

# *ОСНОВЫ SPARQL*

# Интеграция данных

# SPARQL

**SPARQL – SPARQL Protocol And RDF Query Language**

Описывается спецификациями комитета W3C.

**Язык запросов SPARQL для RDF хранилищ данных**

**Рекомендация W3C, 15 января 2008**

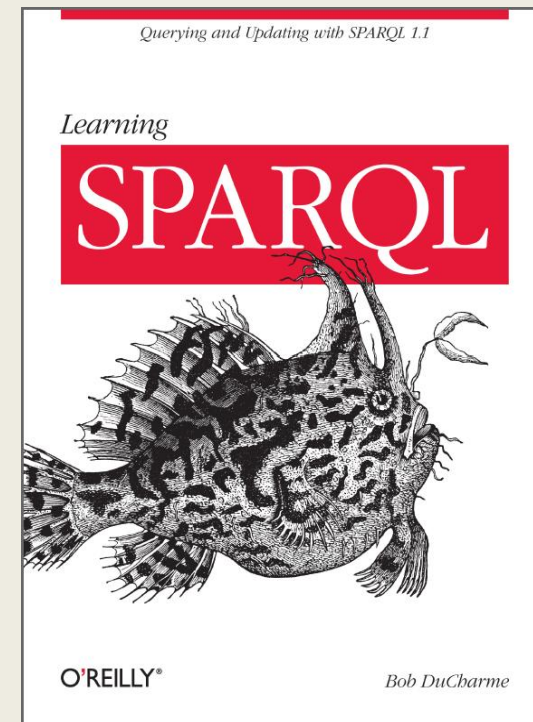
Текущая версия:

<http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>

Последняя версия:

<http://www.w3.org/TR/rdf-sparql11-query/>

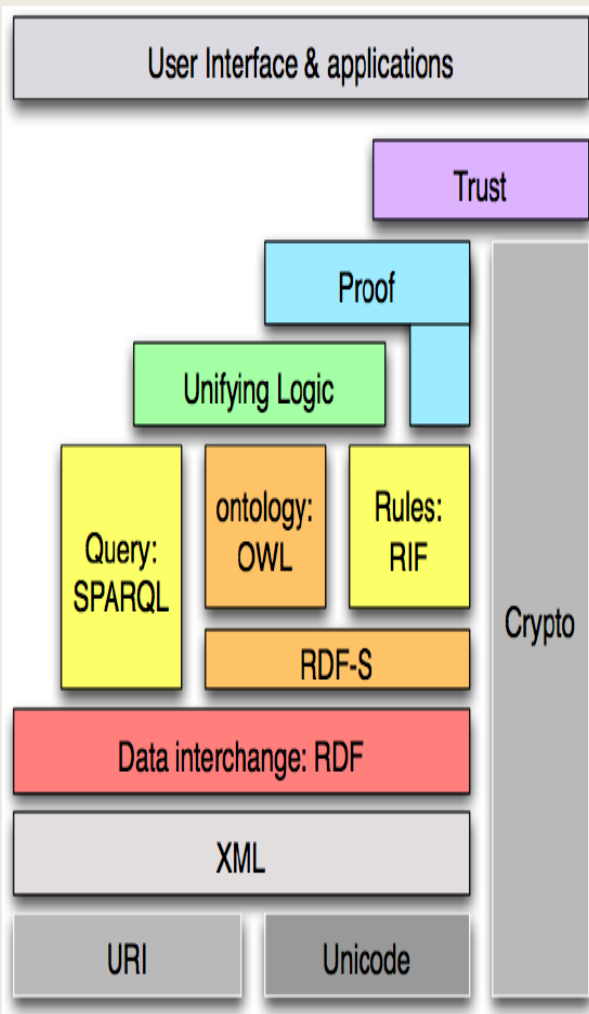
**Хранение информации является важнейшей составляющей современных информационных технологий**



(65 стр., © Губин А.Н., Филиппов Ф.В.  
© ФГОБУВПО СПб ГУТ  
им.проф. М.А. Бонч-Бруевича



**Хранение информации** является важнейшей составляющей современных информационных технологий



Форматы содержания результатов SPARQL  
- запросов



1. **SELECT** — возвращает все или часть переменных, указанных в шаблонах запроса.
2. **CONSTRUCT** — возвращает RDF граф, построенный путем замещения переменных в Construct-шаблоне запроса.
3. **DESCRIBE** — возвращает RDF граф, который описывает найденный результат.
4. **ASK** — возвращает true, если хоть что-то найдено, иначе — false.



# SPARQL

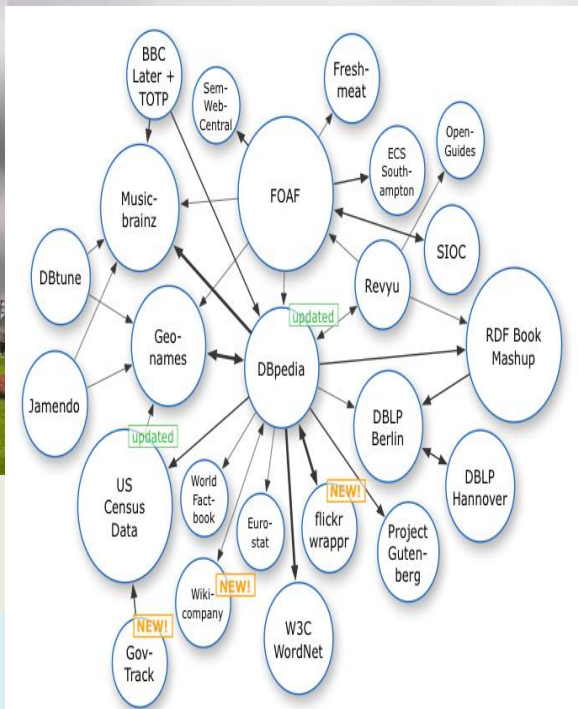


# SPARQL

Системы хранения данных

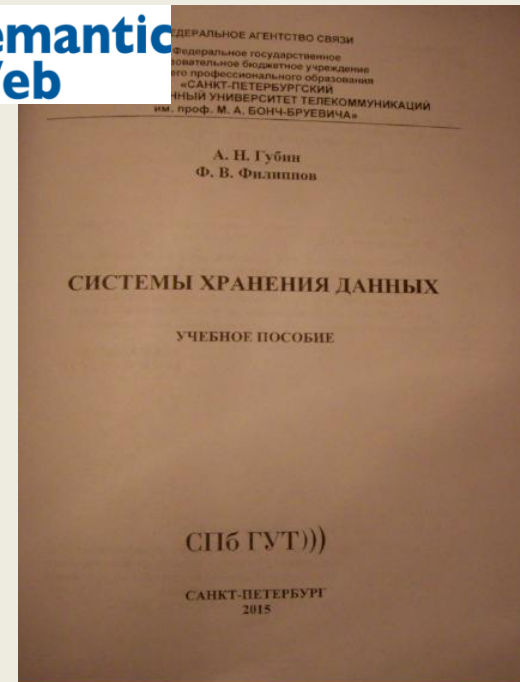
УЧЕБНОЕ ПОСОБИЕ

(65 стр., © Губин А.Н., Филиппов Ф.В.  
© ФГОБУВПО СПб ГУТ  
им.проф. М.А. Бонч-Бруевича



<http://www.sut.ru/>

ции является важнейшей составляющей  
к информационных технологий



# SPARQL



# SPARQL

Запрос CONSTRUCT возвращает граф RDF, заданный шаблоном графа. Этот результирующий граф формируется (конструируется) следующим образом: для каждого результата в последовательности результатов запроса выполняется подстановка переменных из шаблона графа, а затем все триплеты объединяются в единый граф RDF.

## Формирование результатов запроса

Если в результате такого преобразования получается триплет, содержащий несвязанную переменную или неправильную конструкцию RDF, например, литерал в позиции субъекта или предиката, такой триплет не включается в выходной граф RDF.



## Формирование результатов запроса



Шаблон графа может содержать триплеты без переменных, которые выводятся в выходном графе RDF без изменений.



CONSTRUCT извлекает информацию из точки доступа SPARQL в формате [RDF](#) и преобразовывает результаты к определенной форме.



## Синтаксис запроса CONSTRUCT



Шаблон графа может содержать триплеты без переменных, которые выводятся в выходном графе RDF без изменений.



CONSTRUCT извлекает информацию из точки доступа SPARQL в формате [RDF](#) и преобразовывает результаты к определенной форме.

# SPARQL

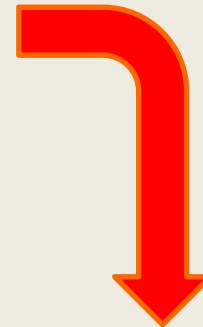
Вид операции	Функции и операторы	Пример
Date/time (SPARQL 1.1)	now, year, month, day, hours, minutes, seconds, timezone, tz	month(now()) < 4
SPARQL tests	isURI, isBlank, isLiteral, isNumeric, bound	isURI(?person)    !bound(?person)
Constructors (SPARQL 1.1)	URI, BNODE, STRDT, STRLANG, UUID, STRUUID	STRLANG(?text, "en") = "hello"@en
Accessors	str, lang, datatype	lang(?title) = "en"

Вид операции	Функции и операторы	Пример
Hashing (1.1)	MD5, SHA1, SHA256, SHA512	<b>BIND(SHA256(?email) AS ?hash)</b>
Miscellaneous	isURI, isBlank, isLiteral, isNumeric, bound	<b>regex(?ssn, “\\d{3}- \\d{2}-\\d{4}”)</b>

В выражениях можно использовать операции отношения: равно ("**=**"), **не равно** ("**!=**"), **больше** ("**>**"), **меньше** ("**<**"), **больше или равно** ("**>=**"), **меньше или равно** ("**<=**"), а также логические операции: **И** ("**&&**"), **ИЛИ** ("**||**") и **НЕ** ("**!**").

Для числовых литералов определены также **одноместные арифметические операции**: **плюс** ("**+**") и **минус** ("**-**"), а также **двуместные арифметические операции**: **сложение** ("**+**"), **вычитание** ("**-**"), **умножение** ("**\***") и **деление** ("**/**").

**В языке SPARQL определены встроенные функции следующих типов: (используются при конструировании фильтров)**



1. функции проверки;
2. функция преобразования;
3. функция определения типа данных;
4. языковые функции;
5. функции выполнения логических операций;
6. функция поиска с использованием регулярного выражения.

К функциям проверки относятся :

- bound()**,
- isIRI()**,
- isURI()**,
- isBlank()**,
- isLiteral()**,
- RDFterm-equal()**,
- sameTerm()**.

Функция  
**bound(переменная)**



*проверяет, является ли переменная, заданная в качестве аргумента функции, связанной. Если это условие выполняется, функция возвращает true, иначе возвращает false.*



# SPARQL

Данные:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@prefix dc:<http://purl.org/dc/elements/1.1/> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
_:a foaf:givenName "Alice".  
_:b foaf:givenName "Bob" .  
_:b dc:date "2005-04-04T04:04:04Z"^^xsd:dateTime .
```



```
SELECT ?givenName  
WHERE  
{  
  ?x foaf:givenName ?givenName .  
  OPTIONAL  
  { ?x dc:date ?date } .  
  FILTER ( bound(?date) )  
}
```



<b>givenName</b>
"Bob"

## Данные:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
_:a foaf:givenName "Alice".
```

```
_:b foaf:givenName "Bob" .
```

```
_:b dc:date "2005-04-04T04:04:04Z"^^xsd:dateTime .
```



```
SELECT ?givenName  
WHERE  
{  
  ?x foaf:givenName ?givenName .  
  OPTIONAL  
  { ?x dc:date ?date } .  
  FILTER ( !bound(?date) )  
}
```

**givenName**

**"Alice"**



Этот запрос выбирает людей с name, но без выраженной date:

**Функции**

***isIRI(аргумент)***

***isURI(аргумент)***



**Проверяют, является ли аргумент IRI(URI). Если это условие выполняется, функция возвращает *true*, иначе возвращает *false*.**

**В качестве аргумента могут быть заданы :**

***IRI(URI),***

***литерал,***

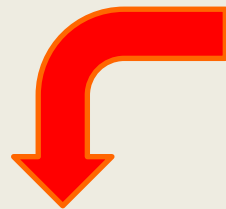
***переменная,***

***пустой узел.***

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Alice".  
_:a foaf:mbox <mailto:alice@work.example> .  
_:b foaf:name "Bob" .  
_:b foaf:mbox "bob@work.example" .
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name ?mbox  
WHERE  
{  
  ?x foaf:name ?name ;  
  foaf:mbox ?mbox .  
  FILTER isIRI(?mbox)  
}
```



name	mbox
"Alice"	<mailto:alice@work.example>

Функция  
***isBlank(аргумент)***



*проверяет, является ли аргумент пустым узлом.*

*Если это условие выполняется, функция возвращает true, иначе возвращает false.*

*В качестве аргумента могут быть заданы IRI(URI), литерал, переменная или пустой узел.*

```
@prefix a: <http://www.w3.org/2000/10/annotation-ns#> .  
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
  
_:a a:annotates <http://www.w3.org/TR/rdf-sparql-query/> .  
_:a dc:creator "Alice B. Toeclops" .  
_:b a:annotates <http://www.w3.org/TR/rdf-sparql-query/> .  
_:b dc:creator _:c .  
_:c foaf:given "Bob" .  
_:c foaf:family "Smith" .
```

```
SELECT ?given ?family  
WHERE  
{  
  ?annot a:annotates <http://www.w3.org/TR/rdf-sparql-query/> .  
  ?annot dc:creator ?c .  
  OPTIONAL { ?c foaf:given ?given ; foaf:family ?family } .  
  FILTER isBlank(?c)
```

given	family
"Bob"	"Smith"

Функция `isLiteral(аргумент)`



проверяет, является ли аргумент литералом.

Если это условие выполняется, функция возвращает `true`, иначе возвращает `false`.  
В качестве аргумента могут быть заданы IRI(URI), литерал, переменная, пустой узел.



```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
_:a foaf:name "Alice".
```

```
_:a foaf:mbox <mailto:alice@work.example> .
```

```
_:b foaf:name "Bob" .
```

```
_:b foaf:mbox "bob@work.example" .
```



```
SELECT ?name ?mbox  
WHERE  
{  
  ?x foaf:name ?name ;  
  foaf:mbox ?mbox .  
  FILTER isLiteral(?mbox)  
}
```



name	mbox
"Bob"	"bob@work.example"

Функция `str(аргумент)`



возвращает строковое представление аргумента.

В качестве аргумента могут быть заданы литерал, переменная, IRI(URI).

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
_:a foaf:name "Alice".
_:a foaf:mbox <mailto:alice@work.example> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@home.example> .
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
{
  ?x foaf:name ?name ;
  foaf:mbox ?mbox .
  FILTER regex(str(?mbox),
"@work.example")
}
```



name	mbox
"Alice"	<mailto:alice@work.example>

Функция `lang(аргумент)`



Возвращает тег языка (language tag) для `ltrl` при наличии такового.  
Если тег языка для `ltrl` отсутствует, возвращается ""

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Robert"@EN.  
_:a foaf:name "Roberto"@ES.  
_:a foaf:mbox <mailto:bob@work.example> .
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name ?mbox  
WHERE  
{  
  ?x foaf:name ?name ;  
  foaf:mbox ?mbox .  
  FILTER ( lang(?name) = "ES" )  
}
```



name	mbox
"Roberto"@ES	<mailto:bob@work.example>

Функция **datatype(аргумент)**



Возвращает datatype IRI для typedLit;  
возвращает xsd:string, если параметр  
является простым (simple) литералом.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@prefix eg: <http://biometrics.example/ns#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>  
_:a foaf:name "Alice".  
_:a eg:shoeSize "9.5"^^xsd:float .  
_:b foaf:name "Bob".  
_:b eg:shoeSize "42"^^xsd:integer .
```



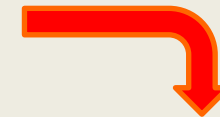
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  
PREFIX eg: <http://biometrics.example/ns#>  
SELECT ?name ?shoeSize  
WHERE  
{  
  ?x foaf:name ?name ;  
  eg:shoeSize ?shoeSize .  
  FILTER ( datatype(?shoeSize) = xsd:integer )  
}
```



name	shoeSize
"Bob"	42



Функция **RDFterm-equal**(*аргумент-1*, *аргумент-2*)



Возвращает *возвращает true*, если оба аргумента являются эквивалентными, иначе *возвращает false*.

*В качестве аргументов могут быть заданы литерал, переменная, пустой узел, IRI(URI).*

*Два литерала являются эквивалентными при выполнении следующих условий: они посимвольно равны; у обоих либо есть, либо нет меток языков; если метки языков есть, то они одинаковые; у обоих либо есть, либо нет типов данных.*

*Два пустых узла считаются эквивалентными, если они указывают на один и тот же узел.*

*Два IRI(URI) считаются эквивалентными, если они посимвольно равны.*

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Alice".  
_:a foaf:mbox <mailto:alice@work.example> .  
_:b foaf:name "Ms A.".  
_:b foaf:mbox <mailto:alice@work.example> .
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
SELECT ?name1 ?name2  
WHERE  
{  
  ?x foaf:name ?name1 ;  
  foaf:mbox ?mbox1 .  
  ?y foaf:name ?name2 ;  
  foaf:mbox ?mbox2 .  
  FILTER (?mbox1 = ?mbox2 && ?name1 != ?name2)  
}
```

name1	name2
"Alice"	"Ms A."
"Ms A."	"Alice"

Функция **datatype(аргумент)**



Возвращает datatype IRI для typedLit;  
возвращает xsd:string, если параметр  
является простым (simple) литералом.

# SPARQL

*проверяет, является ли аргумент литералом. Если это условие выполняется, функция возвращает true, иначе возвращает false. В качестве аргумента могут быть заданы IRI(URI), литерал, переменная или пустой узел.*

*Функция RDFterm-equal(аргумент-1, аргумент-2) возвращает true, если оба аргумента являются эквивалентными, иначе возвращает false. В качестве аргументов могут быть заданы литерал, переменная, пустой узел или IRI(URI). Два литерала являются эквивалентными при выполнении следующих условий: они посимвольно равны; у обоих либо есть, либо нет меток языков; если метки языков есть, то они одинаковые; у обоих либо есть, либо нет типов данных. Два пустых узла считаются эквивалентными, если они указывают на один и тот же узел. Два IRI(URI) считаются эквивалентными, если они посимвольно равны.*

Функция `sameTerm(аргумент-1, аргумент-2)` возвращает `true`, если оба аргумента являются одним и тем же данным, иначе возвращает `false`. В качестве аргументов могут быть заданы литерал, переменная, пустой узел или `IRI(URI)`.

Функция преобразования `str(аргумент)` возвращает строковое представление аргумента. В качестве аргумента могут быть заданы литерал, переменная или `IRI(URI)`.

Функция определения типа `datatype(аргумент)` возвращает уточненный тип данных схемы XML для аргумента. Если аргумент задает простое (нетипизированное) данные, возвращается тип данных `xsd:string`. В качестве аргумента могут быть заданы простой или типизированный литерал, а также переменная.

К языковым функциям относятся функции `lang()` и `langMatches()`.

Функция `lang(аргумент)` возвращает строку кода языка аргумента. Если код языка для аргумента не задан, возвращается пустая строка. В качестве аргумента могут быть заданы литерал или переменная.

Функция `langMatches(метка-языка, диапазон-меток)` возвращает `true`, если *метка-языка* входит в *диапазон-меток*, заданный во втором аргументе, иначе возвращает `false`. Аргумент *диапазон-меток* содержит либо список меток языков, отделенных друг от друга символом "-", либо символ "\*".

Логические функции `logical-or()` и `logical-and()` выполняют логические операции над аргументами.

Функция `logical-or(аргумент-1, аргумент-2)` возвращает результат выполнения операции *аргумент-1* || *аргумент-2* (`true` или `false`). Оба аргумента должны иметь булевский тип.

Функция `logical-and(аргумент-1, аргумент-2)` возвращает результат выполнения операции *аргумент-1* && *аргумент-2* (`true` или `false`). Оба аргумента должны иметь булевский тип.



Функция поиска с использованием регулярного выражения `regex(строка, шаблон, флажки)` вызывает функцию `fn:matches()` языка XPath 2.0 (см. 5.1.1.4.3.5). В аргументе строка задается строковый литерал или строковая переменная, а в аргументе шаблон – шаблон поиска в строке (в соответствии с правилами задания регулярных выражений). В необязательном аргументе задаются флажки "s", "m", "i" и "x" (один флажок или строка, содержащая комбинацию флажков). Функция возвращает `true`, если в строке найдено соответствие шаблону и `false` – в противном случае.

# SPARQL

```
PREFIX dc: http://purl.org/dc/elements/1.1/  
PREFIX : <http://www.libRegistry.org/ns/lib#>  
SELECT ?bookAuthor ?bookTitle  
WHERE {  
  ?book dc:title ?bookTitle;  
  dc:creator ?bookAuthor .  
  FILTER (regex(str(?bookAuthor), "Defoe") ||  
  regex(str(?bookAuthor), "Дефо"))  
}
```

bookAuthor	bookTitle
"Дефо Даниэл"@ru	"Приключения Робинзона Крузо"@ru
"Defoe Daniel"@en	"Robinson Crusoe"@en

# SPARQL

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>  
PREFIX : <http://www.libRegistry.org/ns/lib#>  
SELECT ?bookAuthor ?bookTitle  
WHERE {  
  ?book dc:title ?bookTitle;  
  dc:creator ?bookAuthor .  
  FILTER (lang(?bookTitle) = "uk")
```

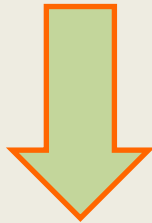
bookAuthor	bookTitle
"Шевченко Тарас"@uk	"Кобзар"@uk

```
SELECT ?bookAuthor ?bookTitle ?bookPubDate
WHERE {
  ?book dc:title ?bookTitle;
  dc:creator ?bookAuthor;
  dc:publisher ?bookPublisher;
  dc:date ?bookPubDate .
  FILTER (?bookPublisher = "Эксмо"@ru &&
  ?bookPubDate > "2005-01-01"^^xsd:date)
}
```

```
-----
| bookAuthor          | bookTitle                                     | bookPubDate          |
=====
| "Дефо Даниэл"@ru  | "Приключения Робинзона Крузо"@ru           | "2007-03-28"^^xsd:date |
-----
```

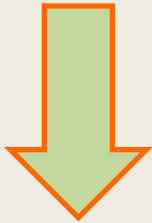


Проверка даты: Анализируемые даты должны находиться между 1 января. 1-й (включительно), 2015 и 1 января. 2016 годов



```
FILTER("2015-01-01"^^xsd:dateTime <= ?dob  
&& ?dob < "2016-01-01"^^xsd:dateTime).
```

**FILTER on values in Labels**



```
SELECT ?human ?humanLabel
WHERE
{
?human wdt:P31 wd:Q15632617; # fictional human
rdfs:label ?humanLabel.
FILTER(LANG(?humanLabel) = "en").
FILTER(STRSTARTS(?humanLabel, "Mr. ")).
}
```

**FILTER NOT EXIST**



```
SELECT ?human ?humanLabel ?image
WHERE
{
  ?human wdt:P31 wd:Q15632617; # fictional human
  rdfs:label ?humanLabel.
  FILTER(LANG(?humanLabel) = "en").
  FILTER(STRSTARTS(?humanLabel, "Mr. ")).
  FILTER NOT EXISTS{ ?human wdt:P18 ?image.}
} # without images
```

MINUS



```
SELECT ?human ?humanLabel ?image
WHERE
{ ?human wdt:P31 wd:Q15632617; # fictional human
rdfs:label ?humanLabel.
FILTER(LANG(?humanLabel) = "en").
FILTER(STRSTARTS(?humanLabel, "Mr. ")).
MINUS{ ?human wdt:P18 ?image. } # without images
}
```



СРАВНЕНИЕ  
ТЕКСТОВЫХ СТРОК



```
SELECT ?workflow
WHERE
{
?workflow rdf:type mecontrib:Workflow ;
mebase:has-content-type ?ct .
?ct dcterms:title ?ct_title
FILTER regex(?ct_title,'^taverna','i')
}
```

Сравнивается значение `?ct_title` с текстовой строкой `'taverna'`. Знак **caret ( ^ )** используется, чтобы указать, что строка для `?ct_title` должен начинаться с `'taverna'`, а не просто иметь его где-то внутри строки.

**СРАВНЕНИЕ  
ТЕКСТОВЫХ СТРОК**

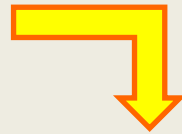


Анализируется значение **результата преобразования ресурса в текстовую строку** на **вхождение в эту строку литерала «Group»**

```
SELECT ?membership ?requester  
WHERE  
{  
?membership rdf:type mebase:Membership ;  
mebase:has-requester ?requester ;  
mebase:has-accepter ?accepter ;  
mebase:accepted-at ?accepted_at  
FILTER regex(str(?requester),'Group','i')  
}
```



ОПЕРАЦИИ С  
ЧИСЛАМИ



```
SELECT DISTINCT ?workflow ?ct_title
WHERE
{
?workflow rdf:type mecontrib:Workflow ;
mebase:has-content-type ?ct .
?ct dcterms:title ?ct_title .
?rating rdf:type meannot:Rating ;
mebase:annotates ?workflow;
meannot:rating-score ?score
FILTER (?score >= 4)
}
```

Выдача всех  
рабочих  
процессов,  
которые имеют  
рейтинг, равный  
4 или больше:

## ФИЛЬТРАЦИЯ ПО НЕСКОЛЬКИМ КРИТЕРИЯМ



```
SELECT DISTINCT ?workflow ?ct_title
WHERE
{
?workflow rdf:type mecontrib:Workflow ;
mebase:has-content-type ?ct .
?ct dcterms:title ?ct_title .
?rating rdf:type meannot:Rating ;
mebase:annotates ?workflow ;
meannot:rating-score ?score
FILTER (?score >= 4 &&
regex(?ct_title, '^Taverna 1'))
}
```

Выдача всех рабочих процессов, которые имеют рейтинг, равный 4 или больше и имеют наименование Taverna 1

**ФИЛЬТРАЦИЯ ПО  
ДАТЕ ОБНОВЛЕНИЯ  
ДАнных**



```
SELECT ?workflow ?added
WHERE
{
?workflow rdf:type mecontrib:Workflow ;
dcterms:created ?added
FILTER ( ?added >= xsd:dateTime('2009-09-01T00:00:00Z') )
}
```

Выдача всех рабочих процессов,, которые были добавлены с начала сентября 2009 года: