Поиск информации в Semantic Web

SPARQL

© Филиппов Ф.В., доцент СПбГУТ
им. проф. М.А.Бонч-Бруевича

Лирическое отступление

Реляционные базы данных и язык SQL

Пример базы данных

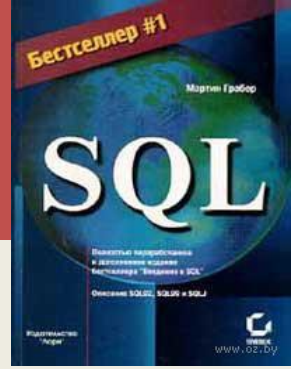


Таблица 1. Salespeople (Продавцы)

snum	sname	city	comm
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rifkin	Barcelona	0.15
1003	Axelrod	New York	0.10

Таблица 2. Customers (Покупатели)

cnum	cname	city	rating	snum
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	San Jose	300	1007
2007	Pireira	Rome	100	1004

Таблица 3. Orders (Заказы)

onum	amt	odate	cnum	snum
3001	18.69	10/03/2008	2008	1007
3003	767.19	10/03/2008	2001	1001
3002	1900.10	10/03/2008	2007	1004
3005	5160.45	10/03/2008	2003	1002
3006	1098.16	10/03/2008	2008	1007
3009	1713.23	10/04/2008	2002	1003
3007	75.75	10/04/2008	2004	1002
3008	4723.00	10/05/2008	2006	1001
3010	1309.95	10/06/2008	2004	1002
3011	9891.88	10/06/2008	2006	1001

SQL запросы

Квалифицированный выбор строк

Ключевое слово WHERE **city = ‘London’**

Условие (предикат)

snum	*sname*	*city*	*comm*
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rifkin	Barcelona	0.15
1003	Axelrod	New York	0.10

```
======= SQL Execution Log =======
| SELECT sname, city             |
| FROM  Salespeople              |
| WHERE city = 'London'          |
| ============================== |
|  sname        city             |
|  -------      ----------       |
|  Peel         London           |
|  Motika       London           |
================================
```

SQL запросы

Квалифицированный выбор строк

cnum	*cname*	*city*	*rating*	*snum*
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	
2008	Cisneros	San Jose	300	
2007	Pireira	Rome	100	

Ключевое слово **WHERE** `rating = 100`

Условие

```
============== SQL Execution Log ==========
| SELECT *                                  |
| FROM  Customers                           |
| WHERE rating = 100;                       |
| ========================================= |
|  cnum    cname      city      rating  snum  |
|  ------  --------   ------    ----    ----- |
|  2001    Hoffman    London    100     1001  |
|  2006    Clemens    London    100     1001  |
|  2007    Pereira    Rome      100     1001  |
=============================================
```

SQL запросы

Составное условие

cnum	*cname*	*city*	*rating*	*snum*
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	San Jose	300	1007
2007	Pireira	Rome	100	1004

city = ‘San Jose’ AND rating > 200

Составное условие

```
========== SQL Execution Log ============
| SELECT *                                |
| FROM  Customers                         |
| WHERE city = 'San Jose'                 |
| AND rating > 200;                       |
=========================================
|  cnum    cname      city       rating   snum  |
|  ------  --------   --------   ----     -----  |
|  2008    Cirneros   San Jose   300      1007   |
=========================================
```

SQL запросы

Составное условие

cnum	*cname*	*city*	*rating*	*snum*
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	San Jose	300	
2007	Pireira	Rome	100	

city = ‘San Jose’ OR rating > 200

Составное условие

```
========== SQL Execution Log ==========
| SELECT *                              |
| FROM  Customers                       |
| WHERE city = 'San Jose'               |
| OR rating > 200;                      |
| ===================================== |
|  cnum   cname     city       rating  snum |
|  ------ --------  --------   ----    ----- |
|  2003   Liu       San Jose   200     1002  |
|  2004   Grass     Berlin     300     1002  |
|  2008   Cirneros  San Jose   300     1007  |
=======================================
```

SQL запросы

Специальный оператор BETWEEN

```
WHERE comm BETWEEN 0.10 AND 0.12;
```

snum	*sname*	*city*	*comm*
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rifkin	Barcelona	0.15
1003	Axelrod	New York	0.10

```
======== SQL Execution Log ==========
| SELECT *                           |
| FROM  Salespeople                  |
| WHERE comm BETWEEN 0.10 AND 0.12;  |
| ===================================|

|  snum     sname       city          comm  |
|  ------    ----------  -----------   -------  |
|  1001     Peel        London        0.12  |
|  1004     Motika       London       0.11  |
|  1003     Axelrod     New York      0.10  |
=======================================
```

SQL запросы

Специальный оператор LIKE *(примеры)*

WHERE cname LIKE ‘P__L%';

snum	cname	city	comm
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rifkin	Barcelona	0.15
1003	Axelrod	New York	0.10

```
SELECT cname
FROM  Salespeople
WHERE cname
LIKE ‘P__L%';
```

cname
Peel

WHERE cname LIKE 'G%';

cnum	cname	city	rating	snum
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002

```
SELECT *
FROM  Customers
WHERE cname LIKE 'G%';
```

cnum	cname	city	rating	snum
2002	Giovanni	Roma	200	1003
2004	Grass	Berlin	300	1002

SQL запросы

Вспомогательный оператор GROUP BY *(пример)*

onum	amt	odate	cnum	snum
3001	18.69	10/03/2008	2008	1007
3003	767.19	10/03/2008	2001	1001
3002	1900.10	10/03/2008	2007	1004
3005	5160.45	10/03/2008	2003	1002
3006	1098.16	10/03/2008	2008	1007
3009	1713.23	10/04/2008	2002	1003
3007	75.75	10/04/2008	2004	1002
3008	4723.00	10/05/2008	2006	1001
3010	1309.95	10/06/2008	2004	1002
3011	9891.88	10/06/2008	2006	1001

```
==== SQL Execution Log =====
| SELECT snum, MAX (amt)      |
| FROM  Orders                |
| GROUP BY snum;              |
| ========================== |
|  snum                       |
|   ------      -----------   |
|   1001        9891.88       |
|   1002        5160.45       |
|   1003        1713.23       |
|   1004        1900.10       |
|   1007        1098.16       |
==============================
```

Максимальная сумма продажи у каждого продавца

SQL запросы

Объединение запросов

Используется специальный оператор UNOIN

snum	*sname*	*city*	*comm*
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rifkin	Barcelona	0.15
1003	Axelrod	New York	0.10

cnum	*cname*	*city*	*rating*	*snum*
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	San Jose	300	1007
2007	Pireira	Rome	100	1004

```
===  SQL Execution Log ===
| SELECT snum, sname          |
| FROM  Salespeople           |
| WHERE city = 'London'       |
| UNION                       |
| SELECT cnum, cname          |
| FROM Customers              |
| WHERE city = 'London';      |
===========================
|  -----    --------          |
|   1001    Peel              |
|   1004    Motika            |
|   2001    Hoffman           |
|   2006    Climens           |
===========================
```

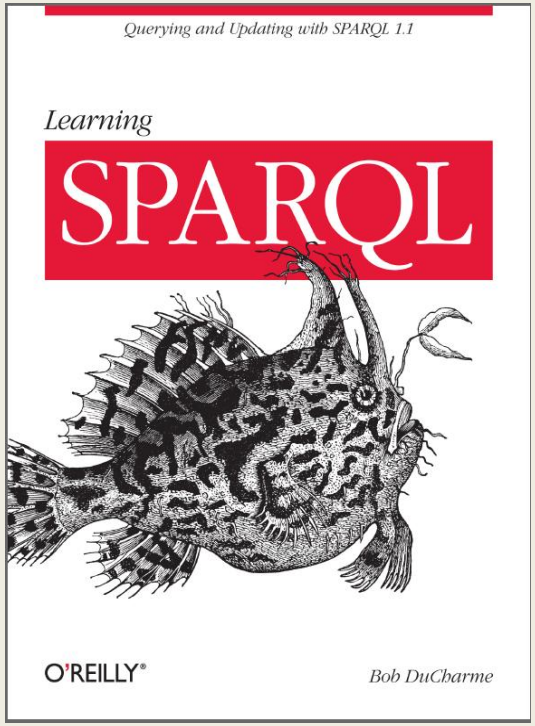
Конец
лирического отступления

Реляционные базы данных и язык SQL

Semantic Web

SPARQL

SPARQL – SPARQL Protocol And RDF Query Language
Описывается множеством спецификаций комитета W3C.



Язык запросов SPARQL для RDF

Рекомендация W3C, 15 января 2008

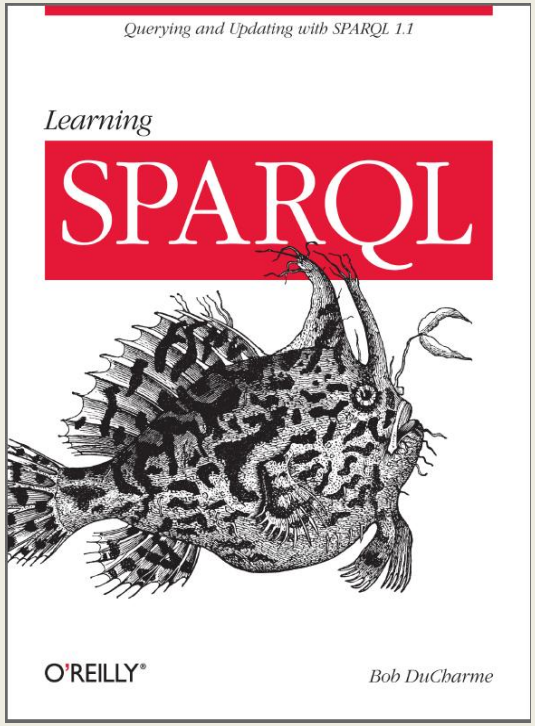
Текущая версия:
http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/
Последняя версия:
http://www.w3.org/TR/rdf-sparql11-query/

SPARQL - язык запросов к данным представленным по модели **RDF**, а также **протокол** для передачи этих запросов и ответов на них.

Semantic Web

SPARQL

SPARQL – SPARQL Protocol And RDF Query Language
Описывается множеством спецификаций комитета W3C.



SPARQL - язык запросов к данным представленным по модели **RDF**, а также **протокол** для передачи этих запросов и ответов на них.

SPARQL

SPARQL позволяет пользователям писать **глобально однозначные запросы**. Например, следующий запрос возвращает имена и адреса электронной почты каждого человека в мире:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
        ?person  a            foaf:Person.
        ?person  foaf:name    ?name.
        ?person  foaf:mbox    ?email.
         }
```

Этот запрос может быть распределен на несколько конечных точек SPARQL и сбор результатов осуществляется процедурой, известной как **федеративный поиск**.

Общая структура SPARQL-запроса

```
PREFIX foo: <http://example.com/resources/>
# префиксные объявления
FROM ...
 # источники запроса
SELECT ...
# состав результата
WHERE {...}
 # шаблон запроса
ORDER BY ...
# модификаторы запроса
```

- **Префиксные объявления** служат для указания сокращений URI.
- **Источники запроса** определяют, какие RDF-графы запрашиваются.
- **Состав результата** определяет возвращаемые элементы данных.
- **Шаблон запроса** определяет, что запрашивать из набора данных.
- **Модификаторы запроса** ограничивают, упорядочивают, преобразуют результаты запроса.

Отличия SPARQL и SQL

```
SELECT  возвращаемые переменные
WHERE  {
          удовлетворяемые условия
          в форме триплетов
        }
```

Возвращаемые переменные:

- любые имена, начинающиеся с ? или $

Удовлетворяемые условия:

- триплеты, объединяемые «.» точкой или «;» точкой с запятой;
- триплеты могут включать сущности (URI), литералы и переменные;
- переменные могут обозначать сущности и литералы.

Синтаксис записи триплетов

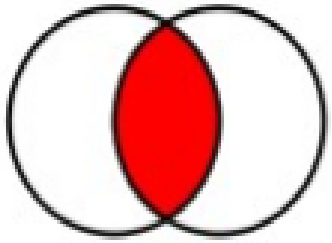
```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE {
        <http://example.org/book/book1>  dc:title  ?title
      }
```

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/book/>
SELECT $title
WHERE {
          :book1  dc:title  $title
          }
```

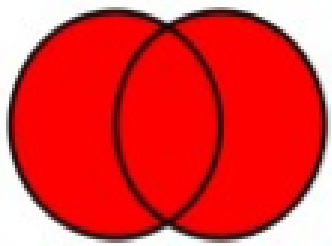
```
BASE <http://example.org/book/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT $title
WHERE {
          <book1>  dc:title  ?title
          }
```

Объединение триплетов

- Тройка, триплет (?X Отношение ?Y)
- Конъюнкция (A ^ B.)
  - SELECT ?x WHERE { ?A Relation1 ?x .
    ?x Relation2 ?B .}



- Дизъюнкция (A or B.)
  - SELECT ?x WHERE { ?A Relation1 ?x .
    union
    ?x Relation2 ?B .}



- Дополнительные шаблоны для уточнения поиска.

Перечни в триплетах

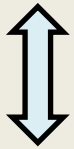
predicate-object

```
?x      foaf:name      ?name ;
        foaf:mbox      ?mbox .
```

```
?x      foaf:name      ?name .
?x      foaf:mbox      ?mbox .
```

objects

```
?x   foaf:nick   "Alice" , "Alice_" .
```



```
?x   foaf:nick   "Alice" .
?x   foaf:nick   "Alice_" .
```

комбинированный

```
?x foaf:name ?name ;
   foaf:nick "Alice" , "Alice_" .
```

```
?x foaf:name ?name .
?x foaf:nick "Alice" .
?x foaf:nick "Alice_" .
```

Ключевые слова в запросах

1) **OPTIONAL** — обозначает необязательный шаблон (триплет).
2) **UNION** — позволяет объединять результаты различных шаблонов.
3) **REDUCED** — сокращает количество результатов.
4) **DISTINCT** — обеспечивает уникальность решений в ответе на запрос.
5) **ORDER BY** — позволяет отсортировать результаты.
6) **LIMIT** — задает максимальное количество выводимых результатов.
7) **FILTER** — позволяет выделить интересующие данные в результате.
8) **OFFSET** — опускает в результате первые n решений.

Ключевое слово OPTIONAL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
WHERE {
        ?x foaf:name ?name .
        OPTIONAL { ?x foaf:mbox       ?mbox } .
        OPTIONAL { ?x foaf:homepage ?hpage }
         }
```

name	mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@work.example>	

OPTIONAL — обозначает необязательный шаблон

Ключевое слово UNION

```
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
_:a   dc10:title      "SPARQL Query Language Tutorial" .
_:a   dc10:creator  "Alice" .
_:b   dc11:title      "SPARQL Protocol Tutorial" .
_:b   dc11:creator  "Bob" .
_:c   dc10:title       "SPARQL" .
_:c   dc11:title       "SPARQL (updated)" .
```

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE {
            {?book  dc10:title  ?title }
            UNION
            {?book  dc11:title   ?title }
            }
```

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

UNION — позволяет объединять результаты различных шаблонов

Ключевые слова REDUCED и DISTINCT

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
        ?x foaf:name ?name
        }
```

name
"Alice"
"Alice"
"Alice"

```
SELECT REDUCED ?name
```

```
SELECT DISTINCT ?name
```

REDUCED — сокращает количество результатов

DISTINCT — обеспечивает уникальность решений в ответе на запрос

Ключевые слова ORDER BY и LIMIT

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
        ?x foaf:name ?name
        }
ORDER BY ?name
LIMIT 5
```

ORDER BY — позволяет отсортировать результат

LIMIT — задает максимальное количество выводимых результатов (ограничение вывода)

Ключевые слова FILTER и OFFSET

```
SELECT ?x
WHERE {
        ?x a foaf:Person .
         FILTER regex(?x, «felix", "i")
         }
LIMIT 10
```

```
SELECT ?x
WHERE {
        ?x a foaf:Person .
         FILTER regex(?x, «felix", "i")
         }
LIMIT 10
OFFSET 8
```

FILTER — позволяет выделить интересующие данные в результате

OFFSET — опускает в результате первые n решений

x	
http://dbpedia.org/resource/Allyson_Felix	1
http://dbpedia.org/resource/Archduke_Felix_of_Austria	2
http://dbpedia.org/resource/Felix_Andries_Vening_Meinesz	3
http://dbpedia.org/resource/Felix_Baumgartner	4
http://dbpedia.org/resource/Felix_Frankfurter	5
http://dbpedia.org/resource/Felix_Graf_von_Bothmer	6
http://dbpedia.org/resource/Felix_Hell	7
http://dbpedia.org/resource/Felix_Luz	8
http://dbpedia.org/resource/Felix_Maria_von_Exner-Ewarten	
http://dbpedia.org/resource/Felix_Nussbaum	

x
http://dbpedia.org/resource/Felix_Maria_von_Exner-Ewarten
http://dbpedia.org/resource/Felix_Nussbaum
http://dbpedia.org/resource/Felix_Otto
http://dbpedia.org/resource/Felix_de_Muelenaere
http://dbpedia.org/resource/Prince_Felix_of_Denmark
http://dbpedia.org/resource/Prince_Felix_of_Schwarzenberg
http://dbpedia.org/resource/Felix_of_Cantalice
http://dbpedia.org/resource/Prince_Felix_of_Bourbon-Parma
http://dbpedia.org/resource/Felix_Alderisio
http://dbpedia.org/resource/Felix_Aylmer

Точки доступа SPARQL
(Endpoints)

Точка доступа SPARQL Endpoint — это **служба, поддерживающая протокол запросов SPARQL**. Точка доступа позволяет пользователю делать запросы к базе знаний. Сервер обрабатывает запрос и возвращает ответ в некотором, обычно машинно-читаемом, формате.

Точки доступа SPARQL в первую очередь являются API к базам знаний, а представление результатов должно быть реализовано программным обеспечением вызывающей стороны.

Различают два вида точек доступа: **общего назначения** и **локальные**:

- точки доступа **общего назначения** могут производить запросы по любым указанным RDF-документам, находящимся в Semantic Web;

- **локальные точки** доступа способны получать данные только от одного ресурса.

SPARQLer - General purpose processor

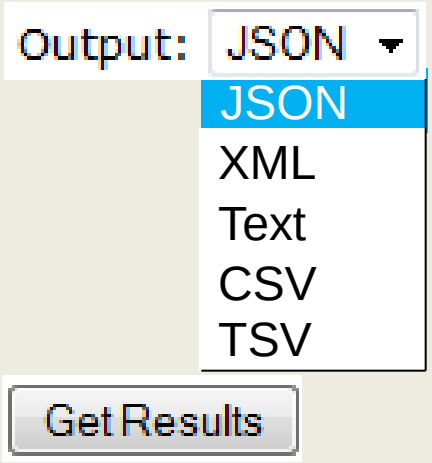
```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?Name ?email
FROM <http://www.w3.org/People/Berners-Lee/card>
WHERE {
        ?person    foaf:name    ?Name .
        ?person    foaf:mbox    ?email  .
        }
```

http://sparql.org/sparql.html

Output: Text

Name	email
"Timothy Berners-Lee"	<mailto:timbl@w3.org>

Форматы результата



SPARQLer - General purpose processor

http://sparql.org/sparql.html

JSON (***J**ava**S**cript **O**bject **N**otation*) — текстовый формат обмена данными, основанный на JavaScript и обычно используемый с этим языком.

```
{"head":{"vars":["Name","email"]},"results":{"bindings":[{"Name":{ "type":"literal","value":"Timothy Berners-Lee" },"email":{"type":"uri","value":"mailto:timbl@w3.org" }}]}}
```

CSV (***C**omma **S**eparated **V**alues* — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных.

```
Name,email
Timothy Berners-Lee,mailto:timbl@w3.org
```

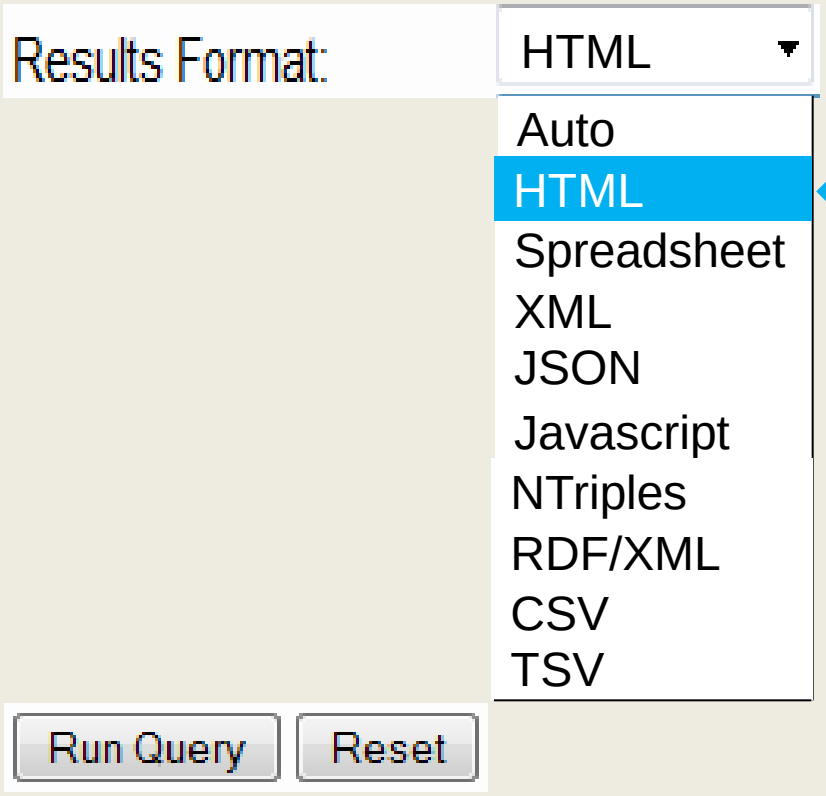
TSV (***T**ab **S**eparated **V**alues* — значения, разделённые табуляцией) — текстовый формат для представления таблиц баз данных.

```
?Name   ?email   "Timothy Berners-Lee“   <mailto:timbl@w3.org>
```

Virtuoso SPARQL Query Editor

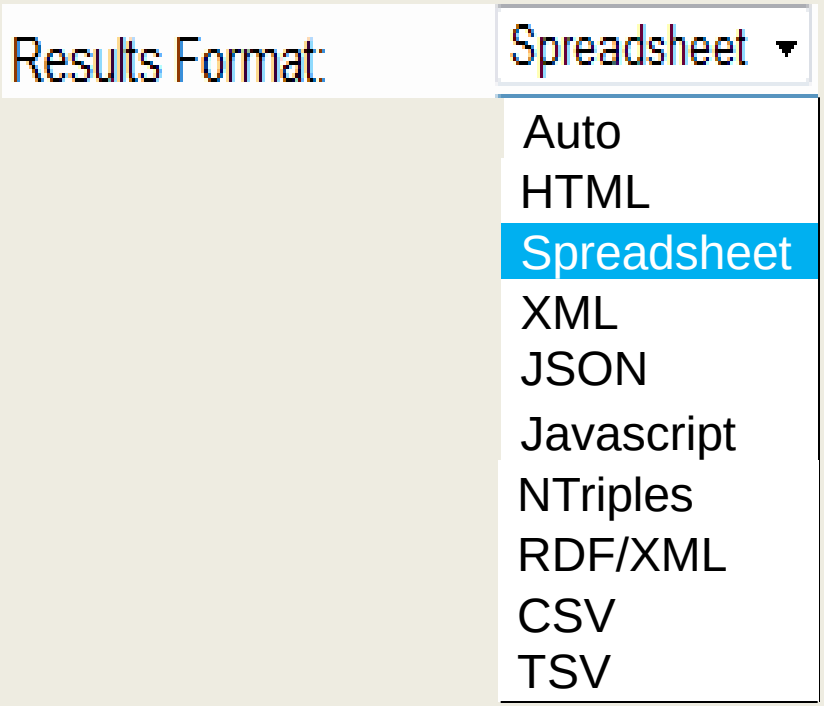
```
SELECT DISTINCT ?Concept
WHERE {
        [ ] a ?Concept
        }
LIMIT 5
```

http://dbpedia.org/sparql



Concept
http://www.w3.org/2002/07/owl#Thing
http://dbpedia.org/ontology/Agent
http://dbpedia.org/ontology/Band
http://dbpedia.org/ontology/Organisation
http://schema.org/MusicGroup

Форматы результата



Virtuoso SPARQL Query Editor

http://dbpedia.org/sparql

```
SELECT DISTINCT ?Concept
WHERE {
        ?xxx a ?Concept
        }
LIMIT 10
```

	A	B
1	Concept	
2	http://www.w3.org/2002/07/owl#Thing	
3	http://dbpedia.org/ontology/Agent	
4	http://dbpedia.org/ontology/Band	
5	http://dbpedia.org/ontology/Organisation	
6	http://schema.org/MusicGroup	
7	http://schema.org/Organization	
8	http://dbpedia.org/class/yago/MusicalGroupsDisestablishedIn1986	
9	http://dbpedia.org/class/yago/DanceBand108249960	
10	http://dbpedia.org/class/yago/Organization108008335	
11	http://dbpedia.org/class/yago/RockGroup108250501	
12		
13		

SPARQL Endpoint

http://rdf.myexperiment.org/sparql

```
SELECT DISTINCT ?x
WHERE {
        [ ] a ?x
      }
```

Results

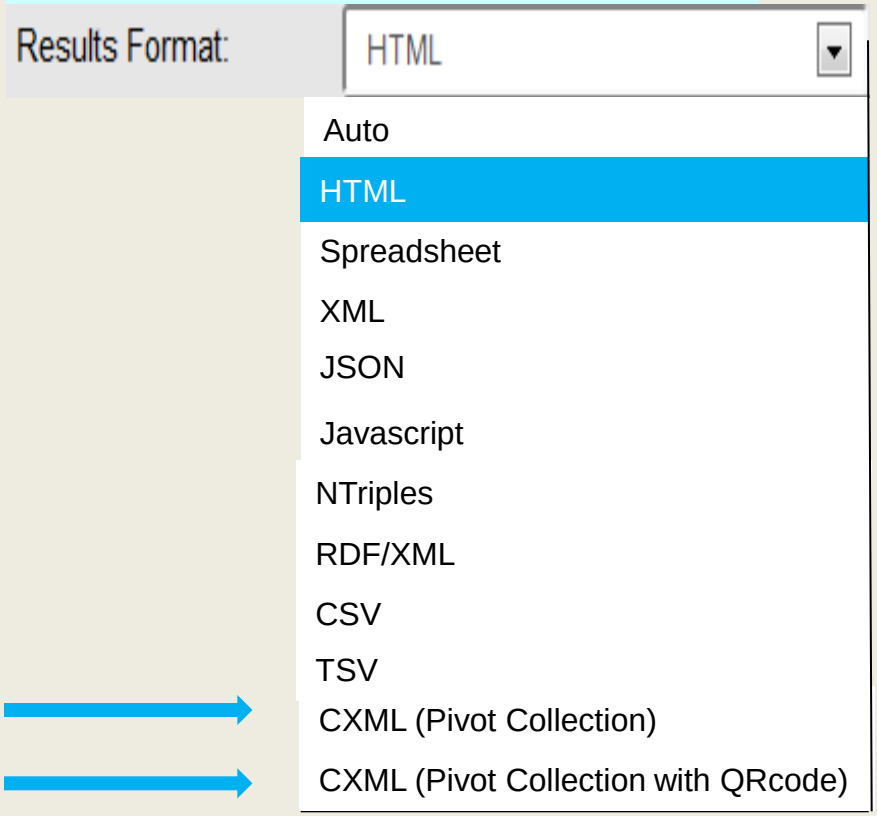
Time Taken: <1 seconds

No. of Results: 48

x
http://www.w3.org/2002/07/owl#AnnotationProperty
http://rdf.myexperiment.org/ontologies/components/Output
http://rdf.myexperiment.org/ontologies/packs/Relationship
http://rdf.myexperiment.org/ontologies/snarm/Access
http://rdf.myexperiment.org/ontologies/packs/RelationshipEntry
http://rdf.myexperiment.org/ontologies/components/WSDLProcessor
http://www.w3.org/2002/07/owl#DatatypeProperty
http://rdf.myexperiment.org/ontologies/packs/PackSnapshot
http://rdf.myexperiment.org/ontologies/components/Link
http://rdf.myexperiment.org/ontologies/base/ContentType
http://rdf.myexperiment.org/ontologies/annotations/Tagging
http://rdf.myexperiment.org/ontologies/annotations/Tag
http://rdf.myexperiment.org/ontologies/base/GroupAnnouncement
http://rdf.myexperiment.org/ontologies/base/Announcement
http://rdf.myexperiment.org/ontologies/components/OtherProcessor
http://rdf.myexperiment.org/ontologies/packs/LocalPackEntry
http://www.w3.org/2002/07/owl#ObjectProperty
http://rdf.myexperiment.org/ontologies/components/Input
http://rdf.myexperiment.org/ontologies/annotations/Comment

OpenLink Virtuoso SPARQL Query Editor

```
SELECT DISTINCT ?Concept
WHERE {
        [ ] a ?Concept
        }
 LIMIT 10000
```



http://demo.openlinksw.com/sparql/

Concept
http://www.openlinksw.com/schemas/virtrdf#QuadMapFormat
http://www.openlinksw.com/schemas/virtrdf#QuadStorage
http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMap
http://www.openlinksw.com/schemas/virtrdf#QuadMap
http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMapFormat
http://www.openlinksw.com/schemas/virtrdf#QuadMapValue
http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMapATable
http://www.openlinksw.com/schemas/virtrdf#array-of-QuadMapColumn
http://www.openlinksw.com/schemas/virtrdf#QuadMapColumn
http://www.openlinksw.com/schemas/virtrdf#QuadMapFText
http://www.openlinksw.com/schemas/virtrdf#QuadMapATable
http://www.openlinksw.com/schemas/virtrdf#array-of-string
http://www.w3.org/1999/02/22-rdf-syntax-ns#Property
http://www.w3.org/2002/07/owl#DatatypeProperty
http://www.openlinksw.com/schema/attribution#DataSource
http://www.w3.org/2002/07/owl#AnnotationProperty
http://www.w3.org/2002/07/owl#Class
http://www.w3.org/2000/01/rdf-schema#Class
http://www.w3.org/2002/07/owl#Ontology
http://www.w3.org/2002/07/owl#OntologyProperty
http://purl.org/ontology/bibo/Webpage
http://bblfish.net/work/atom-owl/2006-06-06/#Content
http://www.openlinksw.com/schemas/opltable#Table
http://www.openlinksw.com/schemas/opltable#Row
http://purl.org/net/provenance/ns#DataCreation
http://purl.org/net/provenance/ns#DataItem

rdfs:label

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
WHERE {
          ?subject rdfs:label ?object
         }
LIMIT 10
```

http://dbpedia.org/sparql

subject	object
http://wikidata.dbpedia.org/resource/Q11026841	"Розкриття інформації емітентом"@uk
http://wikidata.dbpedia.org/resource/Q967886	"Арбатов Георгій Аркадійович"@uk
http://wikidata.dbpedia.org/resource/Q204911	"Російська соціал-демократична робітнича партія"@uk
http://wikidata.dbpedia.org/resource/Q4308551	"Мустафін Олексій Рафаїлович"@uk
http://wikidata.dbpedia.org/resource/Q1189943	"Магістр Європейського права"@uk
http://wikidata.dbpedia.org/resource/Q21204	"Міжнародний фонетичний алфавіт"@uk
http://wikidata.dbpedia.org/resource/Q7929651	"Бондаренко Віктор Вікторович (політик)"@uk
http://wikidata.dbpedia.org/resource/Q7848836	"Щербатих Станіслав Іванович"@uk
http://wikidata.dbpedia.org/resource/Q101751	"Чемпіонат світу з футболу 1994"@uk
http://wikidata.dbpedia.org/resource/Q953761	"Чемпіонат світу з футболу 1994"@uk

rdfs:subClassOf

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
WHERE {
          ?subject rdfs:subClassOf ?object
         }
LIMIT 5
```

http://dbpedia.org/sparql

subject	object
http://dbpedia.org/class/yago/13th-centuryConflicts	http://dbpedia.org/class/yago/Conflict100958896
http://dbpedia.org/class/yago/18th-centuryAmericanPeople	http://dbpedia.org/class/yago/Person100007846
http://dbpedia.org/class/yago/1950sScienceFictionFilms	http://dbpedia.org/class/yago/Movie106613686
http://dbpedia.org/class/yago/1962FIFAWorldCupPlayers	http://dbpedia.org/class/yago/Player110439851
http://dbpedia.org/class/yago/1966FIFAWorldCupPlayers	http://dbpedia.org/class/yago/Player110439851

rdfs:comment

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?subject ?object
WHERE {
            ?subject rdfs:comment ?object
           }
LIMIT 5
```

http://dbpedia.org/sparql

subject	object
http://dbpedia.org/resource/Tillandsia_'Comet'	"'Comet' is a hybrid cultivar of the genus Tillandsia in the Bromeliad family."@en
http://dbpedia.org/resource/Lake_Nuijamaa	"Lake Nuijamaa is a lake on the border between Finland and Russia next to the town of Nuijamaa. It is part of the Saimaa Canal linking Vyborg Bay in the Baltic Sea to Lake Saimaa in the Finnish Lakeland."@en
http://dbpedia.org/resource/Miroslav_Ivani%C5%A1evi%C4%87	"Miroslav Ivanišević is a Montenegrin politician. He was appointed as the Montenegrin Minister of Finance in 1998. In 2007 he was accused of being involved in cigarette smuggling into Italy, but was found not guilty in 2010."@en
http://dbpedia.org/resource/Francisco_Bautista	"Francisco Bautista Cuamatzi is a male long-distance runner from Mexico. He represented his native country at the 2008 Summer Olympics in Beijing, PR China, where he finished in 66th place in the men's marathon event, clocking 2:29.28. Rojas set his personal best in the marathon on March 7, 2004 in Torreón."@en
http://dbpedia.org/resource/American_Legion_Field_(Florence)	"American Legion Field is a baseball venue in Florence, South Carolina, USA. It is home to the Florence Red Wolves of the Coastal Plain League, a collegiate summer baseball league. The Red Wolves have played at the field since 1998. The venue was built sometime before 1981 and has a capacity of 3,500. The field's dimensions are 305 ft. down the foul lines, 335 ft. to the gaps, and 385 ft. to dead center field."@en

Домашнее задание

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT *
WHERE {
        ?subject rdfs: свойство ?x
       }
```

http://dbpedia.org/sparql

На базе представленного SPARQL-запроса изучите различные свойства RDFS

Составьте SPARQL-запросы для изучения конструкций языка OWL Lite

LODUM SPARQL Endpoint

LODUM Blog About Team LIFE Project Partners Results Data SPARQL Endpoint

LODUM
Linked Open Data
University of Münster

LODUM is the University of Münster's Open Data initiative. We make any public information about the university available in machine-readable formats for easy access and reuse. By opening and cross-referencing data from different information systems, LODUM provides a one-stop shop for any data about the University of Münster.

LODUM is hosted at the Institute for Geoinformatics' Semantic Interoperability Lab (MUSIL) and carried out in collaboration with a wide range of partners across the university. The initiative was started on internal funding from the university's innovation funds and is now continued on external funding.

The LODUM team has co-initiated both LinkedUniversities.org and LinkedScience.org.

Explore the Data

News from the Blog

Feb 19, 2014 Two full papers accepted for presentation at AGILE 2014

Dec 9, 2013 Making the Web of Data available via WFS

MSc Thesis Topic: Campus navigation

Campusplan App

The LODUM data also feed the university's Campusplan app with administrative data, navigation instructions, and cafeteria menues. The app is available for iOS and Android, and

SPARQL запросы

SPARQL Editor

Our SPARQL endpoint can also be queried directly using http://data.uni-muenster.de/sparql.
Just copy&paste the URL for the endpoint and you can access and query the data from any other SPARQL client, for example, Twinkle (Desktop) or R.

Show/Hide Prefixes

```
1 SELECT * WHERE {
2         ?a rdf:type foaf:Person .
3 }
4 LIMIT 10
5
```

The editor is based on CodeMirror and JQuery. The triple store running in the backend is powered by OWLIM

Submit

?a
http://data.uni-muenster.de/context/cris/person/12571
http://data.uni-muenster.de/context/cris/person/6648
http://data.uni-muenster.de/context/cris/person/11616
http://data.uni-muenster.de/context/cris/person/10381
http://data.uni-muenster.de/context/cris/person/15681
http://data.uni-muenster.de/context/cris/person/12895
http://data.uni-muenster.de/context/cris/person/16644
http://data.uni-muenster.de/context/cris/person/12806
http://data.uni-muenster.de/context/cris/person/7358
http://data.uni-muenster.de/context/cris/person/7356

http://data.uni-muenst

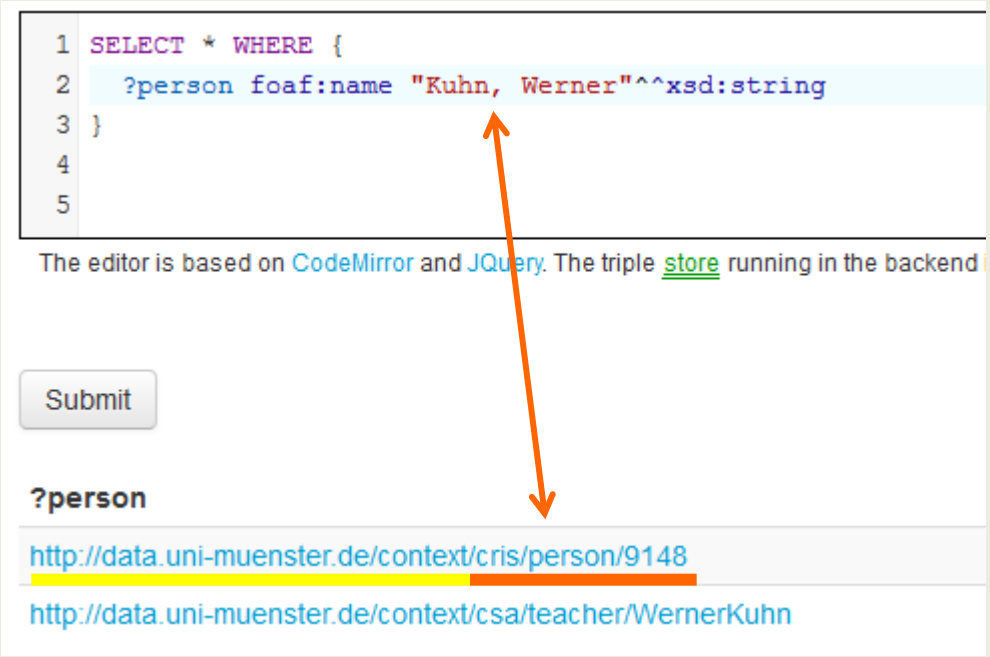
LODUM Data

Show/Hide Prefixes

```
base <http://data.uni-muenster.de/context/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix dct: <http://purl.org/dc/terms/>
prefix dc: <http://purl.org/dc/elements/1.1/>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix isbd: <http://iflastandards.info/ns/isbd/elements/>
prefix skos: <http://www.w3.org/2004/02/skos/core#>
prefix bibo: <http://purl.org/ontology/bibo/>
prefix rda: <http://RDVocab.info/ElementsGr2/>
prefix blt: <http://data.bl.uk/schema/bibliographic#>
prefix bio: <http://purl.org/vocab/bio/0.1/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix event: <http://purl.org/NET/c4dm/event.owl#>
prefix org: <http://www.w3.org/ns/org#>
prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
prefix pv: <http://linkedscience.org/pv/ns#>
prefix fn:<http://www.w3.org/2005/xpath-functions#>
prefix vcard:<http://www.w3.org/2006/vcard/ns#>
prefix aiiso:<http://purl.org/vocab/aiiso/schema#>
prefix teach:<http://linkedscience.org/teach/ns#>
prefix res:<http://www.medsci.ox.ac.uk/vocab/researchers/0.1/>
prefix resume: <http://rdfs.org/resume-rdf/#>
prefix tis:<http://www.ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl#>
prefix ti:<http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl#>
prefix lode:<http://linkedevents.org/ontology/>
prefix wgs84:<http://www.w3.org/2003/01/geo/wgs84_pos#>
prefix tipr:<http://www.ontologydesignpatterns.org/cp/owl/timeindexedpersonrole.owl#>
```

SPARQL запросы к
LODUM Data

1. Query a list of a researcher's publications
2. Query a list of a researcher's coauthors
3. Query a list of a researcher's coauthors publications
4. Embed this data in a website



URI

<cris/person/9148>

http://linkedscience.org/tutorials/tutorial-using-sparql-on-lodum-data/

SPARQL запросы к LODUM Data

1. Query a list of a researcher's publications
2. Query a list of a researcher's coauthors
3. Query a list of a researcher's coauthors publications
4. Embed this data in a website

```
SELECT * WHERE {
?pub bibo:producer <cris/person/9148> .
?pub dct:title ?title .
?pub dct:issued ?year
}
```

?pub	?title	?year
http://data.uni.../26911	"Affordances as Qualities"	"2010"
http://data.uni.../17225	"Cognitive Semantics and..."	"2007"
http://data.uni.../17177	"Electronic GI Marketplaces"	"2004"

SPARQL запросы к LODUM Data

1. Query a list of a researcher's publications
2. Query a list of a researcher's coauthors
3. Query a list of a researcher's coauthors publications
4. Embed this data in a website

```
SELECT ?name WHERE {
?pub bibo:producer <cris/person/9148> .
?pub bibo:producer ?coauthor .
?coauthor foaf:name ?name
}
```

```
SELECT DISTINCT ?name WHERE {
?pub bibo:producer <cris/person/9148> .
?pub bibo:producer ?coauthor .
?coauthor foaf:name ?name
}
```

?name
"Schwering, Angela"
"Scheider, Simon"

SPARQL запросы к LODUM Data

1. Query a list of a researcher's publications
2. Query a list of a researcher's coauthors
3. Query a list of a researcher's coauthors publications
4. Embed this data in a website

```
SELECT DISTINCT ?name ?title ?year WHERE {
?pub bibo:producer <cris/person/9148> .
?pub bibo:producer ?coauthor .
?coauthor foaf:name ?name .
?pub dct:title ?title .
?pub dct:issued ?year
FILTER(?coauthor owl:sameAs <cris/person/9148>)
}
```

?name	?title	?year
"Scheider, Simon"	"Grounding Geographic Categories..."	"2009"
"KeÃŸler, Carsten"	"Semantic Referencing of Geosensor..."	"2011"
"Kauppinen, Tomi"	"Semantic Referencing of Geosensor..."	"2011"

SPARQL запросы к LODUM Data

1. Query a list of a researcher's publications
2. Query a list of a researcher's coauthors
3. Query a list of a researcher's coauthors publications
4. Embed this data in a website

Библиотеки JavaScript

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script type="text/javascript" src="http://data.uni-muenster.de/rdf-spark/jquery.spark.js"></script>
```

Шаблон для заголовка и запроса на SPARQL

```
<span class="spark"
data-spark-format="http://data.uni-muenster.de/rdf-spark/jquery.spark.lodumlist.js"
data-spark-param-head="Заголовок для данных"
data-spark-query="SPARQL запрос">
</span>
```

SPARQL запросы к LODUM Data

```
base http://data.uni-muenster.de/context/
prefix dct: http://purl.org/dc/terms/
prefix person: http://data.uni-muenster.de/context/cris/person/
SELECT DISTINCT  ?title WHERE {
?pub bibo:producer <cris/person/9148> .
?pub bibo:producer ?coauthor .
?coauthor foaf:name ?name .
?pub dct:title ?title .
?pub dct:issued ?year
FILTER(?coauthor owl:sameAs <cris/person/9148>)
}
ORDER BY DESC(?year) ?title ?name
LIMIT 5
```

Prof. Werner Kuhn's coauthors' publications

- Establishing a Publication Outlet Rating for GIScience
- Semantic Referencing of Geosensor Data and Volunteered Geographic Information
- A process-centric ontological approach for integrating geo-sensor data
- Affordances as Qualities
- A Hybrid Semantic Similarity Measure for Spatial Information Retrieval

DBpedia

Набор данных DBpedia представляет из себя большую многодоменную онтологию которая была получена из Википедии. Английская версия DBpedia в настоящее время описывает 4,0 миллиона "сущностей" с 470 миллионами "фактов".

Реализованы локализованные версии DBpedia на 119 языках. Все эти версии вместе описывают 24,9 миллиона объектов. Полный набор данных DBpedia индексирует и описывает 12,6 миллионов уникальных вещей на 120 различных языках, использует 24,6 миллиона ссылок на изображения и 27,6 миллионов HTML-ссылок на внешние веб-страницы. DBpedia использует 45,0 миллионов ссылок на внешние наборы RDF-данных, 67,0 миллионов ссылок на категории Wikipedia и 41,2 миллиона ссылок на YAGO категории. Этот набор данных состоит из 2,46 миллиарда единиц информации (RDF троек), из которых 470 миллионов были взяты из английского издания Википедии, 1,98 миллиарда были извлечены из других языковых разделов.

SPARQL запросы

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX prop: <http://dbpedia.org/property/>
PREFIX yago: <http://dbpedia.org/class/yago/>
 SELECT ?place ?population
WHERE {
            ?place   rdf:type    yago:English-speakingCountriesAndTerritories.
            ?place   rdf:type    dbpedia-owl:Country;
                        prop:populationEstimate   ?population .
}
ORDER BY DESC(?population)
LIMIT 5
```

place	population
http://dbpedia.org/resource/Pakistan	180440005
http://dbpedia.org/resource/Nigeria	170123740
http://dbpedia.org/resource/Kenya	43500000
http://dbpedia.org/resource/Ghana	24200000
http://dbpedia.org/resource/Australia	22859335

SPARQL запросы

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX prop: <http://dbpedia.org/property/>

SELECT ?place ?population
WHERE {
        ?place rdf:type yago:AfricanCountries.
        ?place rdf:type dbpedia-owl:Country;
                prop:populationEstimate ?population .
         }
ORDER BY DESC(?population)
LIMIT 5
```

http://dbpedia.org/sparql

place	population
http://dbpedia.org/resource/Nigeria	170123740
http://dbpedia.org/resource/Ethiopia	91195675
http://dbpedia.org/resource/Egypt	79602000
http://dbpedia.org/resource/Democratic_Republic_of_the_Congo	75507308
http://dbpedia.org/resource/Kenya	43500000

SPARQL запросы

```
PREFIX bpedia-owl: <http://dbpedia.org/ontology/>
SELECT ?Cat
{
  <http://dbpedia.org/resource/Cat> dbpedia-owl:abstract ?Cat
  FILTER(langMatches(lang(?Cat), "RU"))
}
```

Cat
"Ко́шка, или дома́шняя ко́шка — домашнее животное, одно из наиболее популярных «животных-компаньонов». С зоологической точки зрения, домашняя кошка — млекопитающее семейства кошачьих отряда хищных. Ранее домашнюю кошку нередко рассматривали как отдельный биологический вид. С точки зрения современной биологической систематики домашняя кошка (Felis silvestris catus) является подвидом лесной кошки . Являясь одиночным охотником на грызунов и других мелких животных, кошка — социальное животное, использующее для общения широкий диапазон звуковых сигналов, а также феромоны и движения тела. В настоящее время в мире насчитывается около 600 млн домашних кошек, выведено около 256 пород, от длинношёрстных до лишённых шерсти, признанных и зарегистрированных различными фелинологическими организациями. На протяжении 10 000 лет кошки ценятся человеком, в том числе за способность охотиться на грызунов и других домашних вредителей."@ru

Формы SPARQL-запросов

Язык SPARQL определяет четыре различных варианта запросов для различных целей:

SELECT запрос

Извлекает необработанные значения из точки доступа SPARQL и возвращает результаты в формате таблицы.

CONSTRUCT запрос

Извлекает информацию из точки доступа SPARQL в формате RDF и преобразовывает результаты к определенной форме.

ASK запрос

Формирует запрос типа Истина/Ложь.

DESCRIBE запрос

Получает описание RDF-ресурса. Реализация поведения DESCRIBE-запросов определяется разработчиком точки доступа SPARQL.

http://www.w3.org/TR/rdf-sparql-query/

SPARQL-запрос ASK

ASK - правда ли, что

http://api.talis.com/stores/space/items/tutorial/spared.html

```
PREFIX nasa: <http://nasa.dataincubator.org/spacecraft/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

ASK { nasa:1968-089A     foaf:name    "Apollo 7" . }
```

SPARQL-запрос SELECT

SELECT - заполни недостающее

http://dbpedia.org/sparql

```
PREFIX dbpo: <http://dbpedia.org/ontology/>
SELECT *
WHERE {
        ?e dbpo:series <http://dbpedia.org/resource/The_Sopranos> .
        ?e dbpo:releaseDate  ?date .
        ?e dbpo:episodeNumber  ?number .
        ?e dbpo:seasonNumber  ?season
        }
ORDER BY DESC(?date)
```

SPARQL-запрос CONSTRUCT

CONSTRUCT - конструирует новые RDF-графы

CONSTRUCT предназначен для того, чтобы преобразовывать данные. В секции WHERE мы пишем то же, что и писали в WHERE у SELECT-запроса, а сразу после CONSTRUCT мы пишем то, к чему хотим преобразовать наши данные.

SPARQL-запрос CONSTRUCT

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
PREFIX d: <http://learningsparql.com/ns/data#>
CONSTRUCT {?person ?p ?o.}
WHERE
{
    ?person  ab:firstName   «Петя";
             ab:lastName    «Иванов";
             ?p              ?o.
}
```

SPARQL-запрос CONSTRUCT

```
PREFIX gp: <http://wifo-mannheim.de/gutendata/resource/people/>
CONSTRUCT
{
        http://dbpedia.org/resource/Joseph_Hocking    ?dbpProperty    ?dbpValue.
        gp:Hocking_Joseph                             ?gutenProperty ?gutenValue.
}
WHERE
{
  SERVICE <http://DBpedia.org/sparql>
  {
  http://dbpedia.org/resource/Joseph_Hocking  ?dbpProperty  ?dbpValue.
  }

  SERVICE <http://wifo-mannheim.de/gutendata/sparql>
  {
  gp:Hocking_Joseph  ?gutenProperty  ?gutenValue.
  }
}
```

SPARQL-запрос CONSTRUCT

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
CONSTRUCT  { ?course   ab:courseTitle   ?courseName.}
FROM  NAMED  <ex125.ttl>
WHERE
{
   GRAPH  <ex125.ttl>
   {
     ?course   ab:courseTitle   ?courseName
   }
}
```

```
CONSTRUCT { ?s  ?p  ?o }
FROM <http://rdf.freebase.com/rdf/en.joseph_hocking>
WHERE
{ ?s ?p ?o }
```

SPARQL-запрос DESCRIBE

DESCRIBE запрашивает тройки, которые описывают конкретный ресурс. Спецификация SPARQL оставляет для процессора - решить, что необходимо отправить обратно для описания ресурса. Это привело к несогласованности реализаций запросов DESCRIBE.

DESCRIBE - опиши ресурс так как тебя учили

Расскажите мне что-нибудь о Пушкине! DESCRIBE

SPARQL поиск bif:conteins

(вместо regex)

Virtuoso и некоторые другие SPARQL endpoints спецификации SPARQL 1.1 расширены полезными пользовательскими функциями. Одной из них является **bif:contains ‘ ‘**, которая **работает значительно быстрей**, чем regex().

```
SELECT DISTINCT ?film ?studio ?producer
WHERE {
        ?film rdf:type
<http://dbpedia.org/ontology/Film> .
        ?film foaf:name ?film_title .
FILTER regex(str(?film_title), "\\bArgo\\b") .
        ?film dbpprop:studio ?studio .
FILTER regex(str(?studio), "\\bPictures\\b") .
        ?film dbpprop:producer ?producer .
FILTER regex(str(?producer), "\\bBen\\b")
        }
```

```
SELECT DISTINCT ?film ?studio ?producer
WHERE {
        ?film rdf:type
<http://dbpedia.org/ontology/Film> .
        ?film foaf:name ?name .
        ?name bif:contains '"Argo"' .
        ?film dbpprop:studio ?studio .
        ?studio bif:contains '"Pictures"' .
        ?film dbpprop:producer ?producer .
        ?producer bif:contains '"Ben"‘
        }
```


https://coderwall.com/p/4v5b5w

SPARQL поиск bif:conteins

(пример)

```
SELECT DISTINCT ?lbl
WHERE {
        ?country     rdfs:label    ?lbl .
  FILTER(bif:contains(?lbl, "Republic")) .
        ?country     rdf:type      dbpedia-owl:Country .
}
```

lbl
"Autonomous Albanian Republic of Korçë"@en
"Democratic Republic of Afghanistan"@en
"Free Republic of Schwarzenberg"@en
"Government of the Autonomous Republic of Abkhazia"@en
"Mongolian People's Republic"@en
"People's Republic of Kampuchea"@en
"Rauracian Republic"@en

http://dbpedia.org/sparql

Semantic Web

Возможности SPARQL 1.1

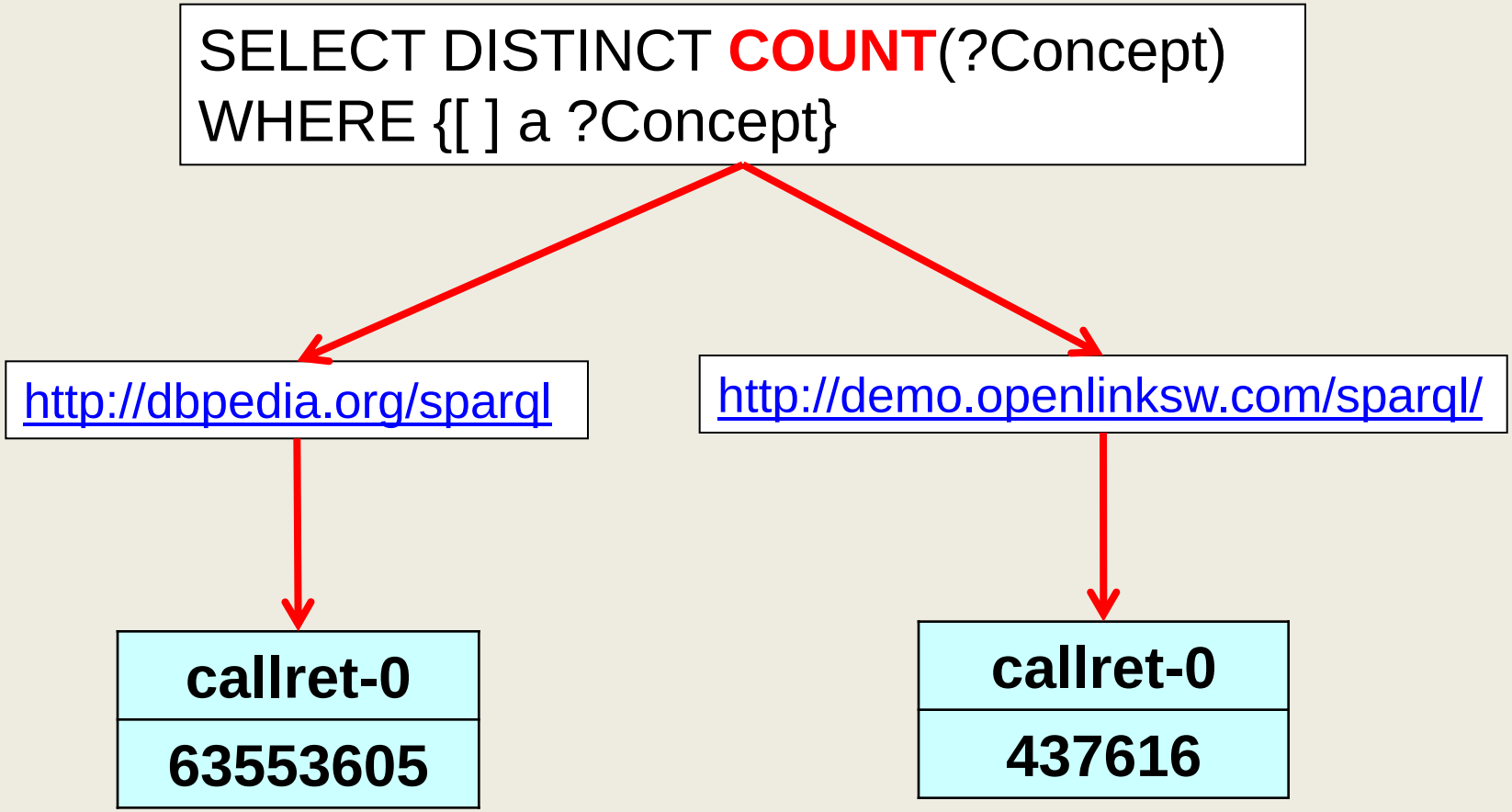
SPARQL Update позволяет делать следующие виды запросов:

- **INSERT DATA** { triples }
- **DELETE DATA** { triples }
- [DELETE { template }] [INSERT { template }] WHERE { pattern }
- **LOAD** uri [**INTO GRAPH** uri]
- **CLEAR GRAPH** uri
- **CREATE GRAPH** uri
- **DROP GRAPH** uri

Добавлены **агрегирующие** функции:

ADD
COUNT

Пример использования *COUNT*



Федеративные запросы

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?actor_name_en ?birth_date
FROM <http://www.w3.org/People/Berners-Lee/card> # placeholder graph
WHERE {
  SERVICE <http://data.linkedmdb.org/sparql> {
    <http://data.linkedmdb.org/resource/film/675>  movie:actor            ?actor .
    ?actor                                         movie:actor_name  ?actor_name
   }
  SERVICE <http://dbpedia.org/sparql> {
    ?actor2 a                       dbpedia:Actor ;
            foaf:name               ?actor_name_en ;
            dbpedia:birthDate       ?birth_date .
   }
}
LIMIT 20
```

http://sparql.org/sparql.html

Semantic Web

Федеративные запросы

actor_name_en	birth_date
"Adriana Chechik"@en	"1991-11-04"^^<http://www.w3.org/2001/XMLSchema#date>
"Chechik, Adriana"@en	"1991-11-04"^^<http://www.w3.org/2001/XMLSchema#date>
"Akihisa Shiono"@en	"1995-01-03"^^<http://www.w3.org/2001/XMLSchema#date>
"Shiono, Akihisa"@en	"1995-01-03"^^<http://www.w3.org/2001/XMLSchema#date>
"Atsushi Maruyama"@en	"1983-06-11"^^<http://www.w3.org/2001/XMLSchema#date>
"Maruyama, Atsushi"@en	"1983-06-11"^^<http://www.w3.org/2001/XMLSchema#date>
"Atsushi Maruyama"@en	"1993-06-11"^^<http://www.w3.org/2001/XMLSchema#date>
"Maruyama, Atsushi"@en	"1993-06-11"^^<http://www.w3.org/2001/XMLSchema#date>
"Ayaka Fukuhara"@en	"1989-12-31"^^<http://www.w3.org/2001/XMLSchema#date>
"Casey Calvert"@en	"1990-03-17"^^<http://www.w3.org/2001/XMLSchema#date>
"CoCo Brown"@en	"1978-09-16"^^<http://www.w3.org/2001/XMLSchema#date>
"Colby Keller"@en	"1980-10-18"^^<http://www.w3.org/2001/XMLSchema#date>
"Keller, Colby"@en	"1980-10-18"^^<http://www.w3.org/2001/XMLSchema#date>
"Hideya Tawada"@en	"1993-11-05"^^<http://www.w3.org/2001/XMLSchema#date>
"Tawada, Hideya"@en	"1993-11-05"^^<http://www.w3.org/2001/XMLSchema#date>
"Himika Akaneya"@en	"1994-07-16"^^<http://www.w3.org/2001/XMLSchema#date>
"Julian Gaertner"@en	"1985-12-08"^^<http://www.w3.org/2001/XMLSchema#date>
"Gaertner, Julian"@en	"1985-12-08"^^<http://www.w3.org/2001/XMLSchema#date>
"Yì Yǔháng"@en	"1985-12-08"^^<http://www.w3.org/2001/XMLSchema#date>
"jik6 jyu5 hong4"@en	"1985-12-08"^^<http://www.w3.org/2001/XMLSchema#date>

http://sparql.org/sparql.html

Semantic Web

Вложенные запросы

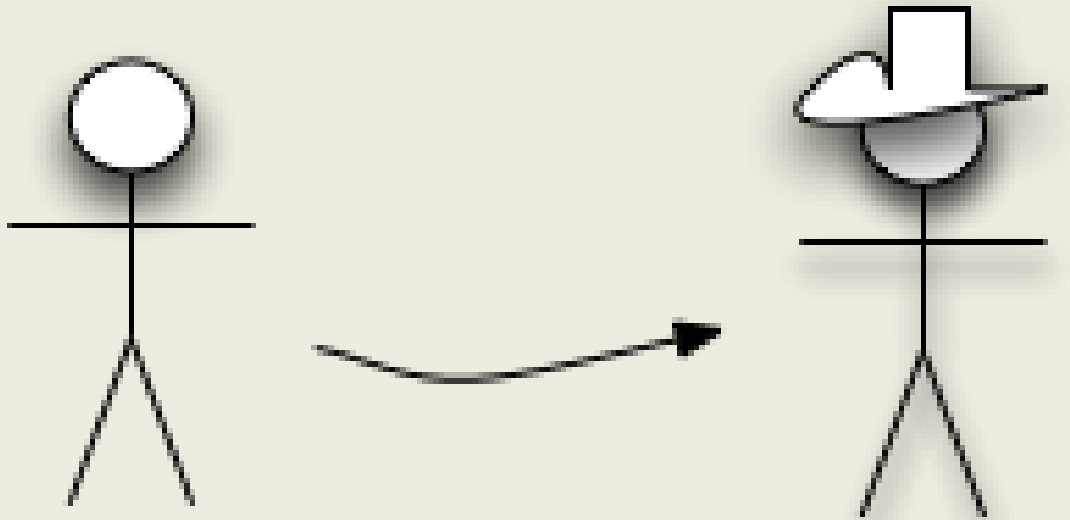
```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
   {     SELECT DISTINCT ?person ?name WHERE {
      ?person foaf:name ?name
    } ORDER BY ?name LIMIT 10 OFFSET 10    }
   OPTIONAL { ?person foaf:mbox ?email }
}
```

name	email
"Dave Beckett"	
"Dean Jackson"	<mailto:dino@grorg.org>
"Dean Jackson"	<mailto:dean@w3.org>
"Edd Dumbill"	<mailto:edd@xmlhack.com>
"Edd Dumbill"	<mailto:edd@xml.com>
"Edd Dumbill"	<mailto:edd@usefulinc.com>
"Eric Miller"	<mailto:em@w3.org>
"Henrik Nielsen"	
"Henry Story"	
"Håkon Wium Lie"	
"Ian Jacobs"	
"Ira Fuchs"	
"Ivan Herman"	

http://spargl.org/spargl.html

SQL и SPARQL

A walkthrough example illustrating the power of SPARQL



XML/SQL

RDF/SPARQL

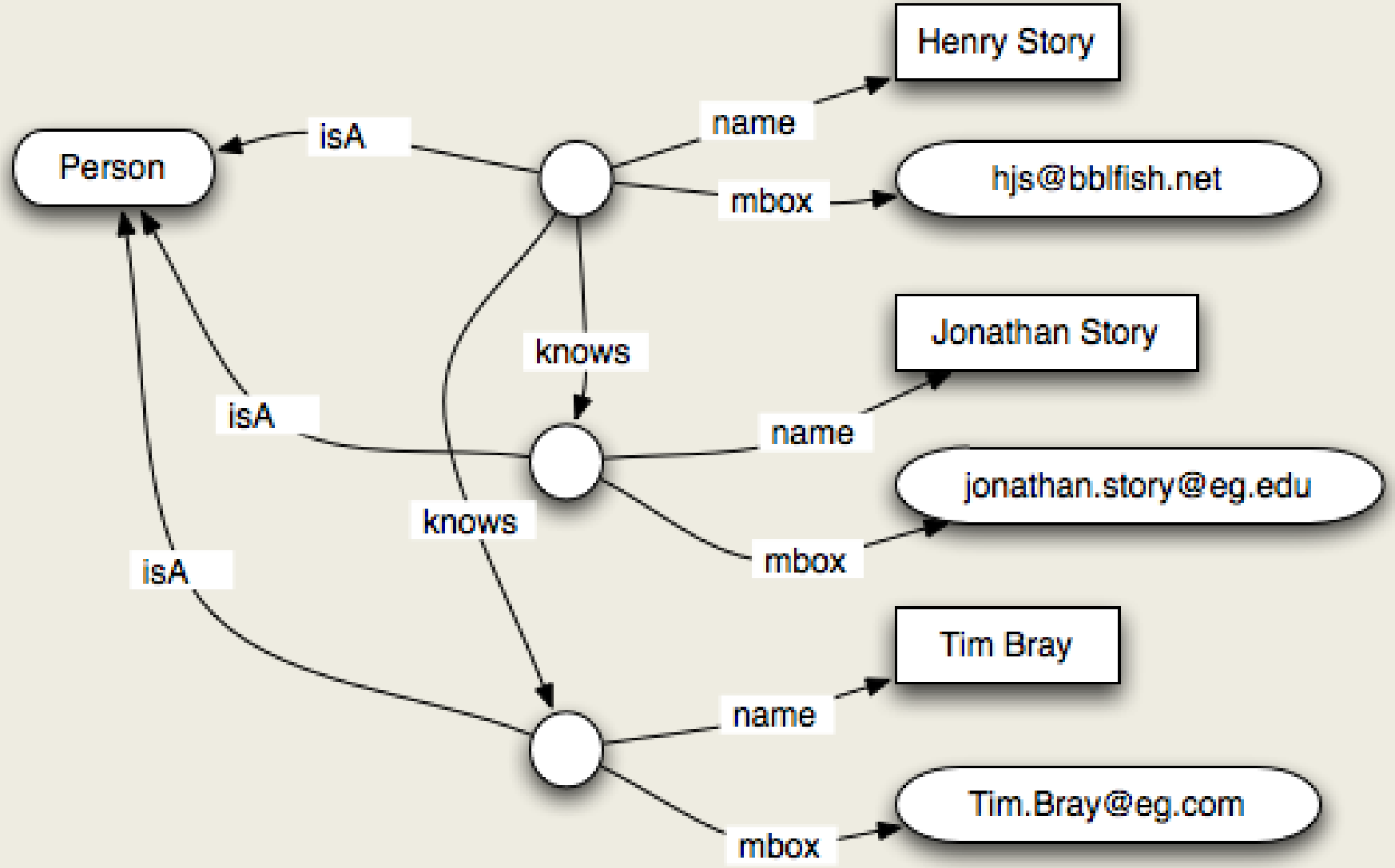
Пример 1: XML

```
<Person>
      <name>Henry Story</name>
      <mbox>hs@bblfish.net</mbox>
      <knows>
         <Person>
                  <name>Tim Bray</name>
                  <mbox>tb@eg.com</mbox>
         </Person>
         <Person>
                  <name>Jonathan Story</name>
                  <mbox>js@eg.edu</mbox>
         </Person>
     </knows>
</Person>
```

Пример 1: SPARQL

```
[  a    :Person;
   :name "Henry Story";
   :mbox <mailto:hs@insead.edu>;
        :knows [     a :Person;
                     :name "Tim Bray";
                     :mbox <mailto:tb@eg.com     ];
        :knows [ a :Person;
                     :name "Jonathan Story";
                     :mbox <mailto:js@eg.edu> ];
   ] .
```

Граф 1


Henry Story
name
isA
Person
hjs@bblfish.net
mbox
knows
Jonathan Story
isA
name
jonathan.story@eg.edu
knows
mbox
isA
Tim Bray
name
Tim.Bray@eg.com
mbox

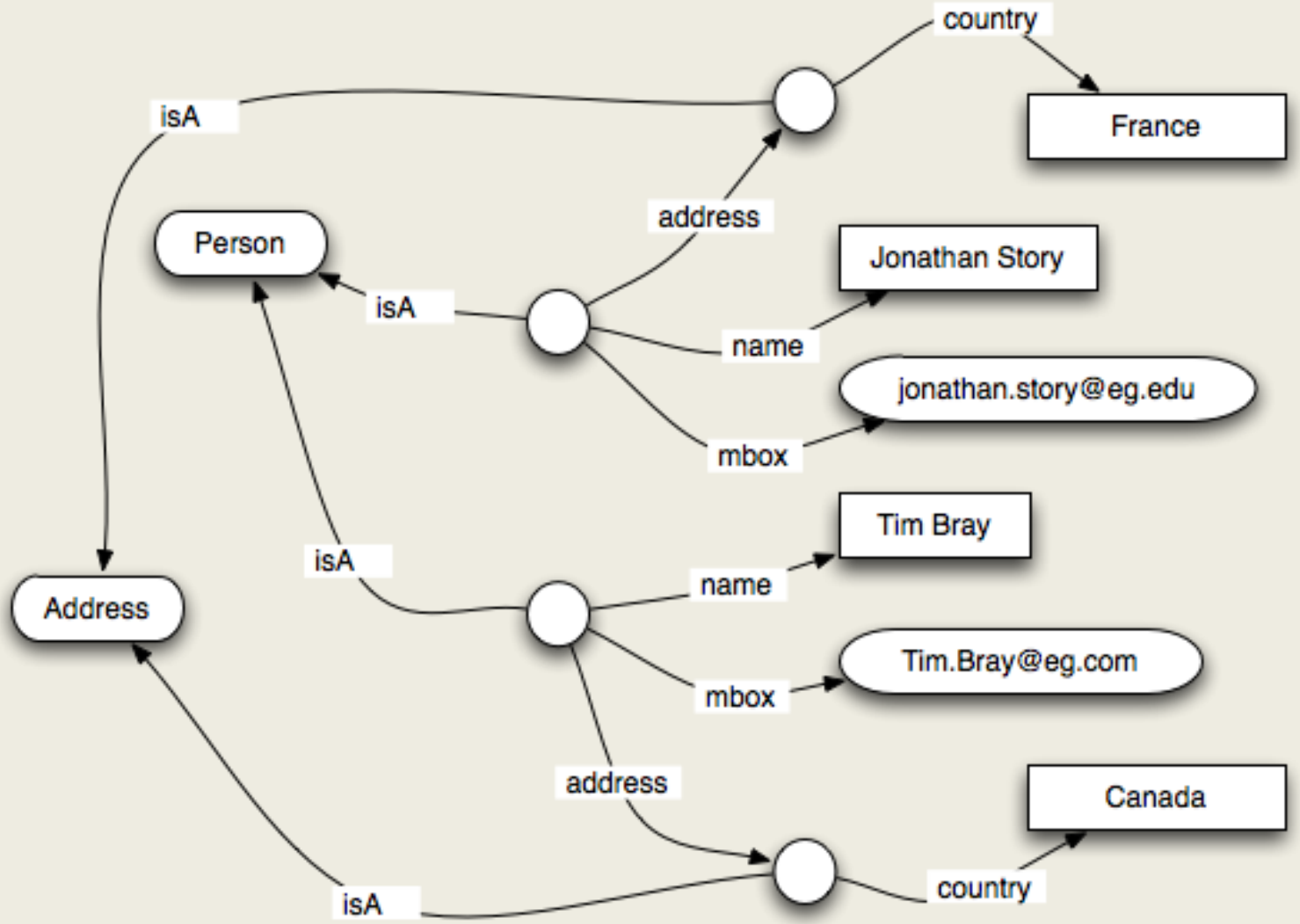
Пример 2: XML

```
<AddressBook>
    <Person>
        <name>Jonathan Story</name>
        <mbox>Jonathan.Story@eg.edu</mbox>
        <address>
            <Country>France</Country>
        </address>
    </Person>
    <Person>
        <name>Tim Bray</name>
        <mbox>Tim.Bray@eg.Com</mbox>
        <address>
            <Country>Canada</Country>
        </address>
    </Person>
</AddressBook>
```

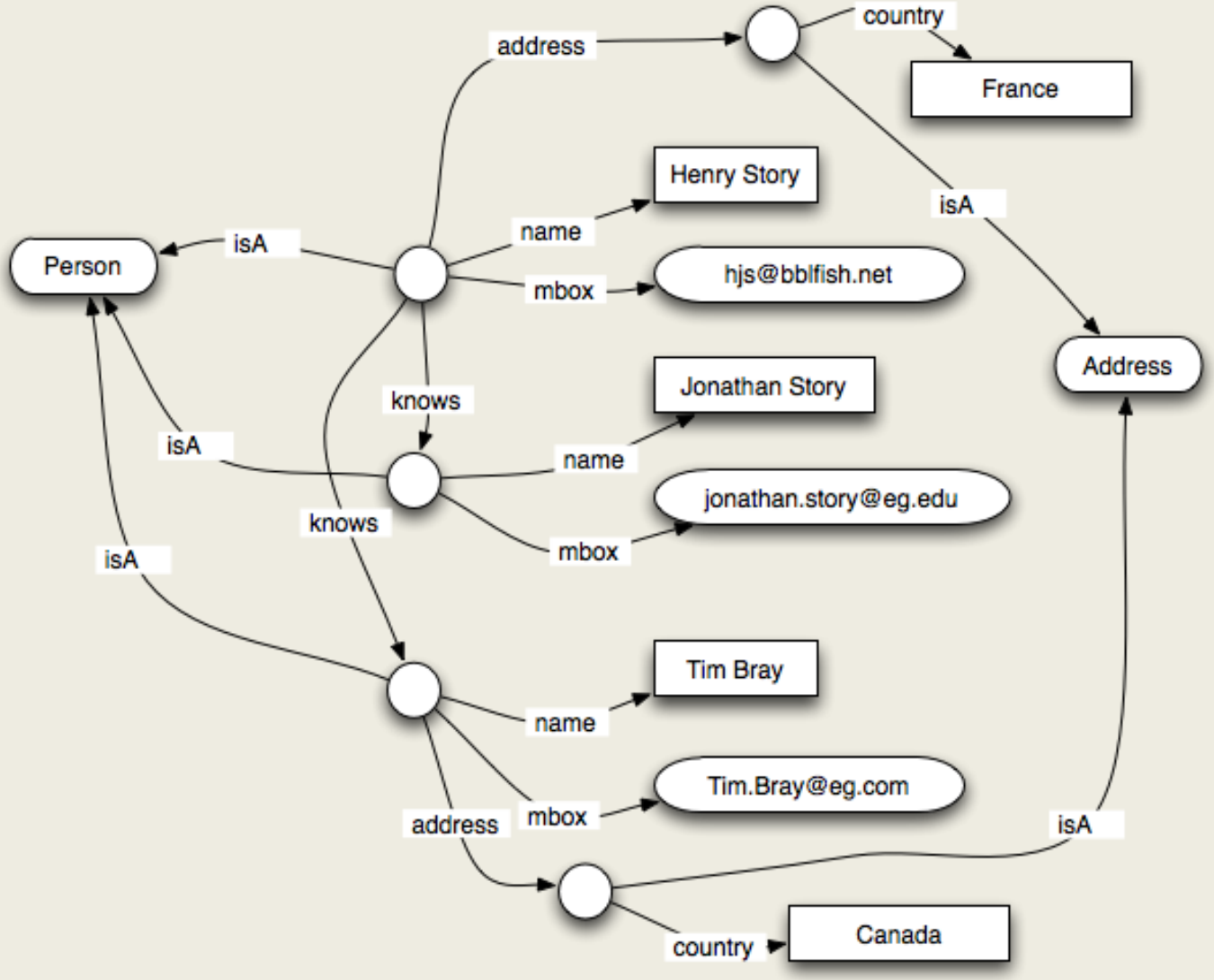
Пример 2: SPARQL

```
[       a :Person;
        :name "Tim Bray";
        :mbox <mailto:Tim.Bray@eg.com>
                :address [      a :Address;
                                :country "Canada"@en ]    ].
[       a :Person;
        :name "Jonathan Story";
        :mbox <mailto:Jonathan.Story@eg.edu>
        :address [
                a :Address;
                :country "France"@en ]
].
```

Граф 2


country
France
isA
address
Person
Jonathan Story
isA
name
jonathan.story@eg.edu
mbox
Tim Bray
isA
name
Address
Tim.Bray@eg.com
mbox
address
Canada
country
isA

Граф 1 + граф 2


address
country
France
isA
Henry Story
name
Person
isA
mbox
hjs@bblfish.net
Address
knows
Jonathan Story
isA
name
jonathan.story@eg.edu
knows
mbox
isA
Tim Bray
name
Tim.Bray@eg.com
address
mbox
isA
country
Canada

Вопрос

Кого знает Генри из Канады? Как его зовут и какой у него e-mail?

На этот вопрос можно ответить только проанализировав два ресурса.

Это невозможно на SQL/XML и просто на SPARQL/RDF.

Ответ

```
SELECT ?name ?mail
WHERE {
        [a :Person;
        :name "Henry Story";
        :knows [      :name ?name;
                      :mbox ?mail;
                      :address [   a :Address;
                                   :country "Canada"@en;
        ] ] ].
 }
```

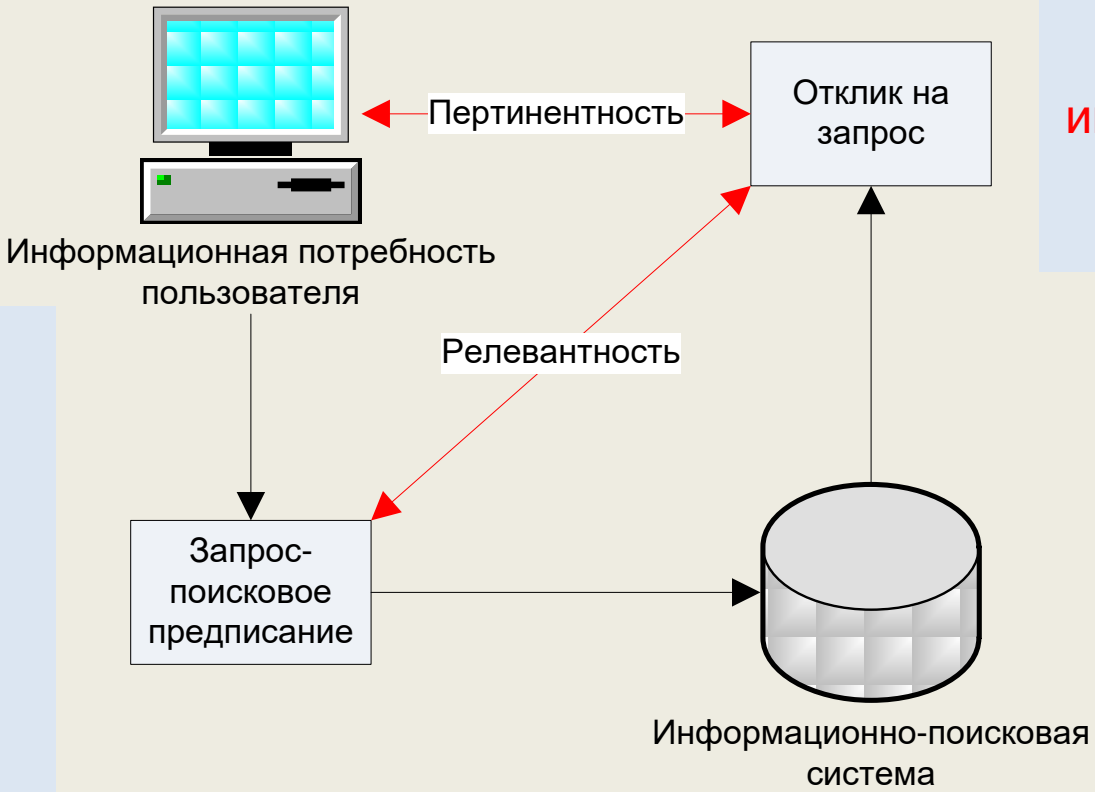
Домашнее задание

http://test.wikivote.ru/index.php/Мастер-класс_по_SPARQL

Задача: **Найдите страны с количеством жителей от 5 до 50 миллионов.**
Датасет: http://dbpedia.org/sparql
Количество жителей - это свойство http://dbpedia.org/property/populationEstimate
Название страны обозначается как http://www.w3.org/2000/01/rdf-schema#label
Страна - это объект со следующим URI: http://dbpedia.org/ontology/Country

Характеристики поиска

Соответствие полученной информации информационной потребности пользователя.



Отношение объема полезной для пользователя информации к общему объему полученной информации, найденной поисковой системой.

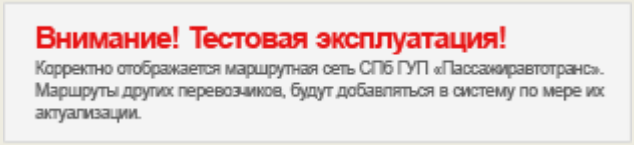
http://www.seonews.ru/analytics/pertinentnost-poiska-novyiy-trend-v-konkurentsii-poiskovyih-sistem/

API разработчика

https://te-st.ru/2014/02/18/open-data-sources-russia/

http://data.mosreg.ru/opendata SPARQL

http://transport.orgp.spb.ru/Portal/transport/main



http://vk.com/apiclub API позволяет создавать интересные, интерактивные и популярные приложения. Встроенные средства монетизации позволяют приносить доход разработчикам приложений, начиная с первых дней запуска.

https://topvisor.ru/api/

http://api.superjob.ru/

http://www.towave.ru/pub/kak-zastavit-besplatnyi-onlain-servis-prinosit-dengi.html