# На пути к большим RDF данным

© А. Г. Марчук
ИСИ СО РАН, НГУ
Новосибирск
mag@iis.nsk.su

## Аннотация

Проблематика Big Data [1] связана с накоплением и использованием больших объемов данных разных предметных областей. То же самое можно сказать о направлении Semantic Web [2], для которого «Семантическая сеть привнесёт структуру в смысловое содержание веб-страниц, тем самым создав среду, в которой программные агенты, переходя со страницы на страницу, смогут без особого труда выполнять замысловатые запросы пользователей» [2, 13]. Существенной конкретизацией Semantic Web является опора на единый механизм структуризации данных и знаний RDF, «окруженный» рядом стандартов и технологий наиболее значимыми из которых представляются OWL [3], Sparql [4], Linking Open Data [5], а также множество сформированных и опубликованных онтологий (vocabularies) [6]. Целью исследования, отраженного в докладе, являлось определение возможностей работы с большими RDF данными. Исследование поддержано грантом РФФИ 11-07-00388а, программой РАН Р-15/10, интеграционным проектом СО РАН М-48.

## 1 Введение

RDF позволяет накапливать и использовать данные произвольной природы, если не прямой их фиксацией, то хотя бы в виде метаинформации. Собственно последнее, похоже имелось ввиду авторами формата структурирования, чье полное название Resource Description Framework. У автора доклада накоплен значительный опыт по использованию RDF в качестве хранилищ данных для информационных систем общего назначения, архивных и музейных систем [7]. В исследовании изучались различные существующие технологии, наборы данных, онтологические построения, однако, работа не претендует на полноту обзорной части и внешние результаты рассматривались под углом зрения решаемых в ИСИ СО РАН фундаментальных и прикладных задач.

Главные вопросы, которые были поставлены: какие технологии обработки RDF лучше подходят для решаемых в коллективе прикладных задач и до каких размеров данных эти технологии будут «работать». Кроме того, интересовали массивы RDF-данных, имеющиеся в открытом пользовании, используемые для больших данных онтологии, сопоставление разных методов выборки данных, наличие в накопленных в мире массивах полезных данных, возможных к использованию в научной деятельности. Эта последняя группа вопросов в данном докладе не отражена.

## 2 Технологии обработки RDF данных

В рамках проектов, ведущихся в ИСИ СО РАН в течение более 10 лет накапливаются данные, в основном по предметной области исторической фактографии [8]. Эти данные собираются в так называемые кассеты [9], объединяющие в себе как собственно документный контент (фотографии, видео, аудио, сканы страниц документов, документы разных форматов), так и базу данных, в виде RDF-документов. Всего накоплено около 500 Гб. кассет и процесс накопления продолжается. Созданная технология, выполненная в виде платформенного решения Sema2012 [7], позволяет выбранный набор кассет активизировать в виде Web-приложения, сочетающего в себе элементы сервиса и разные пользовательские Web-интерфейсы, и использовать приложение для решения задач просмотра информации, поиска и редактирования. В совокупности, накопленный объем RDF данных оценивается в 2 млн. триплетов. Важным требованием прикладных построений к платформе является возможность обеспечения работы «в реальном времени», т.е. со средними задержками меньше одной секунды и с максимальными задержками не более нескольких секунд. Реализованный в платформенном решении

**Труды 15-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL-2013, Ярославль, Россия, 14-17 октября 2013 г.**

«движок» RDF позволяет это делать на имеющихся данных. Однако, предполагается, что объем вовлекаемых в обработку данных может возрасти многократно, так что главная задача исследования было выяснение пределов, до которых подход будет «работать». Поскольку платформенное решение позволяет, через создание адаптера, подключать различные СУБД, все реализации разных подходов были сделаны в рамках единого средства, что в частности, упростило сравнение характеристик решений.

Для подключения конкретной СУБД требуется сформировать специфическую структуру базы данных удобную для существования в ней RDF, а также реализовать:

- создание базы данных;

- загрузку данных или в варианте потока триплетов, или в варианте потока RDF записей;

- реализовать «портретную» выборку данных о сущности по ее идентификатору;

- реализовать поиск по частичному совпадению образца и заданного поля.

Полный API платформенного решения требует также реализации нескольких дополнительных методов, связанных с редактированием данных, но это делалось не для всех решений в силу исследовательского характера работы.

Особо надо сказать о «портретной» выборки данных. Речь идет о выборке и специальном оформлении информации из окрестности заданного узла. Формируется XML-структура, соответствующая части RDF графа окрестности некоторого узла. В простом случае, такая структура содержит информацию об узле, свойствах данных для этого узла, ссылках (объектных свойствах), ведущих от этого узла и обратных ссылках (объектных свойствах), ведущих от других узлов к данному. Вид такого представления:

```
<record id="идентификатор узла" type="тип узла">
  <field prop="свойство данных 1">Значение свойства данных 1</field>
  <field prop="свойство данных 2">Значение свойства данных 2</field>
  …
  <direct prop="объектное свойство 1"><record id="ид1"/></direct>
  <direct prop="объектное свойство 2"><record id="ид2"/></direct>
  …
  <inverse prop="объектное свойство 3"><record id="ид3"/></inverse>
  < inverse prop="объектное свойство 4"><record id="ид4"/></inverse>
  …
</record>
```

В более сложном случае, эта структура формируется с использованием т.н. шаблонного дерева и представляет собой управляемую выборку окрестности узла.

Подход был обоснован и реализован в предыдущих работах [7, 8, 9, 10] и показал свою эффективность. Такой способ выборки данных отличается от рекомендаций Sparql [4], но их сферы применения не полностью пересекаются.

Все решения реализованы в обстановке Microsoft .NET, с использованием библиотеки .NET Framework, на языке C# с активным применением LINQ.

## 3 Использование реляционной СУБД

Движок EngineRDB был реализован вместе с платформенным решением и уже более года применяется в экспериментальных и прикладных работах. Он был реализован с использованием интерфейсов из пространства имен System.Data.Common, что позволяет легко адаптировать его к разным реляционным СУБД. Система испытывалась с MS SQL Server, MySQL, Sqlite.

Структура таблиц достаточно приближена к традиционному решению, когда организуют одну таблицу утверждений (триплетов) и колонками в таблице являются «субъект», «предикат» и «объект». Однако, сделано не столь прямолинейно. Во-первых, утверждения разбиты на объектные (object properties) и свойства данных (datatype properties). И соответственно, организованы две таблицы, также состоящие из трех столбцов, в которые отображаются утверждения. Кроме того, эти таблицы, для эффективности, реализованы в целых значениях для всех трех величин. Поэтому добавляется еще две таблицы для сущностей и данных, устанавливающие соответствие между идентификатором (для сущностей) или строкой (для данных) и их целым индексом. Индексы для сущностей «строгие», т.е. одинаковым идентификаторам обязательно сопоставляется один индекс. Такое требование нужно для того, чтобы сравнивать индексы, а не идентификаторы. Для данных, строгость не обязательна, хотя позволяет экономить память базы данных.

Также для сравнения было спроектировано и реализовано более классическое решение по использованию реляционных СУБД. База данных была спроецирована на две таблицы – таблицу объектных отношений и таблицу свойств данных.

## 4 Использование MongoDB

MongoDB [11] относится к классу NoSQL СУБД. Она ориентирована на обработку больших объемов данных и на эффективную их обработку, в том числе, на кластерных конфигурациях. Данные распределяются по нужному количеству таблиц. Таблицы хранят записи (документы, в терминах MongoDB) с обязательным первичным ключом. В этом смысле, MongoDB может рассматриваться как key-value хранилище. Записи представляют собой динамически сформированный набор колонок, хранящие скалярные и структурные значения. Структуризация осуществляется по принципам JSON (BSON в бинарном представлении) и легко сопрягается с большинством систем программирования, в первую очередь с JavaScript.

Колонки, хранящие скалярные данные и строки могут быть проиндексированы. Замечательным свойством MongoDB является то, что эффективно индексируются и колонки, содержащие вектора скалярных значений.

MongoDB «тяготеет» к укрупнению используемых структур данных. Поэтому была выбрана реализация RDF движка в виде одной таблицы, хранящей отдельные записи RDF-документов. Структура записи (документа) в синтаксисе C# выглядит следующим образом:

```
    public class EntityInfo
    {
        public ObjectId Id { get; set; } //
Системный идентификатор объекта
        public string LastId { get; set; } //
этот идентификатор и есть последний
        public string TypeId { get; set; } //
тип ресурса (сущности, записи) или null если не
известно
        private bool _isremoved = false;
        public bool IsRemoved { get { return
_isremoved; } set { _isremoved = value; } }
        public string[] MergedIds { get; set; }
// множество идентификаторов группы
эквивалентности, включая EntityId
        private DateTime _timestamp =
DateTime.MinValue; // последняя временная
отметка оригинала определения записи LastId
        public DateTime TimeStamp { get {
return _timestamp; } set { _timestamp = value;
} }
        // Следующее поле - для хранения полей
и объектных свойств записи
        public PredicateInfo[] RecordElements {
get; set; }
        // Следующее поле - для начальных слов
полей name
        public string[] FirstWords { get; set;
}
        // Следующее поле - набор внешних из
записи ссылок
        public string[] ExternalLinks { get;
set; }
    }
    public class PredicateInfo
    {
        public bool IsObjectProperty { get;
set; }
        public string Predicate { get; set; }
        public string Value { get; set; } //
это либо идентификатор объекта, либо строка
данных
        public string Lang { get; set; }
    }
```

Запись выглядит несколько усложненной, но это из-за того, что реализовывался вариант RDF, позволяющий осуществлять редактирование данных. В частности для этого используется поле IsRemoved истинность которого означает, что запись уничтожена. Также к редактированию относится поле MergedIds, хранящая цепочку эквивалентных идентификаторов и поле TimeStamp, предназначенное для фиксации времени изменения записи. Индексируется поле LastId для доступа к записи по ее идентификатору. Причем семантика идентификатора та, что он «последний» в наборе эквивалентных идентификаторов. Индексируются также два векторных поля: FirstWords и ExternalLinks. В вектор FirstWords помещаются первые слова вариантов, в том числе и языковых вариантов, имени объекта и используются для поиска по частичному совпадению с образцом. Вектор ExternalLinks имеет принципиальное значение в подходе. В него помещаются все (последние) идентификаторы объектных свойств, хранимых в RecordElements. Эта индексация позволяет экономно получать доступ ко всем записям, ссылающимся на заданный идентификатор.

## 5 Использование Open Link Virtuoso

Virtuoso фирмы Open Link Software [12], является коммерческой СУБД в которой, в частности, реализован RDF-движок с осуществлением запросов через Sparql. Фирма ставит перед собой амбициозные цели и хорошо известна в мире разработчиков, использующих RDF. К счастью, кроме дорогого коммерческого варианта СУБД, имеется свободно распространяемый вариант, который был использован для исследовательских целей.

Отображение RDF на Virtuoso и построение требуемого для платформы API – задача достаточно простая. Единственная существенная сложность оказалась в реализации поискового запроса. Рекомендуемый Sparql способ поиска по неполному совпадению через фильтрацию с использованием регулярного выражения, хоть и работает, но работает не быстро даже на небольших данных. Имеющееся в Virtuoso дополнительное средство имеет свои особенности применения.

## 6 База данных в оперативной памяти – RGraphEngine

Для целей сравнения, был адаптирован к платформенному решению наш «старый» движок RGraph. Это – прямая реализация RDF в виде графа в оперативной памяти. Такой вариант понадобился в исследовании для получения сопоставительных данных как «предельный» по скорости обработки и формирования выходных структур. Кроме того, такой движок вполне годится для проектов

небольшого и среднего размера. В частности, в ИСИ уже три года эксплуатируется публичный интерфейс фотоархива СО РАН. Архив пополняется новыми данными, так что внутренняя база данных, построенная на RGraph перезагружается ежедневно. Кроме того, собственно загрузка выполняется достаточно быстро. Например, база данных, содержащая 700 тыс. триплетов загружается за 6 секунд. Это быстрее, чем «холодный» запуск сложного сайта.

## 7 Использование Apache Cassandra

В качестве еще одной СУБД, которая вошла в систему экспериментов по реализации RDF, явилась Apache Cassandra, созданная и развиваемая компанией DataStax [14] совместно с вовлеченным в проект сообществом программистов. Это решение уже считается «ветераном» среди NoSQL СУБД и использовано в большом количестве коммерческих и некоммерческих систем. К сожалению, нам не удалось подобрать адаптер к Кассандре, позволяющий стабильно работать из C#/.NET. Тем не менее, кое-какие выводы относительно возможности ее использования для реализации RDF-хранилища удалось сделать.

Поскольку Кассандра считается хорошим Key-Value-хранилищем, схема реализации была основана на множестве RDF-записей, идентифицированных текстовыми ключами (использовались URI сущностей):

```
CREATE TABLE DB.Records (id text PRIMARY KEY,
type text, xbody text, invlist set<text>);
```

Для простоты, сама запись сохранялась в виде текста её XML-представления (поле xbody), а для эффективности, были добавлены в виде списка идентификаторы сущностей, ссылающихся на данную (поле invlist). Такая схема не достаточно удобна для расширения графа при его редактировании, но для целей, поставленных в исследовании, является вполне адекватной. Кроме построения информационного портрета, была реализована достаточно быстрая схема поиска сущностей заданных классов по образцу, хотя Кассандра и не приспособлена для подобных действий. Нестабильность адаптера СУБД не позволила загрузить данные большие, чем 2 млн. триплетов.

## 8 Специализированная СУБД FSRDF

Поскольку решения, базирующиеся на свободно распространяемых СУБД оказались неспособными загружать 1 млрд. триплетов, решено было попробовать написать свою упрощенную и специализированную СУБД, основанную на специальных файловых структурах. Такая СУБД была создана и погружена в платформенное решение. В итоге, удалось загрузить тест, имеющий миллиард триплетов и временные характеристики доступа к данным оказались весьма неплохими.

Решение основывается на трех файлах. Первый файл – собственно хранилище триплетов, организован как набор строк формата tsv (Tab Separated Values), где субъект, предикат и объект разделены традиционным символом-разделителем. Строки неупорядочены и записаны в порядке поступления при сканировании RDF-документов. Строка доступна через длинное целое – позицию в файле. Второй файл – бинарная файловая структура, хранящая наборы специально организованной информации для каждого объектного узла. Эта информация имеет ссылки (длинные целые) на триплеты, хранящиеся в первом файле. Один набор группирует множество триплетов в элементах которого идентификатор узла является либо субъектом, либо объектом высказывания.

Третий файл – достаточно просто организованный словарь, устанавливающий соответствие между идентификаторами объектных узлов и наборами второго файла.

Опыт и эксперименты показали, что даже на больших данных, RDF-хранилище работает довольно быстро и тратит мало оперативной памяти.

## 9 Методика проведения экспериментов и некоторые результаты

Проведенное исследование состояло в изучении возможностей реализации загрузки и работы графа RDF для разных подходов и СУБД и второе – изменение скоростей и других характеристик загрузки и обработки.

По первой части, можно констатировать, что почти для каждой СУБД, пришлось формировать адекватный подход к эффективному решению задачи. Причем, кроме логических и программистских задач, приходилось преодолевать значительное количество задач инженерных, связанных с особенностями развертывания, настройки и функционирования той или иной системы.

Для измерения характеристик были сформированы тестовые наборы данных по следующей схеме. Сначала была собрана из ряда имеющихся в ИСИ СО РАН источников тестовая база данных tm (two millions of triples), представляющая реальные, увязанные между собой данные в количестве около 2 млн. триплетов. Потом, путем мультиплицирования с изменением идентификаторов и имен объектов, были сделаны базы данных tm3 и tm10, представляющие 6 и 20 млн. триплетов соответственно. Для базы данных tm, в результате «ручного» просмотра в имеющемся интерфейсе, через навигационные действия и поисковые запросы, была создана последовательность запуска построения информационных портретов и поиска по образцу.

Была произведена трассировка на данном наборе запросов каждого из набора данных. Результаты таких трассировок, фиксировались в лог-файле. Потом были сведены воедино и обработаны.

Измерялись и вычислялись следующие характеристики: время формирования (загрузки) базы данных, время выполнения отдельных запросов на построение информационные портреты и объем получившихся информационных портретов, время выполнения поисковых запросов и количество получившихся вариантов.

Тестирование осуществлялось на рабочей станции с процессором Intel Core i7, 3.5 GHz, 16 Gb RAM, 64-разрядная операционная система Windows-7, .Net Framework 4.0. Результаты расчетов приведены в таблицах 1-3.

| | tm | tm3 | tm10 |
|---|---|---|---|
| mssql | 185 | 638 | 3700 |
| mongo | 26 | 70 | 359 |
| virtuoso | 402 | 800 | 2729 |
| rgraph | 17 | | |
| cassandra | 433 | | |
| fsrdf | 26 | 57 | 204 |

Таб. 1 Время загрузки наборов тестовых данных, сек.

| | tm | tm3 | tm10 |
|---|---|---|---|
| mssql | 360 | 1290 | 5733 |
| mongo | 272 | 816 | 2636 |
| virtuoso | 248 | 644 | 2425 |
| rgraph | 147 | | |
| cassandra | 14.6 | | |
| fsrdf | 1.9 | 1.6 | 2.3 |

Таб. 2 Среднее время поиска по текстовому образцу, мс.

| | tm | tm3 | tm10 |
|---|---|---|---|
| mssql | 92 | 380 | 391 |
| mongo | 1223 | 3190 | 10112 |
| virtuoso | 290 | 267 | 331 |
| rgraph | 3.4 | | |
| cassandra | 209 | | |
| fsrdf | 37 | 42 | 45 |

Таб. 3 Среднее время построения информационного портрета, мс.

## 10 Заключение

В докладе рассматриваются различные схемы реализации работы с RDF-данными. Показано, что традиционные и специализированные схемы отображения базы данных на СУБД позволяют решать основные задачи визуализации, поиска и навигации для данных среднего размера. Произведен сопоставительный анализ характеристик использования разных СУБД. Некоторые решения прошли опытную эксплуатацию и используются в реальных проектах.

Полученные результаты нельзя рассматривать как точную оценку возможностей различных СУБД для решения задач работы с RDF-информацией. Это зависит от слишком многих факторов, включая: примененные схемы реализации RDF-графа и вспомогательные структуры, улучшающие работу с этой информацией, инженерные конфигурационные настройки, качество адаптеров, операционная система и ее разрядность, количество ядер, объем имеющейся в наличии оперативной памяти, скорость работы дисковой памяти и др.

В рамках исследования был поставлен еще один вопрос: какие предельные объемы данных можно обрабатывать в рамках предложенных подходов и схем реализации? В качестве тестового материала был выбран набор данных freebase-rdf-2013-02-10-00-00.nt2, содержащий около 1 млрд. триплетов. При использовании усечения данного набора какой-то длины можно было проследить поведение той или иной СУБД. При росте объемов вводимых данных, увеличивались и трудности работы с конкретными СУБД. В большинстве случаев, все «ломалось» при объемах в десятки миллионов триплетов. Как правило, объемы захватываемой оперативной памяти становились слишком большими.

Все это привело к созданию простой специализированной СУБД FSRDF, ориентированной на работу с графами RDF и реализованную средствами файловой системы и прямого доступа к файлам. Оказалось, что современная стыковка файлов и виртуальной памяти настолько оптимизирована, что простое, но специализированное средство работы с RDF-данными может быть заметно более эффективным, чем настройка на эти цели универсальной СУБД. Приведенные цифры это показывают. Задача загрузки 1 млрд. триплетов была решена. Причем скорость построения упрощенного информационного портрета (множество исходящих из узла дуг и множество входящих) оказалась приемлемой и для работы в традиционном стиле Web-интерфейса.

## Литература


[1] White, Tom (10 May 2012). Hadoop: The Definitive Guide. O'Reilly Media. p. 3. ISBN 978-1-4493-3877-0.

[2] Berners-Lee Tim, Hendler James, Lassila Ora, The Semantic Web. In Scientific American, volume 284(5), pages 34-43, 2001.

[3] Web Ontology Language (OWL). — http://www.w3.org/2004/OWL

[4] SPARQL Query Language for RDF. - http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[5] Sören Auer, Lorenz Bühmann, Christian Dirschl, Orri Erling, Michael Hausenblas, Robert Isele, Jens Lehmann, Michael Martin, Pablo N. Mendes, Bert Van Nuffelen, Claus Stadler, Sebastian Tramp, and Hugh Williams. Managing the Life-Cycle of Linked Data with the LOD2 Stack.. In

Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist (Eds.), International Semantic Web Conference 2, (7650):1-16, Springer, 2012.

[6] Vocabularies, - http://www.w3.org/standards/semanticweb/ontology

[7] А.Г.Марчук, П.А.Марчук Платформа реализации электронных архивов данных и документов // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XIV Всероссийской научной конференции RCDL'2012. Переславль-Залесский, Россия, 15-18 октября 2012 г. – г. Переславль-Залесский: изд-во «Университет города Переславля», 2012, С. 332-338.

[8] Marchuk A.G. Methods and Technologies of Digital Historical Factography // Knowledge Processing and Data Analysis. First International Conference, KONT 2007, Novosibirsk, Russia, September 14-16, 2007,and First International Conference, KPP 2007, Darmstadt, Germany, September 28-30, 2007. Revised Selected Papers. Series: Lecture Notes in Computer Science, Vol. 6581, Subseries: Lecture Notes in Artificial Intelligence, Wolff, K.E.; Palchunov, D.E.; Zagoruiko, N.G.; Andelfinger, U. (Eds.), 2011, ISBN 978-3-642-22139-2, pp 217-231

[9] Марчук А.Г., Марчук П.А. Особенности построения цифровых библиотек со связанным контентом // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Сб.трудов / XII Всеросс. научн. Конф. RCDL'2010, Казань, Россия 13–17 октября 2010 г. — Казань: Казан. ун-т, 2010. — С. 19–23.

[10] Ануреев И.С. и др. Модели и методы построения информационных систем, основанных на формальных, логических и лингвистических подходах // отв. ред. А.Г. Марчук ; Рос. акад. наук, Сиб. отд-ние, Ин-т систем информатики им. А.П. Ершова. – Новосибирск: Изд-во СО РАН, 2009. – 330 с.

[11] Data Modeling Considerations for MongoDB Applications, - http://docs.mongodb.org/manual/core/data-modeling/

[12] Open Link Software, - http://www.openlinksw.com/Donald E. Knuth, Tracy L. Larrabee, and Paul M. Roberts. Mathematical Writing. Mathematical Association of America, 1989. http://www-cs-faculty.stanford.edu/knuth/klr.html

[13] Тим Бернерс-Ли, Джеймс Хендлер и Ора Лассила Семантическая Сеть – перевод [2] Евгения Золина, 2004. http://ezolin.pisem.net/logic/semantic_web_rus.html/

[14] Сайт компании DataStax // http://www.datastax.com

## On the Way to Big RDF Data

Alexander G. Marchuk

The goal of research work, reflected in the report was to determine the possibilities of working with large RDF data.

Some principles and details of different DBMS use for RDF data are presented. Various approaches were used to create efficient solutions for work with such graphs.

It was shown that traditional and specialized schemes of the RDF-graph on the database can solve the basic problems of visualization, search and navigation data of the medium size. A comparative analysis of the characteristics of the use of several database systems was done. Some solutions are currently used in real projects.