Научная статья

УДК 004.89:004.4

DOI 10.25205/1818-7900-2021-19-4-85-95

Разработка интеллектуального редактора SPARQL-запросов

Ирина Алексеевна Турова ¹
Игорь Сергеевич Постановов ²

^{1,2} Пермский государственный национальный исследовательский университет
Пермь, Россия

¹ turovaia@yandex.ru

² ipostanogov@outlook.com, <https://orcid.org/0000-0002-7791-4789>

Аннотация

Обсуждается вопрос создания редактора SPARQL-запросов – запросов к онтологиям в форматах RDF и OWL. Потребность в удобном и функциональном инструменте для составления SPARQL-запросов постоянно растет в связи с распространением концепции семантического веба. Приводится анализ возможностей, а также общих подходов к реализации существующих редакторов SPARQL-запросов. Предлагается концепция инструмента, одновременно предоставляющего интеллектуальные дополняющие подсказки, а также возможности построения визуального представления запроса и сравнения результатов выполнения запросов с использованием различных программ-ризонеров и без них. Языковая поддержка реализуется в соответствии с современным подходом к предоставлению инструментальной поддержки языков программирования, основанного на протоколе LSP. Приводятся скриншоты работы редактора.

Ключевые слова

SPARQL-запросы, онтологии, OWL, языковой сервер, LSP, ризонер, визуальное представление запроса

Для цитирования

Тулова И. А., Постановов И. С. Разработка интеллектуального редактора SPARQL-запросов // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 85–95. DOI 10.25205/1818-7900-2021-19-4-85-95

Development of Intelligent SPARQL Query Editor

Irina A. Turova ¹, Igor S. Postanogov ²

^{1,2} Perm State University
Perm, Russian Federation

¹ turovaia@yandex.ru

² ipostanogov@outlook.com, <https://orcid.org/0000-0002-7791-4789>

Abstract

The paper discusses the development of SPARQL query editor. This is an actual question because of the growth of Semantic Web data – the data presented in RDF/OWL formats. A comparative analysis of different types of editors and their main features is provided. In this paper, we propose a description of a SPARQL editor that combines three most useful features: intelligent completions, query visualization comparison of query results provided by different reasoners or without them. The editor provides SPARQL support as LSP service, this approach is considered a modern way to implement language support. This paper also presents the editor's screenshots.

Keywords

SPARQL queries, ontologies, OWL, language server, LSP, reasoner, query visualization

© Тулова И. А., Постановов И. С., 2021

For citation

Turova I. A., Postanogov I. S. Development of Intelligent SPARQL Query Editor. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 85–95. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-85-95

Введение

В настоящее время активно развиваются технологии искусственного интеллекта. Создаются интеллектуальные бизнес-решения, также интеллектуальные системы широко используются в образовательных целях. Технологии искусственного интеллекта являются одним из приоритетных направлений развития государства¹. Востребованы как методы машинного обучения, так и интеллектуальные системы, основанные на знаниях.

Большую популярность в настоящее время приобрели системы, основанные на онтологиях, – одной из форм представления знаний. Microsoft использует онтологически управляемые решения для создания цифровых двойников умных городов². Онтологии активно используются в медицине: многие системы поддержки принятия врачебных решений основываются на онтологиях [1].

Официальным стандартом для представления онтологий является формат OWL 2, который рекомендован консорциумом Всемирной паутины. SPARQL – язык запросов к онтологиям. Для работы со знаниями инженерам необходимо составлять запросы на языке SPARQL, позволяющие извлекать факты из онтологий, проверять истинность некоторых гипотез, добавлять новые факты в онтологию.

Статистические исследования [2; 3] SPARQL-запросов к разным точкам доступа SPARQL, в которых пользовательские запросы рассматриваются отдельно от программных, показывают, что пользовательские запросы структурно более сложные, чем программные. В работе рассматриваются возможности различных редакторов SPARQL-запросов, а также формулируются требования к собственному редактору SPARQL-запросов для удовлетворения максимального количества потребностей пользователей.

Обзор редакторов SPARQL-запросов

За последние десятилетия многие исследования были посвящены разработке различных редакторов для составления SPARQL-запросов. Наибольшее количество публикаций описывает визуальные редакторы, которые позволяют пользователям составлять запросы, добавляя на рабочую область графические элементы, соответствующие различным частям запроса.

Многие авторы, разрабатывая подобные редакторы, используют аналогичные по назначению определенным в SPARQL графические блоки, а не придумывают визуальный язык с нуля. Однако набор поддерживаемых возможностей обычно значительно меньше, чем в SPARQL. Например, редактор RDF Explorer [4] поддерживает составление запросов, состоящих только из основного графового шаблона запроса³, где концепты могут быть заменены переменными. Авторы работы позиционируют RDF Explorer как редактор для пользователей, не являющихся экспертами.

Поскольку обычно визуальные редакторы SPARQL-запросов значительно ограничены в своей выразительности, более сложные запросы не могут быть сформулированы с их помощью. Хотя такие редакторы и подходят для начинающих пользователей, могут возникнуть трудности при постепенном изучении конструкций языка, поскольку пользователь в процес-

¹ Указ Президента РФ от 10 октября 2019 г. № 490 «О развитии искусственного интеллекта в Российской Федерации»

² URL: <https://techcommunity.microsoft.com/t5/internet-of-things/smart-cities-ontology-for-digital-twins/ba-p/2166585>

³ URL: <https://www.w3.org/TR/rdf-sparql-query/#BasicGraphPatterns>

се обучения столкнется с необходимостью либо искать более выразительный визуальный интерфейс, либо переходить на текстовый язык запросов. Некоторые инструменты, например SPARQLinG [5] и NITELIGHT [6], по своей выразительности близки к полной спецификации SPARQL 1.0, но даже небольшие визуальные запросы выглядят громоздко, что затрудняет их использование начинающими пользователями, а для более продвинутых пользователей составление текстового запроса может оказаться предпочтительней.

Текстовые редакторы SPARQL-запросов не ограничивают пользователя в выразительности языка. Для начинающих пользователей могут быть реализованы дополняющие подсказки, которые упростят процесс составления запросов. Продвинутые пользователи смогут использовать любые возможности языка, даже если для некоторых из них не будет реализовано активной поддержки.

Большинство существующих текстовых редакторов, например YASGUI [7; 8] и основанный на нем Flint SPARQL-editor⁴, предоставляют базовую поддержку при составлении SPARQL-запросов: подсветку и валидацию синтаксиса запроса. Помимо подсказок ключевых слов дополняющие подсказки обычно составляются на основе всех концептов и отношений онтологий. Редактор SparQLed [9] основан на редакторе Flint SPARQL-editor и предоставляет дополняющие подсказки на основе содержимого онтологии, положения вызова подсказки в запросе, а также уже присутствующих в запросе триплетов, но в настоящее время проект закрыт.

Многие инструменты для работы с онтологиями также позволяют составить и выполнить SPARQL-запрос к онтологиям, однако зачастую эти возможности достаточно ограничены. Среди таких инструментов можно выделить Protégé – самый популярный и широко используемый редактор онтологий (около 300 000 пользователей⁵), который обладает богатым, расширяемым за счет разработки и подключения плагинов набором функциональных возможностей. Protégé также имеет возможность составления SPARQL-запросов, что позволяет работать с онтологиями и SPARQL-запросами в единой интегрированной среде. Однако редактор SPARQL-запросов, предоставляемый Protégé, не реализует никакой поддержки пользователей в процессе составления запросов и представляет собой простое текстовое поле, что затрудняет разработку запросов.

Stardog Studio⁶ – IDE для работы с графами знаний (в том числе и онтологиями) и SPARQL-запросами к ним. Stardog Studio имеет возможность составления SPARQL-запросов и предоставляет поддержку пользователей на этапе составления запросов – присутствует подсветка синтаксиса, валидация запроса с учетом синтаксиса и дополнение запроса ключевыми словами SPARQL и ранее использованными в запросе токенами (переменными, названиями элементов онтологии). Иначе говоря, при формировании подсказок не используются знания из целевой онтологии. В отличие от приведенных выше текстовых редакторов поддержка SPARQL выделена в модуль, отдельный от Stardog Studio, но поставляется вместе с ней. Этот модуль представляет собой языковой сервер, реализующий все функции для поддержки SPARQL и взаимодействующий с клиентом (Stardog Studio) по протоколу LSP⁷ (Language Server Protocol).

Помимо концептов и отношений онтологии могут содержать аксиомы, которые могут использоваться для вывода новых фактов на основе присутствующих в онтологии. Ризонер – программное средство для вывода новых фактов из онтологии на основе аксиом. Ризонеры различаются набором поддерживаемых аксиом. Удобной возможностью для редактора SPARQL-запросов может стать сравнение результатов выполнения запросов к одной онтологии с использованием нескольких ризонеров. Существуют работы [10; 11], описывающие системы для автоматизированного сравнения показателей работы нескольких ризонеров

⁴ URL: <https://github.com/TSO-Openup/FlintSparqlEditor/>.

⁵ URL: <https://protege.stanford.edu/community.php>

⁶ URL: <https://www.stardog.com/studio/>.

⁷ URL: <https://microsoft.github.io/language-server-protocol/>.

на пользовательских онтологиях, но ни один из рассмотренных редакторов SPARQL-запросов не предоставляет такой возможности.

Среди рассмотренных редакторов не было найдено ни одного инструмента, одновременно предоставляющего возможности составления текстового SPARQL-запроса, просмотра визуального представления запроса и сравнения результатов выполнения запроса с разными ризонерами. Визуальные редакторы SPARQLinG и NITELIGHT предоставляют возможность конвертации визуального запроса в SPARQL, однако текстовый вариант запроса доступен только для чтения.

Концепция редактора SPARQL-запросов

На основании обзора рассмотренных ранее редакторов было решено разработать собственный текстовый редактор SPARQL-запросов, который должен иметь следующие возможности:

- дополняющие подсказки на основе контекста вызова и содержимого онтологии;
- визуальное представление запроса с возможностью исследования связанных элементов онтологии;
- сравнение результатов выполнения запроса к одной и той же онтологии с использованием различных ризонеров.

Ключевой особенностью разрабатываемого решения является наличие всех этих возможностей в одном инструменте. Кроме того, эти возможности будут вынесены в отдельные модули, которые в дальнейшем могут быть легко переиспользованы. Каждый модуль будет реализован в виде сервиса, клиентом для этих сервисов будет выступать расширение для редактора Visual Studio Code⁸, в интерфейсе которого пользователю будет предложено работать со SPARQL-запросами.

Далее будут подробнее описаны особенности реализации модуля языкового сервера и правила составления дополняющих подсказок, а также модули визуального представления и сравнения ризонеров. В качестве сквозного примера рассматривается SPARQL-запрос к фрагменту географических данных онтологии Dbpedia (данные трех стран о столицах и официальных языках и восьми городов) для получения всех городов стран, в которых официальным языком является французский (рис. 1).

```
1  # sparql : http://localhost:3030/countries\_ds
2  PREFIX countries: <http://www.semanticweb.org/irina/ontologies/2021/5/countries#>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT ?city
6  WHERE {
7    ?country a countries:Country .
8    ?city a countries:City .
9    ?country countries:city ?city .
10   ?lang a countries:Language .
11   ?lang rdfs:label "French"@en .
12   ?country countries:officialLanguage ?lang .
13 }
14
```

Рис. 1. Пример SPARQL-запроса

Fig. 1. Sample SPARQL query

⁸ URL: <https://code.visualstudio.com>

Языковой сервер

Основной модуль редактора – языковой сервер, реализующий поддержку пользователя при составлении текстового SPARQL-запроса. Кроме генерации дополняющих подсказок этот модуль отвечает за подсветку и валидацию синтаксиса SPARQL-запроса. Для взаимодействия этого модуля с остальной системой можно использовать протокол LSP.

Language Server Protocol (LSP) – протокол для взаимодействия редакторов кода или IDE с языковыми серверами. Протокол основан на JSON-RPC – одном из протоколов удаленного вызова процедур – и определяет форматы сообщений для основных функций поддержки языков программирования. LSP поддерживается большинством современных редакторов кода и IDE.

В ходе проведенного исследования удалось найти один языковой сервер, реализующий поддержку SPARQL по протоколу LSP, – сервер, реализованный компанией «Stardog» в рамках разработки редактора Stardog Studio. Данное программное обеспечение предоставляет базовую поддержку языка SPARQL: подсветка и проверка синтаксиса, дополнение введенного запроса на основе синтаксиса (ключевыми словами) и ранее введенных токенов, однако отсутствуют подсказки не присутствующих в запросе элементов онтологии. Поскольку сервер обладает базовой структурой для реализации полной поддержки SPARQL, было решено реализовать собственный языковой сервер на основе существующего, а в качестве клиента для демонстрации использовать редактор Visual Studio Code.

Использование LSP в качестве протокола для коммуникации между языковым клиентом и языковым сервером позволит в дальнейшем переиспользовать языковой сервер во многих других редакторах, часть из которых использует LSP в качестве основной модели для подключения языковой поддержки (например, Visual Studio Code), другие имеют возможность подключения языковых серверов по протоколу LSP (например, IntelliJ IDEA, Sublime Text) с помощью плагинов для этих сред.

Дополняющие подсказки на основе содержимого онтологии

Наличие дополняющих подсказок в современных текстовых редакторах является неотъемлемой возможностью. Большинство рассмотренных выше редакторов реализует дополняющие подсказки ключевых слов SPARQL и элементов онтологии, но в основном без учета контекста. При отсутствии в редакторе SPARQL-запросов функции дополнения элементами онтологии пользователю приходится запоминать структуру онтологии либо параллельно с составлением запроса использовать средства для навигации по онтологии. Наличие дополняющих подсказок, основывающихся на структуре целевой онтологии, позволяет совместить эти два процесса. В работе [12] обсуждается необходимость интеллектуальных (использующих структуру онтологии) дополняющих подсказок для разных групп пользователей SPARQL.

Базовым содержимым SPARQL-запроса является основной графовый шаблон – набор триплетов, участвующих в запросе. Каждый триплет состоит из субъекта, предиката и объекта, в роли которых могут выступать элемент онтологии, переменная или константа (литерал). Для упрощения составления корректного графового шаблона необходимо поддерживать как дополнение имеющейся части триплета недостающими элементами онтологии, непротиворечащими ее структуре и другим триплетам запроса, а также проверку каждого триплета из графового шаблона на соответствие некоторому факту в онтологии. Кроме этого, полезной будет подсказка о несвязности графового шаблона, так как, вероятнее всего, SPARQL-запрос с несвязанным графовым шаблоном был составлен по ошибке.

В зависимости от места вызова дополняющей подсказки в триплете предлагаемые для дополнения элементы должны отличаться. В качестве дополняющих элементов могут высту-

пать как элементы онтологии, так и переменные. Формирование дополнительных подсказок происходит в три этапа:

- 1) определение типа дополняющей подсказки в зависимости от места вызова;
- 2) формирование и выполнение вспомогательного SPARQL-запроса для определения первичных дополняющих подсказок;
- 3) формирование вторичных дополняющих подсказок, содержащих переменные, на основе результатов вспомогательного SPARQL-запроса.

Рассмотрим алгоритм составления дополнительных подсказок для субъекта последнего триплета рассматриваемого в разделе SPARQL-запроса (рис. 2).

```

1  # sparql : http://localhost:3030/countries_ds
2  PREFIX countries: <http://www.semanticweb.org/irina/ontologies/2021/5/countries#>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT ?city
6  WHERE {
7    ?country a countries:Country .
8    ?city a countries:City .
9    ?country countries:city ?city .
10   ?lang a countries:Language .
11   ?lang rdfs:label "French"@en .
12   ?country

```

Рис. 2. Дополняющие подсказки
Fig. 2. Completion hints

При вызове подсказки курсор находится в 12-й строке после переменной *?country*, перед которой находится токен «.» (точка), который означает конец триплета. Преобразование текста запроса в последовательность токенов, а также определение их типов осуществляется с помощью лексического анализатора библиотеки Millan, разработанной компанией «Star-dog» для своего языкового сервера и основанной на Chevrotain⁹ – популярной библиотеке для создания парсеров синтаксических деревьев по заданной грамматике. Таким образом, можно понять, что в текущем триплете присутствует только субъект в виде переменной, а предикат и объект требуют дополнения.

Так как в данном примере на месте субъекта стоит переменная, а не конкретный элемент онтологии, то в общем случае дополняющими элементами могут быть все присутствующие в онтологии комбинации предикат-объект. Однако выше в запросе присутствует триплет *?country a countries:Country*, определяющий тип допустимых значений переменной *?country*. Таким образом, поиск может быть сужен до тех комбинаций предикат-объект, субъектом для которых выступает элемент типа *Country*. Поскольку дополнение SPARQL-запроса на основе структуры онтологии требует получения из онтологии элементов, отвечающих некоторым критериям, удобно для формирования подсказок использовать SPARQL-запросы, сформиро-

⁹ URL: <https://sap.github.io/chevrotain/docs/>.

ванные программно и выполняемые в фоновом режиме, к той же самой онтологии, к которой пользователь составляет запрос в редакторе. Помимо поиска комбинаций предикат-объект также с помощью вспомогательного SPARQL-запроса определяется и тип объекта для формирования вторичных подсказок (рис. 3).

```

1  PREFIX countries: <http://www.semanticweb.org/irina/ontologies/2021/5/countries#>
2
3  SELECT DISTINCT ?p ?o ?oc
4  WHERE {
5      ?s ?p ?o .
6      ?s a countries:Country .
7      ?o a ?oc
8  }
```

Рис. 3. Вспомогательный SPARQL-запрос
Fig. 3. Service SPARQL query

Приведем результаты выполнения вспомогательного SPARQL-запроса (единообразные результаты заменены многоточием):

?p	?o	?oc
capital	Monaco-Ville	City
capital	Oslo	City
capital	...	City
city	Bergen	City
city	Toronto	City
city	...	City
officialLanguage	Norwegian	Language
officialLanguage	French	Language
officialLanguage	...	Language
type	Country	Class

Первичные дополняющие подсказки составлены на основе уникальных пар значений переменных *?p* и *?o*. Вторичные дополняющие подсказки в данном примере составляются путем подмены объекта (значения переменной *?o*) в первичных подсказках на подходящую переменную. Для определения подходящих переменных используется тип значения объекта (значение переменной *?oc*), и среди всех используемых в запросе переменных выбираются те, значения которых ограничены нужным типом. В рассматриваемом примере на основе пар из предиката *capital* и объектов типа *City* будет составлена дополняющая подсказка *capital ?city*, так как значение переменной *?city* ограничено типом *City* с помощью триплета *?city a countries:City*. Аналогично будет составлена подсказка *officialLanguage ?lang*.

Визуальное представление

Помимо дополняющих подсказок и проверки корректности фактов в запросе для поиска семантических ошибок удобно использовать графическое представление запроса. Тело SPARQL-запросов типа SELECT представляет собой графовый шаблон, поэтому он естественным образом представляется в виде графа, где вершинам и ребрам в соответствии поставлены элементы онтологии или переменные из запроса. Поскольку протокол LSP не поддерживает работу с графическими элементами, передача информации о графе на клиент происходит посредством протокола HTTP.

Наличие в редакторе дополняющих подсказок позволяет составлять запросы, не обращаясь за информацией о структуре онтологии к редакторам онтологий. Однако полезной возможностью может стать получение дополнительной информации о структуре онтологии из визуального представления без необходимости редактировать SPARQL-запрос с целью вызова необходимой подсказки. Поскольку онтология может содержать большое количество концептов, которые могут быть достаточно сильно связаны между собой отношениями, чтобы не перегружать визуальное представление, выводятся только концепты и отношения, связанные с выбранной пользователем вершиной, которая может быть как вершиной, так и элементом онтологии (рис. 4). В случае если выбран элемент онтологии, в качестве соседних отображаются концепты и отношения, связанные с этим элементом. А в случае если выбрана переменная, – концепты и отношения, связанные с любым из возможных значений этой переменной.

Рис. 4. Визуальное представление SPARQL-запроса
Fig. 4. Vizualization of SPARQL query

Сравнение результатов нескольких ризонеров

При выполнении SPARQL-запроса к онтологии в результат могут попасть только те факты, которые непосредственно присутствуют в описании онтологии. Чтобы учесть факты, которые могут быть выведены с помощью аксиом онтологии, необходимо использовать ризонер. Ризонеры различаются набором поддерживаемых типов аксиом, временем выполнения, подходом к выводу новых фактов (жадный вывод с кэшированием или ленивый вывод по необходимости). Для сравнения нескольких ризонеров с целью выбора ризонера, наиболее подходящего под задачу, а также в учебных целях для демонстрации различных техник логического вывода удобной возможностью редактора SPARQL-запросов представляется выполнение запроса к онтологии и сравнение результатов, полученных несколькими ризонерами.

Для реализации этой возможности необходимо иметь несколько точек доступа SPARQL: для исходной онтологии и по одной для каждого ризонера, который будет работать с этой онтологией. Как для удаленных онтологий (если они открыто доступны), так и для локальных задача подключения к ним ризонеров решается созданием собственных точек доступа с необходимой онтологией и ризонером.

После выполнения запросов ко всем точкам доступа необходимо предоставить возможность сравнения результатов. Поскольку результат SELECT-запросов можно рассматривать как таблицу, то задача сводится к сравнению нескольких таблиц. Наглядным образом пред-

ставить сравнение нескольких таблиц достаточно сложно, поэтому было решено организовать попарное сравнение результатов различных ризонеров и исходной онтологии.

При выполнении рассматриваемого SPARQL-запроса к точке доступа онтологии без использования ризонера в результат не попадут города-столицы, так как отношение *capital* наследуется от отношения *city*, а в онтологии для столиц отсутствуют факты с использованием отношения *city*. Однако при использовании ризонера города-столицы будут присутствовать в результате, так как ризонер учитывает транзитивность отношения наследования. В таблице выводится результат сравнения результатов, полученных с ризонером (все пять строк), относительно результатов без ризонера (три строки) (рис. 5).

Select one or two tables to compare: | | Daff table ☒

@@	city
	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Fontvieille
+++	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Monaco-Ville
+++	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Ottawa
	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Toronto
	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Vancouver

Рис. 5. Сравнение результатов ризонеров
Fig. 5. Comparison of reasoners results

Архитектура редактора SPARQL-запросов

Полная система состоит из языкового клиента, языкового сервера и точек доступа SPARQL (рис. 6). Сторонние модули отмечены пунктирными границами, а разработанные самостоятельно – сплошными. Модули языкового сервера не зависят от используемого в качестве клиента редактора кода или IDE. При необходимости использовать другой инструмент в качестве клиента необходимо будет реализовать модули, объединенные в языковой клиент, с использованием API другого редактора.

Рис. 6. Архитектура системы
Fig. 6. System architecture

Редактор может работать с любыми удаленными точками доступа SPARQL, такими как Dbpedia, Wikidata и др. Для работы с локальными онтологиями используется SPARQL-сервер Fuseki, который позволяет настроить несколько точек доступа с разными онтологиями и ризонерами. Для упрощения настройки Fuseki было разработано приложение для генерации файла конфигураций Fuseki, позволяющее в графическом интерфейсе выбрать файлы с онтологиями и необходимые ризонеры.

Заключение

В работе был проведен обзор редакторов SPARQL-запросов, проанализированы возможности различных типов редакторов. На основании обзора сформулированы требования к собственному редактору.

Разработанный редактор реализует три основные возможности для работы со SPARQL: дополняющие подсказки, визуальное представление запроса и сравнение результатов выполнения. Ключевой особенностью разработанного редактора является его модульность, каждый модуль реализует одну из возможностей. Такое архитектурное решение позволяет переиспользовать отдельные модули в других инструментах.

Разработанный редактор может быть использован как в образовательных целях на занятиях, посвященных работе с онтологиями, изучению SPARQL или работе ризонеров, так и в практических и исследовательских целях.

Список литературы / References

1. **Dissanayake P. I., Colicchio T. K., Cimino J. J.** Using clinical reasoning ontologies to make smarter clinical decision support systems: a systematic review and data synthesis. *Journal of the American Medical Informatics Association*, 2020, vol. 27, no. 1, pp. 159–174. DOI 10.1093/jamia/ocz169
2. **Rietveld L., Hoekstra R.** Man vs machine: Differences in SPARQL queries. In: Proceedings of the 4th USEWOD Workshop on Usage Analysis and the Web of Data, ESWC, 2014.
3. **Warren P., Mulholland P.** Using SPARQL – the practitioners’ viewpoint. In: European Knowledge Acquisition Workshop. Cham., 2018, pp. 485–500. DOI 10.1007/978-3-030-03667-6_31
4. **Vargas H., Buil-Aranda C., Hogan A., López C.** RDF Explorer: A Visual SPARQL Query Builder. In: International Semantic Web Conference. Cham., 2019, pp. 647–663. DOI 10.1007/978-3-030-30793-6_37
5. **Hogenboom F., Milea V., Frasincar F., Kaymak U.** RDF-GL: A SPARQLBased Graphical Query Language for RDF. In: Emergent Web Intelligence: Advanced Information Retrieval, 2010, pp. 87–116. DOI 10.1007/978-1-84996-074-8_4
6. **Smart R., Russell A., Braines D., Kalfoglou Y., Bao J., Shadbolt R.** A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In: Knowledge Engineering and Knowledge Management (EKAW), 2008, pp. 275–291. DOI 10.1007/978-3-540-87696-0_25
7. **Rietveld L., Hoekstra R.** YASGUI: not just another SPARQL client. In: Extended Semantic Web Conference. Berlin, 2013, pp. 78–86. DOI 10.1007/978-3-642-41242-4_7
8. **Rietveld L., Hoekstra R.** The YASGUI family of SPARQL clients. *Semantic Web*, 2017, vol. 8, no. 3, pp. 373–383. DOI 10.3233/sw-150197
9. **Campinas S., Perry T. E., Ceccarelli D., Delbru R., Tummarello G.** Introducing RDF graph summary with application to assisted SPARQL formulation. In: 23rd International Workshop on Database and Expert Systems Applications, 2012, pp. 261–266. DOI 10.1109/dexa.2012.38

10. **Gardiner T., Horrocks I., Tsarkov D.** Automated benchmarking of description logic reasoners. In: Proceedings of the International Workshop on Description Logics (06) CEUR, 2006, vol. 189, pp. 167–174.
11. **Alaya N., Yahia S.B., Lamolle M.** RakSOR: Ranking of ontology reasoners based on predicted performances. In: IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016, pp. 1076–1083. DOI 10.1109/ictai.2016.0165
12. **Rafes K., Abiteboul S., Cohen-Boulakia S., Rance B.** Designing scientific SPARQL queries using autocompletion by snippets. In: IEEE 14th International Conference on e-Science, 2018, pp. 234–244. DOI 10.1109/escience.2018.00038

Информация об авторах

Ирина Алексеевна Турова, студентка магистратуры
Игорь Сергеевич Постановов, старший преподаватель

Information about the Authors

Irina A. Turova, Master's Student
Igor S. Postanogov, Senior Lecturer

*Статья поступила в редакцию 09.09.2021;
одобрена после рецензирования 01.11.2021; принята к публикации 01.12.2021
The article was submitted 09.09.2021;
approved after reviewing 01.11.2021; accepted for publication 01.12.2021*