Интеграция баз данных на основе онтологий с использованием технологий Semantic Web

А. Ф. Тузовский[1]

[1]Институт кибернетики Томского политехнического университета
Отдел проблем информатики ТНЦ СО РАН
Россия, г. Томск, 634034 , ул. Советская, 84/3
tuzovskyaf@tpu.ru

Аннотация. Описан подход к разработке систем виртуальной интеграции на основе использования технологий Semantic Web. Рассмотрены используемые языки RDF и SPARQL, а также существующие подходы федерирования запросов к источникам RDF данных. На примере системы SemWIQ показана архитектура построения системы интеграции и решения основных проблем ее функционирования.

Ключевые слова: интеграция, посредники, адаптеры, онтология, RDF, SPARQL.

1 Введение

Цель систем интеграции данных состоит в том, чтобы освободить пользователей от необходимости искать источники данных релевантные запросу, взаимодействовать с каждым из источников в отдельности и вручную объединять данные из разных источников. Каждый источник содержит некоторые структурированные (или полу-структурированные) данные и имеет свое описание модели хранимых данных (схему данных). Например, реляционную модель (таблицы/колонки), XML модель или онтологическую модель. В общем случае, источники данных являются распределенными, т.е. расположенными на разных компьютерах, к которым есть доступ по компьютерной сети (по интранет или Интернет). Разные источники данных поддерживаются разными группами пользователей (т.е. являются автономными).

Имеются следующие два базовых подхода к решение задачи интеграции данных [1]:

1. **Консолидация данных** (материализованная интеграция). В соответствии с этим подходом создается единое хранилище данных,

которое периодически пополняется из интегрируемых источников данных с помощью специальных модулей, выполняющих извлечение, преобразование и загрузку данных (архитектура «хранилище – ETL модули»).

2. **Федерирование данных** (виртуальная интеграция). В соответствии с этим подходом центральный элемент системы интеграции (посредник[1]) выполняет преобразование поступающих запросов в подзапросы к источникам данных, которые поддерживаются специальными модулями «адаптерами» (wrappers), облегчающими взаимодействие посредника и источника данных (архитектура «посредник-адаптеры»).

Несмотря на различие данных подходов многие решаемые в них проблемы совпадают, и в последнее время отмечается все большее их сближение. Основным отличием федерирования данных от консолидации является обработка поступающих запросов, т.е. их преобразование в подзапросы к источникам данных и обработка полученных результатов.

Интеграция данных на основе федерирования данных по многим причинам [1] (как например, возможность оперативно учитывать изменения в источниках данных, техническая трудность копирования полных объемов источников данных, ограничения прав доступа к данным) считается наиболее перспективной. Далее будет рассматриваться только виртуальная интеграция данных.

2 Системы виртуальной интеграция данных

Стандартная архитектура системы виртуальной интеграции данных основана на использовании посредников и адаптеров. Посредник это центральная компонента системы интеграции, а адаптеры – компоненты, обеспечивающие единообразное взаимодействие посредника с источниками данных, как показано на рис. 1.

Основными компонентами посредника являются:

- центральный интерфейс доступа, используемый приложениями системы интеграции, либо с помощью промежуточного программного обеспечения (такого, как например, ODBC, Web Service, SPARQL и т.п.) или с помощью прикладного интерфейса API;
- центральный каталог метаданных (или база знаний[2]в случае систем, основанных на онтологиях), хранящий глобальную модель данных, которая может описываться и поддерживаться явно или неявно, как объединение всех локальных моделей данных;

- процессор обработки запросов, отвечающий за разделение (федерирование) глобальных запросов поступающих в систему на подзапросы к источникам данных, их оптимизацию и выполнение;
- журнал регистрации, используемый для регистрации и де-регистрации источников данных (обычно регистрация требует предоставления метаданных об источнике и отображения его модели).

Рис. 1 Обобщенная архитектура «посредника-адаптеров» систем интеграции данных

Адаптеры (wrappers) систем интеграции обычно прикрепляются к источникам данных для решения проблем технической разнородности и разнородностей метамоделей. В зависимости от используемого подхода к описанию отображения, адаптеры могут дополнительно решать проблемы других уровней разнородности.

Системы виртуальной интеграции могут дополнительно включать также средства формирования отображения моделей, которые позволяют решать задачу отображения моделей с помощью графического интерфейса пользователей GUI и такие дополнительные возможности, как сопоставление моделей, проверка и чистка данных. Некоторые системы предоставляют также графический анализатор данных, браузер результатов запросов и другие виды средств визуализации.

Кроме этих компонентов, системы виртуальной интеграции включают: используемую глобальную метамодель; глобальный язык запросов применяемый для описания глобальных запросов; интерфейс и протокол взаимодействия между адаптерами и посредниками.

Исследования по разработке методов и архитектур построения систем виртуальной интеграции данных ведутся с начала 90-х годов. Развитие

таких систем ведется путем перехода к использованию более выразительных мета-моделей для описания схем данных и к использованию технологий сетей Web и Semantic Web.

Можно выделить следующие этапы развития федеративного подхода[1]:

1. системы, основанные на объектных моделях (в соответствии со стандартом ODMG-93), использующие язык запросов OQL (либо его версии, как например GQL));
2. системы, основанные на реляционных моделях (например, E-R моделях), использующие язык запросов SQL;
3. системы, основанные на концептуальных моделях (онтологиях, описанных на языке OWL и RDF), использующие язык запросов SPARQL.

Исследования в области использования онтологий ведутся в двух основных (пока достаточно независимых) направлениях:

- Исследование проблем преобразования запросов к глобальной онтологии в запросы к источникам данных с помощью средств логического вывода на онтологической модели (интенсиональной компоненты TBox). Далее полученные подзапросы выполняются относительно экстенсиональной компоненты онтологии ABox. Если ABox поддерживается реляционной СУБД, то обработка созданных конъюнктивных запросов выполняется самой СУБД (вернее ее подсистемой обработки SQL запросов) с использованием уже хорошо разработанных стратегий оптимизации запросов [2, 3].
- Исследование архитектур построения систем интеграции с использованием базовых технологий Semantic Web, таких, как языки RDF и SPARQL.

Так как первый подход детально рассмотрен в статье из данного сборника [4], то далее будет рассматриваться поход к виртуальной интеграции данных на основе указанных технологий Semantic Web.

3 Основные технологии Semantic Web

3.1 Стандарты языков RDF и SPARQL

Двумя основными стандартами Semantic Web (SW) являются язык RDF (для описания онтологий и метаданных) [5] и язык SPARQL (для описания запросов к RDF данным) [6].

Базовой моделью данных для всех приложений Semantic Web является язык RDF (Resource Description Framework). Любая информация более высокого уровня, например, описания онтологий с помощью таких языков, как RDF Schema и OWL, представима в языке RDF. Описание ресурсов на языке RDF выполняется в виде триплетов (s, p, o), где s называется субъектом (subject), p – предикатом (predicate), а o – объектом (object). Такие триплеты также называются RDF утверждениями. Набор триплетов обычно рассматривается, как RDF граф – направленный, размеченный граф, сформированный из неупорядоченного набора триплетов.

В виде RDF утверждений могут быть описаны данные, имеющие очень разные форматы. Наиболее широко используемой программной средой для работы с RDF данными является Jena Semantic Web Framework для языка Java [7].

Для работы с данными, представленными на языке RDF, разработан стандартный язык запросов SPARQL. Он может использоваться для поиска информации, содержащейся в RDF графах, структурным способом. SPARQL соответствует выражению SPARQL Protocol and RDF Query Language, т.е. данное название не только определяет декларативный язык запросов, подобно тому, что SQL делает для реляционной модели, но кроме этого определяет RESTful протокол, который используется для отправки запросов и получения результатов на основе протокола Hypertext Transfer Protocol (HTTP).

Для описания SPARQL запросов используются шаблоны триплетов, имеющие такой же формат, что и триплеты, но в которых вместо конкретных значений могут быть записаны свободные переменные. В языке SPARQL перед переменными шаблона стоит префикс в виде символа вопроса '?'. Например: (<http://scott.com/foaf.rdf#me> foaf:knows ?person). Такие шаблоны триплетов ограничивают рассматриваемый граф подмножеством соответствующих им триплетов. При сопоставлении шаблона триплетов с RDF графом формируется набор ***решающих соответствий*** переменных запроса.

Основной формой SPARQL запросов являются оператор SELECT (Рис. 2). Кроме этой формы также имеются три другие формы запроса: DESCRIBE, CONSTRUCT и ASK. Все формы SPARQL запросов основываются на сопоставлении графовых шаблонов, однако каждая форма определяет, как обрабатываются решающие соответствия после того, как они будут получены, и как они окончательно представляются в качестве результата.

Все запросы SPARQL включают базовые графовые шаблоны (БГШ, basic graph pattern (BGP), другой перевод «простой шаблон подграфа»). Под базовым графовым шаблоном `bgp` понимается набор шаблонов триплетов `tp`, т.е. $bgp = (tp_1, tp_2, \ldots, tp_n)$. Базовый графовый шаблон, включает набор шаблонов триплетов, которые могут содержать совпадающие свободные переменные.

БГШ может рассматриваться как коньюнктивный запрос, в котором все шаблоны триплетов должны быть сопоставлены активному RDF графу. При сопоставлении, каждый шаблон триплетов БГШ сопоставляется с RDF графом, что эквивалентно внутреннему связыванию в алгебре SQL. Формально, решающие соответствия одного шаблона триплетов соединяются (сливаются) с решающими соответствиями другого шаблона триплетов, если они содержат общие переменные. Слияние означает, что два совместимых решающих соответствия объединяются в одно решающее соответствие, если они совместимы, т.е. все их общие переменные имеют одинаковые значения. Если решающие соответствия являются не совместимыми, то они отбрасываются (семантика связывания), а если совсем нет общих переменных, то результатом такого связывания будет перекрестное связывание (cross join) обоих последовательностей решений.

Несколько БГШ могут быть объединены и при этом могут использоваться три разных семантики: семантика связывания (join semantics), если не задана операция между БГШ; семантика левого связывания (left join semantics), если задан оператор OPTIONAL; семантика объединения (union semantics), если задан оператор UNION.

Кроме базовых графовых шаблонов, к алгебраическим операторам более высокого уровня относятся: `Filter` (фильтрация), `Join` (соединение), `Optional` (необязательное, левое связывание), `Union` (объединение) и `Graph` (построение графа). Кроме этого к алгебраическим операторам относятся модификаторы решений, такие, как: `Order-by` (упорядочить), `Project` (проецировать), `Distinct` (выделить различные), `Reduced` (сократить) и `Slice` (разделение на части).

Для SPARQL запроса может быть сформирован план запроса (алгебраический план), описывающий порядок его выполнения. Пример алгебраического плана для примера SPARQL запроса, показанного на рис. 2.

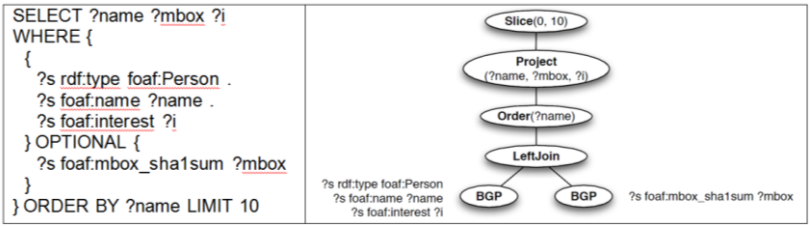

Рис. 2 Запрос SPARQL и соответствующий ему алгебраический план

Планы запросов часто изображаются в виде графов, в которых операторы рисуются в виде вершин, соединенных дугами (Рис. 2). Однако, описание планов запросов может выполняться и с помощью композиционной записи. Например, план запроса, показанный на рис. 2 может быть записан следующим образом:

Slice((0; 10),

Project((?name ?mbox ?i),

Order((?name),

LeftJoin(

BGP((?s rdf:type foaf:Person),(?s foaf:name ?name), (?s foaf:interest ?i)),

BGP((?s foaf:mboxsha1_sum ?mbox))

))))

Оператор может задавать список аргументов, за которым следует список под-операторов.

Одной из первых программных реализаций полностью согласованной со стандартом SPARQL была система ARQ[1], которая является частью среды Jena Semantic Web Framework для языка Java [7]. ARQ это система обработки запросов для среды Jena, которая поддерживает язык SPARQL. ARQ предоставляет такие возможности, как: обработка запросов на стандартном языке SPARQL; полнотекстовый; обработка запросов на языке SPARQL/Update ; доступ к SPARQL алгебре и ее расширение; поддержка собственных функций фильтрации; поддержка функций

[1] http://jena.sourceforge.net/downloads.html

свойств для специальной обработки семантических взаимосвязей; агрегирование GROUP BY и задание значений в виде SPARQL расширений; поддержка федеративных запросов; поддержка расширений для использования других систем хранения; поддержка клиентов для удаленного доступа к любой конечной точки SPARQL.

3.2 Дополнения стандарта SPARQL

При разработки систем интеграции с использованием языка SPARQL потребовались различные расширения, позволяющие разрабатывать на его основе эффективно работающие системы. К таким расширениям относятся добавление форм запросов и используемых операторов:

- Формы запросов DESCRIBE SERVICE и DESCRIBE DATASET для получение описаний сервисов и исходных наборов данных. Запрос DESCRIBE SERVICE возвращает описание конечной SPARQL точки сервиса и его возможности, а запрос DESCRIBE DATASET возвращает метаданные и статистические параметры RDF данных для обслуживаемого источника.

- Форма запроса EXPLAIN для получения подробностей выполнения плана запроса, например, EXPLAIN SELECT * WHERE { ... }.

- Оператор SERVICE, используемый для передачи локального под-плана адаптеру источника данных.

- Оператор BINDINGS для задания начальных связываний для эффективной обработки распределенных связываний (т.е. связей между данными из разных источников).

3.3 Преимущества использования RDF в качестве глобальной метамодели систем интеграции данных

Язык RDF имеет много особенностей, которые делают его очень удобным для использования в качестве глобальной метамодели в системах интеграции данных, которые имеют дело с разнородностью метамоделей:

- RDF по определению ориентирован на работу в Web и является хорошо масштабируемым, так как взаимосвязанные RDF онтологии могут быть распределены по всей Web-сети.

- RDF онтологии могут быть опубликованы любым пользователем в Web сети для того, чтобы расширить существующие понятия (концепты) взаимосвязями с новыми понятиями, если это потребуется, например, при добавлении нового источника данных к системе интеграции, информация которого не описывается текущей глобальной моделью, по существу являющейся объединением всех опубликованных онтологий, использованных для описания источников данных.

- В качестве идентификаторов для всех понятий используются URI идентификаторы, что облегчает управление глобальным пространством имен URI, путем использования Системы Доменных Имен (Domain Name System, DNS). Каждый владелец домена, является владельцем пространства определения, т.е. он может создавать URI идентификаторы) на основе своего собственного имени домена. Это не означает введения предположения об уникальности имен (Unique Name Assumption, UNA), которое способствует фрагментации, основанной на понятиях, т.е. любой может добавить утверждения о конкретном ресурсе и частичные RDF графы могут объединяться из разных мест.

- Языки RDF, RDF-Schema, OWL и другие стандарты W3C разрабатываемые на их основе являются стандартизированными и широко используемыми языками представления знаний.

- RDF граф может быть описан простым способом в виде набора триплетов, что облегчают слияние не полных, фрагментированных графов из распределенных источников.

- RDF-Schema, OWL и другие словари построенные над RDF Core предоставляют все типичные средства описания понятий, известные из других сред (например, E-R Model и UML), которые требуются для моделирующих онтологий.

- RDF-Schema и OWL поддерживают терминологические утверждения (TBox) и утверждения о фактах (ABox), а также многие дескриптивные логики, которые могут использоваться

для выполнения логического вывода над данными и наложения ограничений.

- Хотя RDF метамодель (RDF Core) остается настолько простой, насколько это возможно, все средства RDF Schema и версии языка OWL Full сами могут быть описаны на основе RDF метамодели, которая позволяет выполнять мета-моделирование и нарушает дихотомию «экземпляр-тип» .

3.3 Web-сеть данных или Связанные Данные

Подход к разработке систем интеграции на основе технологий SW близок к подходу создания Web сети Данных (Web of Data или Linked Data). В то время, как сеть WWW состоит из HTML документов и мультимедиа ресурсов, связанных гиперссылками, сеть Semantic Web состоит из ресурсов любого вида, связанных свойствами (или семантическими связями). Чтобы сделать RDF графы интуитивно доступными и просматриваемыми было предложены 4 правила проектирования [8]:

1. Для идентификации объектов должны использоваться URI идентификаторы.

2. Точнее, HTTP URI идентификаторы должны использоваться таким образом, чтобы пользователи могли выполнять поиск объектов.

3. Если ищется некоторый URI идентификатор, то с помощью стандартов RDF и SPARQL должна предоставляться описывающая его информация.

4. Другие семантические связи должны указывать на другие URI, чтобы пользователи могли находить все большее количество объектов.

В 2007 г. было организовано несколько проектов для поддержки этой идеи и способствования широкому принятию парадигмы Связанных Данных (Linked Data) [9]. С тех пор было опубликовано множество открыто доступных, сгенерированных пользователями и открытых государственных наборов данных.

Такая идея построения глобально-масштабируемой открытой Web-сети Данных в дополнении к традиционной сети World Wide Web и принципы ее построения полностью подходят для систем интеграции информации. На рис. 3 приведена современная инфраструктура Web сети

Данных (Web of Data). В центре рисунка показано клиентское приложение (обычно это Web-браузер или специализированный для Linked Data браузер, такой, как Tabulator [8]). По краям рисунка показано: несколько простых RDF ресурсов, распределенных по Web, некоторые множества RDF данных с конечными SPARQL точками и возможно интерфейсом Linked Data Interface, система поиска для Semantic Web (например, Swoogle [10] или SWSE [11]), сервис поиска путем просмотра (например, Sindice [12]), посредник (например, SemWIQ – система расмотренная в данной статье).

Краткое описание всех этих компонент приведено ниже.

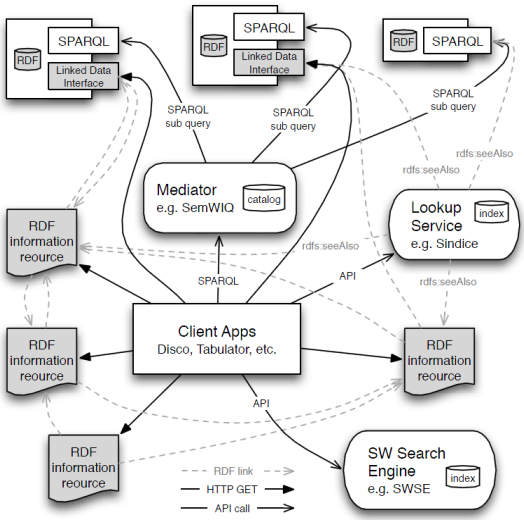

Рис. 3 Текущая инфраструктура Связанных Данных (Linked Data)

- ***Клиентские приложения*** – в настоящее время наиболее популярными клиентами являются браузеры связанных данных. Они выполняют разбор простых RDF данных и предоставляют хорошо организованное, структурированное представление данной информации, обычно в виде групп триплетов.

- ***Ресурсы RDF данных*** – это Web документы, содержащие RDF данные. Каждый RDF ресурс идентифицируется абсолютным

HTTP идентификатором URI. Он также может автоматически генерироваться из базы данных, но внешне выглядеть, как отдельный Web документ. Он содержит RDF утверждения, описывающие один или несколько информационных или не информационных ресурсов, возможно с внешними RDF связями .

- ***Конечные SPARQL точки*** (точки доступа, endpoints) – предоставляют доступ к наборам RDF данных с помощью языка и протокола SPARQL на основе HTTP. Точки доступа являются сервисами (хотя и не обязательно Web-сервисами), которые принимают и обрабатывают SPARQL запросы и возвращают результаты в разных форматах, в зависимости от формы запроса. Точки доступа, которые доступны через HTTP должны соответствовать протоколу SPARQL (http:/www.w3.org/TR/rdf-sparql-protocol).

- ***Интерфейс связанных данных*** (Linked Data Interface) – так как конечные SPARQL точки предоставляют только декларативный доступ, то невозможно выполнять просмотр ресурсов, которые описаны вне пределов данной конечной точки без обертывания данной информации, например, с помощью DESCRIBE запросов. Интерфейс Linked Data Interface в действительности является адаптером RDF-to-HTML, который прикреплен к конечной точке (например, Pubby [13]). Он в реальном времени находит URI идентификаторы ресурсов, выполняет соответствующий запрос и возвращает HTTP ответ (возможен результат с кодом 303-redirect).

- ***Поисковые системы Semantic Web*** – Такие поисковые системы используют специализированную программу систематического просмотра web-сети, которая ищет RDF данные в Web сети, обычно идентифицируемых по заголовку Content-type в HTTP ответе. Она также предоставляет Web интерфейс и/или API с несколькими возможносями поиска. Например, система SWSE предоставляет поиск по ключевым словам, который возвращает список RDF ресурсов (как информационных, так и не информационных). На втором шаге пользователь может ограничить показ только тех RDF ресурсов, тип которых известен.

- ***Сервис поиска путем просмотра*** (look-up service, например Sindice [12]). В отличие от поисковых систем, которые обычно

хранят в кэше все найденые во время автоматического просмотра сайтов данные (в основном RDF и OWL файлы), система Sindice хранит только специальные индексы. Например, индексы для поиска URI идентификаторов ресурсов, инверсные функциональные свойства и их значения, ключевые слова.

- ***Посредники*** – системы, как например, SemWIQ, которые могут использоваться для выполнения запросов к распределенным наборам RDF данных из одной точки доступа. Они могут рассматриваться в качестве еще одного сервиса инфраструктуры Web of Data, которая может использоваться другими клиентами приложениями. Сервисы подобные посредникам, как например Semantic Web Client Library [14] или SQUIN [15], которые позволяют выполнять перемещение по распределенному RDF графу, также могут быть отнесены к данному типу сервисов.

Такая инфраструктура является хорошей основой для появляющейся Web сети Данных. Однако, количество доступных наборов данных и открытых конечных SPARQL точек все еще является очень небольшой в сравнении с традиционной Web сетью.

4 Системы федерирования RDF данных

В настоящее время известно уже достаточно много систем виртуальной интеграции на основе семантических технологий, например, такие, как например: DARQ [16], SemWIQ [17, 18], ADERIS [19], FeDeRate [20]. Все они основаны на выполнении разделения (федерирования) SPARQL запросов к глобальной онтологической модели на подзапросы к адаптерам RDF данных.

В качестве примера рассмотрим две системы интеграции: DARQ и SemWIQ.

4.1 Система федерирования SPARQL запросов DARQ

Система DARQ (Distributed ARQ) [16], предназначена для выполнения федерированных SPARQL запросов к RDF источникам данных. Она предоставляет прозрачное выполнение запросов к множеству распределенных конечных SPARQL точек, как если бы запрос выполнялся к одному RDF графу (Рис. 4).

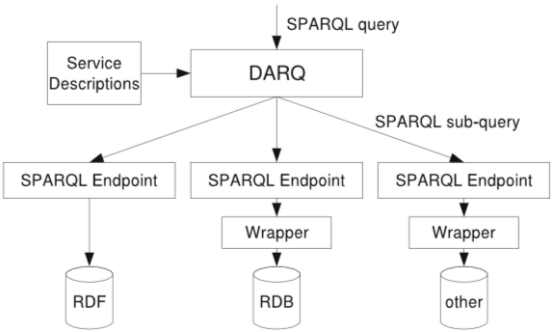

Рис. 4 Архитектура системы интеграции DARQ

Система DARQ позволяет приложениям (или пользователям) видеть единый интерфейс запроса, оставляя подробности разделения запроса на части подсистеме обработки запросов. Она расширяет систему ARQ (процессор обработки SPARQL запросов среды Jena) путем добавления нового алгоритма планирования запросов и модифицированной подсистемы выполнения запросов. Работа в DARQ включает язык описания сервисов и базовый алгоритм оптимизации запросов.

Федерирование запросов DARQ основывается только на свойствах шаблонов триплетов, что приводит к созданию очень больших планов запросов. DARQ выполняет запросы к подходящим источникам данных на основе отдельных шаблонов триплетов с учетом возможностей, которые описаны в файле описания сервисов (метаданные системы, конфигурационный статический файл). Хотя шаблоны триплетов при возможности могут объединяться в под-планы БГШ, в реальных условиях это делается редко. Несколько шаблонов триплетов может объединяться в один локальный БГШ запрос, только в том случае, если для всех шаблонов триплетов существует только одна подходящая SPARQL конечная точка.

Более того, система DARQ может выполнять только запросы со связанными предикатами. Например, невозможно выполнить такие запросы к концептуальному уровню, как: `{ :s ?p [ ] }`, который должен вернуть все свойства используемые для описания заданного ресурса. Это связано с тем, что выбор источников данных DARQ основывается на сопоставлении предикатов шаблона запроса с предикатами в шаблонах возможностей.

Обработка запросов передаваемых федератором запросов DARQ конечным SPARQL точкам основывается просто на сопоставлении

шаблонов триплетов. Для каждого шаблона триплетов из БГШ, федератор выбирает подходящие конечные SPARQL точки и выполняет к ним запрос в виде шаблона триплетов, даже если полный БГШ может быть выполнен удаленно. Выбор подходящих SPARQL конечных точек основывается на описаниях сервисов, которые составляются на языке RDF и включают в виде триплетов информацию о мощностях, селективностях и ограничениях на доступ.

Все описания сервисов хранятся в статическом конфигурационном файле, который загружается системой DARQ при ее запуске. Во время выполнения системы нет возможности обновить эту информацию. Кроме этого, статистическая информация должна вводиться вручную, так как генератор такой статистики отсутствует (т.е. нет программных средств автоматического сбор нужных данных и их записи в статические конфигурационные файлы).

Планирование запросов в DARQ выполняется отдельно для каждого БГШ из глобального запроса. Алгоритм, описанный в [16] пытается объединить шаблоны триплетов в отдельные БГШ под-запросы, если это возможно. Однако, такая возможность появляется только в том случпае, если для набора шаблонов триплетов подходит только одна конечная точка, на основе анализа возможностей описанных ранее. В реальных условиях такое условие выполняется очень редко.

Связывание распределенных шаблонов триплетов является основным слабым местом подхода DARQ. Для оптимизации таких запросов шаблоны триплетов переупорядочиваются на основе их ожидаемой мощности (количестве элементов). Количество элементов шаблонов триплетов SPARQL запроса может оцениваться с помощью оценок базовой мощности и избирательностей (selectivities) для связанных субъектов и объектов.

В некоторых ситуациях, соединяющее связывание (bind join) является неприменимым или более дорогим, чем связывания с помощью вложенных циклов (nested loop join, NLJ). В связи с этим [16] определили простую стоимостную модель, как для оценки соединяющего связывание (bind join), так и для оценки связывания с помощью вложенных циклов (NLJ). Для вычисления стоимостей планов БГШ (если комбинация шаблонов триплетов является возможной) авторы также определили функцию оценки мощности для базовых графовых шаблонов. Лучший вариант выполнения связывания выбирается с помощью алгоритма динамического программирования.

Система DARQ является интересным подходом, выполняющим интеграцию данных с использованием SPARQL на основе свойств. Однако, остается еще много задач, связанных с процессором обработки запросов, которые в ней не решены.

4.2 Система интеграции данных SemWIQ

Другим примером системы интеграции RDF данных является система SemWIQ [17, 18]. Данная система разработана на основе компонента Jena ARQ [7], которая была расширена федератором и статистическим оптимизатором запросов на основе статистических данных.

В отличие от других систем, которые используют RDF только в качестве уровня метаданных надстроенного над существующей глобальной схемой, систему SemWIQ позволяет на основе использования RDF и других возможностей языка OWL реализовать мощную, распределенную, уровня Web сети, систему интеграции информации, которая может выполнять объединение информации из большого набора различных информационных систем, на основе федерированных SPARQL запросов.

В системе SemWIQ разнородные источники данных виртуально интегрируются посредством адаптеров и центрального процессора обработки федеративных запросов, который является ответственным за формирование ответов на запросы, путем делегирования под-запросов адаптерам. Процесс обработки запросов основывается на использовании конвейеров (pipelining), что позволяет уменьшить время ответа и гарантирует масштабируемость системы путем потоковой передачи результатов через конвейер от исходных информационных систем до отправившего запрос клиента.

Ключевой особенностью системы SemWIQ является федерирование глобальных запросов на основе статистических данных, что делает данную систему хорошо масштабируемой. В традиционных системах, основанных на посредниках-адаптерах, выбор подходящих (релевантных) источников данных основывается на описании глобальной схемы, а не на фактическом наборе данных. Для заданного запроса, к каждому источнику данных, являющемуся релевантным с точки зрения экспортируемой им схемы, явно выполняется запрос, чтобы определить, не имеет ли он каких-либо данных, которые могут быть добавлены в результаты. Это требует выполнении, по крайней мере, одного сетевого соединения, чтобы вызвать на выполнение стоимостные функции источника данных. В системе SemWIQ возможно интегрировать большое

количество источников данных, так как релевантные источники данных могут быть выявлены автономно, без необходимости выполнять взаимодействие с соответствующей конечной SPARQL точкой.

Целью федерирования глобальных запросов является включение в план глобального запроса только тех источников данных, которые потенциально могут добавить какие-либо результаты в его обработку. В отличие от традиционных систем интеграции, система SemWIQ не требует явного описания и поддержки глобальной схемы. В связи с чем, в системе SemWIQ нет недостатка, который имеется в обычных систем интеграции, использующих подход Global-as-View, а именно, необходимости обновлять глобальную схему при каждом добавлении нового источника данных. Вместо поддержки глобальной схемы, которая описывает все интегрируемые локальные схемы, система SemWIQ основывается на сводках статистических данных.

Основными архитектурными принципами данной системы являются следующие:

- Система использует виртуальную глобальную онтологию (виртуальный граф G_G), которая формируется динамически на основе онтологий, используемых при описании источников данных $G_G = G_{S1} \cup G_{S2} \cup \ldots \cup G_{Sn}$. Она также может быть представлена в виде большого набора триплетов, которые были виртуально соединены из всех графов источников данных:

$$G_G = (t_{s1,1}, t_{s1,2}, \ldots, t_{s1,k1}, t_{s2,1}, t_{s2,2}, \ldots, t_{s2,k2}, \ldots t_{sn,1}, t_{sn,2}, \ldots, t_{sn,km}).$$

- Каждый источник данных S_i добавляет некоторый подграф G_{Si} в виртуальный глобальный RDF граф G_G. В том случае, если используется не RDF источник данных, то под-граф G_{Si} виртуально определяется исходным набором данных D_{Si} источника данных S_i и описанием отображения M_i.

- Информация о всех зарегистрированных источниках данных хранится в каталоге метаданных, являющимся также RDF хранилищем, содержащим описания источников с помощью словаря voiD , их текущее состояние возможности их использования и статистические данные RDFStats [21]. Такие статистические данные используются для интеграции и оптимизации глобальных SPARQL запросов. Данный каталог метаданных может быть сохранен в любом, основанном на Jena,

RDF хранилище, но лучшая производительность может быть достигнута с помощью Jena TDB [7]. Система SemWIQ включает много-поточный монитор источников данных, который наблюдает в фоновом режиме за зарегистрированными источниками данных, основываясь на конфигурируемых профайлах и обновляет текущее состояние их доступности, а также статистические данные RDFStats.

- Процессор обработки запросов системы SemWIQ получает глобальный SPARQL запрос и формирует разделенный на части план запроса (подзапросы) на основе состояния каталога метаданных. Процессор разработан на основе компонента Jena ARQ.

- Новые источники данных могут быть зарегистрированы и удалены с помощью компонента журнала регистрации, используя прикладной интерфейс журнала или с помощью Web-сервис. Источник данных регистрируется в виде соответствующего URI идентификатора конечной точкой SPARQL предоставляемого соответствующим адаптером.

В ходе регистрации монитор источников данных будет пытаться найти метаданные в формате voiD [22] в Web, которые могут включать описание системы, которая поддерживает набор данных, лицензионную информацию, и предметные термины, связанные с этим набором данных. Кроме этого данный монитор собирает статистические данные (RDFStats), если она предоставляются системой, а в противном случае он будет использовать встроенный RDFStats компонент и генерировать статистические данные удаленно.

На рис. 5 показана общая архитектура системы SemWIQ. Для каждого источника данных и соответствующей модели исходных данных используется специфический адаптер для выполнения отображения и преобразования данных в RDF формат за один шаг.

Основными элементами данной архитектуры являются клиенты, показанные над посредником, который находится в центре, и несколько источников данных, показанные в нижней части рисунка. Пользователи системы SemWIQ выполняют запросы к посреднику с помощью клиента (самостоятельная программа или Web-приложение, основанное на Web-сервисе) передают на обработку глобальные SPARQL запросы (1).

Глобальный SPARQL запрос может содержать не только любые терминологические понятия, такие, как классы и свойства, но также и

экземпляры понятий, которые существуют в глобальном виртуальном графе. Грамматический разборщик запросов (2) формирует канонический план запроса, который преобразуется и оптимизируется федератором запросов (3) на основе информации, которая содержится в каталоге метаданных (4). Каждый источник данных описывается в каталоге метаданных системы SemWIQ в виде экземпляра OWL класса sdv:Datasource с помощью словаря источников данных системы (SDV), который в свою очередь основан на основан на словаре voiD. Данный словарь может использоваться для предоставления дополнительных метаданных, включая создателя, издателя и т.п. [22].

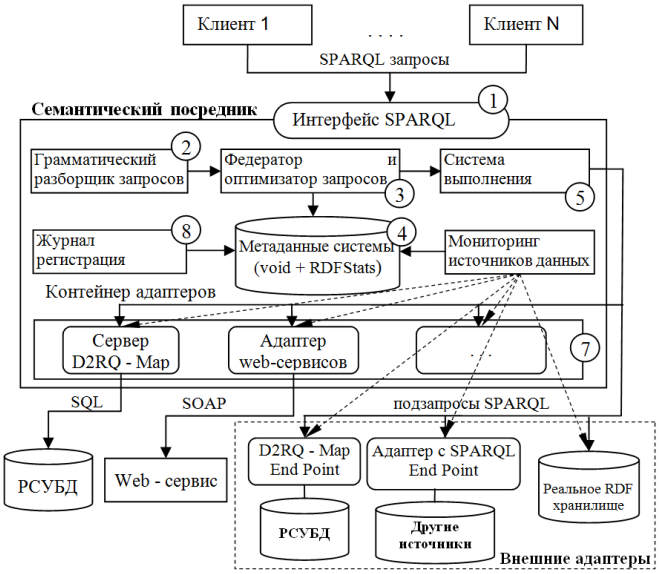

Рис. 5 Архитектура посредников-адаптеров в SemWIQ

Работа с каталогом метаданных, который сам является RDF графом, выполняется с помощью компонента Jena TDB [7], позволяющего выполнять работу с базой данных, содержащей такие данные, как void:Dataset [22] и статистические данные RDFStats [21] для каждого зарегистрированного источника данных.

Разделенный на части и оптимизированный план запроса окончательно обрабатывается подсистемой выполнения запросов (5). Глобальный план запроса состоит из нескольких локальных под-планов, которые далее объединяются высокоуровневыми алгебраическими операциями, которые выполняются в посреднике. Любая операция, которая может быть

непосредственно выполнена адаптером удаленно (там, где физически находится источник данных) включается в план локального запроса.

Удаленные адаптеры размещаются непосредственно над удаленными источниками данных (6). Для источников данных, которые являются полностью автономными, соответствующие адаптеры могут помещаться в специальный контейнер, который находится в посреднике (7).

Журнал регистрации источников данных (8) за регистрацию и де-регистрацию источников данных. Источник данных может быть зарегистрирован и де-регистрирован путем отправки HTTP POST запроса с соответствующим URI кончной точки SPARQL. Монитор источников данных (9) периодически выполняет проверку доступности зарегистрированных источников данных собирает метаданные в формате voiD и текущие статистические данные RDFStats.

Система SemWIQ позволяет решать все уровни разнородности между схемами источников данных.

4.3 Посредники системы SemWIQ

Посредник системы SemWIQ является центральным компонентом рассматриваемой системы интеграции (Рис. 6).

Рис. 6 Структура посредника системы интеграции SemWIQ

Журнал источника данных и монитор управления метаданным создают статистические данные RDFStats, которые используются федератором и оптимизатором запросов для формирования планов запросов к источникам данных.

Задачей федератора посредника является такое преобразование плана запроса, чтобы подсистема обработки запросов могла создать правильные результаты, имея такой виртуальный граф.

Система RDF интеграции основываются на сопоставлении распределенных графовых шаблонов SPARQL. В ходе обработки глобальных (федерирования), они декомпозируется на несколько локальных под-запросов $q_{S_{i,j}}$ и оставшейся глобальной части , которая объединяет локальные подпланы. Локальный подплан $q_{S_{i,j}}$ это план запроса, выполняемый относительно исходного графа G_{Si} соответствующего источника данных. Глобальная часть $q_G^{'}$ плана исходного запроса выполняется в посреднике.

Имея шаблон триплета tp, федератор должен определить, с каким из виртуальных триплетов $t_{si,kj}$ он может быть сопоставлен. Источник данных является релевантным, если граф источника G_{Si} (который является виртуальным в случае источника данных, использующего адаптер) содержит триплеты, которые соответствуют шаблону триплетов tp.

Так как каждый адаптер используемый в системе SemWIQ использует подход прямого отображения метамодель-метамодель, то описание отображения является специфичной для каждого адаптера.

Семантика выполнения запросов в SemWIQ зависит подхода к федерированию запросов. В системе доступны два разных способа федерирование запросов: на основе триплетов и на основе экземпляров. При федерировании на основе триплетов обычно оценивается каждый шаблон триплетов из БГШ относительно глобального графа G_G. Федерирование на основе экземпляров требует выполнения некоторых специальных ограничений на структуру глобального графа G_G.

4.5 Обработка глобальных запросов в посреднике

Общая схема работы подсистемы обработки запросов в посреднике и подсистемы обработки запросов в источниках данных (или в адаптерах источников данных) приведена на рис. 7.

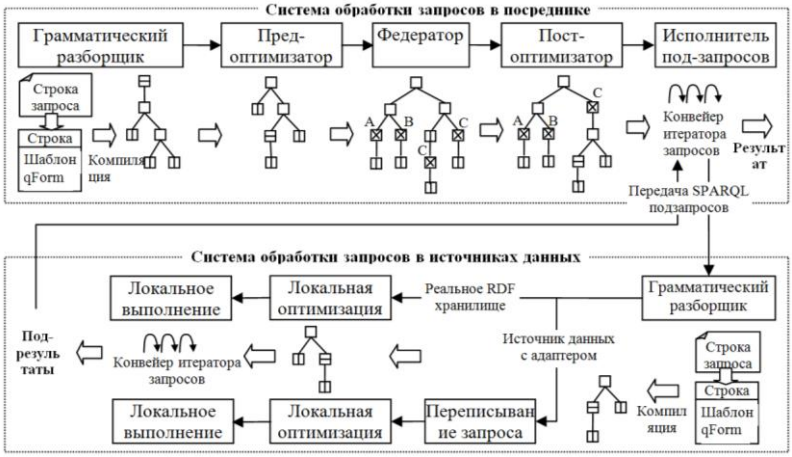

Рис. 7. Общая схема обработки запросов системы SemWIQ

В верхней части рисунка показана подсистема обработки запросов в посреднике. Процесс обработки запросов начинается с грамматического анализа текстовой строки запроса и заканчивается выполнением запроса с помощью итеративного конвейера запросов. Итераторы запросов формируют следующие друг за другом (последовательные) ответы (решения) на запросы (т.е. выполняют конечное связывание переменных), которые могут быть получены в цикле пользователями системы интеграции, отправившими запрос. Такой конвейерный подход требуется для экономии ресурсов и обеспечения высокой производительности и масштабируемости системы.

Обе подсистемы обработки запросов, прежде всего, должны выполнить грамматический разбор строки запроса в некоторое внутреннее описание на языке Java, прежде чем их базовый графовый шаблон будет преобразован в алгебраический план запроса. В подсистеме обработки запросов в посреднике, пред-оптимизатор применяет несколько логических упрощений и оптимизаций, таких, как например, проталкивание фильтров (показанные на схеме, как ⊟), насколько можно ближе к базовым графовым шаблонам (конечным вершинам, показанные на схеме, как ⊡).

Компонент Федератор (federator) декомпозирует план запроса (разбивает план запроса на запросы к отдельным источникам данных) и вставляет операторы Service (показанные на схеме, как ⊠). Все подпланы операторов Service являются локальными подпланами, которые передаются источникам данных (A, B и C в показанном примере). Пост-оптимизатор выполняет дальнейшую логическую оптимизацию запросов, такую, как объединение локальных подпланов, а также повторное проталкивание фильтров. Последним шагом работы подсистемы

обработки запросов посредника является потоковое выполнение составленного плана запроса.

В ходе выполнения запроса, подсистема обработки запросов посредника организует соединения с удаленными интерфейсами запросов источников данных и передает им под-планы, которые должны быть выполнены непосредственно на источниках данных.

Существуют два варианта обработки запроса в подсистема обработки запросов источников данных, которые показаны на рис. 7 в виде разветвления: верхняя ветвь показывает обработку запросов в источниках данных, которые являются реальными хранилищами RDF данных, а нижняя ветвь показывает обработку запросов в адаптерах, работающих с источниками данных других (не RDF) форматов, который должны переписывать запрос перед его выполнением.

Дополнительно, в обоих случаях, применяется локальная оптимизация, так как подсистема обработки запросов обычно имеет более точные статистические данные и метаданные (например, имеются индексы), чем посредник.

Необходимая для этого информация поддерживается компонентом RDFStats в форме сводок статистических данных, как будет описано далее.

Для каждого БГШ федератор создает модифицированный под-план, который содержит операторы Service, чтобы передать локальные планы интерфейсам обработки запросов зарегистрированным источникам данных.

5.6 Адаптеры источников данных

Каждый адаптер системы SemWIQ должен эффективно обрабатывать SPARQL под-запросы, для добавления решающих соответствий к результатам обработки глобального запроса. Обобщенная схема работы адаптеров также показана на рис. 7.

Адаптер системы интеграции должен быть способен представлять содержание источника данных S_i, использующего некоторую модель исходных данных отличную от RDF (реляционную, XML, электронную таблицу и т.п.), в виде RDF графа G_{Si} на основе использования отображения M_i. Более того, адаптер должен быть способен формировать ответы на SPARQL запросы к такому виртуальному графу G_{Si}. Для отображения произвольных структур данных на RDF граф требуется определение метамодели на языке RDF.

Базовый язык RDF Core (без семантики RDF Schema) имеет только возможности структурного моделирования, такие, как описание базовых (не типизированных) отношений, задание типов данных и составление коллекции. Возможности более высокоуровневого концептуального

моделирования предоставляются языками RDF Schema и OWL, которые также основываются на базовом языке RDF Core, и поэтому вся RDF метамодель в действительности описывается RDF графом.

RDF адаптеры могут быть разработаны на основе определений SPARQL запросов и порядка их выполнения. Обработка SPARQL запроса начинается с сопоставления базовых графовых шаблонов. В реализациях, основанных на триплетах, каждый шаблон триплета, входящий в состав БГШ, т.е., $tp_i \in bgp$ сопоставляется с активным графом G' набора RDF данных и всем переменным (шаблона трилета) tp_i задаются все возможные значения из (графа триплетов) G'. Результатом является решающая последовательность для каждого шаблона tp_i. Для оценки БГШ, совместимые соответствия двух решающих последовательностей объединяются. Далее решения полученные для Базовых Графовых Шаблонов объединяются операторами более высокого уровня и окончательный результат представляет собой конечный результат обработки запроса.

Обычно реализации такого процесса сопоставления основываются на итераторах и методах конвейерного выполнения, чтобы оптимизировать использование памяти и время ответа.

Общий подход к разработке виртуальных RDF адаптеров, которые могут отвечать на SPARQL запросы, заключается в реализации компонента, выполняющего сопоставление шаблонов триплетов. Данный компонент должен быть способен создавать (определять) для заданного шаблона триплетов tp_i решающие соответствия, на основе содержания исходного источника данных, который имеет схему S_i и спецификацию (описание) отображения M_i.

В связи с этим, базовая функциональность RDF адаптеров может быть достаточно просто реализована на основе существующих систем обработки SPARQL запросов (например, системы Jena ARQ [7]), которые отвечает за все операции более высокого уровня.

Источники данных имеющие различные метамодели могут быть преобразованы в RDF модель. В настоящее время разработано большое количество программ, позволяющих преобразовывать данные разных форматов в RDF формат, либо предоставлять доступ к данным в других форматах с помощью SPARQL конечных точек без физического преобразования. В табл. 1 приведены примеры приложений, которые позволяют выполнить преобразование из исходных форматов в RDF формат.

Таблица 1 Инструменты для преобразования исходных форматов в RDF данные

Приложение	Web-сайт	Описание

Aperature	http://aperture.sourceforge.net/	Aperature является средой для поиска, извлечения и индексирования данных различных форматов, которая позволяет разработчика преобразовать, как сами данные, так и их метаданные в RDF формат. В настоящее время имеется около 20 извлекаемых типов данных начиная с файлов в формате JPEG и MP3 и заканчивая документами в формате PDF, Word или Visio.
Flickcurl	http://librdf.org/flickcurl/	Созданная Dave Beckett на языке C реализация преобразования Flickr данных, таких, как метаданные для фото, тэги и местоположения в RDF данные.
Javadoc RDFizer	http://simile.mit.edu/wiki/JavadocRDFizer/	Проект, разработанный группой Simile из MIT, создает doclet, который будет преобразовывать любые совместимые с javadoc-данные в RDF данные. doclet это программа, которая реализует doclet API, для преобразования javadoc-ов в любой выбранный пользователем формат. Более подробную информацию о doclet-ах можно найти по адресу http://java.sun.com/javase/6/docs/technotes/guides/javadoc/doclet/overview.html.
RDF123	http://rdf123.umbc.edu/	Исследовательский проект (выполненный в University of Maryland, Baltimore County), в котором было разработано приложение, которое может обрабатывать простую табличную информацию, как например таблицы размеченные с помощью HTML тегов и файлов, содержащих текст разделенный запятыми и возвращать данные в RDF формате.
Torrent2RDF	http://www.inf.unideb.hu/~jeszy/rdfizers/torrent2rdf-0.3.zip	Torrent2RDF предоставляет базовый Java код для чтения torrent файла или torrent URL адреса и извлечения информации в виде RDF данных. В настоящее время вывод выполняется в выходной поток stdout, но GNU GPL лицензия дает разработчика возможность расширять программу, как требуется.

4.7 Реализация адаптеров с помощью среды Jena ARQ

Так как среда Jena ARQ [7] является очень хорошо расширяемой, то она может использоваться в качестве основы для разработку любых видов виртуальный RDF-адаптеров. В данной среде имеется много возможностей для подключения к системе обработки запросов дополнительной функциональности. Например, имеется возможность преобразовать сгенерированный алгебраический план и заменить

стандартные операторы языка SPARQL собственными операторами. Более того, имеется возможность предоставить собственные реализации итераторов запросов, как для стандартных, так и собственных алгебраических операторов.

Наиболее простым способом подключения собственного обработчика БГШ к стандартной подсистеме обработки запросов ARQ является реализация собственного генератора этапов выполнения сопоставления. Генератор этапов создает корневые решающие связывания для базовых графовых шаблонов. Например, для реализации адаптера RDB-to-RDF может быть реализован специальный генератор этапов, который будет формировать решения на основе описания отображений и данных, поучаемых из базы данных с помощью выполнения SQL запросов. Основной задачей является составление правильных и эффективных SQL запросов для заданного шаблона триплетов tp_i и отображения M, а также преобразование полученных результатов обработки SQL запросов в соответствующие RDF триплеты для tp_i.

Сгенерированный SQL запрос, являющийся комбинацией существующих утверждений отображения (полей базы данных в понятия глобальной схемы) $g \rightsquigarrow q_S$, может рассматриваться, как RDF представление схемы источника данных S_i в соответствии с описанным отображением M_i.

Общий подход к разработке адаптеров на основе Jena ARQ и собственного оценивания БГШ требует использования некоторых способов оптимизации, для достижения приемлемой производительности при больших объемах данных. Такая оптимизация должна включать:

- При оценивании БГШ, система обработки запросов должна выполнять новое упорядочение шаблоны триплетов на основе ожидаемой мощности решающих соответствий, для уменьшения количества промежуточных связываний, а следовательно потребления памяти и времени работы процессора. Например, если имеется набор данных с FOAF описание 1000 человек, из которых два человека старше 30 лет, то дешевле будет сперва выполнить сопоставление с шаблоном `{?s foaf:age ?age}`, а уже затем выполнять сопоставление с шаблоном `{?s a foaf:Person}` для каждого значения `?s` из решающих соответствий первого шаблона, чем оценивать эти триплеты в обычном порядке.
- Должны использоваться любые низкоуровневые индексные структуры, которые доступны, как часть исходной информационной системы.

Прежде чем выражения фильтров и упорядочения могут быть учтены на уровне БГШ, они должны продвинуты когда это возможно в алгебраическом плане.

4.8 Адаптеры для реляционных баз данных

С момента появления концепции Semantic Web большое внимания уделяется отображению реляционных БД в RDF формат [23] и возможности оперативного выполнения запросов и преобразования данных. Достаточно точно можно предполагать, что в настоящее время большинство информации хранится в реляционных базах данных. К современным технологиям разработки адаптеров для РСУБД относятся: D2RQ-Map [24], R2O [25], OpenLink Virtuoso RDF Views [26], подходы предложенные в [27 и 28], SquirrelRDF [29].

Различие между технологиями состоит в том, что технологии D2RQ-Map, R2O и Virtuoso RDF Views основываются на семантике предметных областей, в них используются отображения для связывания произвольной реляционной модели с глобальной онтологией предметной области, а технология SquirrelRDF непосредственно генерирует RDF данные из БД и не позволяет выполнять отображения на заранее составленные онтологии.

В системе интеграции SemWIQ адаптеры реляционных СУБД строятся с использование системы D2RQ-Map и сервера D2R (Рис. 8), который позволяет публиковать содержание РСУБД в Semantic Web, глобальном информационном пространстве, содержащем Связанные Данные (linked data). Сервер D2R использует настраиваемые D2RQ отображения для представления содержания БД в формате RDF и предоставляет возможность просматривать RDF данные и выполнять в них поиск. Хотя средства отображения в системе D2RQ-Map могут использоваться в виде библиотеки приложениями Semantic Web, сервер D2R-Server добавляет конечную SPARQL точку, которую могут использовать Web приложения и интерфейс пользователя, называемый Snorql.

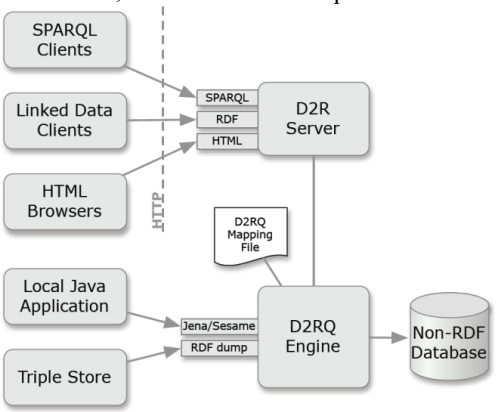

Рис. 8 Архитектура системы D2RQ

Технология D2RQ-Map основана на описании отображений вида Global-as-View. Поэтому, составление D2R отображения является в основном процессом создания утверждений в форме: $g \rightsquigarrow q_S$, как это было описано ранее. Для одного глобального понятия g, которое может быть RDF классом или свойством, описывается запрос q_S относительно схемы базы данных источника S.

D2RQ-Map не поддерживает непосредственно SQL запросы, однако, она поддерживает все, что требуется для создания 1 : 1, 1 : n и n : n соответствий, включая функциональные соответствия (основанные на SQL выражениях и функциях), таблицы преобразования значений и условия (с использованием SQL выражений). Соответствия свойств (называемые мостами свойств, property bridges) описываются в виде карт классов (class maps). D2RQ-Map использует основанному на понятиях подход, в котором, в общем случае, предполагается, что все результирующие ресурсы являются экземплярами некоторого RDF класса, хотя также возможно создать и не типизированные ресурсы. Карта классов может рассматриваться как группа одинаковых целевых ресурсов (как, например, люди или документы), имеющих общие свойства, а поэтому и общие мосты свойств.

Сами отображения описываются на языке RDF, что облегчает запись ссылок на внешние RDF понятия онтологий. Отображение D2RQ позволяет выполнять одношаговое преобразование во время выполнения и облегчает использование низкоуровневых индексов и других способов оптимизации во время обработки запросов.

Имеются разные способы управлять формированием RDF вершин из результатов обработки SQL запросов. В основном возможно генерировать URI ресурсы (URI идентификаторы), пустые вершины и типизированные литералы. Также имеется несколько возможностей обработки исходных значений с помощью SQL выражений и функций, и возможность создания URI идентификаторов на основе специальных строковых шаблонов.

Такими возможностями можно управлять с помощью специальных свойств словаря отображений D2RQ. Для обработки операций SPARQL более высокого уровня (таких, как фильтры, необязательные элементы, объединения и модификаторы решений), система D2RQ-Map использует подсистема обработки запросов, предоставляемую средой Jena ARQ.

В связи с этим важным является оптимизация планов запросов, путем переупорядочения шаблонов триплетов на основе ожидаемых объемов получаемых результатов (мощностей) и проталкивания фильтров в сгенерированный SQL запрос, везде, где это возможно.

Оценивание шаблонов триплетов сравнимо с низко-уровневыми операциями доступа с помощью языка SQL. Фильтры, связывания и другие операции выполняются на более высоком этапе работы конвейера обработки запроса, через который проходит поток промежуточных связываний до тех пор, пока результирующее связывание будет

возвращено в качестве решения (результата обработки запроса). Таким образом, система D2RQ-Map полностью соответствует процессу выполнения запросов в системе SemWIQ.

Система D2RQ-Map использует достаточно мощные средства (формирования) отображения, которые могут справиться практически со всеми видами разнородностей моделей источников данных.

4.9 Формирование статистики об источниках данных

Оптимизация обработки запросов в значительной степени зависит от статистических данных источников данных. Использование таких данных позволяет решать проблему распределенности данных и уменьшать объемы передаваемых по сети данных, котрые требуются для обработки связей между источниками данных.

Для формирования сводок статистических данных в системе SemWIQ был разработан компонент RDFStats [21]. Данный компонент состоит из RDF генератора статистических данных для наборов RDF данных, доступного через его конечную SPARQL точку, и набора функций оценки мощностей (количества возвращаемых элементов). Эти функции используются для оценки ожидаемого минимального, среднего и максимального количества результатов для шаблонов триплетов, базовых графовых шаблонов и полных планов запроса.

Так как компонент RDFStats предоставляет максимальные оценки) (т.е. верхние границы), то возможно дать гарантированные ответы на вопрос, вернет ли конкретный источник данных вообще какие-либо результаты (верхняя граница оценки должна быть больше нуля). Компонент RDFStats также поддерживает оценку для запросов, содержащих интервальные ограничения, описанных с помощью шаблонов триплетов и базовых графовых шаблонов, содержащих фильтрующие выражения (FILTER).

Для каждого источника данных, посредник поддерживает набор данных RDFStats, который включает сводку данных для под-графа, который добавляется к системе интеграции соответствующим источником.

При обработке запросов, посредник (компонент федератор) должен уметь определять, какие источники данных являются релевантными обрабатываемому запросу, а также уметь оценивать количество решающих связываний (промежуточных решений) для рассматриваемых под-планов запроса. Также, как и в реляционных СУБД в системах интеграции для оптимизации запросов используются гистограммы распределения значений. Под гистограммой некоторого атрибута понимается распределение его значений по n взаимно не пересекающихся

под-множеств (интервалов, которые также называются контейнерами, bins).

Система RDFStats [21] состоит из генератора статистических данных и прикладного интерфейса RDFStatsModel API для управления и доступа к статистике RDF графа, включая функции оценивания для мощностей SPARQL шаблонов запросов. Генератор статистических данных собирает статистику для RDF графов, которые имеются у локальных или удаленных конечных SPARQL точек (а также для RDF графов, загружаемых из файлов) и формирует несколько гистограмм.

В RDFStats включено два генератора статистики: для конечных SPARQL точек и для локальных RDF документов. Генераторы статистики вычисляют одну гистограмму для URI идентификаторов субъектов, они учитывают субъекты, соответствующие пустым вершинам графа, для всех свойств создают гистограммы (по одной гистограмме для каждого определенного URI идентификатора свойства и типа объектов). Для основных типов данных XML, таких, как булевый, целый, вещественный, даты и строки разработаны специальные построители гистограмм.

Компонент RDFStats предоставляет прикладной интерфейс RDFStatsModel API для управления множеством статистических данных для разных источников и интерфейс RDFStatsDataset API для получения доступа к статистическим данным конкретного источника данных. Прикладной интерфейс Histogram API предоставляет все низко-уровневые функции работы с гистограммами.

Описание статистических данных выполняется с помощью словаря (простой онтологии) статистики RDFStats, основанного на словаре SCOVO (Statistical Core Vocabulary) [30], который позволяет описывать статистическую информацию о сети связанных данных (Web of Data).

4.10 Решение проблемы потери соответствия

Так как между данными интегрируемых источниках потенциально могут быть связи (например, значение `dc:creator` для некоторого документа должно быть правильно связано с описаниями авторов, которые могут быть в другом источнике данных), то полезным является использование инструментов связывания данных из различных источников. Такие системы предназначены для решения проблемы потери соответствия (impedance mismatch) между описаниями экземпляров понятий глобальной модели и данными из различных источников.

Существуют общие системы, как например Febrl [31], которые могут использоваться для решения задач связывания записей в СУБД, а также имеется системы специальные предназначенные для Semantic Web, как например Silk [32].

Система Silk Link Discovery Framework предназначена для задания явных RDF связей между элементами данных из разных источников. Она реализована на языке Scala, который может выполняться в Java Virtual Machine.

В системе Silk используется декларативный язык описания связей (Silk – Link Specification Language, Silk-LSL), с помощью которого разработчики систем интеграции данных могут описывать типы RDF связей, которые должны определяться между источниками данных, а также условия, которым должны удовлетворять элементы данных, чтобы быть связанными. Для описания условий связывания могут использоваться различные метрики сходства, а также может учитываться граф, окружающий элементы данных, что может быть задано с помощью языка описания путей в RDF данных.

Система Silk получает доступ к источникам данных, которые должны быть взаимосвязаны также с помощью протокола SPARQL и поэтому, могут использоваться как для локальных, так и для удаленных конечных SPARQL точек.

Основными особенностями системы обнаружения связей Silk (Silk link discovery engine) являются: поддержка формирования RDF связей (связи `owl:sameAs`, а также другие типы отношений); гибкий, декларативный язык для описания условий задания связей; возможность использования в распределенных средах (путем получения доступа к локальным и удаленным конечным SPARQL точкам); возможность смешивания терминов из разных словарей, когда не существует согласованной RDFS или OWL схемы.

Литература

1. Ballard C., Davies N. Gavazzi M., Stephani J., Lurie M. IBM Informix: Integration Through Data Federation. IBM International Technical Support Organizat, 2003. - 270 p. (http://www.iiug.org/library/ids/technical/sg247032.pdf)
2. Calvanese D., De Giacomo G., Lembo D. et al. Ontologiesand Databases: The DL-LiteApproach. Semantic Technologies for InformationsSystems -5th Int. Reasoning Web Summer School (RW 2009). LNCS, Vol. 5689, 2009.
3. Бездушный А.А. Математическая модель интеграции данных на основе дескриптивной логики, диссертация к.ф.-м.н., 2008.
4. Когаловский М.Р. Системы доступа к данным, основанные на онтологиях. // Труды Второго Симпозиума «Онтологическое моделирование». – М: ИПИ РАН, 2010.
5. Hayes P., McBrien B. RDF Semantics W3C Recommendation, 2004, http://www.w3.org/TR/rdf-mt/.
6. Prud'hommeaux E., Seaborne A. SPARQL Query Language for RDF, W3C Recommendation, 2008, http://www.w3.org/TR/rdf-sparql-query/.

7. Jena Community: Joseki – A SPARQL Server for Jena, http://www.joseki.org/. 2009.
8. Berners-Lee T., Chen Y., Chilton L., Connolly, D. Tabulator: Exploring and analyzing linked data on the SemanticWeb, Proceedings of the ISWC Workshop on Semantic Web User Interaction, CEUR Workshop Proceedings, 2006.
9. W3C SWIG, Linking Open Data W3C SWEO Community Project, 2009, http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
10. Ding L., Finin T., Joshi A. et al. Swoogle: a search and metadata engine for the semantic web, CIKM '04: Proceedings of the thirteenth ACM international conference on Information and know-ledge management, ACM, New York, NY, USA, 2004, pp. 652–659.
11. Harth A., Umbrich J., Decker, S. MultiCrawler: A Pipelined Architecture for Crawling and Indexing SemanticWeb Data, 5th International Semantic Web Conference, Athens, USA, 2006.
12. Tummarello G., Delbru R., Oren E. Sindice.com: Weaving the Open Linked Data, Proceedings of the 6th International Semantic Web Conference (ISWC), 2007.
13. Cyganiak R., Bizer C. Pubby – A Linked Data Frontend for SPARQL End-points, 2009, http://www4.wiwiss.fu-berlin.de/pubby/.
14. Bizer C., Gauß T., Cyganiak R., Hartig O. Semantic Web Client Library – Querying the complete Semantic Web with SPARQL, 2009, http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/.
15. Hartig O., Bizer C., Freytag J.-C. Executing SPARQL Queries over theWeb of Linked Data, 8th International Semantic Web Conference (ISWC2009), 2009.
16. Quilitz B., Leser U. Querying Distributed RDF Data Sources with SPARQL. ESWC 2008. LNCS, vol. 5021, pp. 524–538.
17. Langegger A., Woss A. A Semantic Web Middleware for Virtual Data Integration on the Web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 493–507.
18. Langegger A. A Flexible Architecture for Virtual Information Integration based on Semantic Web Concepts, Phd Thesis, 2010.
19. Lynden S., Kojima I., Matono A., Tanimura Y. ADERIS: Adaptively Integrating RDF Data from SPARQL Endpoints, 2010.
20. Cheung K.-H., Frost R., Marshall S., et el. A journey to Semantic Web query federation in the life sciences, 2009.
21. Langegger A., Wöß W. RDFStats - An Extensible RDF Statistics Generator and Library," DEXA, 2009 20th International Workshop on Database and Expert Systems Application, 2009, pp.79-83.
22. Alexander K., Cyganiak R., Hausenblas M., Zhao, J. Describing Linked Datasets, in C. Bizer, T. Heath, T. Berners-Lee and K. Idehen (eds), Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW2009), CEUR Workshop Proceedings, 2009.
23. Berners-Lee, T.: Relational Databases on the Semantic Web, 1998, http://www.w3.org/DesignIssues/RDB-RDF.html.
24. Bizer C., Cyganiak R.D2R Server – Publishing Relational Databases on the Semantic Web, 5th International Semantic Web Conference, 2006.
25. Rodriguez J. B., Corcho O., Gomez-Perez, A. R2O, an Extensible and Semantically Based Databaseto-ontology Mapping Language, 2004.
26. Erling, O., Mikhailov, I. RDF Support in the Virtuoso DBMS, CSSW, 2007, pp. 59–68.

27. Chen H., Wu Z., Mao Y. Rewriting Queries Using Views for RDF-Based Relational Integration, 17th IEEE International Conference on Tools with Artificial Intel-ligence (ICTAI'05), IEEE Computer Society, Los Alamitos, CA, 2005, pp. 260–264.
28. Laborda C. P., Conrad, S. Bringing Relational Data into the SemanticWeb using SPARQL and Relational.OWL, ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops, IEEE Computer Society, Washington, DC, USA, 2006, p. 55.
29. Steer D. SquirrelRDF Homepage, http://jena.sourceforge.net/SquirrelRDF/.
30. Ayers D., Feigenbaum L., Halb W., et el. The Statistical Core Vocabulary (SCOVO), 2009, http://purl.org/NET/scovo.
31. Christen P. Febrl – an open source data cleaning, deduplication and record linkage system with a graphical user interface, KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, 2008, pp. 1065–1068.
32. Volz J., Bizer C., Gaedke M., Kobilarov G. Silk - A Link Discovery Framework for the Web of Data, Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW2009), CEUR Workshop Proceedings, 2009.

Abstract

Ontology-based data integration using Semantic Web technologies

A. F. Tuzovsky

The development approach of virtual data integration using Semantic Web technologies is described. The basic Semantic Web standards (RDF and SPARQL) and state-of-the-art in the field of federated query processing over RDF data are explained. The architecture of virtual integration systems and methods of problem solving are shown on example of DARQ and SemWIQ systems.