Представлено дослідження і аналіз методів оптимізації швидкодії виконання запитів до онтологічних баз знань, використовуючи сервер даних на основі «Virtuoso». Встановлено залежність часу виконання запиту до сервера від ключових факторів впливу. Визначено такі фактори: частота запитів до бази знань, об'єм інформації, який необхідно обробити, правильність формування запиту, використання допоміжних інструментів та фреймворків

Ключові слова: запит, оптимізація, триплет, SPARQL, Jena Framework, Caching, Virtuoso, OWL, RDF, BGP, SPARUL, Pattern

Представлены исследования и анализ методов оптимизации быстродей-ствия выполнения запросов к онтологическим базам знаний, используя сервер данных на основе «Virtuoso». Установлена зависимость времени выполнения запроса к серверу от ключевых факторов влияния. Определены следующие факторы: частота запросов к базе знаний, объем информации, который необходимо обработать, правильность формирования запроса, использования вспомогательных инструментов и фреймворков

Ключевые слова: запрос, оптимизация, триплет, SPARQL, Jena Framework, Caching, Virtuoso, OWL, RDF, BGP, SPARUL, Pattern

УДК 519.7:007.52
DOI: 10.15587/1729-4061.2014.28553

ОПТИМИЗАЦИЯ БЫСТРОДЕЙСТВИЯ ОНТОЛОГИЧЕСКИХ БАЗ ЗНАНИЙ, ПОСТРОЕННЫХ НА ОСНОВЕ «VIRTUOSO»

И. Е. Бибичков*
E-mail: bibi4kov@gmail.com
В. В. Сокол
Инженер*
E-mail: sokol@sw-expert.com
А. Ю. Шевченко
Кандидат технических наук, доцент*
*Кафедра искусственного интеллекта
Харьковский национальный
университет радиоэлектроники
пр. Ленина, 14, г. Харьков, Украина, 61000
E-mail: shevchenko@sw-expert.com

1. Введение

Как показывает практика, в настоящее время, время обилия информации, остро стоит вопрос о ее систематизации и хранении для последующего эффективного использования.

Хорошо зарекомендовавшая реляционная модель хранения данных, при помощи которой решались (и сейчас решается) обилие задач, не всегда удовлетворяет многим требованиям, которые выдвигаются к моделям данных. Преимуществом использования реляционных баз данных является простота в применении, большое обилие инструментов для проектирования и разработки моделей данных, огромное количество различных серверных систем, поддерживающих данный подход к работе с данными. Однако недостатком реляционного подхода к хранению данных является недостаточная гибкость и расширяемость. То есть, чаще всего, даже для внесения незначительных изменений в имеющуюся структуру данных, требуется внесение множества изменений в таблицы, которые представляют базу данных, а также внесение изменений в связи между этими таблицами и, возможно, необходимость создания большого количества дополнительных таблиц и связей между ними.

Именно для устранения подобных недостатков, которые присущи реляционным базам данных, им на смену приходят базы знаний. (База знаний – это особого рода база данных, разработанная для оперирования знаниями (метаданными). База знаний содержит структурированную информацию, покрывающую некоторую область знаний [1]).

В данной работе речь пойдет об особенностях применения на практике баз знаний в виде OWL-онтологий, с использованием сервера «Virtuoso»[2], а именно будут рассмотрены возможные пути оптимизации скорости и времени работы с базой знаний. База знаний «Virtuoso» на данный момент является одной из наиболее пригодных для использования в коммерческих информационных системах [3]. В частности, будут рассмотрены инструменты работы с OWL-базами знаний, а именно язык запросов SPARQL и Jena Framework [4]. Большой проблемой использования онтологических баз знаний в коммерческих проектах является их крайне низкое быстродействие. Универсальная структура онтологических баз знаний превосходно подходит для хранения в упорядоченном виде разнородной информации, но снижение быстродействия при увеличении объемов хранимой информации негативно сказывается на возможностях использования этой технологии. В статье рассмотрены методы позволяющие увеличить быстродействие базы знаний и сделать её пригодной для решения задач в рамках всеобъемлющего решения проблемы обеспечения хранения и анализа разнородной информации и быстрой её выборки из сверхбольших баз знаний, получившей название “Big Data”.

2. Возможные пути решения проблемы низкого быстродействия онтологических систем при больших объемах сохраненных знаний

Для работы с базами знаний на практике используются специальные инструменты. В данной работе будут описаны особенности применения сервера «Virtuoso», а также использования языка запросов SPARQL и Jena Framework, фирмы «Apache» (который, по сути, является обверткой к языку запросов SPARQL).

Методы оптимизации быстродействия запросов регламентированы особенностями самого сервера и инструментов, при помощи которых производится работа с базой знаний. Для решения проблем, связанных с быстродействием, предлагается использовать ряд специальных методов: кэширование данных [5], применение особенностей языка SPARQL версии 1.1, разбиение сложного запроса на языке SPARQL на более мелкие, следование рекомендациям по построению SPARQL-запросов, использование BGP (Basic Graph Pattern) [6], и многие другие, некоторые из которых будут описаны в данной работе.

Кэширование данных. То есть сохранение промежуточных результатов выполнения запросов во временное хранилище на сервере в виде реляционной таблицы, либо специальных файлов. Данный метод применяется для сокращения объема получаемых данных из базы знаний и уменьшения количества обращений к серверу [5].

Применение особенностей языка SPARQL версии 1.1, везде, где это возможно. Этот метод эффективен для применения на практике, так как многие системы используют SPARQL версии 1.0, несмотря на то, что SPARQL версии 1.1 имеет ряд дополнительных функций, которые позволяют экономить время, которые будут описаны позже.

Разбиение сложного запроса на языке SPARQL на более мелкие. Чем сложнее запрос, тем больше времени требуется для его обработку и выполнения сервером.

Следование рекомендациям по построению SPARQL-запросов, которые направлены на оптимизацию времени выполнения запроса. Основной принцип данного метода заключается в оптимизации самой структуры запроса. Например, переписывание переменных, которые расположены в секции «FILTER» SPARQL-запроса, перемещение опции «FILTER» вверх.

Использование BGP (Basic Graph Pattern) – базового графического паттерна для оптимизации SPARQL-запросов [6].

3. Оптимизация SPARQL-запросов кэшированием

Для реализации кэширования в SPARQL-запросах предполагается создать небольшой прокси-слой, который должен быть расположен между приложением Semantic Web и SPARQL/SPARUL [7]. Все SPQRQL-запросы и SPARUL-обновления проходят через этот прокси-слой. После того как прокси-сервер получает запрос, он проверяет, закэширован ли результат выполнения данного запроса в локальном хранилище. Если да, то результат незамедлительно передается клиенту, без вызова хранилища триплетов. Если запрос не был ранее сохранен и не найден в сохраненных пользовательских правилах, то он перенаправляется в локальное хранилище триплетов, и, прежде чем пользователь получит результат, он кэшируется в локальное хранилище. Структурная схема, согласно которой реализовано кэширование данных, представлено на рис. 1.

Для того чтобы определить выигрыш от применения кэширования, построим запрос на языке SPARQL (листинг 1) и определим время его выполнения без применения кэширования и с применением кэширования.

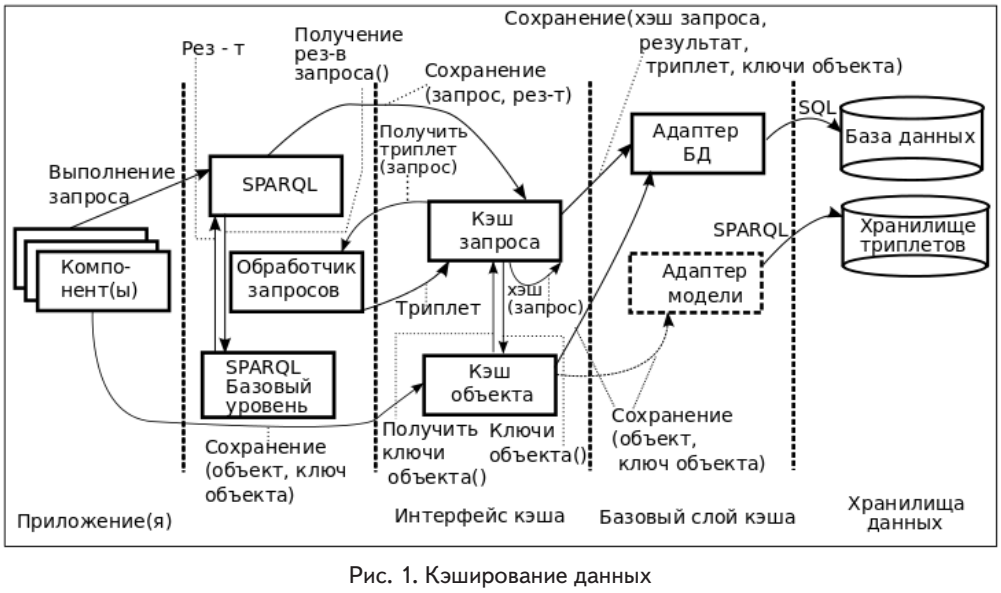

Рис. 1. Кэширование данных

```
1 PREFIX aksw: <http://aksw.org/people#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 SELECT  ?classUri ?classLabel FROM <http://aksw.org/people#>
4 WHERE { ?classUri  rdfs:subClassOf  aksw:People .
5         ?classUri  aksw:sort  ?sort .
6         OPTIONAL { ?classUri  rdfs:label  ?classLabel } }
```

Листинг 1 — пример запроса на языке SPARQL

Практика показывает, что применение кэширования в запросах SPARQL позволяет повысить производительность почти на 90 %, при выборке 25 миллионов триплетов. При том, что в кэше хранится от 8 % до 25 % от общего числа триплетов.

При выполнении операций добавления и обновления данных (UPDATE), выигрыш, полученный от кэширования, менее существенный, всего около 5 %.

4. Применение особенностей SPARQL 1.1

Применение особенностей SPARQL-запросов версии 1.1, за счет дополнительных опций в запросах, о которых будет изложено ниже, позволяет в значительной степени оптимизировать время выполнения запроса. Пример запроса с использованием оператора SERVICE приведен в листинге 2.

```
PREFIX  iuphar:     <http://iuphar.example/ns#>
PREFIX  entrez:     <http://entrez.example/ns#>
PREFIX  void:       <http://rdfs.org/ns/void#>
PREFIX  dcterms:    <http://purl.org/dc/terms/>

SELECT ?service ?id ?iuphar
WHERE {
  # Find the service with the expertise.
  ?service dcterms:subject ?gene
  FILTER (?gene = entrez:h2550 || ?gene = entrez:h9568

  # Query that service for species and iuphar.
  SERVICE ?service {
    ?receptor  iuphar:species    ?species .
    ?species   iuphar:name       ?iuphar .
    ?species   entrez:id         ?id .
  }
}
```

Листинг 2 — Пример запроса с использованием оператора SERVICE

Федеративное расширение (Federation Extension) [8] в SPARQL 1.1 включает два новых оператора: SERVICE и BINDINGS (рис. 2).

Оператор SERVICE позволяет пользователю включать подзапросы в запросах. То есть пользователь может составлять сложные запросы из подзапросов, и, следовательно, такой запрос будет выполняться порциями, что позволит уменьшить время его выполнения.

Оператор BINDINGS позволяет накладывать дополнительное пользовательское ограничение на запрос, результат которого можно использовать прямо в текущем запросе (при этом не создавая, прежде, новый запрос). Такой запрос представляется на языке SPARQL.

Пример запроса с использованием оператора BINDINGS приведен в листинге 3.

```
PREFIX  entrez:     <http://entrez.example/ns#>
PREFIX  med:        <http://med.example/testDrug#>
PREFIX  study:      <http://study.example/affects#>

SELECT ?med ?species ?iuphar
WHERE {
  ?study  entrez:id            ?id .
  ?study  study:species        ?species
  ?study  med:ication          ?med
  ?study  study:change         ?change .
  FILTER (?change < -.2)
} BINDINGS ?human ?iuphar ?id {
  ("human" "GABBR1" "2550")
  ("human" "GABBR2" "9568")
}
```

Листинг 3 — Пример запроса с использованием оператора BINDINGS

Выигрыш от применения данных операторов достигается посредством систематизации данных путем распределения по определенным признакам в отдельные группы.

Время обработки запросов, применяя SPARQL 1.1: федеративные запросы приведено в табл. 2. Ключевые характеристики тестируемых запросов приведены в табл. 1, 2 [10].

Таблица 1

Количественные характеристики триплетов

	Количество триплетов	Количество внешних ссылок	Количество внутренних ссылок	Namespace	Limit
DB pedia	1 000 000 000	16 480 998	11 531 095	http://dbpedia.org/resource/	1000
NYTimes	345 889	23 700	21 289	http://data.nytimes.com/	1000
LinkedMDB	6 148 121	162 756	1 883	http://data.linkedmdb.org/resource/	1000
Geonames	93 896 732	–	1 028 252	http://sws.geonames.org/	1000
World Factbook	38 640	–	665 005	http://www4.wiwiss.fu-berlin.de/factbook/resource/	1000
El Viajero	9 462 339	12 750	–	http://webenemasuno.linkeddata.es/elviajero/resource/	1000

Таблица 2

Характеристики запросов (количество ключевых элементов)

№	Joins	OPT	FILTER	UNION	Patterns	Variables	Solution Modifiers	Bounded Subject	Bounded Predicates	Bounded Objects
Q1	0	0	0	1	2	2	0	1	1	1
Q2	1	0	0	0	2	3	0	1	3	1
Q3	1	0	0	0	2	3	0	0	5	1
Q4	1	0	0	0	2	2	0	0	5	1
Q4b	0	1	1	0	3	5	0	0	6	1
Q5	1	0	0	0	2	4	0	0	4	1
Q6	0	0	0	0	1	3	0	0	4	1
Q7	1	0	0	0	2	2	0	0	4	1
Q8	0	1	1	0	2	3	0	0	3	1
Q9	1	1	0	0	3	4	0	0	7	2

Результаты применения к триплетам оптимизационных приемов отображены в табл. 3 [10].

Таблица 3

Результаты оптимизации

Запрос	Время выполнения не оптимизированного запроса	Время выполнения оптимизированного запроса
Q1	2.117 ms	2.150 ms
Q2	2.370 ms	2.200 ms
Q3	9.659 ms	8.085ms
Q4	8.341 ms	7.291 ms
Q4b	10.562 ms	9.771 ms
Q5	5.687ms	3.759ms
Q6	887ms	0.878ms
Q7	11.625ms	7.994ms
Q8	15.544ms	14.373ms
Q9	19.658ms	17.192ms

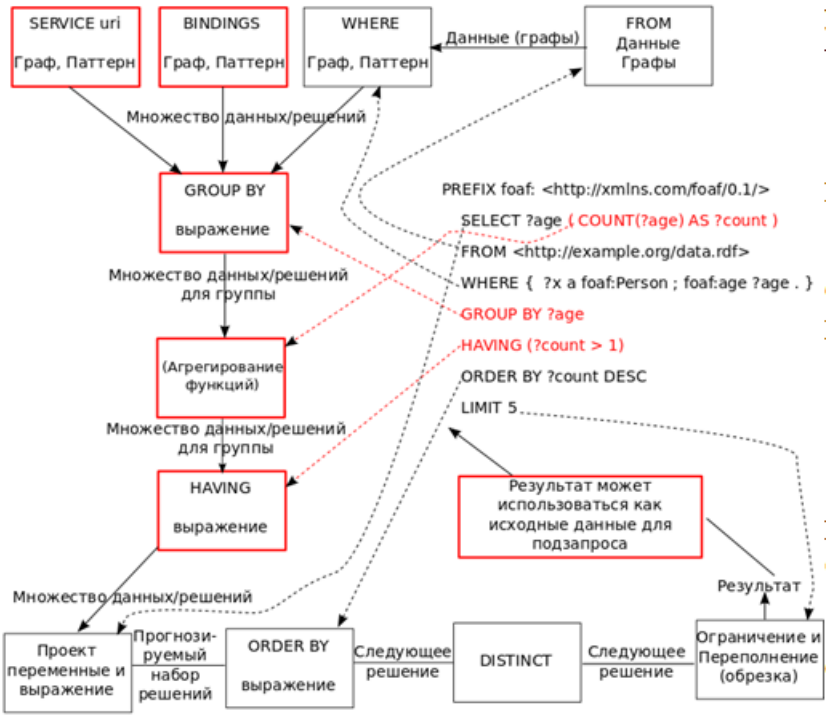

Рис. 2. Федеративное расширение

5. Применение Jena Framework

В случае применения Jena Framework программист экономит время на составление SPARQL-запросов. Jena Framework, по своей сути, является «обверткой» для SPARQL. Среди преимуществ фреймворка также есть создание удобной модели для работы с данными не только в виде триплетов, но так же и для работы с данными как с объектами, что позволяет не задумываться о том, как получить ресурс и какой именно.

Среди недостатков – создание фреймворком модели данных по полученным триплетам, что занимает значительное количество машинного времени (около 1 с).

В случае, когда запрос требует множественного обращения к серверу, применение Jena Framework с точки зрения затрат более выгодно. Для этого программисту необходимо минимизировать количество обращений к серверу для получения данных и построения модели. В некоторых случаях применение данного фреймворка быстрее, чем применение SPARQL. Это преимущество имеет место, когда необходимо произвести множественную выборку конкретных ресурсов.

В таком случае SPARQL действует путем отправки множества обращений на сервер (в данном случае применяется сервер «Virtuoso»), что приводит к большому количеству затраченного времени, напротив, Jena Framework позволяет минимизировать количество обращений к серверу. Это происходит благодаря тому, что данный фреймворк строит модель данных по существующим триплетам. Хотя сам фреймворк большую часть времени тратит на построение самой модели, но, благодаря тому, что данное действие производится единоразово, результаты достигаются лучше, чем в случае применения SPARQL запросов.

Исследование проводилось на модели данных, состоящей из 377 420 триплетов, созданной при помощи BSBM. Запрос приведен в листинге 4.

Данный код приведен для случая использования языка программирования Java. Суть в том, что пользователь запрашивает 1000 ресурсов по изменяющимся именам продуктов.

Листинг аналогичного запроса, написанного при помощи Jena Framework, представлен в листинге 5.

Выборка производится, как и в предыдущем случае, циклично.

```
String myQuery =
        "PREFIX bsbm-inst: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/instances/> "+
        "PREFIX bsbm: <http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/> "+
        "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> "+
        "PREFIX dc: <http://purl.org/dc/elements/1.1/> "+
        "SELECT ?producer "+
        "FROM <"+myGraph+"> "+
        "WHERE { <"+
            productURI+"> bsbm:producer ?p . " +
                "?p rdfs:label ?producer . "+
            "}";
```

Листинг 4 — Тестируемый SPARQL запрос

```
OntModel ontModel = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, new VirtModel(dataSet));
Property labelProperty = ontModel.getProperty(productProperty+"label");
Property producerProperty = ontModel.getProperty(bsbmProperty+"producer");
Individual ind = ontModel.getIndividual(productURI);
propertyProducerValue = ind.getPropertyValue(producerProperty).toString();
Resource producerResource = ontModel.getResource(propertyProducerValue);
Statement propertyProducerLabel =  producerResource.getProperty(labelProperty);
propertyOfProducer = propertyProducerLabel.getObject().toString();
```

Листинг 5 — Запрос 1000 продуктов при помощи Jena Framework

В результате произведенных исследований были получены такие данные: при использовании SPARQL запросов время выполнения составило 3812 ms, Jena Framework – 2834 ms

6. Выводы

Применение на практике представленных в данном исследовании приемов оптимизации позволяет увеличить быстродействие построенных запросов, в зависимости от конкретных случаев применения. В частности, применение кэширования в построении SPARQL запросов позволяет повысить производительность почти на 90 %, при выборке из 25 миллионов триплетов. А применение Jena Framewok в случае выборки 1000 ресурсов из базы знаний позволяет экономить до 30 % времени выполнения программы. Применение особенной SPARQL 1.1, в частности федеративных запросов, при написании запросов, содержащих «Join», «Bounded Predicates» «Patterns», достигается уменьшение времени выполнения запроса от 20 % до 35 %.

В итоге самым лучшим приемом для оптимизации SPARQL запросов является кэширование промежуточных результатов запроса, так как это позволяет экономить до 90 % времени выполнения, в других двух рассмотренных случаях, выгода составляет до 35 %.

Работа актуальна, так как на данный момент существует острая необходимость в оптимизации быстродействия работы с базами знаний. Приемы, описанные выше, помогают значительно, а именно до 90 %, экономить время выполнения запросов, а, следовательно, увеличивать показатели быстродействия приложений.

Литература

1. Википедия — свободная энциклопедия [Электронный ресурс] / Базы знаний. – Режим доступа: https://ru.wikipedia.org/wiki/База_знаний — 18.08.2014. – Загл. с экрана.
2. OPENLINK software — Making Technology Work For You [Electronic resource] / Virtuoso — Universal Server. – Available at: http://virtuoso.openlinksw.com/ – 2014. – Title from the screen.
3. Шевченко, О. Ю. Сравнительный анализ современных систем управления онтологическими базами знаний [Текст] / О. Ю. Шевченко, О. Л. Шевченко // Вісник СевНТУ. Збірник наукових праць. Серія Інформатика, електроніка, зв'язок. – 2012. – № 131. – С. 82–86.
4. Apache Jena [Electronic resource] / A free and open source Java framework for building Semantic Web and Linked Data applications. – Available at: https://jena.apache.org/ – 2014. – Title from the screen.
5. Martin, M. Improving the Performance of Semantic Web Applications with SPARQL Query Caching [Text] / M. Martin, J. Unbehauen, S. Auer // The Leipzig University of computer science. – 2010. – P. 304–318. doi:10.1007/978-3-642-13489-0_21
6. Stocker, M. SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation [Text] / M. Stocker, A. Seaborne, A. Bernstein , C. Kiefer , D. Reynolds // Proceeding of the 17th International Conference on World Wide Web - WWW '08, 2008, 2–9. doi:10.1145/1367497.1367578
7. Andy Seaborne and Geetha Manjunath. SPARQL [Electronic resource] / Update - a language for updating RDF graphs. – Available at: http://www.w3.org/Submission/SPARQL-Update/ – 2008. – Title from the screen.
8. Buil-Aranda, C. Semantics and Optimization of the SPARQL 1.1. Federation Extension [Text] / C. Buil-Aranda, M. Arenas, O. Corcho // The Semanic Web: Research and Applications Lecture Notes in Computer Science. – 2011. – Vol. 6644. – P. 1–15. doi:10.1007/978-3-642-21064-8_1
9. Beckett, D. Learn about SPARQL 1.1. [Electronic resource] / SPARQL 1.1 Query Execution. — Available at: http://www.dajobe.org/talks/201105-sparql-11/ – 2011. – Title from the screen.
10. Buil-Aranda, C. Federated Query Processing for the Semantic Web [Text] / C. Buil-Aranda, O. Corcho Garc´a // A thesis submitted for the degree of PhD Thesis. – 2012. – Vol. 1. – P. 3–33.