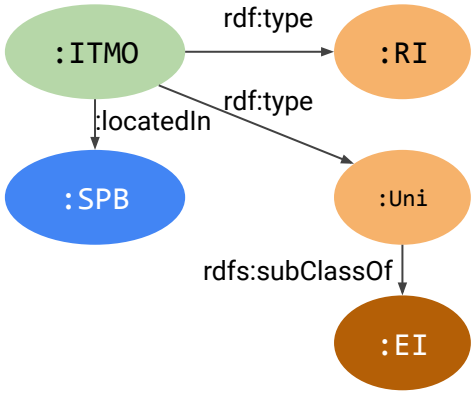



```
:ITMO_University   rdf:type              :University .
:ITMO_University   rdf:type              :Research_Institution .
:ITMO_University   :locatedIn            :Saint_Petersburg .
:University        rdfs:subClassOf       :Educational_Institution .
```

```
SELECT ?type WHERE {
   :ITMO_University rdf:type ?type .
}
```

?type
:University
:Research_Institution

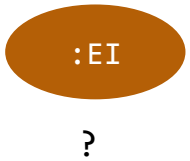
SPARQL Graph Pattern Matching




```
:ITMO_University   rdf:type              :University .
:ITMO_University   rdf:type              :Research_Institution .
:ITMO_University   :locatedIn            :Saint_Petersburg .
:University        rdfs:subClassOf       :Educational_Institution .
```

```
SELECT ?type WHERE {
   :ITMO_University rdf:type ?type .
}
```

?type
:University
:Research_Institution



:Educational_Institution как суперкласс :University тоже подходит, но требуется ризонинг для материализации

```
:ITMO_University rdf:type :Educational_Institution .
```

SPARQL Query Structure

Префиксы

Тип запроса

Возвращаемые переменные

Источник (граф)

BGP

Модификаторы

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

SELECT ?type ?city

FROM <named_graph>

WHERE {
    ?s     rdf:type      ?type .
    ?s     :locatedIn    ?city .
    ?city  :population   ?num .
}

ORDER BY <> LIMIT <num> OFFSET <num>
```

SPARQL Query Structure - Префиксы

Префиксы

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX dbo:  <http://dbpedia.org/ontology/> .
```

- Используются для улучшения читаемости
- Во время обработки запроса префикс переписывается в полный URI
- Как правило, используются `#-URIs` или `/-URIs`

```
?x rdf:type ?type
?x <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type

?s dbo:city ?city
?s <http://dbpedia.org/ontology/city> ?city
```

SPARQL Query Structure - Типы запросов - SELECT

Префиксы

Тип запроса

Возвращаемые переменные

SELECT - возвращает таблицу со значениями переменных (projections)

```
SELECT ?s WHERE {
     ?s :knows ?o .
}
```

→

```
:Alice :knows :Bob .
:Alice :knows :Ann .
```

→

?s
:Alice
:Alice

SELECT DISTINCT - возвращает уникальные значения

```
SELECT DISTINCT ?s WHERE {
     ?s :knows ?o .
}
```

?s
:Alice

SPARQL Query Structure - Типы запросов - ASK

Префиксы

Тип запроса

Возвращаемые переменные

ASK - возвращает True/False если данный BGP существует в графе

```
ASK WHERE {
    ?s :knows :Bob .
}
```

→

```
:Alice :knows :Bob .
:Alice :knows :Ann .
```

→ True

```
ASK WHERE {
    :Alice :knows :John .
}
```

False

```
ASK WHERE {
    ?s ?p ?o .
}
```

True

SPARQL Query Structure - Типы запросов - CONSTRUCT

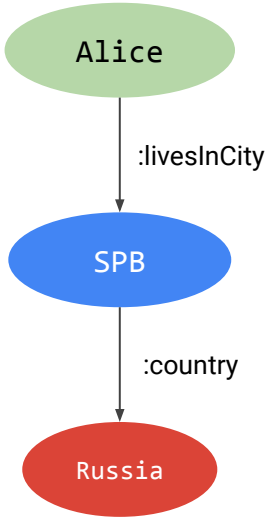
Префиксы

Тип запроса

Возвращаемые переменные

CONSTRUCT - возвращает граф со значениями переменных из BGP

```
CONSTRUCT {
   ?x :livesInCountry ?country .
} WHERE {
   ?x    :livesInCity ?city .
   ?city :country     ?country .
}
```



Исходный граф:

```
:Alice :livesInCity :Saint_Petersburg .
:Saint_Petersburg :country :Russia .
```

SPARQL Query Structure - Типы запросов - CONSTRUCT

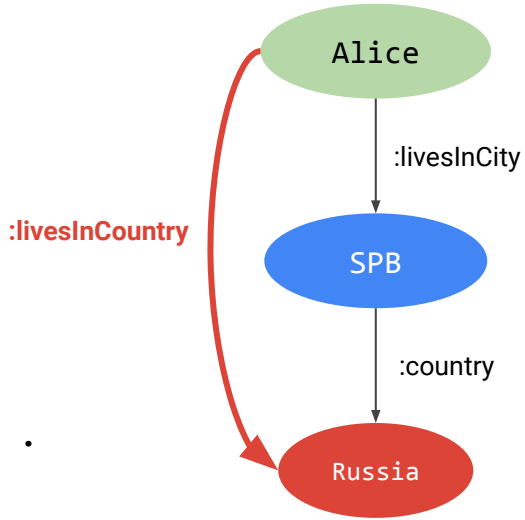
Префиксы

Тип запроса

Возвращаемые переменные

CONSTRUCT - возвращает граф со значениями переменных из BGP

```
CONSTRUCT {
   ?x :livesInCountry ?country .
} WHERE {
   ?x    :livesInCity ?city .
   ?city :country     ?country .
}
```



Исходный граф:

```
:Alice :livesInCity :Saint_Petersburg .
:Saint_Petersburg :country :Russia .
```

```
:Alice :livesInCountry :Russia .
```

SPARQL Query Structure - Типы запросов - DESCRIBE

Префиксы

Тип запроса

Возвращаемые переменные

DESCRIBE - возвращает описание ресурса.
Реализация зависит от конкретной СУБД, чаще всего - паттерн `?p ?o`

```
DESCRIBE :Alice
```

```
DESCRIBE ?x WHERE {
     ?x :knows :Bob .
}
```

```
:Alice :knows :Bob .
:Alice :knows :Ann .
```

SPARQL Query Structure - Типы запросов - SPARQL 1.1

Префиксы

Тип запроса

Возвращаемые переменные

Стандарт SPARQL 1.1 определяет новые типы запросов:

- **INSERT** - вставка триплета в граф.
- **DELETE** - удаление триплета или паттерна из графа.

```
INSERT DATA
{ <http://example/book1>  dc:title  "War and Peace" }

DELETE DATA
{ <http://example/book1>  dc:title  "War and Peace" }


DELETE { ?person foaf:givenName 'Bill' }
INSERT { ?person foaf:givenName 'William' }
WHERE { ?person foaf:givenName 'Bill' }
```

SPARQL Query Structure - FROM

Префиксы

Тип запроса

Возвращаемые переменные

Источник (граф)

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

SELECT ?type ?city

FROM <graphURI>
```

- `FROM` указывает на именованный граф, к которому будут отправляться запросы
- Если не указывать `FROM` - выполнение по всем именованным графам в СУБД

SPARQL Query Structure - BGP

Префиксы

Тип запроса

Возвращаемые переменные

Источник (граф)

BGP

`Basic Graph Pattern (BGP)` - конъюнкция graph patterns, объединенные минимум одной общей переменной (по которой будет производиться JOIN).

```
SELECT ?s ?num

WHERE {
    ?s     rdf:type      ?type .
    ?s     :locatedIn    ?city .
    ?city  :population   ?num .
}
```

SPARQL Query Structure - Solution Modifiers

Префиксы

Тип запроса

Возвращаемые переменные

Источник (граф)

BGP

Модификаторы

Модификаторы результатов изменяют вывод запроса:

- **`ORDER BY ASC/DESC`** - сортировка результатов по возрастанию или убыванию

```
SELECT ?x WHERE { :Alice :knows ?x . } ORDER BY ASC(?x)
```

- **`LIMIT`** - ограничивает число выводимых результатов

```
SELECT * WHERE { ?s ?p ?o . } LIMIT 10
```

- **`OFFSET`** - сдвигает внутренний счетчик числа результатов, часто используется вместе с LIMIT для страничной обработки

```
SELECT * WHERE { ?s ?p ?o . } OFFSET 20 LIMIT 10
```

SPARQL Query Structure - Example

Префиксы

Тип запроса

Возвращаемые переменные

Источник (граф)

BGP

Модификаторы

Авторы и названия их заметных произведений, отсортированные по возрастанию по авторам, с 10 по 110 значения

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT DISTINCT ?author ?title
FROM <http://dbpedia.org>
WHERE {
     ?author   rdf:type           dbo:Writer ;
               rdfs:label         ?author_name ;
               dbo:notableWork    ?work .
     ?work     rdfs:label         ?title .
} ORDER BY ASC(?author) LIMIT 100 OFFSET 10
```

SPARQL Query Structure - Blank Nodes

Blank Nodes

Неименованные сущности (blank nodes) могут находиться в шаблоне подграфа только на позиции субъекта или объекта, при этом их нельзя материализовать как результат SELECT-запроса.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?name ?date
FROM <http://dbpedia.org>
WHERE {
     _:x  rdf:type dbo:Writer ;
          dbp:award [ dbp:awardName ?name ;
                      dbp:awardDate ?date ]}
```

SPARQL FILTER

- Привязаны к graph pattern
- Операторы и функции
- String, numerical, regexp сравнения
- Трехзначная логика оценивания True/False/Error

Для фильтрации graph pattern используется ключевое слово **FILTER**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT DISTINCT ?author ?title
FROM <http://dbpedia.org>
WHERE {
      ?author    rdf:type         dbo:Writer ;
                 rdfs:label       ?author_name ;
                 dbo:notableWork  ?work .
      ?work      rdfs:label ?title FILTER (lang(?title)=”en”);
                 dbo:numberOfPages ?numPages FILTER (?numPages > 500) .
} ORDER BY ASC(?author) LIMIT 100 OFFSET 10
```

SPARQL FILTER I

```
FILTER (?a = 5 && ?b < 0)
```

```
FILTER (( ?a * 2) < 100 )
```

```
FILTER (?numPages < 100 )
```

```
FILTER ( isLiteral(?a) )
```

- Логические: `!` (отрицание), `&&` (И), `||` (ИЛИ) для логических типов
- Математические: `+`, `-`, `*`, `/` , для числовых типов
- Сравнительные: `>`, `<`, `>=`, `<=`, `=`, `!=`
- Унарные функции: `isURI()`, `isLiteral()`, `isBlank()`, `bound()`

SPARQL FILTER II

```
FILTER (?a = 5 && ?b < 0)
```

```
FILTER (( ?a * 2) < 100 )
```

```
FILTER (?numPages < 100 )
```

```
FILTER ( isLiteral(?a) )
```

```
FILTER (lang(?title)=”en”)
```

```
FILTER (regex(?name,”Joe”))
```

```
FILTER (bif:contains(?a,”str”))
```

```
FILTER (NOT EXISTS
    {?s :knows :Bob} )
```

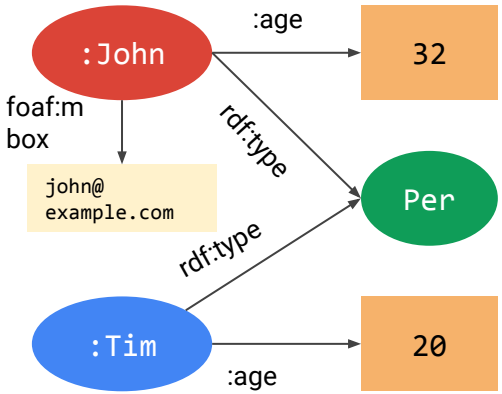
- Логические: `!` (отрицание), `&&` (И), `||` (ИЛИ) для логических типов
- Математические: `+, -, *, /` , для числовых типов
- Сравнительные: `>, <, >=, <=, =, !=`
- Унарные функции: `isURI(), isLiteral(), isBlank(), bound()`
- `str(), lang(), datatype(), sameTerm(), langMatches()`
- Регулярные выражения `regex(string, pattern [, flag])`
- Встроенные специфичные для СУБД функции, например, `bif:contains()` для проверки подстроки в Virtuoso
- Работа с отрицаниями: `NOT EXISTS {pattern}, MINUS {pattern}`

SPARQL FILTER III - Example

Комбинация разнообразных фильтров

```
:John :age 32 .
:John foaf:name "John"@en .
:Tim  :age 20.
:Tim  foaf:name "Tim"^^xsd:string .


SELECT DISTINCT ?friend WHERE {
?friend foaf:name ?name
   FILTER (regex(?name, “im”) || ?age > 25)
}
```

SPARQL OPTIONAL



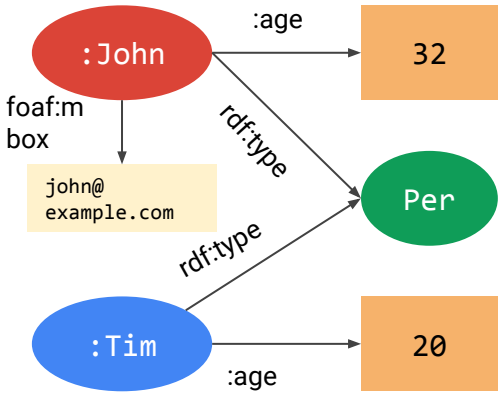

```
:John :age 32 .
:John foaf:mbox
“john@example.com”
:John rdf:type foaf:Person
:Tim rdf:type foaf:Person
:Tim :age 20.
```

У двух экземпляров одного класса может быть разных набор предикатов (исходящих ребер)

```
SELECT ?s WHERE {
      ?s rdf:type foaf:Person.
      ?s foaf:mbox ?mbox. }
```

?s
:John

SPARQL OPTIONAL II




```
:John :age 32 .
:John foaf:mbox
“john@example.com”
:John rdf:type foaf:Person
:Tim rdf:type foaf:Person
:Tim :age 20.
```

У двух экземпляров одного класса может быть разных набор предикатов (исходящих ребер)

```
SELECT ?s WHERE {
      ?s rdf:type foaf:Person.
      ?s foaf:mbox ?mbox. }
```

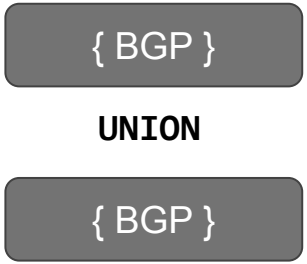
?s
:John

- **OPTIONAL** позволяет сделать некоторый шаблон опциональным.
- По своей работе **OPTIONAL** похож на **LEFT OUTER JOIN** в SQL.

```
SELECT ?s WHERE {
      ?s rdf:type foaf:Person.
      OPTIONAL {?s foaf:mbox ?mbox.}}
```

?s
:John
:Tim

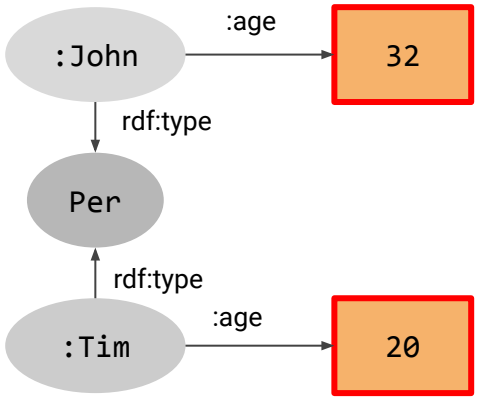
SPARQL UNION



- **UNION** - логическое ИЛИ в запросах
- Позволяет объединять несколько BGP
- Graph patterns могут различаться

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/resource/>

SELECT ?influencer ?influenced WHERE {
 {dbr:Stephen_King dbo:influenced ?influenced}
 UNION
 {dbr:Stephen_King dbo:influencedBy ?influencer}
}
```

SPARQL 1.1 - Aggregations



SPARQL 1.1 - с 2013 года, новые возможности для:

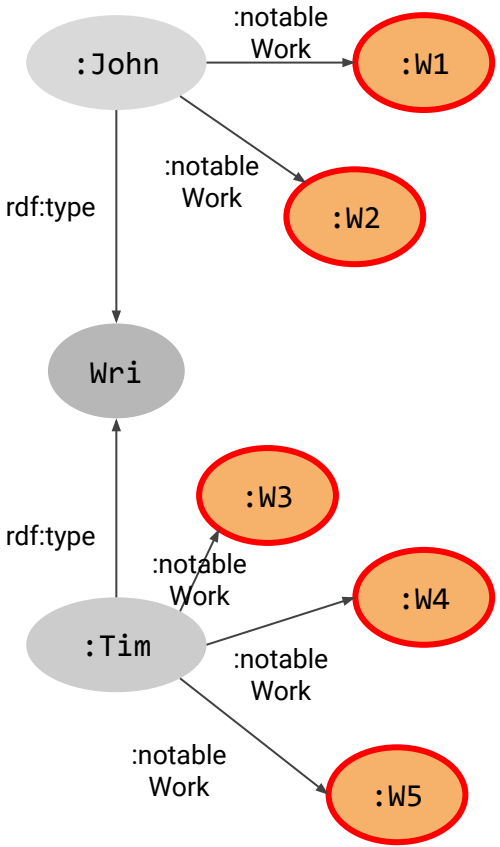
- Агрегации результатов
  - `min()`, `max()`, `avg()`, `sum()`, `count()`, `sample()`, `group_concat()` - агрегирующие функции

```
SELECT AVG(?age) as ?av_age WHERE {
   ?person rdf:type :Person ;
                :age      ?age .
}
```

  - Например, подсчет всех триплетов в графе

```
SELECT COUNT(*) as ?num
WHERE { ?s ?p ?o . }
```

SPARQL 1.1 - Aggregations



SPARQL 1.1 - с 2013 года, новые возможности для:

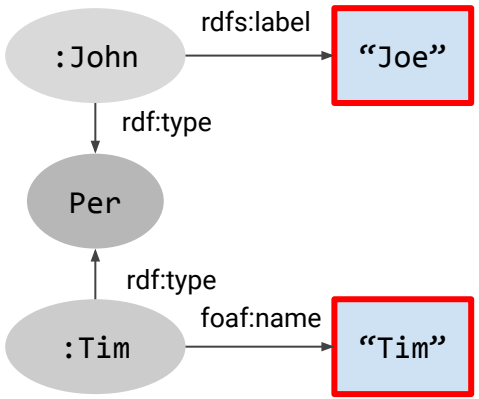
- Агрегации результатов
  - **GROUP BY** - для группировки результатов

```
SELECT ?author COUNT(?work) as ?num WHERE {
     ?author rdf:type dbo:Writer ;
             dbo:notableWork ?work
} GROUP BY ?author ORDER BY DESC(?num)
```

  - **HAVING** - ограничение на graph pattern и агрегация

```
SELECT ?author COUNT(?work) as ?num WHERE {
     ?author rdf:type dbo:Writer ;
             dbo:notableWork ?work
} GROUP BY ?author
HAVING COUNT(?work) = 3
ORDER BY DESC(?num)
```

SPARQL 1.1 - Property Paths - Alternate

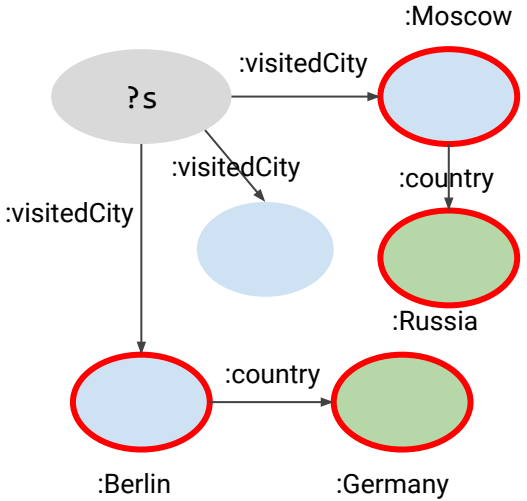


- Property Path - возможная последовательность или комбинация предикатов между двумя вершинами.
  - SPARQL - переменные в path не разрешаются
  - LPG / Cypher - переменные разрешаются
  - **Alternate path** - предикат1 | предикат2

```
SELECT ?name WHERE {
   ?person rdfs:label|foaf:name ?name . }
```

  - Sequence - path длиной >1
  - Inverse path
  - Negated path

SPARQL 1.1 - Property Paths - Sequence

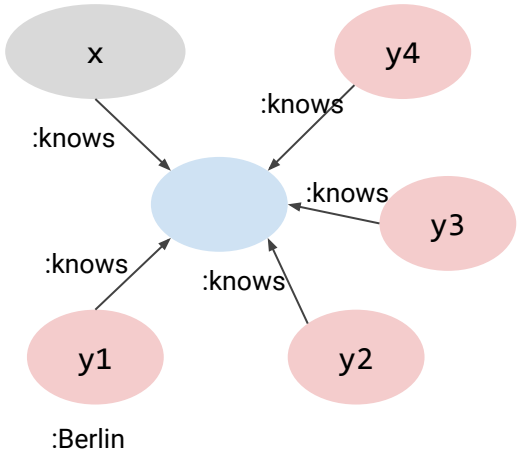


- Property Path - возможная последовательность или комбинация предикатов между двумя вершинами.
  - SPARQL - переменные в path не разрешаются
  - LPG / Cypher - переменные разрешаются
  - **`Alternate path`** - предикат1 | предикат2

  - **`Sequence`** - path длиной >1
    - + : > 1 предиката
    - * : ≥ 0 предиката
    - / : строгая последовательность

```
SELECT ?country WHERE {
   ?s :visitedCity/:country ?country . }

SELECT ?name WHERE {
   ?s :knows*/:name ?name . }
```

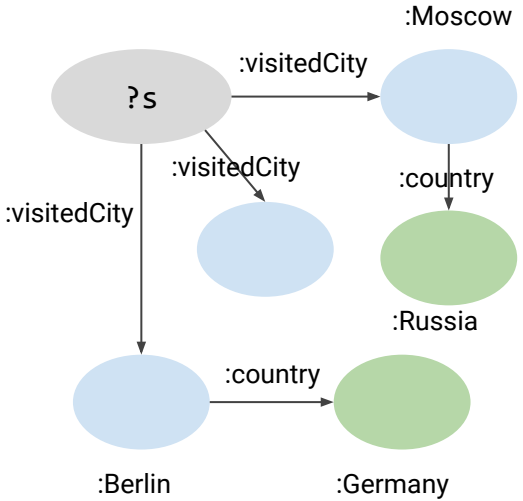
SPARQL 1.1 - Property Paths - Inverse



- Property Path - возможная последовательность или комбинация предикатов между двумя вершинами.
  - SPARQL - переменные в path не разрешаются
  - LPG / Cypher - переменные разрешаются
  - **`Alternate path`** - предикат1 | предикат2
  - **`Sequence`** - path длиной >1
  - **`Inverse path`** - инверсные пути

```
SELECT ?y WHERE {
 ?x foaf:knows/^foaf:knows ?y
 FILTER (?x != ?y) }
```

SPARQL 1.1 - Property Paths - Negated

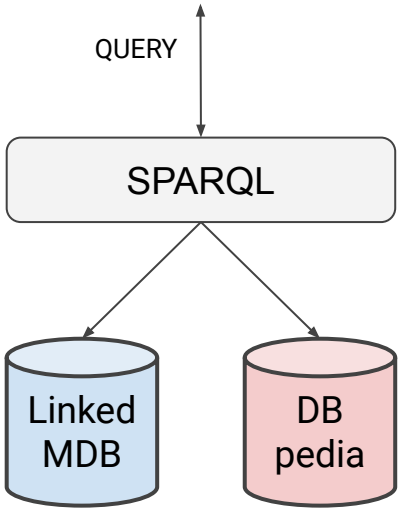


- Property Path - возможная последовательность или комбинация предикатов между двумя вершинами.
  - SPARQL - переменные в path не разрешаются
  - LPG / Cypher - переменные разрешаются
  - **`Alternate path`** - предикат1 | предикат2
  - **`Sequence`** - path длиной >1
  - **`Inverse path`** - инверсные пути

  - **`Negated path`** - указание на отсутствие конкретного пути

```
SELECT ?country WHERE {
   ?s !(:visitedCity/:country) ?country . }
```

SPARQL 1.1 - Federated Querying



- Федеративные запросы (federated queries) - позволяют в рамках одного запроса опрашивать другие хранилища, поддерживающие SPARQL с помощью ключевого слова **SERVICE**

```
SELECT ?film ?label ?subject WHERE {
    SERVICE <http://data.linkedmdb.org/sparql> {
        ?movie rdf:type movie:film .
        ?movie rdfs:label ?label .
        ?movie owl:sameAs ?dbpediaLink
        FILTER (regex(str(?dbpediaLink), “dbpedia”))
     }
    SERVICE <http://dbpedia.org/sparql> {
        ?dbpediaLink dct:subject ?subject .
     }
}
```

Advanced SPARQL - Algebra

```
SELECT ?s WHERE {
    ?s dbo:director dbr:Stephen_Spielberg }
```

```
SELECT ?s WHERE {
    ?s ?p ?o FILTER (?p=dbo:director &&
?o=dbr:Stephen_Spielberg) }
```

Advanced SPARQL - Algebra

```
SELECT ?s WHERE {
    ?s dbo:director dbr:Stephen_Spielberg }
```



```
(base <http://example/base/>
  (bgp (triple ?s dbo:director
dbr:Stephen_Spielberg )))
```

FAST

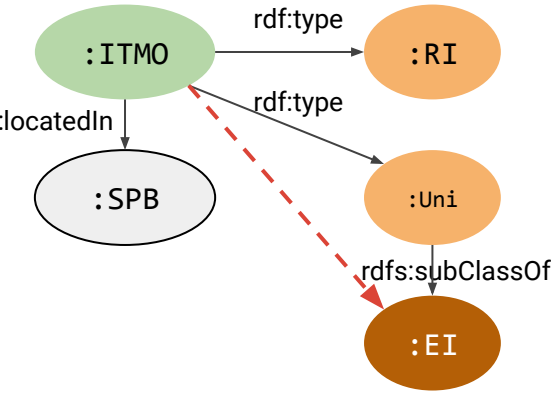
```
SELECT ?s WHERE {
    ?s ?p ?o FILTER (?p=dbo:director &&
?o=dbr:Stephen_Spielberg) }
```



```
(base <http://example/base/)
    (filter (&& (= ?p dbo:director) (= ?o
dbr:Stephen_Spielberg))
        (bgp (triple ?s ?p ?o))))))
```

SLOW

Advanced SPARQL - Reasoning




```
:ITMO_University   rdf:type            :University .
:ITMO_University   rdf:type            :Research_Institution .
:ITMO_University   :locatedIn          :Saint_Petersburg .
:University        rdfs:subClassOf     :Educational_Institution .
(:ITMO_University  rdf:type            :Educational_Institution .)
```

```
SELECT ?t WHERE {
  :ITMO_University a ?t }
```

- Стандартный SPARQL не поддерживает ризонинг - все триплеты должны быть материализованы
- Некоторые СУБД позволяют производить ризонинг (логически выводят новые триплеты в памяти)
  - RDFS (subClassOf, range, domain)
  - OWL 2 RL / QL
  - SWRL
  - owl:sameAs

Люди, которые умеют писать SPARQL-запросы

Люди, которые умеют писать запросы на SQL

Люди, которые могут вбить в поле ввода термин для фильтрации

HOW TO STORE RDF DATA ?

https://commons.wikimedia.org/wiki/File:A_view_of_the_server_room_at_The_National_Archives.jpg

RDF & Graph Databases

[Faye et al. A survey of RDF storage approaches, 2012]

- Графы нужно где-то хранить
- Native (NoSQL) vs Non-native (SQL)
- RDF СУБД vs LPG СУБД

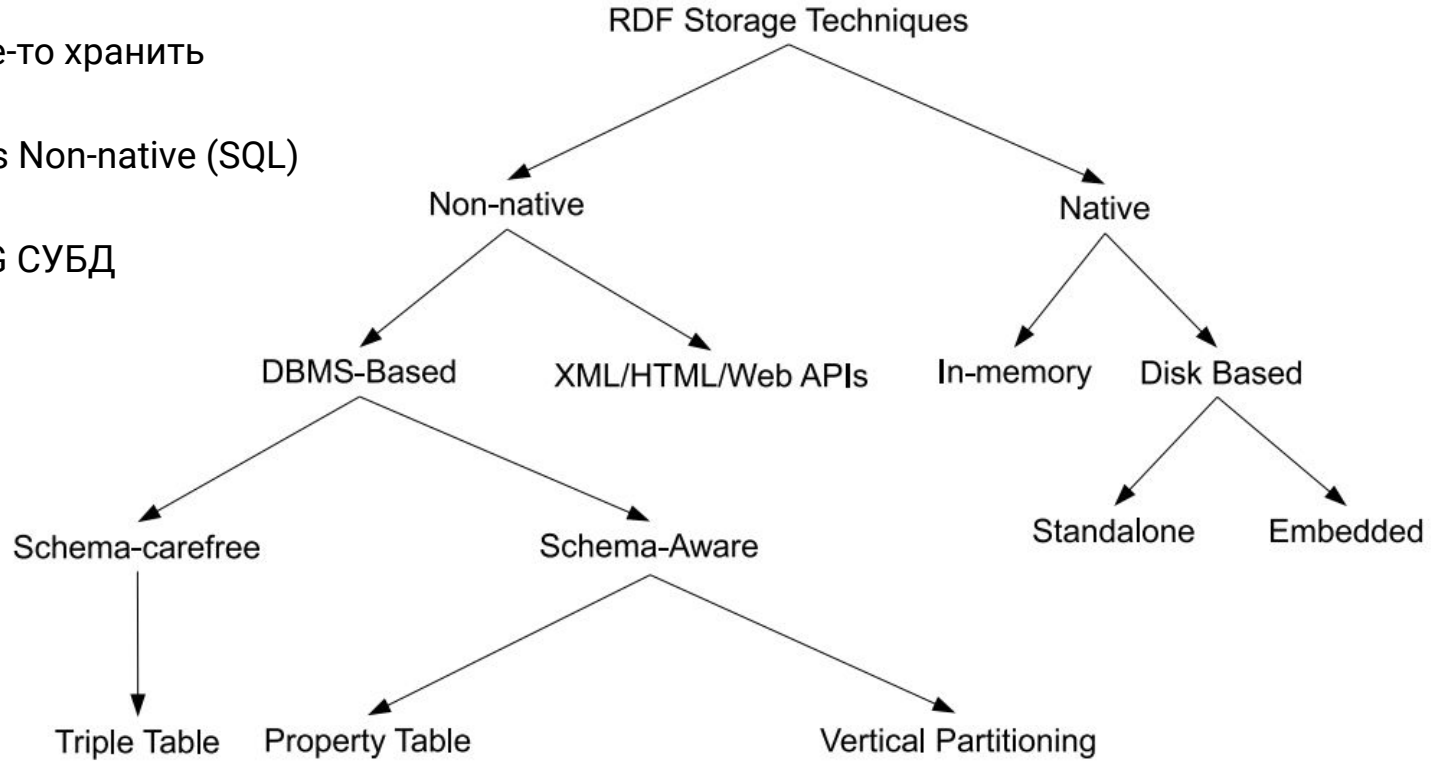
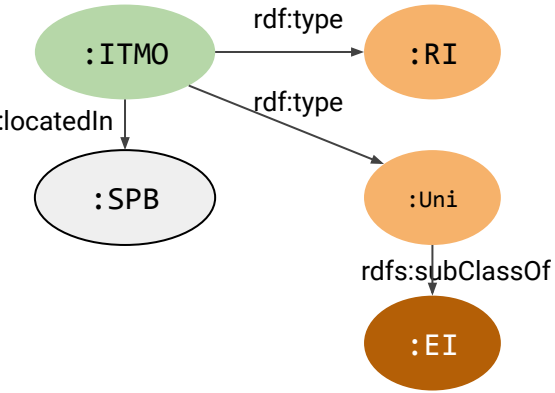


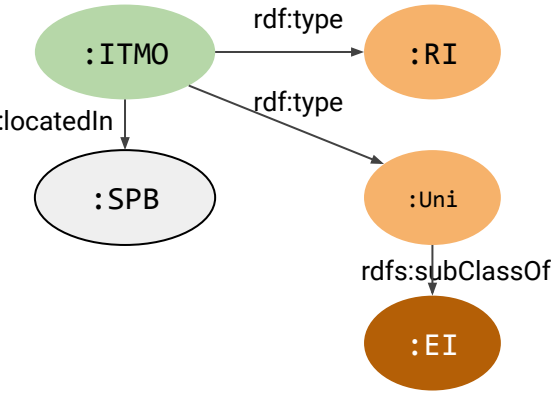
Figure 2. *A classification of RDF data storage approaches*

RDF Databases - RDBMS



- Триплеты хранятся в реляционной СУБД в таблице s - p - o
- Преимущества:
  - Можно использовать все достижения RDBMS за 40 лет
- Недостатки:
  - Необходимо транслировать SPARQL в SQL и обратно
  - Сложная графовая аналитика
  - Эффективность зависит от способа индексирования
- Организация в виде:
  - Triple Table
  - Property Table

RDF Databases - RDBMS - Triple Table



- Одна таблица для всех триплетов
- **Относительно просто реализовать**
- **Большое количество self-joins при обработке запросов**
- Oracle, 3store, Redland, RDFStore, rdfDB

```
SELECT ?city WHERE {
  ?s rdf:type :University ;
     :locatedIn ?city. }
```

```
SELECT t2.o AS city
FROM triples AS t1, triples AS t2
WHERE      t1.p = `type`          AND
           t1.o = `University`    AND
           t2.p = `locatedIn`     AND
           t1.s = t2.s
```

s	p	o
:ITMO_University	rdf:type	:University
:ITMO_University	rdf:type	:Research_Institution
:ITMO_University	:locatedIn	:Saint_Petersburg
:University	rdfs:subClassOf	:Educational_Institution

RDF Databases - RDBMS - ID-based Triple Table

Entity	ID
:ITMO_University	1
rdf:type	2
:University	3
:Research_Institution	4
:locatedIn	5
:Saint_Petersburg	6
rdfs:subClassOf	7
:Educational_Institution	8

- Еще одна таблица для хранения ID всех сущностей и литералов

- **Относительно просто реализовать**
- **Экономия памяти и увеличение производительности**

- **Большое количество self-joins при обработке запросов**

s	p	o
:ITMO_University	rdf:type	:University
:ITMO_University	rdf:type	:Research_Institution
:ITMO_University	:locatedIn	:Saint_Petersburg
:University	rdfs:subClassOf	:Educational_Institution

RDF Databases - RDBMS - ID-based Triple Table

Entity	ID
:ITMO_University	1
rdf:type	2
:University	3
:Research_Institution	4
:locatedIn	5
:Saint_Petersburg	6
rdfs:subClassOf	7
:Educational_Institution	8

- Еще одна таблица для хранения ID всех сущностей и литералов

- **Относительно просто реализовать**
- **Экономия памяти и увеличение производительности**

- **Большое количество self-joins при обработке запросов**

s	p	o
1	2	3
1	2	4
1	5	6
3	7	8

RDF Databases - RDBMS - Property Table

Entity	ID
:ITMO_University	ID1
rdf:type	ID2
:University	ID3
:Research_Institution	ID4
:locatedIn	ID5
:Saint_Petersburg	ID6
rdfs:subClassOf	ID7
:Educational_Institution	ID8

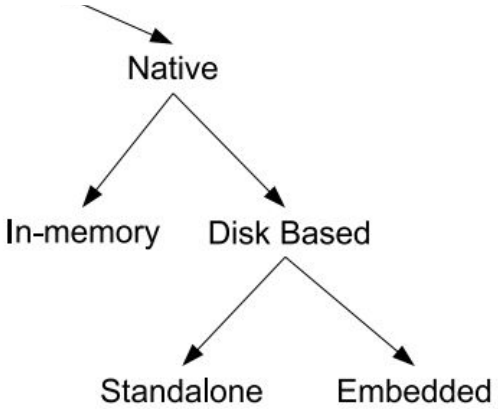
- Комбинирование предикатов и похожих субъектов в одной таблице
- **Меньше self-joins**
- **Если граф идеально структурирован - реляционная таблица**
- **Много NULL**
- **Нетривиальная кластеризация**
- **Сложная поддержка предикатов с несколькими значениями**

ID	type	:locatedIn	rdfs:subClassOf
ID1	ID3	ID6	NULL
ID1	ID4	ID6	NULL

ID	type	:locatedIn	rdfs:subClassOf
ID3	NULL	NULL	ID8
ID4	NULL	NULL	NULL

RDF4J
Jena SDB
RDFSuite
4store

RDF Databases - Native

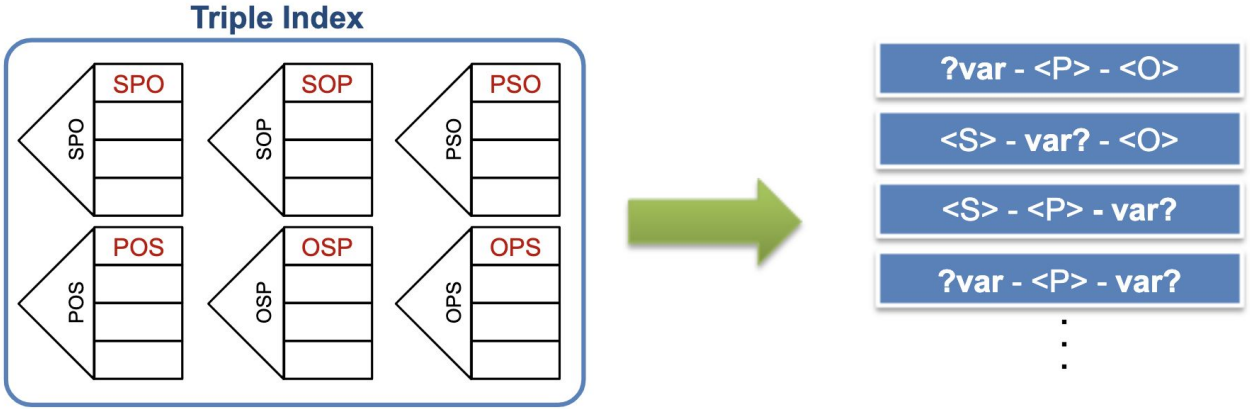


Jena TDB
AllegroGraph
GraphDB
Stardog
RDF3X
Virtuoso

- Нативные RDF хранилища работают напрямую с графовым представлением данных
- **In-memory** - данные целиком расположены в оперативной памяти
  - **Высокая скорость работы**
  - **Надежность**
- **Disk-based** - хранят данные на жестких дисках
  - **Standalone** - не зависят от конкретного приложения и могут быть использованы сторонними сервисами
  - **Embedded** - программно привязаны к конкретному приложению и не могу быть использованы для других сервисов
- B+ Tree - based
- LSM Tree - based

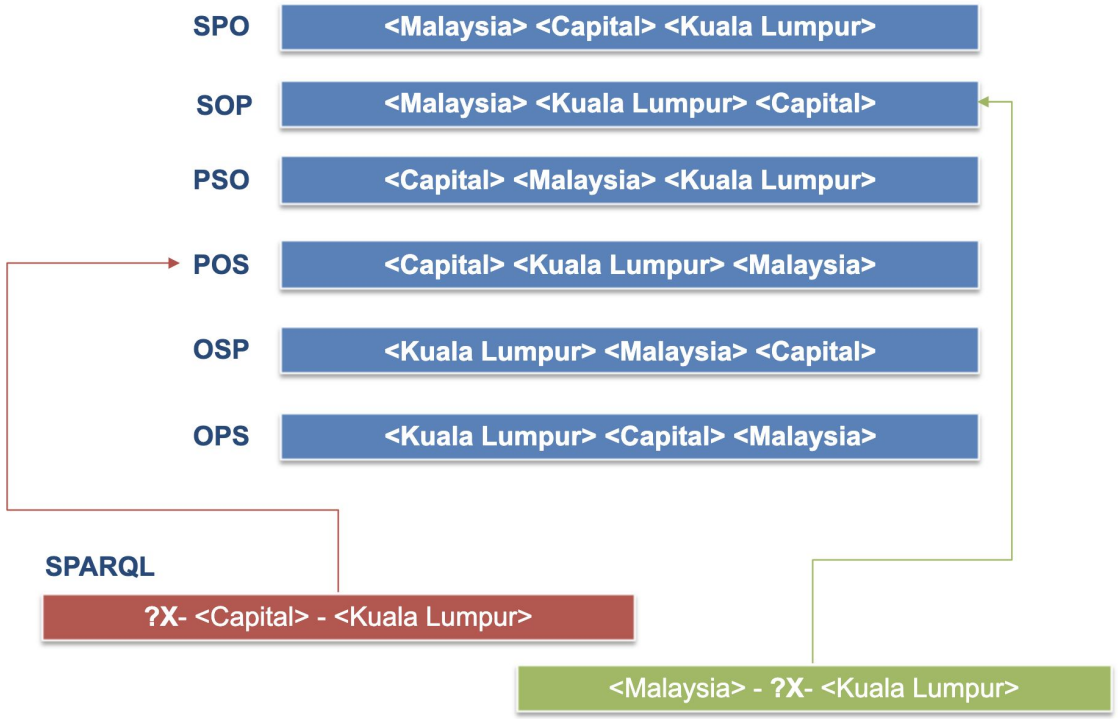
RDF Databases - Native - B+ Trees - RDF-3X

- Six separate indexes
  - (SPO, SOP, OSP, OPS, PSO, POS)
  - Stored in the leaf pages of the clustered B+ tree



[Neumann et al. RDF-3X: a RISC-style engine for RDF, 2008]

RDF Databases - Native - B+ Trees - RDF-3X



[Neumann et al. RDF-3X: a RISC-style engine for RDF, 2008]

RDF Databases - Native - B+ Trees - RDF-3X

- Store collation order
  - Neighboring indexes are very similar
  - Stores the change between triples

S	P	O
0	1	2
0	1	3
0	1	5

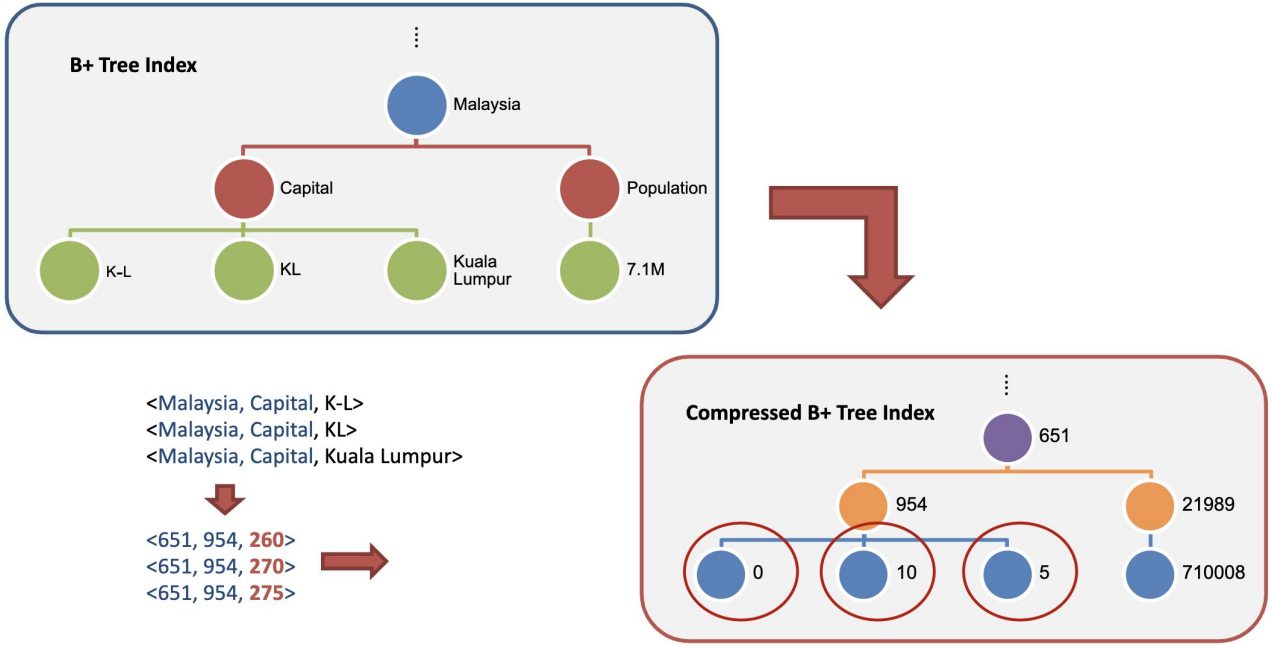
Compression using LZ77

S	P	O
0	1	(2-2) = 0
		(3-2) = 1
		(5-3) = 2

[Neumann et al. RDF-3X: a RISC-style engine for RDF, 2008]

RDF Databases - Native - B+ Trees - RDF-3X

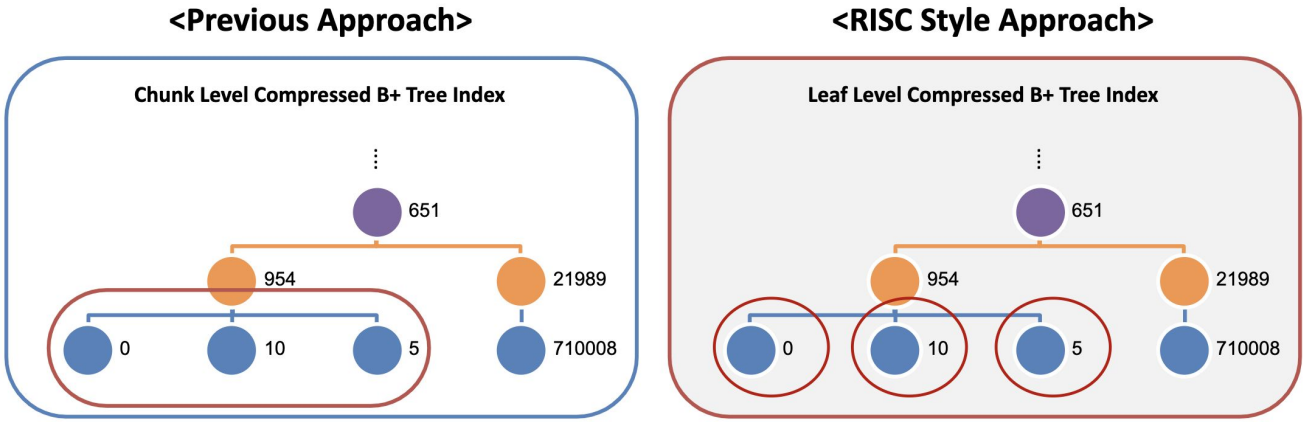
- Compression
  - Stores only the change (δ) between triples



[Neumann et al. RDF-3X: a RISC-style engine for RDF, 2008]

RDF Databases - Native - B+ Trees - RDF-3X

- Leaf level Compression
  - Directly read triple (less decompression cost)
  - Easy update
  - Better concurrency control and recovery



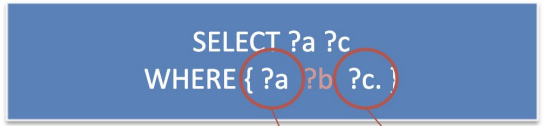
[Neumann et al. RDF-3X: a RISC-style engine for RDF, 2008]

RDF Databases - Native - B+ Trees - RDF-3X

- For many SPARQL patterns
  - Indexing partial triples rather than full triples would be sufficient
  - SELECT ?a ?b WHERE { ?a **?b** ?c }

- Two-value indexes
  - Two of three columns of a triple
  - (value1, value2, count)
  - (SP, PS, SO, OS, PO, OP)

- One-value indexes
  - One of three columns of a triple
  - (value, count)
  - (S, P, O)



Aggregated Indexes

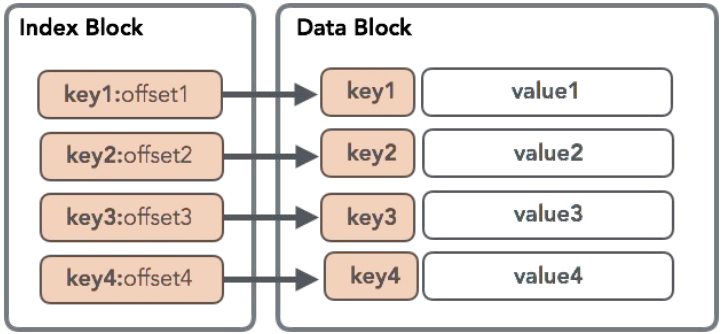
Value 1	Value 2	Value3
Malaysia	Capital	Kuala Lumpur
IDB	PartOf	SNU
Malaysia	Religion	Islamic

Aggregated Indexes

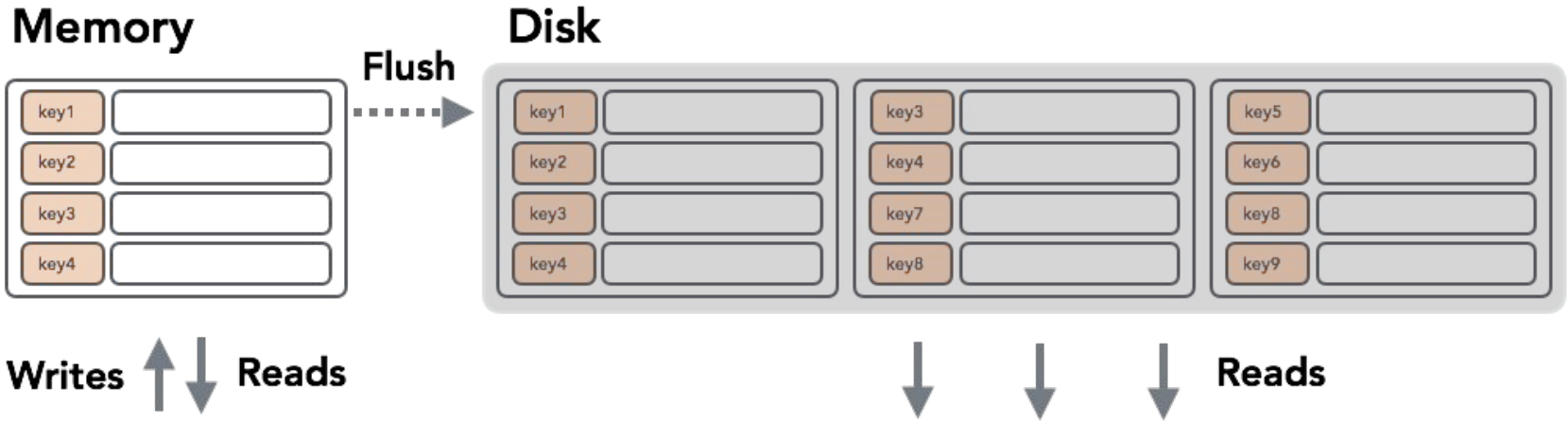
Value 1	Value 2	Count
Malaysia	Kuala Lumpur	4
IDB	SNU	2
Malaysia	Islam	5

[Neumann et al. RDF-3X: a RISC-style engine for RDF, 2008]

RDF Databases - Native - LSM Trees

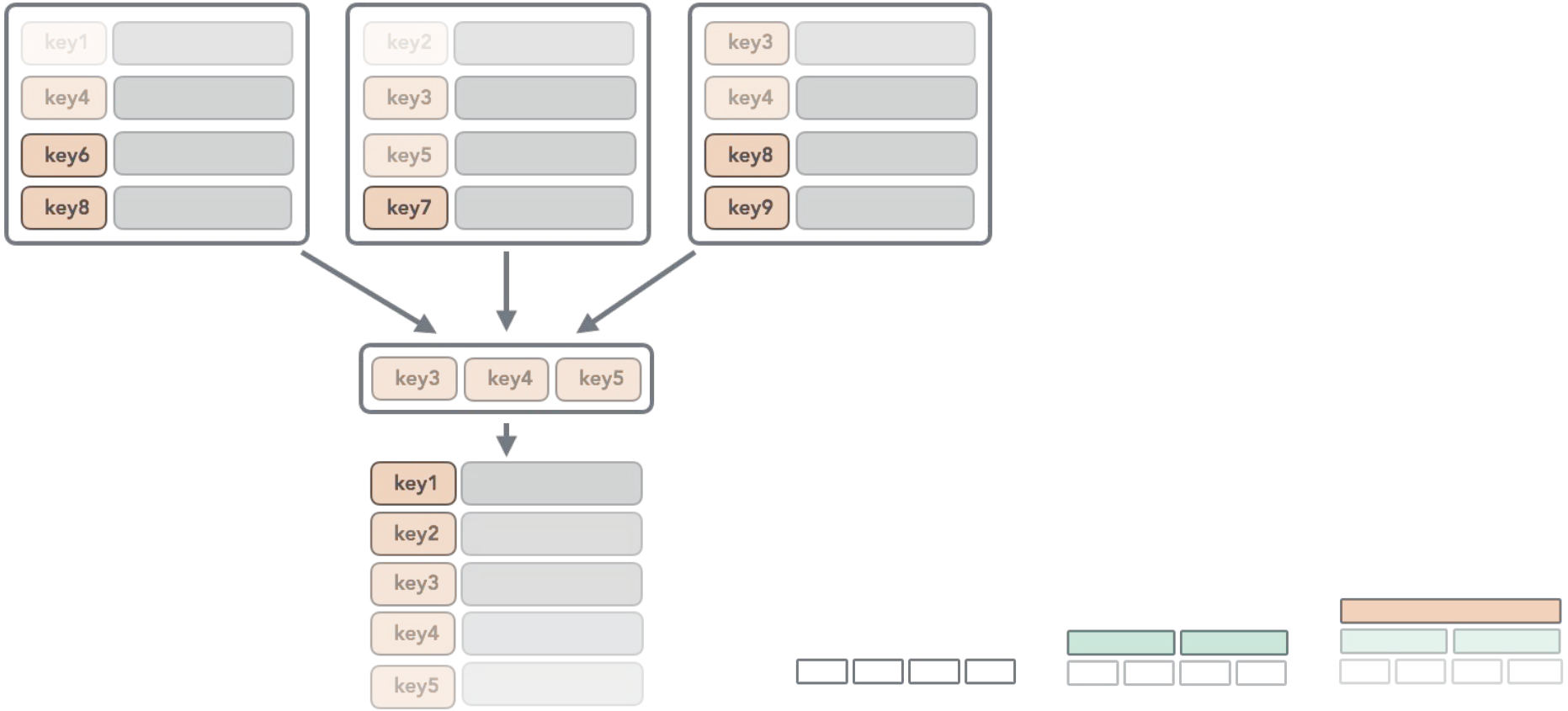


- B+ медленно обрабатывают записи и удаления
- **Log-structured merge (LSM) trees** оптимизированы под записи и удаления
- В LSM простые Data Blocks могут быть B+ деревьями
- Data Blocks - отсортированные списки
- Реализации: RocksDB, Hbase, Stardog



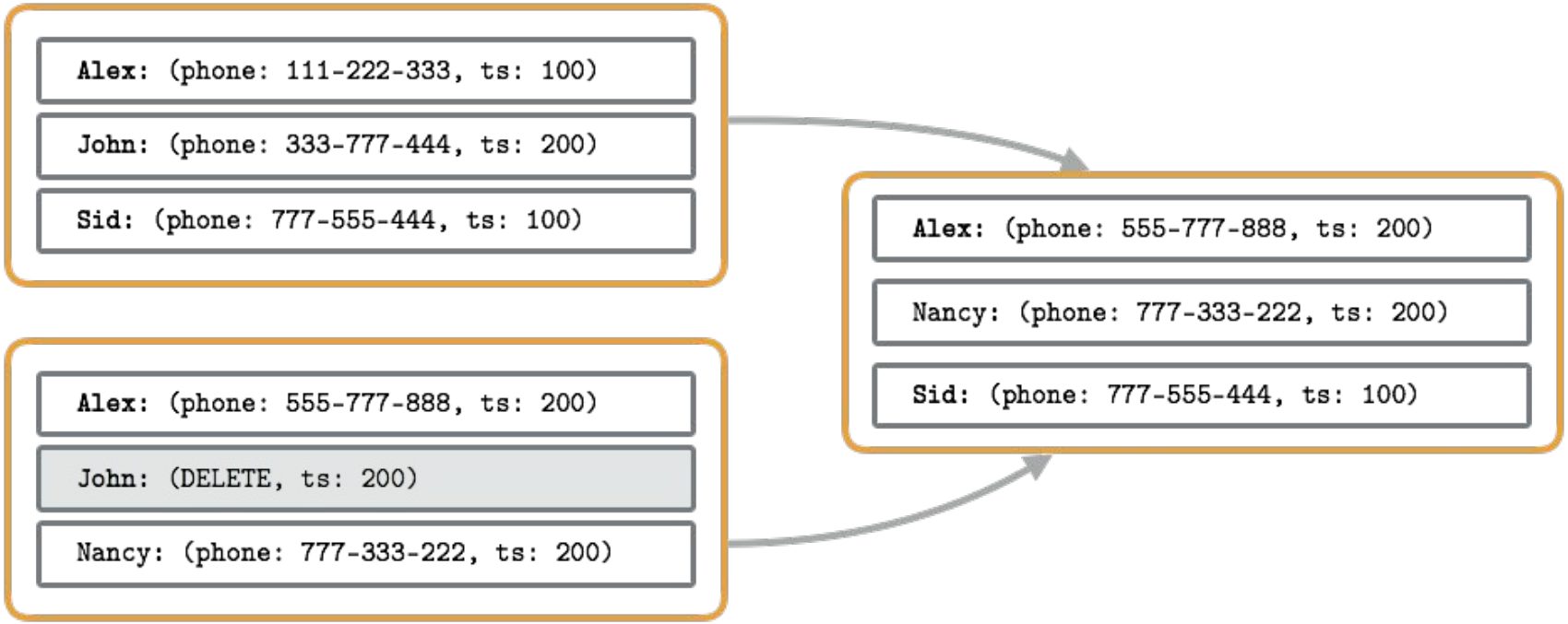
https://medium.com/databasss/on-disk-io-part-3-lsm-trees-8b2da218496f

RDF Databases - Native - LSM Trees - Compaction



https://medium.com/databasss/on-disk-io-part-3-lsm-trees-8b2da218496f

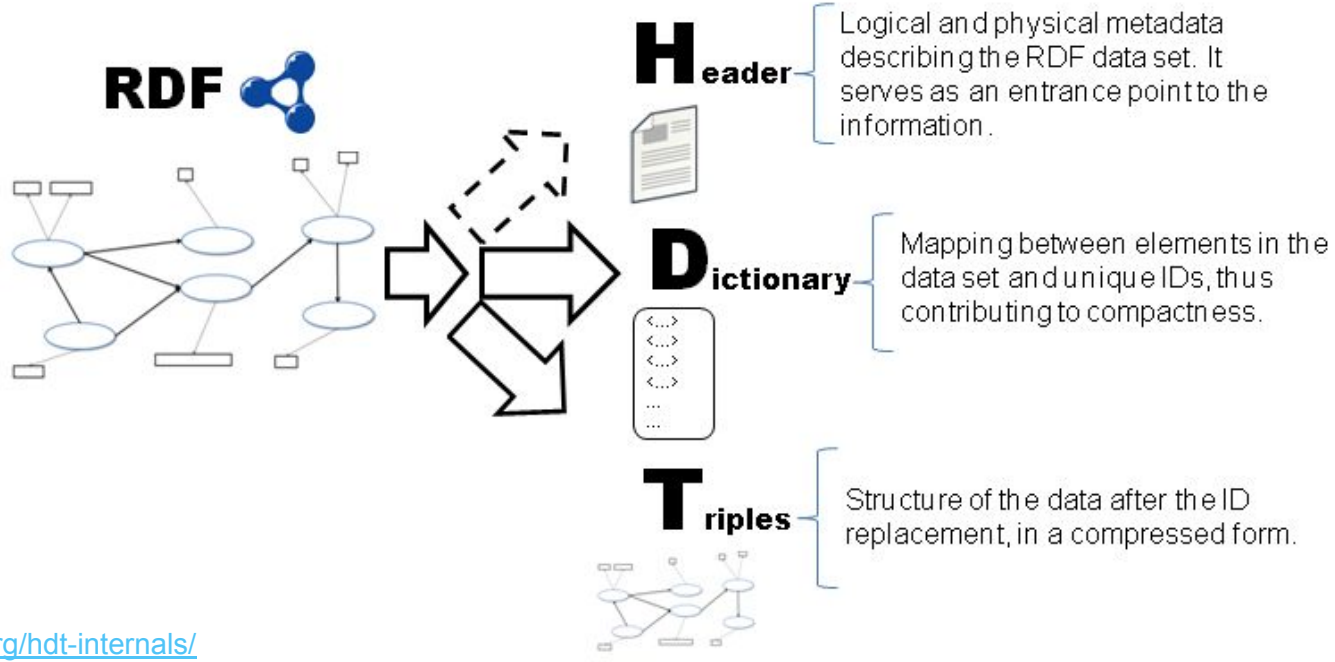
RDF Databases - Native - LSM Trees - Summing Up



https://medium.com/databasss/on-disk-io-part-3-lsm-trees-8b2da218496f

RDF Databases - HDT

- HDT - один из самых эффективных форматов по сжатию RDF графов
- Header - метаданные о графе
- Dictionary - словарь, назначающий уникальные идентификаторы всем RDF terms
- Triples - сжатое представление графа

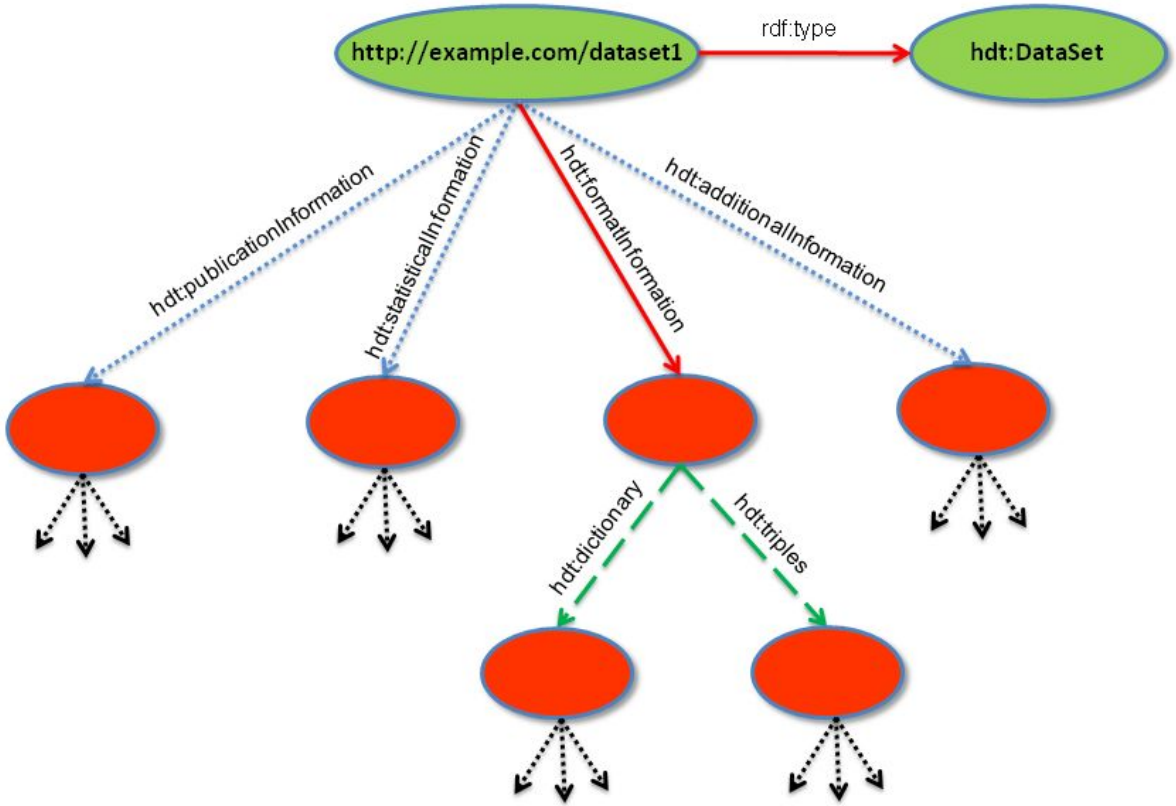


http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Header

Метаданные:

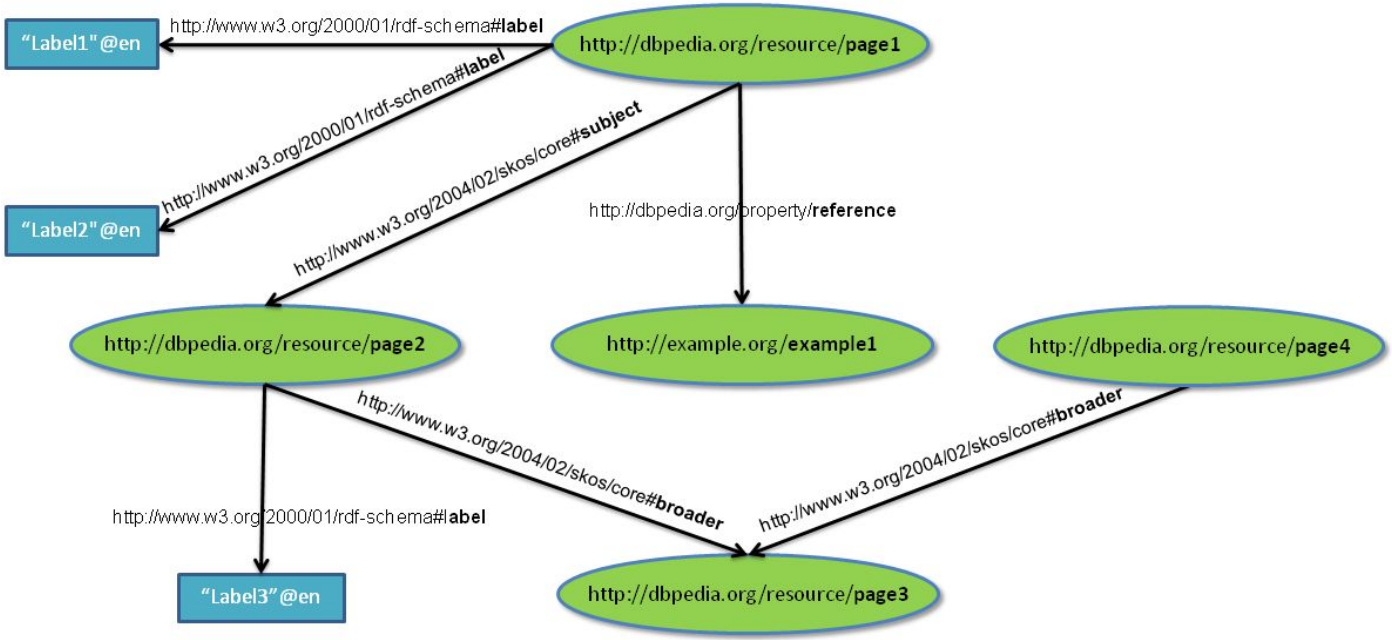
- Исходный размер
- Количество триплетов
- Количество уникальных предикатов
- Информация об авторе и публикации
- Ссылки на объекты dictionary и triples



http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Dictionary

	ID	RDF Term
S-O	1	<../page2>
S	2	<../page1>
	3	<../page4>
O	2	<../page3>
	3	<../example1>
	4	“Label1”@en
	5	“Label2”@en
	6	“Label3”@en
P	1	<../reference>
	2	<../#label>
	3	<../#broader>
	4	<../#subject>

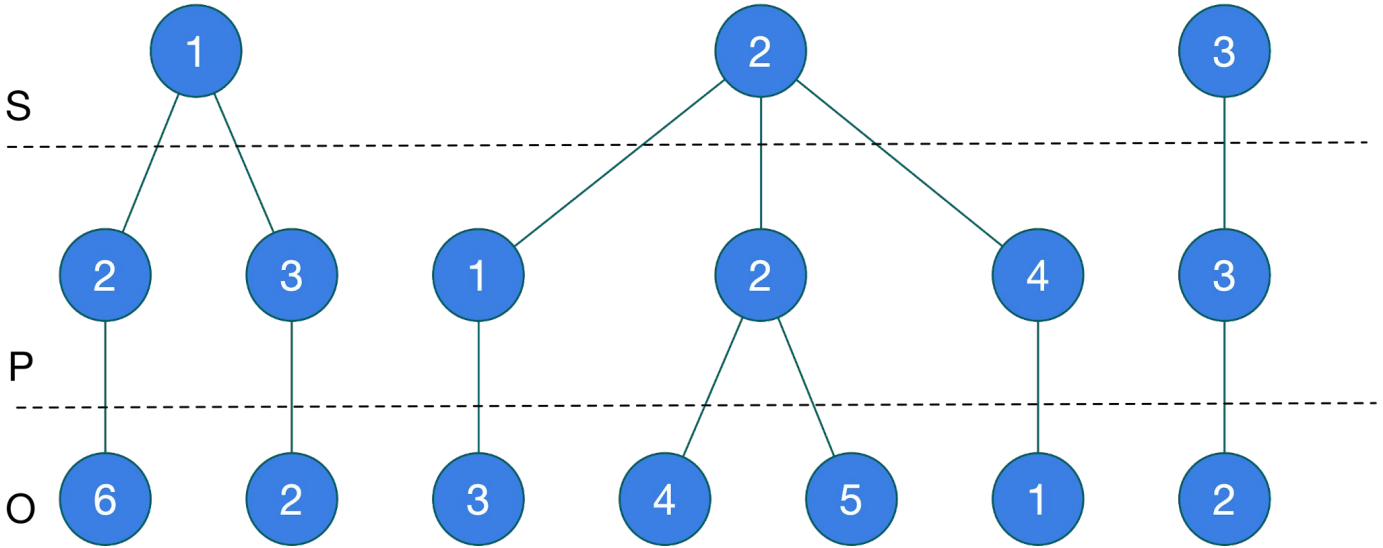


Лексикографическая сортировка без дубликатов

http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Triples

1 2 6
1 3 2
2 1 3
2 2 4
2 2 5
2 4 1
3 3 2



http://www.rdfhdt.org/hdt-internals/

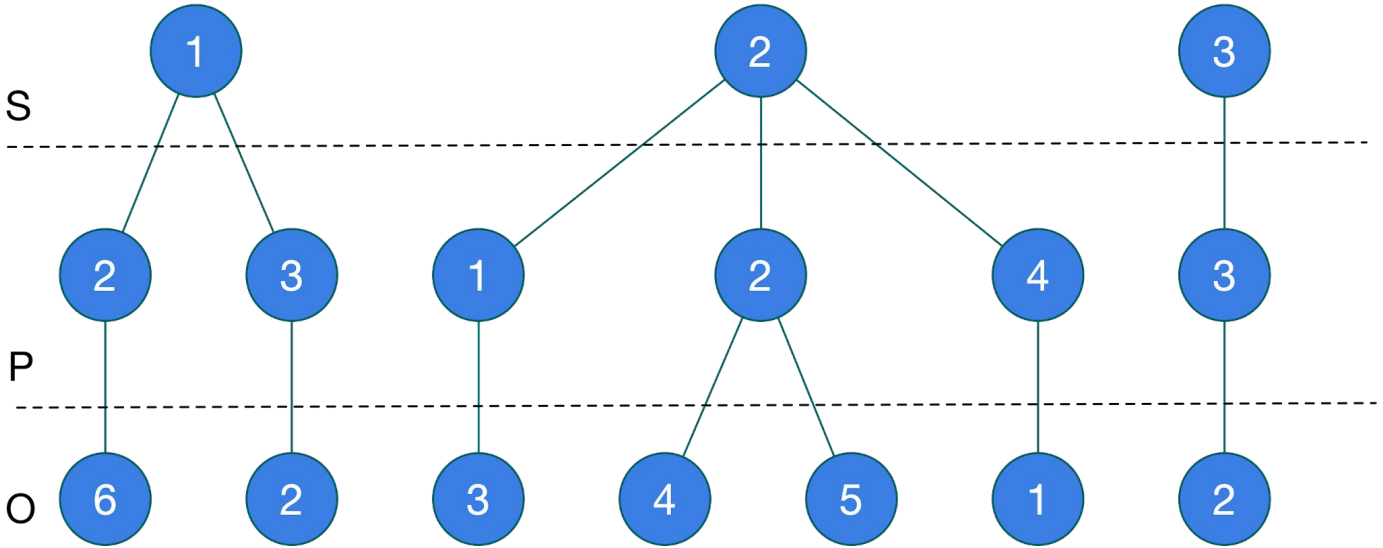
RDF Databases - HDT - Triples - Adjacency Lists

Array	2	3	1	2	4	3
Bitmap	1	0	1	0	0	1

http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Triples Model

1 2 6
1 3 2
2 1 3
2 2 4
2 2 5
2 4 1
3 3 2



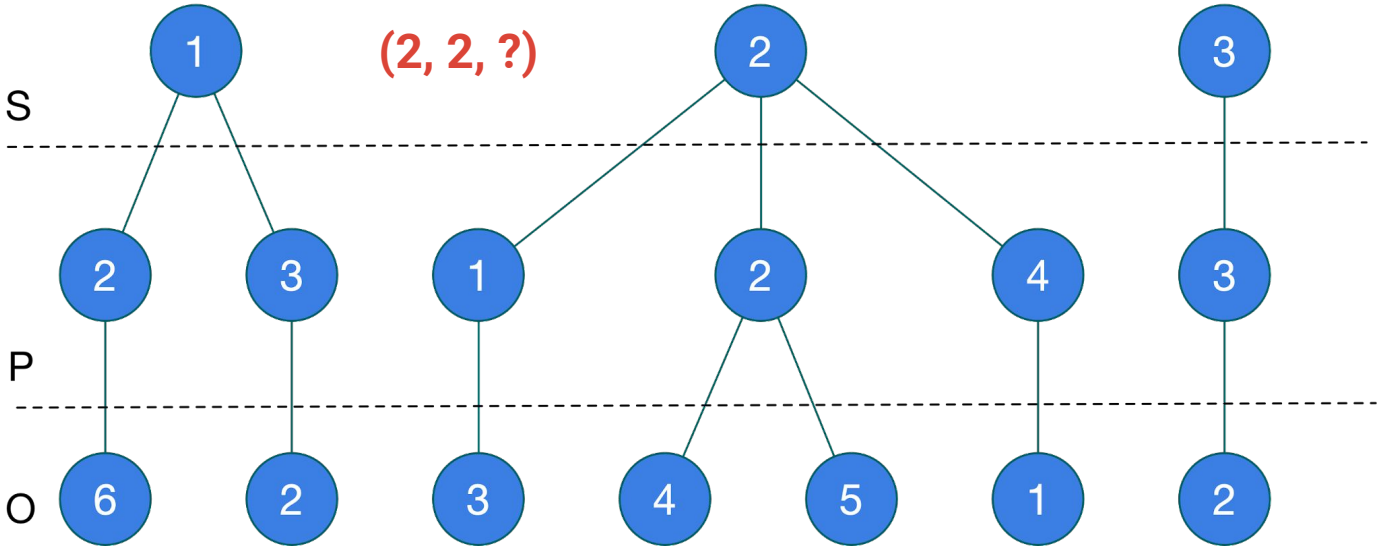
Array P	2	3	1	2	4	3
Bitmap P	1	0	1	0	0	1

Array O	6	2	3	4	5	1	3
Bitmap O	1	1	1	1	0	1	1

http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Search by Subject

1 2 6
1 3 2
2 1 3
2 2 4
2 2 5
2 4 1
3 3 2



(S, ?, ?)
(S, P, ?)
(S, ?, O)
(S, P, O)

Array P	2	3	1	2	4	3
Bitmap P	1	0	1	0	0	1

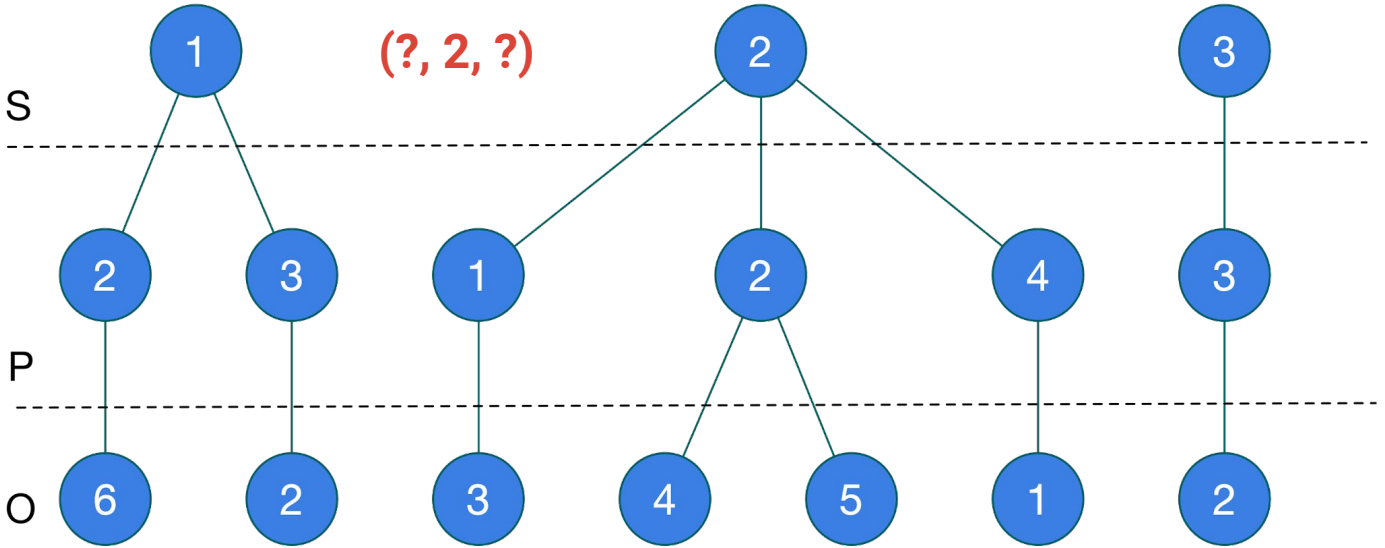
Array O	6	2	3	4	5	1	3
Bitmap O	1	1	1	1	0	1	1

http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Search by Predicate

1 2 6
1 3 2
2 1 3
2 2 4
2 2 5
2 4 1
3 3 2

(?, P, ?)

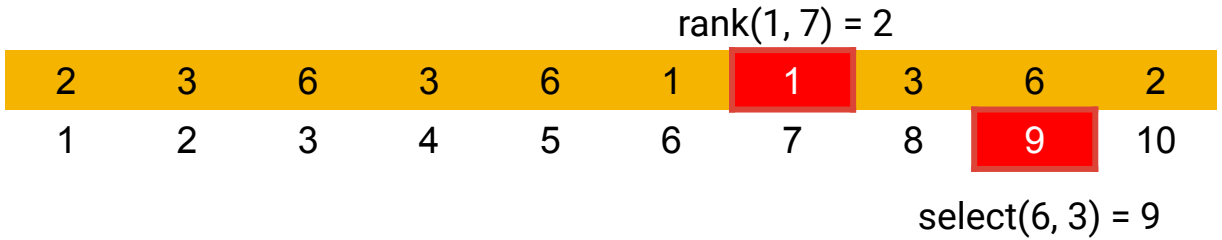


Array P	2	3	1	2	4	3
Bitmap P	1	0	1	0	0	1

Array O	6	2	3	4	5	1	3
Bitmap O	1	1	1	1	0	1	1

http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Wavelet Tree



Compact sequence of integers {0, σ}

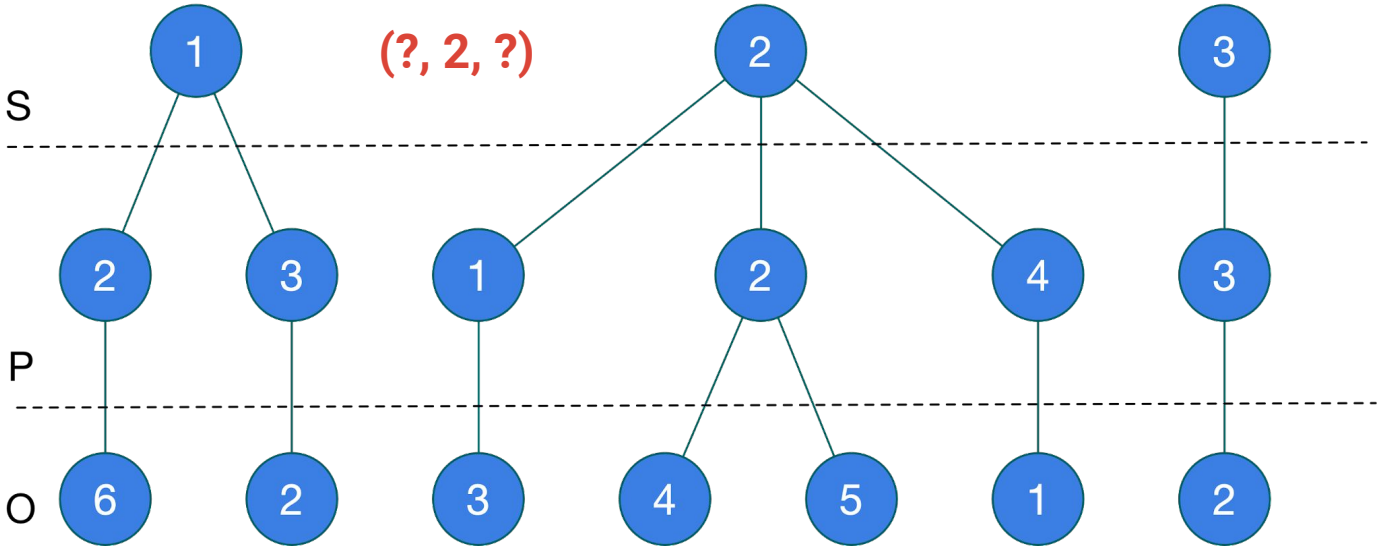
- `access(position)` = value at “position”, **O(log σ)**
- `rank(entry, position)` = Number of appearances of “entry” up to “position”, **O(log σ)**
- `select(entry, i)` = Position where “entry” appears for i-th time, **O(log σ)**

http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Search by Predicate

1 2 6
1 3 2
2 1 3
2 2 4
2 2 5
2 4 1
3 3 2

(?, P, ?)

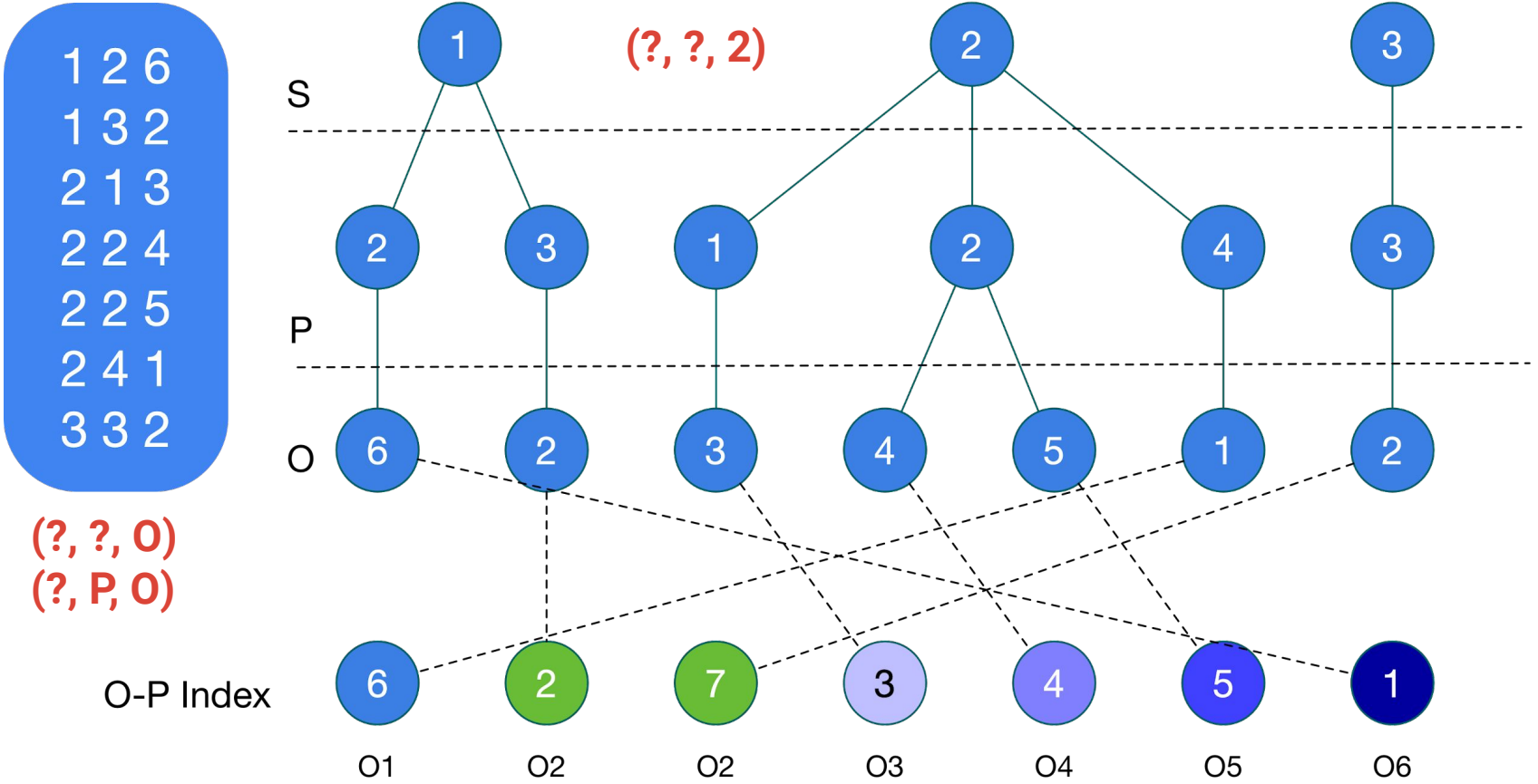


Wavelet P	2	3	1	2	4	3
Bitmap P	1	0	1	0	0	1

Array O	6	2	3	4	5	1	3
Bitmap O	1	1	1	1	0	1	1

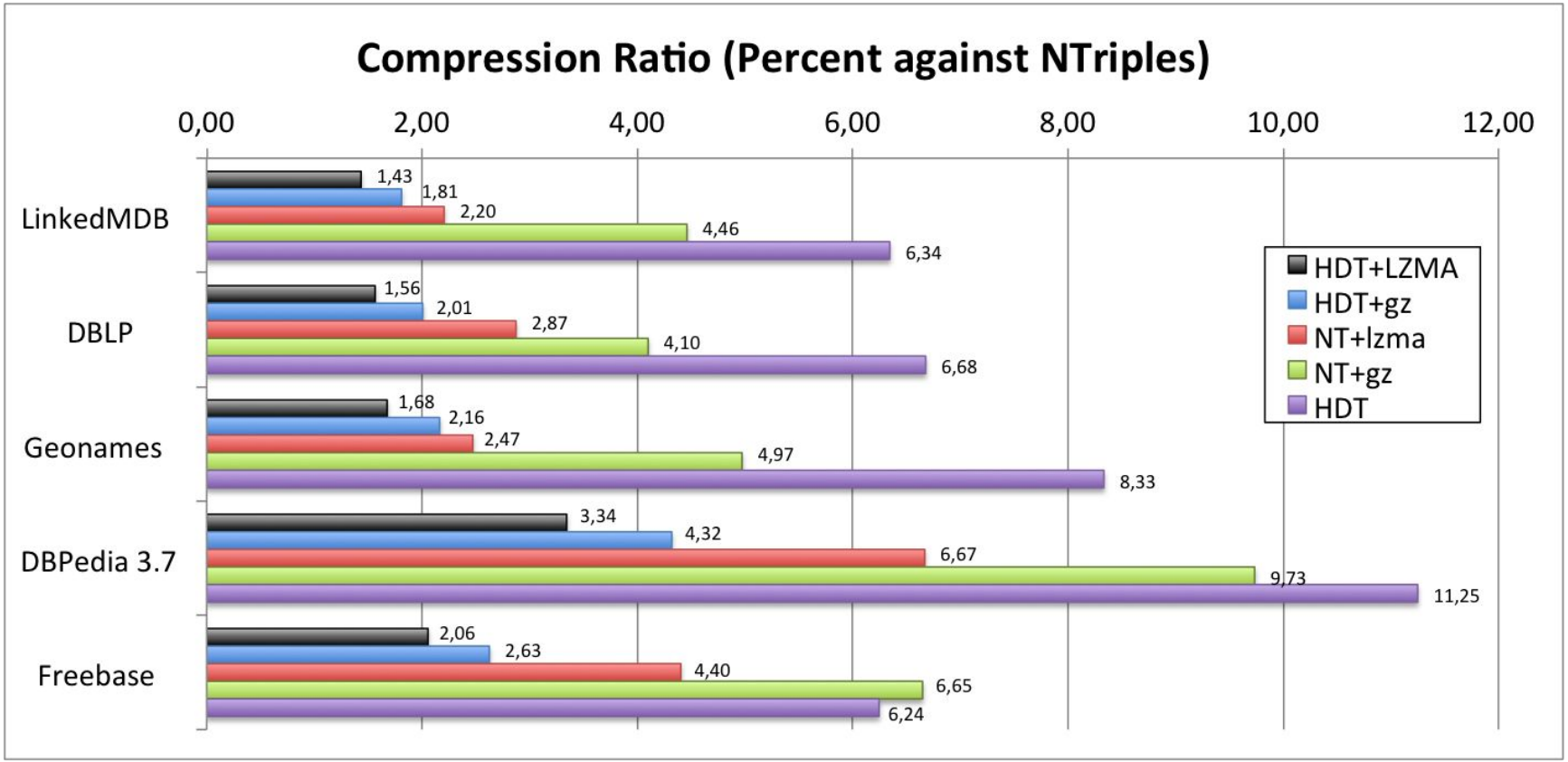
http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Search by Object



http://www.rdfhdt.org/hdt-internals/

RDF Databases - HDT - Compression



RDF Databases - HDT - Indexing

	RDF-3x	HDT-FoQ
LinkedMDB	1 min 51 sec	**1.91 sec**
DBLP	23 min 7 sec	**16.7 sec**
Geonames	44 min 51 sec	**44.9 sec**
DBPedia 3.7	2 hour 11 min	**2 min 9 sec**
Freebase	7 hour 5 min	**4 min 46 sec**

Table 1. Client-side indexing Time.

	LZMA+RDF3x	HDT+LZMA
LinkedMDB	4.1 min	**9.21 sec**
DBLP	27 min	**2.02 min**
Geonames	49.2 min	**3.04 min**
DBPedia 3.7	3 hour 9 min	**17.3 min**
Freebase	8 hour 43 min	**23.4 min**

Table 2. Total time to start querying a dataset.

RDF Databases - HDT - DB Size

Dataset	Triples	Index Size(Mb)			
		Virtuoso	RDF3x	HDT	HDT-FoQ
LinkedMDB	6.1M	518	337	49	**68**
DBLP	73M	3982	3252	695	**850**
Geonames	112M	9216	6678	1028	**1435**
DBPedia 3.7	296M	–	19416	5600	**6626**
Freebase	639M	–	34038	5420	**7713**

Table 3. Total index size.

RDF Databases - HDT - Compression

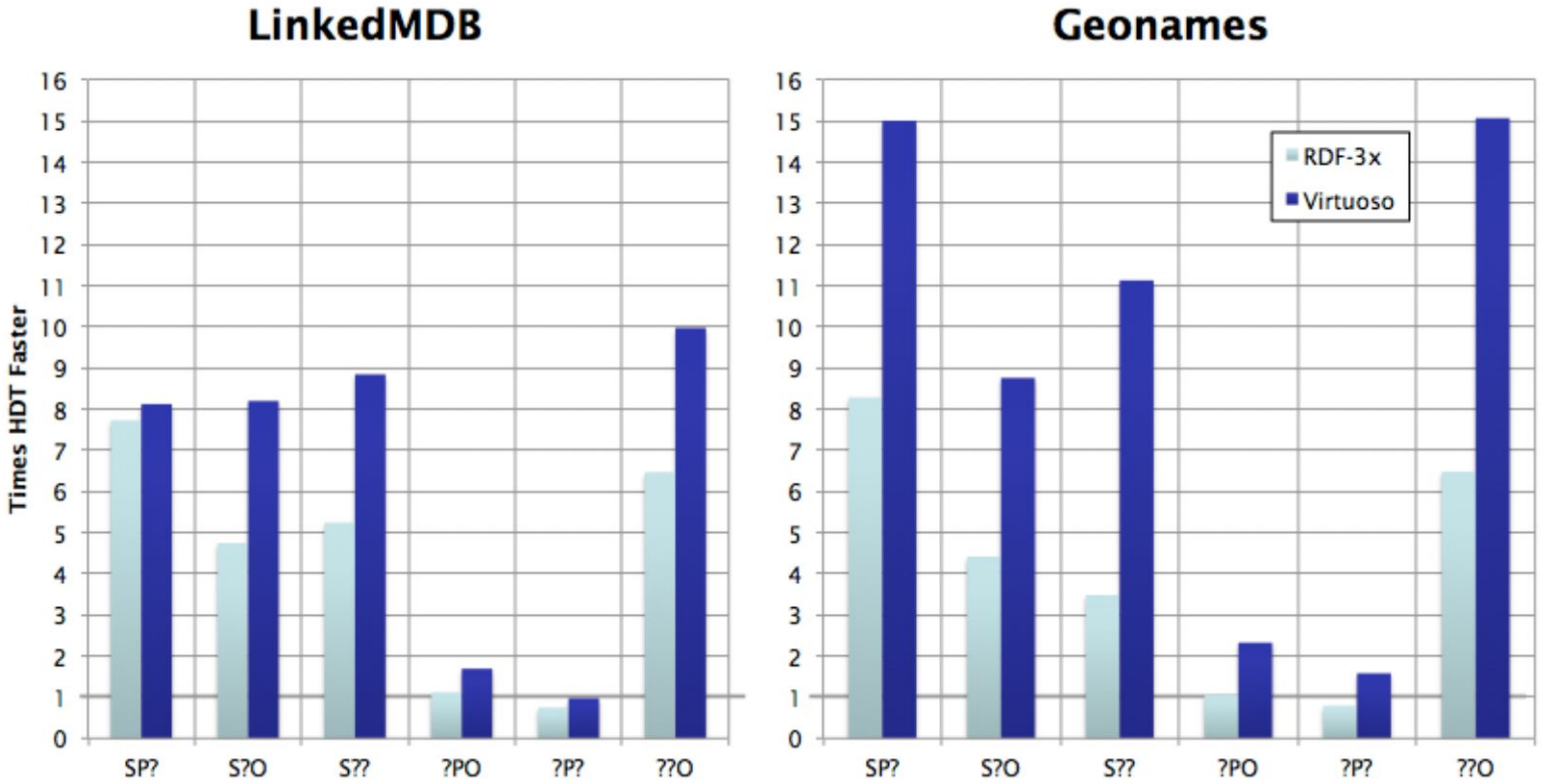


Figure 8: Time HDT is faster solving simple triple patterns, compared to state-of-the-art solutions. Higher means HDT is better.

RDF Databases - HDT + Linked Data Fragments

SPO, SP?, S??, S?O	Original HDT
?P?	Wavelet Tree
?PO, ??O	O-P Index

От HDT до HDT-FoQ (полная поддержка SPARQL)

- ❏ Преобразовать Array P в Wavelet tree
- ❏ Сгенерировать O-P Index

Linked Data Fragments поддерживает организацию SPARQL запросов к HDT - датасетам

- Выводит ответы кумулятивно по мере их обработки
- Перекладывает часть вычислительной нагрузки на SPARQL - клиент
- Поддерживает федеративность - опрос нескольких HDT-источников

http://linkeddatafragments.org/

RDF Databases - HDT + Linked Data Fragments

Linked Data Fragments поддерживает организацию SPARQL запросов к HDT - датасетам

#LD
Linked Data Fragments

http://linkeddatafragments.org/

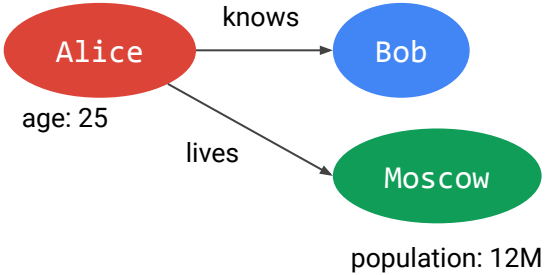
- Выводит ответы кумулятивно по мере их обработки
- Перекладывает часть вычислительной нагрузки на SPARQL - клиент
- Поддерживает федеративность - опрос нескольких HDT-источников

The axis of Linked Data Fragments types

The Linked Data Fragments vision allows us to visualize different HTTP interfaces for Linked Data *together*.

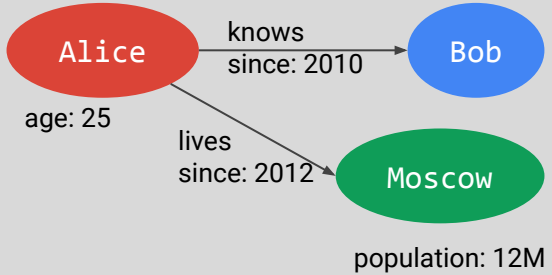
Графовые СУБД: RDF vs LPG

RDF



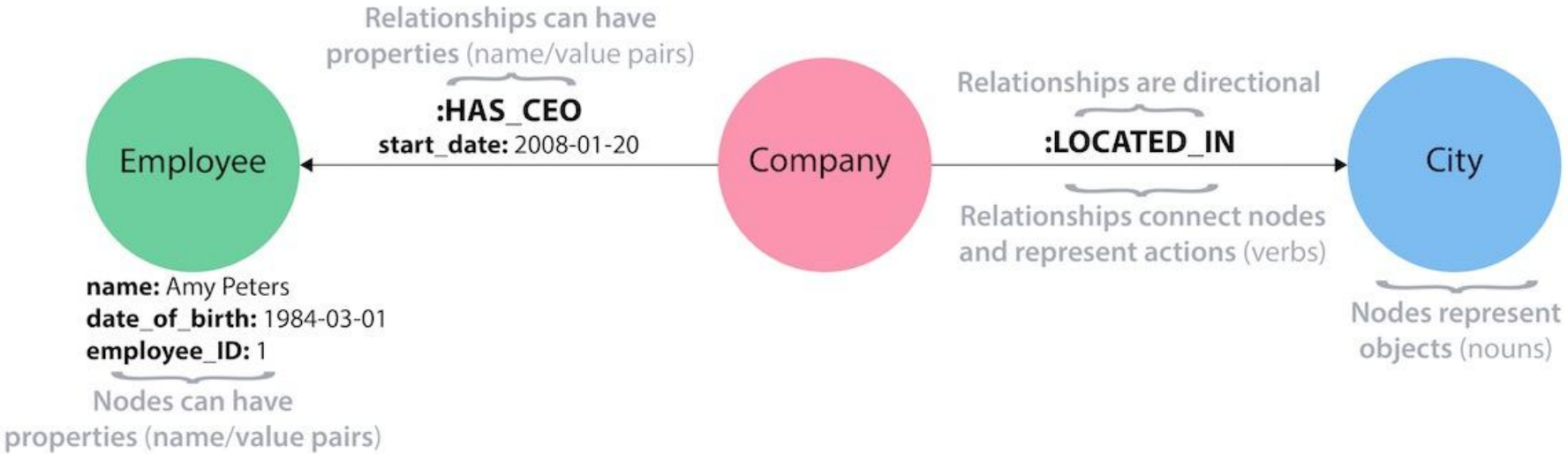
- Язык запросов: SPARQL
- Атрибуты предикатов ограничены RDFS/OWL
- Схема графа - семантическая
- Возможен логический вывод в процессе выполнения запроса

LPG (Labeled Property Graph)



- Язык запросов: Cypher, Gremlin
- Атрибуты предикатов не ограничены
- Схема графа - не семантическая
- Не способны к логическому выводу
- Разные виды графов: (не) направленные, взвешенные, гиперграфы

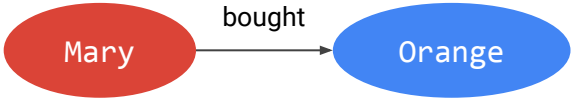
Labeled Property Graph (LPG) Databases



https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/

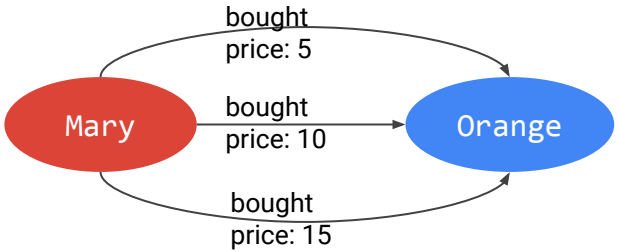
Графовые СУБД: RDF vs LPG

RDF



- Нет понятия уникальных предикатов, разрешенные атрибуты из RDFS, OWL
- Есть именованные графы
- Поддержка логического вывода
- **Reification**

LPG (Labeled Property Graph)

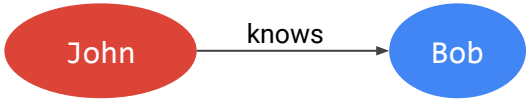


- Уникальные предикаты с собственными атрибутами
- Нет понятия именованных графов
- Нет логического вывода

LPG - Cypher

- Стандартный язык запросов СУБД Neo4j
- Почти все конструкты Cypher могут быть представлены в SPARQL

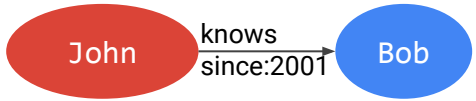
Cypher	SPARQL
MATCH (s:Person) WHERE s.name = “John” RETURN s;	SELECT ?s WHERE { ?s a :Person; :name “John” }
MATCH (s:Person)-[:knows]-(friend) WHERE s.name = “John” RETURN s, friend ;	SELECT ?s ?friend WHERE { ?s a :Person; :name “John” ; :knows ?friend }

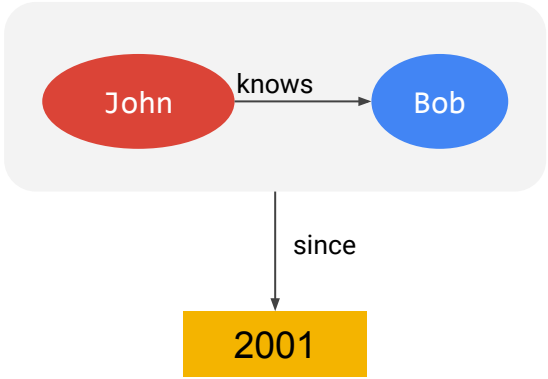


LPG - Cypher

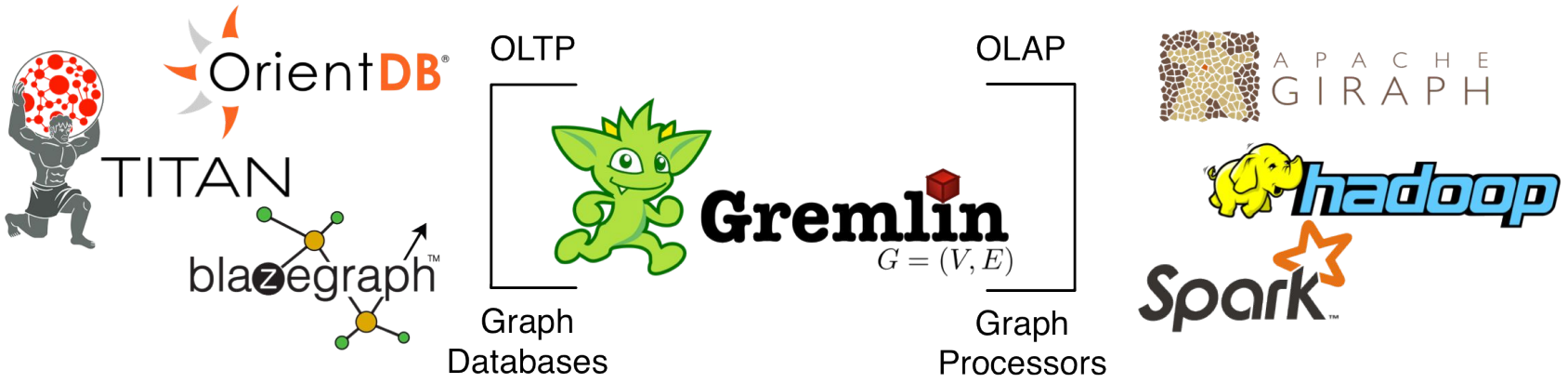
- Стандартный язык запросов СУБД Neo4j
- Почти все конструкты Cypher могут быть представлены в SPARQL

Cypher	SPARQL* (Reification)
`MATCH (s:Person)-[:knows {since:2001}] -> (js)` `RETURN s;`	`SELECT ?s WHERE {` `<<?s :knows :js>> :since 2001 }`





Gremlin



- Разработан в рамках проекта Apache Tinkerpop
- Язык обхода графов, полный по Тьюрингу
- Уровень абстракции между графовыми СУБД и аналитическими платформами
- Поддерживается в Neo4j, Amazon Neptune, Stardog, GraphDB, Spark, Apache Giraph

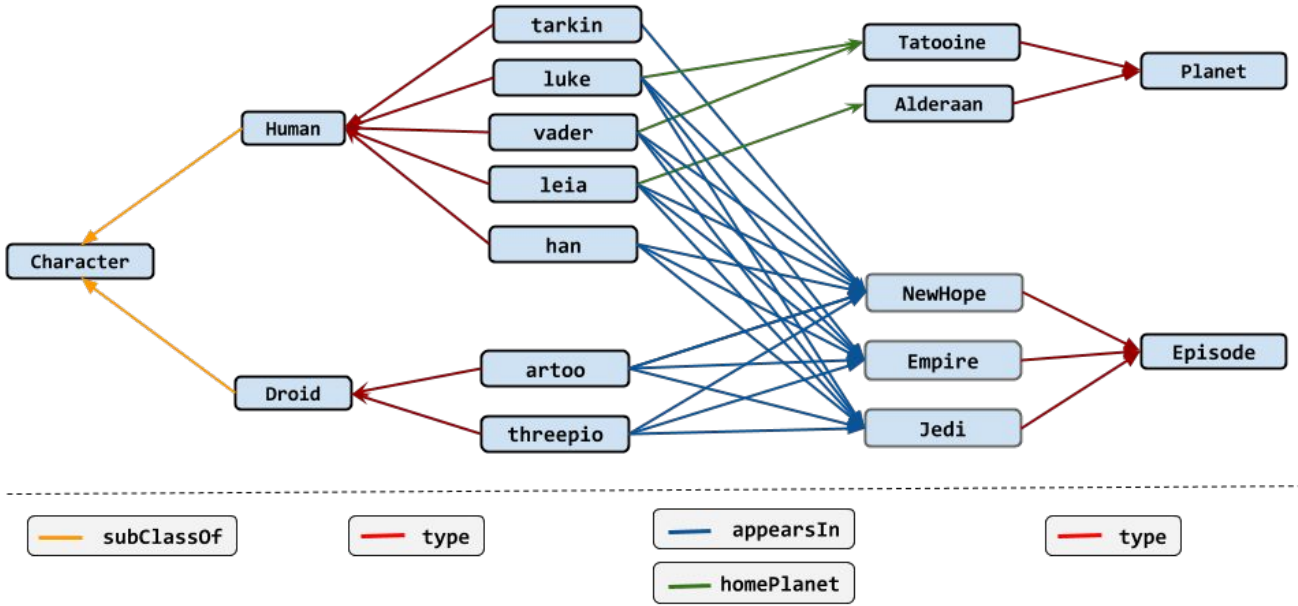
```
public class GremlinTinkerPopExample {
  public void run(String name, String property) {

    Graph graph = GraphFactory.open(...);
    GraphTraversalSource g = graph.traversal();

    double avg = g.V().has("name",name).
                   out("knows").out("created").
                   values(property).mean().next();

    System.out.println("Average rating: " + avg);
  }
}
```

http://tinkerpop.apache.org/gremlin.html

GraphQL

```
{
   Human {
      name
   }
}
```

- Разработан в Facebook больше как замена REST API
- Выразительность позволяет транслировать GraphQL в Cypher (Neo4j) или SPARQL (Stardog / GraphDB)




```
{ “data”: [
   {
      “name”: “tarkin”
   },
   {
      “name”: “luke”
   },
   {
      “name”: “vader”
   },
   {
      “name”: “leia”
   },
   {
      “name”: “han”
   }
]}
```

https://www.stardog.com/docs/#_graphql_queries

В следующей серии

1. Introduction
2. Представление знаний в графах - RDF & RDFS & OWL
3. Хранение знаний в графах - SPARQL & Graph Databases
4. **Однородность знаний - Reification & RDF* & SHACL & ShEx**
5. Интеграция данных в графы знаний - Semantic Data Integration
6. Введение в теорию графов - Graph Theory Intro
7. Векторные представления графов - Knowledge Graph Embeddings
8. Машинное обучение на графах - Graph Neural Networks & KGs
9. Некоторые применения - Question Answering & Query Embedding