УДК 004.75

ФУНКЦИОНАЛЬНЫЕ ОСОБЕННОСТИ И ПРОИЗВОДИТЕЛЬНОСТЬ СИСТЕМЫ ХРАНЕНИЯ RDF-ДАННЫХ В РЕЛЯЦИОННЫХ СУБД GRAFFITI

Д.С. БОРОДАЕНКО

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки 6, 220013, Минск, Беларусь

Поступила в редакцию 30 марта 2010

Представлена разработанная автором система хранения RDF-данных Graffiti, реализующая динамическое преобразование RDF-запросов на основе графовых шаблонов в запросы SQL. Проведено сравнение функциональных особенностей и производительности Graffiti с существующими аналогами.

Ключевые слова: модель данных RDF, реляционные базы данных, метрика производительности BSBM.

Введение

В то время как гипертекстовые технологии Web объединили миллионы разнообразных документов в единое информационное пространство, возможности автоматизированной обработки доступных через Web данных остаются ограниченными. Это обусловлено тем, что форматы данных, используемые в Web, такие как HTML, рассчитаны на использование людьми, а не автоматическими агентами. Для решения этой проблемы консорциум W3C предложил парадигму Semantic Web, предполагающую расширение Web средствами семантической разметки, которая призвана обеспечить автоматическую интеграцию данных из различных источников.

Краеугольный камень Semantic Web — язык описания Web-ресурсов RDF (Resource Description Framework — инфраструктура описания ресурсов). Среди основных задач RDF: формирование открытых метаданных для Web-ресурсов; вынос обработки данных приложений за пределы среды, в которой они были созданы; совместная обработка данных, скомбинированных из различных приложений; автоматическая обработка информации, доступной через Web, при помощи программных агентов. Модель данных RDF включает графовую структуру данных на основе триплетов "субъект–предикат–объект", словарь идентификаторов URIref, типы данных, литералы, факты и правила логического следования [1].

Семантические системы, поддерживающие RDF, могут предоставлять доступ к RDF-данным посредством прикладных программных интерфейсов (API) либо языков запросов. Языки запросов обеспечивают более высокий уровень абстракции доступа к данным, чем API. Благодаря этому даже пользователи, не располагающие глубокими познаниями в области программирования и RDF, могут осуществлять поиск и извлечение RDF-данных, а опытные программисты могут коротко и ясно выражать сложные критерии выборки данных при построении семантических приложений.

Наиболее перспективным подходом к хранению RDF-данных считается интеграция RDF и реляционных данных [2, 3], позволяющая обеспечить семантический доступ к существующим базам данных и в то же время использующая испытанные технологии реляционных СУБД для повышения эффективности хранения и обработки RDF-данных.

Разработанная автором система хранения RDF-данных Graffiti обеспечивает доступ к данным посредством преобразования RDF-запросов на языке Squish в запросы к реляционной СУБД на языке SQL. При этом для доступа при помощи RDF-запросов может быть адаптиро-

вана любая таблица, соответствующая первой нормальной форме. Таким образом, любая существующая база данных может быть адаптирована для доступа через RDF, не теряя при этом обратной совместимости с существующими SQL-запросами. Алгоритмы преобразования запросов и логического вывода, используемые в Graffiti, подробно рассмотрены в [4], в данной статье предлагается анализ функциональных возможностей и производительности RDF-хранилища Graffiti по сравнению с ближайшими аналогами.

Функциональные возможности Graffiti

Подход на основе динамического преобразования RDF-запросов в SQL, использованный в системе хранения RDF-данных Graffiti, также был реализован проектами Federate [5], D2RQ [6] и Virtuoso [7]. Высокоуровневое сравнение возможностей языка RDF-запросов, реализованных данными проектами, приведено в табл. 1.

Таблица 1. **Выразительные возможности языка RDF-запросов**

Возможность	Graffiti	Federate	D2RQ	Virtuoso
Синтаксис SPARQL	Нет	Нет	Да	Да
Простой графовый шаблон	Да	Да	Да	Да
Условия фильтрации	Да	Нет	Да	Да
Необязательный графовый шаблон	Да	Да	Частично	Да
Вложенные группы графовых шаблонов	Частично	Нет	Нет	Да
Неопределенные предикаты	Нет	Нет	Нет	Да
Реификация утверждений	Да	Нет	Нет	Нет
Логический вывод для RDFS	Частично	Нет	Нет	Частично
Логический вывод для OWL	Частично	Нет	Нет	Частично
Агрегация результатов	Да	Нет	Нет	Да
Язык обновления данных	Да	Нет	Нет	Да

Из приведенного сравнения видно, что Graffiti, будучи одним из первых решений, реализовавших динамическое преобразование запросов из RDF в SQL, остается также одним из самых развитых с точки зрения языка запросов. Единственным существенным недостатком разработанной системы является отсутствие совместимости со стандартом SPARQL [8]. В то же время, существуют аспекты, в которых данная система превосходит возможности аналогов, например — реификация утверждений RDF.

Также видно, что из рассмотренных аналогов только RDF-хранилище Virtuoso обеспечивает сравнимый набор выразительных возможностей языка запросов. Вместе с тем, между системами Graffiti и Virtuoso есть и существенные различия как в используемых алгоритмах преобразования запросов, так и в архитектуре программной реализации.

Так, Virtuoso не поддерживает реификацию утверждений RDF, а для логического вывода на правилах для словарей метаданных RDFS и OWL использует прямой перебор вариантов при помощи операции объединения (UNION) языка SQL. Такой подход к логическому выводу обеспечивает большую гибкость, чем используемые в Graffiti хранимые процедуры, но является крайне неэффективным с точки зрения производительности, особенно в случае обработки транзитивных отношений.

Отличия Virtuoso и Graffiti в архитектуре программной реализации еще более существенны. Virtuoso — массивная и в большой степени монолитная система хранения данных, решающая ряд задач из целого набора предметных областей, в том числе и не имеющих отношения к RDF и Semantic Web. Минимальный комплект исходных кодов Virtuoso занимает более 60 Мб, немалую часть этого объема занимает собственная реализация реляционной СУБД, а также локально модифицированные копии стороннего свободного ПО. Основной недостаток такого подхода – сравнительно высокая сложность и трудоемкость внедрения и поддержки, что делает использование Virtuoso нецелесообразным, если предполагается использовать только небольшую часть предлагаемого данным проектом комплекта функциональных возможностей, или если часть реализованных в Virtuoso возможностей пересекается с уже внедренными в существующую информационную систему программными средствами.

В отличие от Virtuoso, система хранения RDF-данных Graffiti — отдельный модуль, насчитывающий всего 1000 строк кода на языке Ruby и 200 строк кода на PL/SQL, и решающий одну задачу: преобразование RDF-запросов в запросы к реляционной СУБД. Решение смежных задач делегируется внешним программным средствам, что существенно облегчает интеграцию Graffiti в существующие информационные системы.

Оценка производительности RDF-хранилища

Для оценки производительности разработанной системы хранения RDF-данных была выбрана метрика BSBM (Berlin SPARQL Benchmark) [9], разработанная специально для тестирования систем на основе отображения реляционных данных. Еще одним фактором в пользу данной метрики является выбор электронной коммерции в качестве предметной области и использование типовой последовательности запросов в вычислении оценки производительности, что делает метрику BSBM особенно показательной при оценке практических возможностей семантических систем.

Метрика BSBM создает набор данных о производителях, поставщиках, торговых предложениях и обзорах для выбранного количества товаров, и обеспечивает возможность генерации тестовых наборов данных не только в виде триплетов RDF, но и в виде реляционной базы данных. В качестве совокупного показателя производительности измеряется скорость обработки специально подобранной последовательности запросов, соответствующей типовому сценарию использования приложения электронной коммерции.

Во время адаптации методики BSBM для тестирования RDF-хранилища Graffiti в процесс тестирования были внесены незначительные изменения по сравнению с эталоном. Стандартный тестовый драйвер BSBM, написанный на Java и использующий язык запросов SPARQL, не совместим с Graffiti, поэтому вместо него был разработан драйвер, дублирующий функциональность стандарного драйвера, со следующими изменениями: использование запросов на языке Squish вместо SPARQL, использование параметризованных запросов, использование интерфейса Ruby API вместо протокола SPARQL, отсутствие квалификационного режима. Вместо автоматической квалификации на уровне тестового драйвера, результаты выполнения тестовых запросов проверялись вручную.

Использовались следующие версии программного обеспечения: ОС Debian squeeze/sid, ядро Linux 2.6.30, СУБД PostgreSQL 8.4.0, интерпретаторы Java 1.6.0.15 и Ruby 1.9.0, библиотеки JDom 1.1, ruby-pg 0.8.0, ruby-dbi 0.4.2, ruby-dbd-pg 0.3.8, SynCache 1.0 и Graffiti 1.0. В тестовом стенде было использовано следующее аппаратное обеспечение: процессор Intel Core 2 Duo E6850 2,66 ГГц с 4 Мб кэша L2; 2 Гб ОЗУ PC3-10600; жесткий диск SATA 500 Гб 10000 rpm. Данный стенд соответствует стенду, использованному в официальных тестах BSBM, по всем параметрам, кроме объема и модели ОЗУ (2 Гб ОЗУ вместо 8 Гб, DDR3 вместо DDR2), и количества ядер процессора (2 вместо 4, что не должно было повлиять на однопоточные тесты).

Тестирование проводилось на 5 наборах данных: помимо стандартных 1, 25 и 100 млн. триплетов, были протестированы наборы в 250 тыс. и 5 млн. триплетов, что позволило более точно оценить изменение производительности Graffiti с ростом объема обрабатываемых данных, как в пределах доступного объема ОЗУ, так и при выходе за эти пределы. Количественные параметры использованных тестовых наборов данных приведены в табл. 2.

Таблица 2. **Тестовые наборы данных**

Набор данных	250K	1M	5M	25M	100M
Количество товаров	666	2810	14208	70812	284826
Объем данных на диске	65 Мб	260 Мб	1,1 Гб	5,2 Гб	21 Гб

Совокупная производительность разработанной системы для полной серии запросов, а также производительность на отдельных запросах метрики, приведены в табл. 3.

Использование аналогичной конфигурации тестового стенда позволяет сравнивать полученные результаты с опубликованными в [9] результатами тестирования систем хранения RDF-данных Jena SDB и Sesame NS, использующих таблицы триплетов, и систем D2RQ и Virtuoso, реализующих преобразование RDF-запросов в реляционную модель данных. На

наиболее показательном наборе данных 1M Graffiti превосходит ближайшую по производительности систему Sesame на 40%. На наборе данных 25M отрыв от ближайшего конкурента (на этом наборе среди аналогов лидирует система Virtuoso) достигает 44%, несмотря на то, что объем данных этого набора в 2,6 раз больше объема ОЗУ стенда, использованного при тестировании разработанной системы, но в 1,5 раз меньше объема ОЗУ стенда, использованного разработчиками метрики BSBM. Только на наборе данных 100M производительность разработанной системы существенно снижается, но и на этом наборе из всех аналогичных систем только Virtuoso показывает более высокую совокупную производительность.

Таблица 3. **Производительность RDF-хранилища Graffiti по метрике BSBM**

Набор данных	250K	1M	5M	25M	100M
Полных серий запросов в час	25896	25287	23493	18735	2246
Сокращенных серий запросов в час	28184	28986	27534	24884	2356
Запросов в секунду, запрос 1	245	245	198	197	76
Запросов в секунду, запрос 2	253	263	249	237	96
Запросов в секунду, запрос 3	240	243	199	196	101
Запросов в секунду, запрос 4	180	178	140	118	63
Запросов в секунду, запрос 5	94	74	47	23	9
Запросов в секунду, запрос 6	999	953	993	931	809
Запросов в секунду, запрос 7	60	60	60	55	4
Запросов в секунду, запрос 8	332	342	344	190	10
Запросов в секунду, запрос 9	406	351	407	406	55
Запросов в секунду, запрос 10	732	725	753	532	14
Запросов в секунду, запрос 11	375	362	370	213	17
Запросов в секунду, запрос 12	533	526	538	527	34

Среди причин более высокой измеренной производительности RDF-хранилища Graffiti по сравнению с результатами аналогичных систем, полученными разработчиками метрики BSBM, можно выделить следующие:

1) построение более эффективных SQL-запросов за счет учета в алгоритмах преобразования частных случаев, позволяющих сократить количество реляционного соединения (что особенно заметно на запросах BSBM 2 и 8);

2) выполнение правил логического вывода на уровне хранимых процедур СУБД (наиболее показательный для этой особенности случай транзитивных свойств не задействован в запросах BSBM);

3) сокращение объема передаваемых между процессами данных за счет использования для взаимодействия между тестовым драйвером и RDF-хранилищем программного интерфейса Ruby вместо сетевого протокола HTTP;

4) разделение преобразования и выполнения запроса между двумя процессами (интерпретатор Ruby и сервер PostgreSQL), позволяющее задействовать два ядра процессора даже в однопоточных тестах;

5) более эффективное кэширование преобразованных запросов за счет использования параметризованных запросов;

6) использование полнотекстового индекса PostgreSQL для поиска по подстроке повышает производительнось обработки запроса 6 метрики BSBM на два порядка.

Заключение

Разработанная автором система хранения RDF-данных Graffiti превосходит большинство существующих систем отображения реляционных данных на RDF по выразительным возможностям реализованного языка RDF-запросов. Использованная программная архитектура существенно упрощает внедрение Graffiti в существующие информационные системы. Тестирование производительности по метрике BSBM показало, что разработанная система заметно превосходит все аналоги по производительности обработки RDF-запросов, что позволяет внедрять семантические системы на платформах с ограниченными аппаратными ресурсами (встроенные системы, виртуальные серверы).

DISTINGUISHING FEATURES AND PERFORMANCE OF THE GRAFFITI RELATIONAL DBMS BACKED RDF STORE

D.S. BORODAENKO

Abstract

The article presents Graffiti RDF store, which implements dynamic translation of graph pattern based queries into SQL, and compares functionality and performance of the presented RDF store with similar solutions.

Литература

1. *Klyne G., Carroll J.J.* // Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. 2003.
2. *Malhotra A.* // W3C RDB2RDF Incubator Group Report. W3C Incubator Group Report. 2009.
3. *Auer S., Dietzold S. Lehman J. et al.* // Proc. of the 2009 WWW Conference. 2009.
4. *Borodaenko D.* // Proc. of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems. 2009. Vol. 5. P. 413.
5. *Prud'hommeaux E.* // RDF Access to Relational Databases. W3C Technical Report. 2003.
6. *Bizer C., Seaborne A.* // D2RQ-Treating non-RDF databases as virtual RDF graphs. In: ISWC 2004 (posters). 2004.
7. *Erling O., Mikhailov I.* // Proc. of the 1st Conference on Social Semantic Web. Vol. P-113 of GI-Edition — Lecture Notes in Informatics (LNI). Bonner Ko llen Verlag. 2007.
8. *Prud'hommeaux E., Seaborne A.* // SPARQL Query Language for RDF. W3C Recommendation. 2008.
9. *Bizer C., Schultz A.* // Proc. of the 4th International Workshop on Scalable Semantic Web knowledge Base Systems. 2008.