УДК 004.75

ОТОБРАЖЕНИЕ РЕЛЯЦИОННЫХ ДАННЫХ НА СЕМАНТИЧЕСКУЮ МОДЕЛЬ RDF ПРИ ПОМОЩИ ДИНАМИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ЗАПРОСОВ

Д.С. БОРОДАЕНКО

ООО "Алатис"
Мележа 5, к. 2, Минск, 220090, Беларусь

Поступила в редакцию 26 ноября 2009

Описывается алгоритм динамического преобразования RDF-запросов, позволяющий отображать любые реляционные структуры данных на модель RDF и выполнять запросы к комбинации отображенных реляционных данных и произвольных триплетов RDF с производительностью на уровне реляционных систем управления базами данных (СУБД). Выразительные возможности реализованного предложенным алгоритмом языка запросов включают необязательные и отрицательные графовые шаблоны, вложенные шаблоны и логический вывод на правилах для подмножества словарей метаданных RDFS и OWL.

Ключевые слова: модель данных RDF, реляционные СУБД, язык запросов

Введение

Для повышения степени автоматизации доступа к данным в Web консорциум W3C, ведающий стандартизацией технологий Web, с 1997 г. работает над набором технологий Semantic Web. Semantic Web — глобальная сеть знаний, представленных таким образом, чтобы автоматические программные агенты могли собирать данные из разных источников, обрабатывать их и обмениваться результатами с другими программами, не требуя при этом активного управления со стороны конечного пользователя. В основе представления знаний в сети Semantic Web лежит формат семантических данных RDF.

Среди препятствий на пути массового внедрения RDF в Web наиболее существенные — необходимость переноса уже накопленной информации на модель данных RDF и неудовлетворительная производительность существующих семантических систем при обработке больших объемов RDF-данных. Наиболее актуальным и перспективным подходом к решению обеих проблем считается интеграция RDF и реляционных баз данных [1, 2], 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.

Для решения вышеописанной задачи разработан описанный в данной статье алгоритм преобразования RDF-запросов в запросы к реляционной базе данных на языке SQL. В следующих разделах статьи описаны: метод адаптации реляционных данных, обеспечивающий приведение данных к виду, совместимому с описанным алгоритмом; алгоритмы частичного логического вывода, расширяющие выразительные возможности реализуемого данным алгоритмом языка RDF-запросов; алгоритм преобразования графового шаблона RDF-запроса, реализующий основную часть преобразования запроса.

Метод адаптации реляционных данных для отображения на модель RDF

Предложенный метод адаптации реляционных данных не накладывает дополнительных ограничений на используемую схему реляционной базы данных сверх ограничений стандарта SQL. Любая таблица в первой нормальной форме может быть отображена для доступа при помощи RDF-запросов на языке Squish [3]. Таким образом, любая существующая база данных может быть адаптирована для доступа через RDF, не теряя при этом обратной совместимости с существующими SQL-запросами.

Метод 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 терминов *rdfs:subClassOf, rdfs:subPropertyOf* и *owl:TransitiveProperty* языков описания метаданных RDFS [4] и OWL [5].

Следующие изменения схемы базы данных обязательны во всех случаях:

– создать таблицу ресурсов, отображенную на суперкласс *rdfs:Resource*, с автоматически генерируемым первичным ключом;

– заменить первичные ключи таблиц, отображенных на подклассы класса *rdfs:Resource*, на внешние ключи, ссылающиеся на таблицу ресурсов (может потребоваться также обновить существующие внешние ключи, чтобы отразить это изменение);

– зарегистрировать хранимые процедуры логического вывода на правилах для *rdfs:subClassOf* для обновления таблицы ресурсов и поддержки целостности внешних ключей при выполнении операций над таблицами подклассов.

Следующие изменения могут потребоваться для поддержки дополнительных возможностей алгоритма преобразования запросов:

– зарегистрировать хранимые процедуры для прочих случаев логического вывода на правилах для *rdfs:subClassOf*;

– создать таблицу триплетов (необходимо для представления RDF-данных, не отображенных на реляционную схему, и реификации утверждений RDF);

– добавить атрибуты различения подотношений, ссылающиеся на записи в таблице ресурсов, хранящие идентификаторы URIref соответствующих отношений, для каждого атрибута, отображенного на подотношение;

– создать таблицы транзитивных замыканий и зарегистрировать хранимые процедуры логического вывода на правилах для *owl:TransitiveProperty*.

Алгоритмы частичного логического вывода на уровне хранимых процедур реляционной СУБД

Разработанная система хранения RDF-данных реализует правила логического следования для следующих предикатов RDFS и классов OWL: *rdfs:subClassOf, rdfs:subPropertyOf, owl:TransitiveProperty*. Для минимизации влияния логического вывода на скорость обработки запросов используются хранимые процедуры.

Хранимые процедуры логического вывода на правилах для *rdfs:subClassOf* вызываются для каждой операции вставки в или удаления из таблицы подкласса. Когда вставляется кортеж без первичного ключа, шаблонный кортеж вставляется в таблицу суперкласса, а полученный при этом первичный ключ подставляется на место первичного ключа таблицы подкласса. Операция удаления распространяется каскадом на все связанные таблицы подклассов и суперклассов.

Логический вывод для *rdfs:subPropertyOf* выполняется во время преобразования запроса, и опирается на хранимую процедуру, возвращающую значение атрибута в случае, если атрибут различения подотношения установлен, и пустое значение NULL в противном случае. Логический вывод для *owl:TransitiveProperty* использует отдельную таблицу транзитивного замыкания для каждого атрибута, отображенного на транзитивное отношение. Таблицы транзитивных замыканий поддерживаются хранимыми процедурами, вызываемыми для каждой операции вставки, обновления и удаления, в которой вовлечены соответствующие атрибуты.

Входными данными для разработанного автором алгоритма обновления транзитивного замыкания являются:

– направленный помеченный граф $G = \langle N, A \rangle$, где N — множество вершин, представляющих RDF-ресурсы, A — множество дуг $a = \langle s, p, o \rangle$, представляющих триплеты RDF-утверждений;

– транзитивное отношение τ;

– подграф $G_\tau \subseteq G$, такой что $a_t = \langle s, p, o \rangle$ OG_t Ы a_t OG Щp = t ;

– граф G_τ^+, содержащий транзитивное замыкание G_τ ;

– операция обновления $\omega \in$ *insert*,*update*,*delete* и ее параметры $a_{old} = \langle s_\omega, \tau, o_{old} \rangle$, $a_{new} = \langle s_\omega, \tau, o_{new} \rangle$, такие что $G'_\tau = G_\tau \setminus a_{old} \cup a_{new}$.

Алгоритм преобразует G_τ^+ в транзитивное замыкание G'_τ. Предполагается, что граф G_τ не содержит и не должен содержать циклов. Более подробно алгоритм обновления транзитивного замыкания рассмотрен в [6].

Алгоритм преобразования графового шаблона RDF-запроса в запрос SQL

Приведенное в предыдущих разделах описание отображаемых реляционных данных и средств частичного логического вывода позволяет сформулировать 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, выразимых 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 $\mathrm{id}(\mathit{Resource})$ 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 $K_\Pi : \Pi_{nodes} \to \mathrm{K}$, $K_N : N_{nodes} \to \mathrm{K}$, $K_\Omega : \Omega_{nodes} \to \mathrm{K}$, $K(n) = K_\Pi(n) \cup K_N(n) \cup K_\Omega(n)$. Используется двухпроходный алгоритм разметки связных компонентов [7], 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 $\Psi^{n}_{nodes} = \langle s,p,o \rangle \big| n \in s,o$ и определить ее режим связывания $\beta_{node} : \Psi_{nodes} \to \Pi, N, \Omega$ такой что $\beta_{node}\ n = \max\ \beta_{arc}\ c\ \ \forall c \in \Psi^{n}_{nodes}$, где $\beta_{arc}\ c$ отражает, который из графовых шаблонов Π, N, Ω содержит дугу *c*, а оператор *max* использует порядок следования $\Pi > N > \Omega$.

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

```
1: for all n ∈ Ψ_nodes do
2:     for all c = ⟨s, p, o⟩ ∈ Ψ_arcs | s = n ∧ C_a(c) = ∅ do
3:         if ∃c' = ⟨s', p', o'⟩ | n ∈ {s', o'} ∧ C_a(c') ≠ ∅ ∧ M_rel(p') = M_rel(p)
   then
4:             C_a(c) ← C_a(c')
5:         else
6:             a = max(𝔸) + 1; 𝔸 ← 𝔸 ∪ {a}; C_a(c) ← a
7:             A_node(a) ← n, A_filter(a) ← ∅
8:             if M_trans(p) = ∅ then
9:                 A_rel(a) ← M_rel(p)
10:                A_β(a) ← β_node(n)
11:            else
12:                A_rel(a) ← M_trans(p)
13:                A_β(a) ← β_arc(c)
14:            end if
15:        end if
16:    end for
17: end for
18: for all c ∈ Ψ_arcs do
19:    A_filter(C_a(c)) ← A_filter(C_a(c)) ∪ F_c(c)
20: end for
```

Рис. 1. Алгоритм определения псевдонимов таблиц

4. Определить связки $B : \Psi_{nodes} \to \mathrm{B}$, где $\mathrm{B} = \{\{\langle a, f \rangle \mid a \in \mathrm{A}, f \in \Phi\}\}$, отображающие вершины графового шаблона на наборы пар псевдонимов таблиц и атрибутов, таких что

$\langle a, f \rangle \in B(n) \Leftrightarrow \exists c \in \Psi^{n}_{arcs} : C_a(c) = a, M^{pos}_{attr}(c, n) = f$.

Преобразовать графовый шаблон Ψ в граф реляционного запроса $Q = \langle \mathrm{A}, J \rangle$, где вершины 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 на рис. 2. Составить список связанных вершин $G \subseteq \Psi_{nodes}$, такой что

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

1: $\exists b = \langle a, f \rangle \in B(n)$
2: **if** n — внутренний ресурс и $\rho(n) = i$ **then**
3: $\quad A_{filter}(a) \leftarrow A_{filter}(a) \cup (b = i)$
4: **else if** n — параметр запроса или литерал **then**
5: $\quad A_{filter}(a) \leftarrow A_{filter}(a) \cup (b = n)$
6: **else if** n — идентификатор URIref **then**
7: $\quad \mathbb{A} \leftarrow \mathbb{A} \cup \{a_r\}$; $A_{node}(a_r) = n$; $A_{rel}(a_r) = Resource$; $A_\beta(a_r) = \beta_{node}(n)$
8: $\quad B(n) \leftarrow B(n) \cup \langle a_r, \mathrm{id}(Resource)\rangle$; $J \leftarrow J \cup \{\langle b, \langle a_r, \mathrm{id}(Resource)\rangle, n\rangle\}$
9: $\quad A_{filter}(a_r) = A_{filter}(a_r) \cup (\langle a_r, literal\rangle = f \wedge \langle a_r, uriref\rangle = t \wedge \langle a_r, label\rangle = n)$
10: **end if**

Рис. 2. Алгоритм связывания фиксированных вершин

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 $\beta_{node}\ n = \Pi$). 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 $\beta_{tree}\ P_i = N$ 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 $\beta_{node}\ n = N$. Добавить в W глобальное условие F_g.

Разработанный алгоритм был реализован в системе хранения RDF-данных Samizdat и показал высокую производительность обработки запросов. На рис. 3 приведен график производительности Samizdat, измеренной по методике BSBM [8], в сравнении с другими системами хранения RDF-данных.

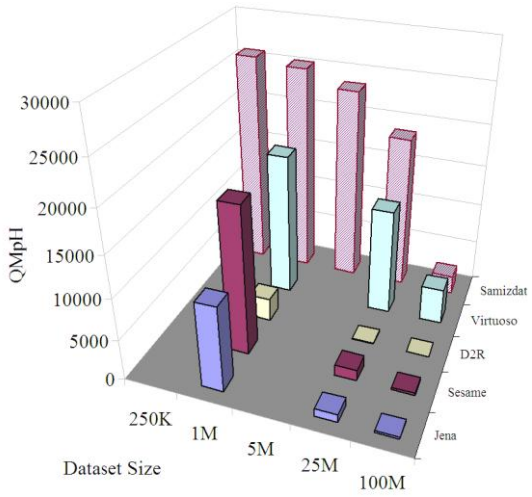

Рис. 3. Производительность обработки RDF-запросов

Заключение

Разработан набор алгоритмов для преобразования RDF-запросов, использующих графовые шаблоны, в запросы SQL, использующие операции реляционной алгебры. Разработаные алгоритмы позволяют производить поиск, извлечение и обновление данных по графу данных RDF, составленному из отображенных реляционных данных и произвольных реифицируемых утверждений RDF, на совместимом с моделью данных RDF языке запросов. В отличие от существующих аналогов, разработанное решение поддерживает необязательные и отрицательные графовые шаблоны, автоматическое распознавание рекурсивных подшаблонов, а также логический вывод на правилах для подмножества словарей RDFS и OWL.

MAPPING RELATIONAL DATA TO THE RDF SEMANTIC DATA MODEL USING ON-DEMAND QUERY TRANSLATION

D.S. BORODAENKO

Abstract

The algorithm for on-demand translation of RDF queries that allows to map any relational data structures to RDF data model and to perform queries over a combination of mapped relational data and arbitrary RDF triples, enabling performance on par with relational databases is presented. The algorithm supports a wide range of query capabilities including optional and negative graph patterns, nested sub-patterns, and inference over a subset of RDFS and OWL metadata vocabularies.

Литература

1. *Malhotra, Ashok* // W3C RDB2RDF Incubator Group Report. W3C Incubator Group Report. 2009.
2. *Auer S., Dietzold S., Lehman J. et al.* // WWW 2009, Madrid, Spain, 2009.
3. *Miller L., Seaborne A., Reggiori A.* // In: Horrocks I., Hendler J. (Eds) ISWC 2002. Springer, Heidelberg, 2002. LNCS Vol. 2342. P. 423–435.
4. *Brickley D., Guha R.V.* // RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. 2003.
5. *Patel-Schneider P.F., Hayes P., Horrocks I.* // OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation. 2004.
6. *Borodaenko D.* // ICIS 2009 Proceedings. Shanghai, 2009. (In print.)
7. *Shapiro L., Stockman G.* // Computer Vision. Prentice-Hall, 2002. P. 69–73.
8. *Bizer C., Schultz A.* // International Journal on Semantic Web and Information Systems (IJSWIS). 2009.Vol. 5, Issue 2.