УДК 004.822:514

АППАРАТНО-ПРОГРАММНАЯ ПОДДЕРЖКА МЕТОДА ОБРАБОТКИ ДАННЫХ И ЗНАНИЙ В ИНТЕРНЕТЕ С ЭЛЕМЕНТАМИ ШИФРОВАНИЯ

В.А. ВИШНЯКОВ, Д.С. БОРОДАЕНКО

Минский институт управления Лазо, 12, Минск, 220012, Беларусь

ЕПАМ, Радиальная, 40, Минск, 220070, Беларусь

Поступила в редакцию 18 марта 2014

Модификация алгоритма динамического преобразования RDF-запросов в методе отображения реляционных структур данных на модель RDF позволяет поддерживать шифрование путем использования операций объединения для кодирования запросов с неоднородным отображением на реляционную модель. Предложены два варианта поддержки модифицированного метода. При небольших объемах запросов пользователей (до десятка тысяч) представлена программная реализация, описанная в терминах классов. Для больших объемов запросов пользователей (десятки и сотни тысяч) – рассмотрена аппаратная реализация подключения к Web-серверу специализированного процессора на основе ПЛИС.

Ключевые слова: обработка данных и знаний, аппаратно-программная поддержка, ПЛИС, спецпроцессор, шифрование.

Введение

Для обработки данных и знаний в интернете широко используется модель данных RDF, которая опирается на следующие ключевые понятия [1]: графовая структура данных, словарь идентификаторов URIref, типы данных, литералы, факты, правила логического следования. Технологии RDF поддерживает представление знаний в Web-пространстве для автоматической обработки. Огромное количество накопленной информации, хранимой в интернете (в том числе и закрытой), представлено в реляционных базах данных (БД), быстрый доступ к которым семантическими средствами затруднен. Среди препятствий на пути массового внедрения RDF в Web: необходимость переноса уже накопленной информации на модель знаний RDF и невысокая производительность семантических систем при обработке больших объемов RDF информации. Перспективным подходом к решению обоих препятствий является интеграция RDF и реляционных баз данных [1] с эффективной программной или аппаратной поддержкой.

Методика эксперимента

Для отображения на структуру RDF и обеспечивается обработка RDF-запросов на доступ и обновление реляционных данных, используется модель адаптации реляционных данных, представленная в виде двойки [2]: $\{M, N\}$, где M – отображения реляционной модели данных на модель RDF, позволяющие создавать утверждения RDF на основе значений полей записей реляционных таблиц (реляционная таблица соответствует классу RDF-ресурсов, запись – RDF-ресурсу, значение первичного ключа – субъекту, имя поля – предикату, значение – объекту утверждения RDF); N – единое пространство имен (первичных ключей) для всех RDF-ресурсов, отображенных из записей реляционных таблиц, а также RDF-ресурсов, описываемых утверждениями, хранимыми в таблице триплетов, что позволит вносить в нее утверждения, использующие в позициях субъекта, предиката и объекта любые RDF-ресурсы. Обработка RDF-запросов определяется тройкой [2]: $\{Aq, An, P\}$, где: Aq – алгоритм

преобразования запросов к данным RDF в запросы SQL; *An* – алгоритм обновления реляционных данных по запросу RDF; *R* – разбор и преобразование RDF-запросов и команд обновления данных в запросы и команды к реляционной СУБД на стандартном языке SQL.

По мере востребованности в конкретных приложениях от системы хранения RDF-данных также может потребоваться поддержка дополнительных возможностей. Набор алгоритмов, входящих в метод, обеспечивает поддержку следующих расширений:

– реификация (представление в виде самостоятельных ресурсов) утверждений RDF, что позволяет составлять утверждения об экземпляре утверждений [3];

– применение правил логического вывода при преобразовании RDF-запросов для учета в результатах выполнения запросов отношений подкласс-суперкласс, заданных предикатом *rdfs:sub Class Of* (так вышеупомянутое создание единого пространства имен равносильно включению всех отображаемых классов ресурсов RDF на суперкласс; *rdfs: Resource;*

– применение правил логического вывода для учета подотношений, определенных при помощи предиката *rdfs: sub Property Of,* указывающего, что все утверждения, верные для подотношения, также верны и для базового отношения;

– применение правил логического вывода для учета транзитивных отношений, входящих в класс предикатов *owl'TransitiveProperty* (примером практического применения транзитивного отношения может быть выборка всех комментариев к заданному сообщению вне зависимости от уровня вложенности).

В работе [3] представлена структура разработанного метода семантического доступа к данным на основе отображения реляционных БД на модель данных RDF. Суть метода заключается в интеграции новой модели адаптации реляционных данных. Для реализации обработки разработаны новые алгоритмы преобразования запросов к данным RDF в запросы SQL и обновления реляционных данных по запросу RDF. Следует отметить, что существующие системы хранения RDF-данных ограничиваются применением правил логического вывода на уровне приложения, что упрощает реализацию таких систем, но существенно снижает производительность обработки запросов.

Разработанный метод семантического доступа, в отличие от аналогов, использует реализацию логического вывода на уровне хранимых процедур реляционной СУБД. Механизм хранимых процедур позволяет в процессе обновления данных создавать и поддерживать вспомогательные структуры, обеспечивающие выборку данных с учетом всех заданных правил логического вывода посредством одного запроса SQL. Наиболее известный пример такой структуры – транзитивное замыкание, сводящее проверку истинности транзитивного отношения до одной операции выборки.

Модель адаптации реляционных данных не накладывает дополнительных ограничений на используемую схему реляционной базы данных сверх ограничений стандарта SQL. Любая таблица *T* в первой нормальной форме может быть отображена для доступа при помощи RDF-запросов. Таким образом, любая существующая база данных может быть адаптирована для доступа через RDF, не теряя при этом обратной совместимости с существующими SQL-запросами.

Процесс адаптации включает добавление в базу данных атрибутов, внешних ключей, таблиц и хранимых процедур, необходимых для преобразования запросов RDF и поддержки дополнительных возможностей, предлагаемых разработанной системой, таких как реификация утверждений и логический вывод на правилах для *rdfs:sub Class Of rdfs: sub Property Of и owhTransitiveProperty* [4]. Разработанная модель подробно описана в [2].

Алгоритм преобразования RDF-запроса в запрос SQL. Приведенное в работе [3] описание отображаемых реляционных данных и средств логического вывода позволяет задать следующие входные данные алгоритма:

– набор отображений $M = \langle M_{rel}, M_{attr}, M_{sub}, M_{trans} \rangle$, где $M_{rel} : P \to R$, $M_{attr} : P \to \Phi$, $M_{sub} : P \to S$, $M_{trans} : P \to T$; *P* – множество отображенных отношений RDF, *R* – множество реляционных таблиц, Φ – множество реляционных атрибутов, $S \subset P$ – подмножество отношений RDF, для которых заданы подотношения, $T \subset R$ – множество транзитивных замыканий;

– графовый шаблон $\Psi = \langle \Psi_{nodes}, \Psi_{arcs} \rangle = \Pi \cup N \cup \Omega$, где Π, N и Ω – основной, отрицательный и необязательный графовые шаблоны соответственно, такие что Π, N и Ω не имеют общих дуг и при этом Π, $\Pi \cup N$ и $\Pi \cup \Omega$ образуют связные графы;

– глобальное условие фильтрации $F_g \in F$ и локальные условия $F_c : \Psi_{arcs} \to F$, где F – множество всех условий над литералами, формируемыми в синтаксисе языка запросов Squish.

Результатом алгоритма является выражение реляционного соединения F и условие W, готовые для включения в разделы FROM и WHERE запроса на языке SQL соответственно. Помимо вышеуказанных входных данных и результатов, в описании алгоритма также используются следующие обозначения: $\mathrm{id}(r)$ – первичный ключ таблицы $r \in R$; $\rho(n)$ – значение $\mathrm{id}(Resource)$ для фиксированной (не являющейся переменной) вершины $n \in \Psi_{nodes}$, если такое значение известно во время трансляции запроса.

Алгоритм преобразования разделен на следующие шаги.

1. Пометить каждый связный компонент Π, N и Ω разными цветами K, такими что $K_\Pi : \Pi_{nodes} \to \mathbb{K}$, $K_N : N_{nodes} \to \mathbb{K}$, $K_\Omega : \Omega_{nodes} \to \mathbb{K}$, $K(n) = K_\Pi(n) \cup K_N(n) \cup K_\Omega(n)$. Используется двухпроходный алгоритм разметки связных компонентов [3], модифицированный для исключения вершин, присутствующих в Π, из списков соседей, используемых при разметке N и Ω. Данная модификация гарантирует, что части N и Ω, связанные только через вершину, входящую в Π, будут помечены разными цветами.

2. Отобразить каждую дугу $c = \langle s, p, o \rangle \in \Psi_{arcs}$ на реляционную модель данных в соответствии с M: определить отображение $M_{attr}^{pos} : \Psi_{arcs} \times \Psi_{nodes} \to \Phi$ такое что $M_{attr}^{pos}(c, s) = \mathrm{id}(M_{rel}(p))$, $M_{attr}^{pos}(c, o) = M_{attr}(p)$; заменить каждую дугу с неотображенным предикатом на ее реификацию и отобразить утверждения реификации в соответствии с M; для каждой дуги, предикат которой является подотношением, добавить дугу, отображенную на соответствующий атрибут различения подотношений. Для каждой вершины $n \in \Psi_{nodes}$, найти смежные дуги $\Psi_{nodes}^n = \{\langle s, p, o \rangle \mid n \in \{s, o\}\}$ и определить ее режим связывания $\beta_{node} : \Psi_{nodes} \to \{\Pi, N, \Omega\}$ такой что $\beta_{node}(n) = \max(\beta_{arc}(c) \forall c \in \Psi_{nodes}^n)$, где $\beta_{arc}(c)$ отражает, который из графовых шаблонов $\{\Pi, N, \Omega\}$ содержит дугу c, а оператор max использует порядок следования $\Pi > N > \Omega$.

3. Отобразить каждую вершину в Ψ на набор псевдонимов таблиц $a \in \mathbb{A}$ в соответствии с алгоритмом, описанным на рис. 1. Указанный алгоритм определяет отображение $C_a : \Psi_{arcs} \to \mathbb{A}$, связывающее каждую дугу в Ψ с псевдонимом, и набор отображений $A = \langle A_{rel}, A_{node}, A_\beta, A_{filter} \rangle$, где $A_{rel} : \mathbb{A} \to R$, $A_{node} : \mathbb{A} \to \Psi_{nodes}$, $A_\beta : \mathbb{A} \to \{\Pi, N, \Omega\}$, $A_{filter} : \mathbb{A} \to F$, которые задают таблицу, вершину, режим связывания и условие фильтрации для каждого псевдонима.

4. Определить связки $B : \Psi_{nodes} \to \mathbb{B}$, где $\mathbb{B} = \{\{\langle a, f \rangle \mid a \in \mathbb{A}, f \in \Phi\}\}$, отображающие вершины графового шаблона на наборы пар псевдонимов таблиц и атрибутов, таких что $\langle a, f \rangle \in B(n) \Leftrightarrow \exists c \in \Psi_{arcs}^n : C_a(c) = a, M_{attr}^{pos}(c, n) = f$.

Преобразовать графовый шаблон Ψ в граф реляционного запроса $Q = \langle \mathbb{A}, J \rangle$, где вершины $\mathbb{A}$ – определенные выше псевдонимы таблиц, а грани $J = \{\langle b_1, b_2, n \rangle \mid b_1 = \langle a_1, f_1 \rangle \in B(n), b_2 = \langle a_2, f_2 \rangle \in B(n), a_1 \neq a_2\}$ – условия реляционного соединения. Связать фиксированные (не являющиеся переменными) вершины графового шаблона со значениями в соответствии с алгоритмом, представленным в [3]. Составить список связанных вершин $G \subseteq \Psi_{nodes}$, такой что

$n \in G \Leftrightarrow n \in F_g \vee \exists \langle b_1, b_2, n \rangle \in J \vee \exists b \in B(n) \exists a \in \mathbb{A} : b \in A_{filter}(a)$.

5. Вычислить для графа реляционого запроса Q упорядоченное связное минимальное покрытие P деревьями с непересекающимися множествами граней, такое что $\forall P_i \in P$ $\forall j = \langle b_{j1}, b_{j2}, n_j \rangle \in P_i \;\; \forall k = \langle b_{k1}, b_{k2}, n_k \rangle \in P_i : K(n_j) \cap K(n_k) \neq \varnothing \wedge \beta_{node}(n_j) = \beta_{node}(n_k) = \beta_{tree}(P_i)$, начинающееся с P_1 такого что $\beta_{tree}(P_1) = \Pi$ (из определений графа Ψ и шага 4 алгоритма следует, что P_1 – единственное такое дерево и что оно покрывает все условия соединения $\langle b_1, b_2, n \rangle \in J$ такие что $\beta_{node}(n) = \Pi$).

6. Записать P_1 в виде корневого внутреннего соединения. Записать остальные деревья из P, имеющие не менее одной грани, в виде подзапросов. Сформировать выражение F как операцию левостороннего внешнего соединения P_1 со всеми подзапросами, а также с псевдонимами таблиц, представляющими вырожденные деревья из P, состоящие из одной вершины. Для каждого P_i такого что $\beta_{tree}(P_1) = \Pi$, найти связку $b = \langle a, f \rangle \in P_i$ такую что $a \in P_1 \cap P_i$ и добавить в W условие (b IS NULL). Для каждой несвязанной вершины $n \notin G$ такой что $\langle a, f \rangle \in B(n) \wedge a \in P_1$, добавить в W условие (b IS NOT NULL), если $\beta_{node}(n) = \Pi$, либо условие (b IS NULL), если $\beta_{node}(n) = \Pi$. Добавить в W глобальное условие F_g.

Для поддержки шифрования адаптируется модуль хранения RDF к более широком спектру проблемных областей. В этом случае алгоритм преобразования запросов модифицируется добавлением операций объединения для кодирования запросов с неоднородным отображением на реляционную модель. Конфигурирование реляционной схемы приводится к более общему виду, включая поддержку композитных ключей и более гибкие хранимые процедуры для логического вывода и реификации утверждений.

Результаты и их обсуждение

Программная реализация. Разработка архитектуры программной реализации велась с использованием UML [5] языка графического описания для объектного моделирования в области разработки ПО. Язык UML уже более 10 лет является стандартом де-факто в области проектирования ПО. Структура разработанной программной системы хранения и доступа к RDF-данным Graffiti представлена в виде диаграммы классов на рис. 1. Приведем описание основных классов разработанной системы. Высокоуровневая структура разработанной программной системы хранения и доступа к RDF-данным Graffiti представлена в виде диаграммы классов: RdfConfig – конфигурация отображения RDF; Store – реализация прикладного программного интерфейса выполнения запросов; SquishQuery, SquishSelect, SquishAssert – разбор запросов и команд обновления, SqlMapper, SqlExpression, SqlNodeBinding реализуют преобразование запроса в SQL. В класс RdfConfig записывается конфигурация отображения RDF, описанная в [2]. Помимо хранения списка пространств имен, используемых для сокращения идентификаторов URIref, и отображения отношений RDF на таблицы и поля реляционной схемы данных, данный класс также предоставляет ряд сопутствующих служебных функций, таких как преобразование идентификаторов URIref из сокращенной формы в полную и обратно. Класс Store обеспечивает прикладной программный интерфейс, позволяющий внешним приложениям выполнять RDF-запросы. При инициализации экземпляру класса передаются в качестве входных параметров открытое соединение с реляционной СУБД и содержимое конфигурационного файла, достаточное для формирования экземпляра класса RdfConfig.

Таким образом, реализация класса не привязана ни к соединению к СУБД, ни к конкретной конфигурации отображения RDF, что позволяет свободно рекомбинировать разные СУБД и разные схемы отображения без изменения реализации класса Store. Синтаксис языков запросов Squish и SQL реализован классом SquishQuery и его подклассами SquishSelect, реализующим разбор запроса на извлечение данных, и SquishAssert, реализующим разбор и выполнение запроса на обновление данных. Оба подкласса используют алгоритм преобразования графовых шаблонов RDF в условия выборки данных на языке SQL, реализованный в классе SqlMapper.

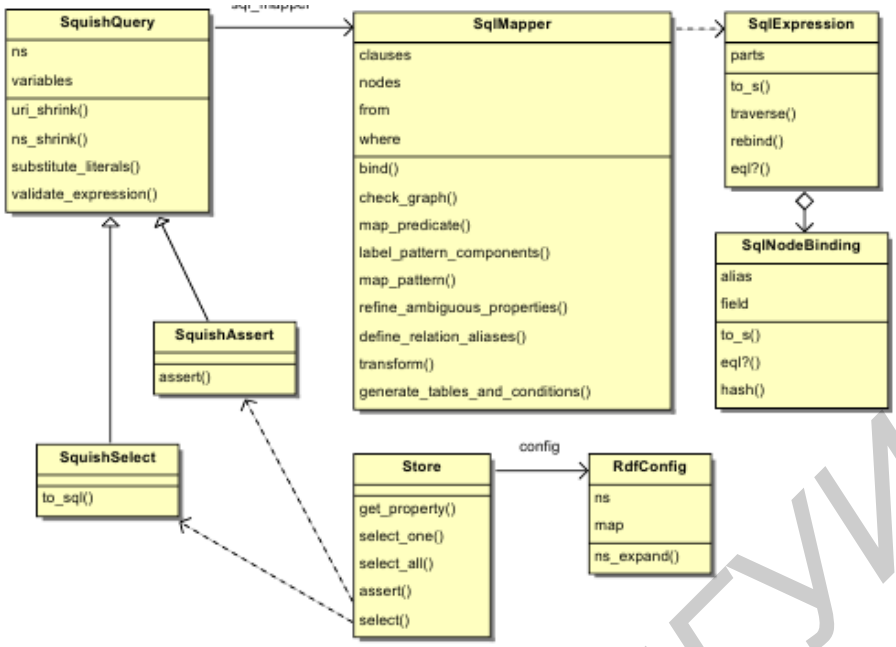

Рис. 1. Диаграмма классов разработанной системы хранения данных

Методы класса SqlMapper, представленные в данной последовательности, соответствуют шагам алгоритма преобразования запроса, описанного выше. Алгоритм преобразования запроса достаточно сложен и вычислительно емок и может оказывать влияние на производительность всей системы. Чтобы компенсировать потенциальные потери производительности, разработанная система хранения RDF-данных поддерживает внутренний кэш преобразованных запросов и позволяет использовать параметризованные запросы, которые увеличивают частоту попадания в кэш, защищают приложение от атак класса «SQL injection» (вставка в запросы пользовательского SQL-кода вместо данных).

Аппаратная реализация. Для решения задач отображения реляционных БД на модель RDF с поддержкой шифрования предлагается вычислительная платформа, как аппаратный ускоритель к Web-серверу для обработки знаний в реальном времени. Аппаратный ускоритель –это специализированный процессор, реализующий модифицируемый алгоритм преобразования запросов на доступ к RDF в запросы SQL, который подключается к серверу через высокоскоростной интерфейс. Основные требования, которые предъявляются к такой реализации: аппаратный ускоритель должен быть специализирован под решение сложной вычислительной задачи для достижения максимальной производительности; возможность изменения алгоритма обработки без переделки спецпроцессора; спецпроцессор должен иметь высокоскоростной интерфейс для обмена с сервером. Спецпроцессор может быть реализован на ПЛИС, что позволит изменить алгоритм обработки данных путем коррекции последовательности, загружаемой в интегральную схему. Имеется большое количество плат для создания прототипов устройств на базе ПЛИС, которые могут быть использованы как спецпроцессор, однако они обладают избыточностью. В качестве ПЛИС для спецпроцессора можно рекомендовать кристалл XC3S 400 Spartac III Xilins [6], который выбран из соображений максимальной емкости (400000 логических вентилей). Требованиям интерфейса для обмена данными с сервером удовлетворяет USB 2.0 (до 480Мб/с). Для хранения корректируемой последовательности будет использоваться постоянное запоминающее устройство (ПЗУ), последовательность заносится в него через интерфейс JTAG, что позволит конфигурировать спецпроцессор из сервера. Задача шифрования передаваемых данных (криптографические алгоритмы реализуются на ПЛИС, которая включена в состав платформы.

Схема обработки RDF-знаний с аппаратной поддержкой приведена на рис. 2. Данная схема – наиболее универсальный вариант развертывания, предполагающий разделение системы между максимально возможным числом серверов и процессов. Такое разделение позволяет оптимизировать масштабируемость отдельных компонентов системы, но не является обязательным.

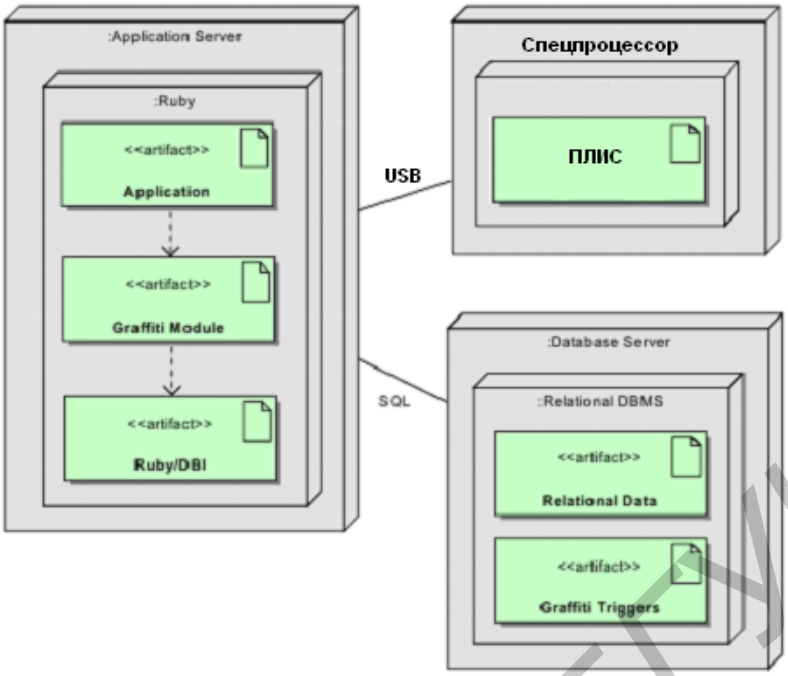

Рис. 2. Схема обработки RDF-знаний с аппаратной поддержкой

Заключение

В результате выполненной разработки для метода обработки данных и знаний в интернете с поддержкой шифрования [2] предложено два варианта поддержки. Это может быть программная реализация, а для обслуживания большого количества пользователей (десятки, сотни тысяч) и аппаратная. Высокоуровневая структура разработанной программной системы хранения и доступа к RDF-данным Graffiti представлена в виде диаграммы классов. Для решения задач отображения реляционных БД на модель RDF с поддержкой шифрования предлагается вычислительная платформа (спецпроцессор на базе ПЛИС), как аппаратный ускоритель к Web-серверу для обработки знаний в реальном времени.

HARD- AND SOFTWARE SUPPORT OF DATA AND KNOWLEDGE PROCESSING METHOD WITH CIPHERING ELEMENTS FOR INTERNET

V.A. VISHNIAKOV, D.S. BORODAENKO

Abstract

The algorithm modification of dynamic transformation RDF queries in method for mapping relational DB to the RDF model allows to support the ciphering by the way of unity operations for queries coding with inhomogeneous mapping on relational model. Two variants of method of mapping relational DB to the RDF model are proposed. The software realization for medium volume of user queries (until ten thousand) in class terms is proposed. The hardware realization of this method for large volume of user queries (tens and hundreds of thousands) by including to web-server the special processor on PLIS is proposed.

Список литературы

1. *Berners-Lee T., Hendler J., Lassila O.* // Scientific American. May, 2001. P. 28–37.
2. *Вишняков В.А., Бородаенко Д.С.* // Докл. БГУИР. 2013. № 8 (62). С. 89–94.
3. *Бородаенко Д.С.* // Докл. БГУИР. 2010. № 2 (48). С. 84–89.
4. *Brickley D.* RDF Vocabulary Description Language 1.0.
5. *Фаулер М.* UML основы. СПб, 2005.
6. *Кузелин М.О., Кнышев Д.А., Зотов В.Ю.* Современные системы ПЛИС фирмы Xilins. М., 2004.