П. А. Марчук

ИСПОЛЬЗОВАНИЕ НЕСПЕЦИФИЧЕСКИХ ОНТОЛОГИЙ ДЛЯ ХРАНЕНИЯ ФАКТОГРАФИЧЕСКИХ ДАННЫХ

ВВЕДЕНИЕ

В настоящее время существует множество подходов к созданию информационных ресурсов. Наиболее распространённым подходом является создание ресурса на базе единого хранилища с общим администрированием и присоединение к нему разделённых данных. Однако для многих целей такой подход неприменим, поскольку требует согласия всех участников быть зависимыми от этого ресурса. При этом данные участников не просто становятся формально доступными для пользования в едином хранилище, но и фактически передаются. В процессе работы новая версия данных может находиться либо в главном хранилище, и тогда за новейшей информацией приходится обращаться к нему, либо, как и в начале, у участника, и тогда в главном хранилище оказываются устаревшие данные. Периодическая синхронизация данных решает проблему их устаревания, однако сама по себе является нетривиальной задачей и не снимает проблему собственности.

При переходе от единого хранилища к распределённой модели хранения данных [1] создаётся единое информационное поле, в котором участники контролируют свои данные, предоставляют к ним доступ и при этом получают возможность расширить свои данные за счёт привлечения данных из смежных областей. Элементы перехода к распределённому хранению данных и его практическое применение для электронных архивов является целью данного исследования.

Уже вошедшая в обиход концепция Resource Description Framework (RDF) [2] предлагает несколько вариантов решения проблемы распределённого хранения данных и возможности их связывания в единую информационную систему (или системы) нового поколения. Однако для учёта многих специфических деталей работы с электронными архивами необходимо дополнить эту концепцию собственной методологией.

С точки зрения программного обеспечения существует несколько решений на базе RDF для доступа к данным, например, Sesame [3] или Jena [4].

В нашей работе эти программные продукты не обеспечивают полноту решения и могут применяться лишь для части подзадач.

1. ПОДДЕРЖИВАЕМЫЕ ДАННЫЕ И МЕСТА ХРАНЕНИЯ

Определим исходные данные, которые должны содержаться в информационной системе.

Первый вид данных — это собственно объекты хранения, т.е. предметы, которые представлены в архивах, библиотеках, музеях и т.д.; например, фото-, видео- и аудиодокументы или ссылки на труды и артефакты. В случае со ссылками первичные объекты — это то, на что они ссылаются.

Второй вид данных — это информация о первичных объектах, метаданные. Метаданные включают названия, описания, идентификаторы, а также связи с другими объектами второго вида. Для каждого первичного объекта создаётся метаобъект, и уже метаобъекты связываются в семантическую сеть [5]. Помимо этого, метаобъекты создаются для объектов реального мира, которые должны быть представлены в информационной системе. Объекты реального мира — это персоны, организации, мероприятия и события, географические объекты и т.д.

Программные продукты для операций с объектами могут различаться. Они могут иметь различные версии, располагаться на разных серверах или виртуальных машинах и нести разную операционную нагрузку. Это могут быть серверы для публичного представления, внутреннего редактирования, резервного хранения, тестирования новых версий онтологий и программного обеспечения и т.п. На каждом из этих серверов или виртуальных машин могут содержаться разные виды данных, конфигурационные файлы и версии программного обеспечения. Однако все они осуществляют поддержку фактографической информационной системы.

2. ИНТЕГРАЦИЯ ДАННЫХ ИЗ РАЗНЫХ ИСТОЧНИКОВ

При формировании информационной системы осуществляется введение имеющейся информации. При этом порой необходимо интегрировать информацию из разных источников. Интеграция информации представляет определённые проблемы, например, могут различаться форматы данных, идентификация объектов, а информация может дублироваться или быть противоречивой.

Первостепенной является задача унификации формата данных. Для этой цели в данной работе применяется модификация интероперабельного стандарта метаданных Dublin Core [7].

Поскольку наиболее важными в контексте нашей системы являются фактографические данные, онтология системы основывается именно на этих параметрах. В качестве базовых понятий взяты следующие типы метаобъектов:

1) персона — это, как правило, человек, хотя может быть и другое живое существо;
2) организационная система — организация, мероприятие, отдел, ассоциация, клуб и т.п.;
3) географическая система —страна, регион, город (с точки зрения места), океан, море, река, локальное природное место и т.п.;
4) документ —текст, фото, видео, аудио и т.п.

Множества, обозначаемые этими понятиями, в принципе могут пересекаться и не позволяют описать весь мир, однако достаточны в качестве фактографической базы для нашей системы.

Онтология описана на языке OWL (Web Ontology Language) [8]. На её основе программные системы могут генерировать интерфейсы для работы с данными. Таким образом возможно получение редактора данных, достаточно универсального, пока онтология описана соответствующим образом, а данные соответствуют онтологии.

Данные в системе хранятся в виде XML-файлов формата RDF. При добавлении новых данных в систему из какого-либо источника возможно создание нового файла для этих данных, что даёт возможность повторной интеграции путём замены непосредственно этого файла и учёта этих данных как данных из конкретного источника.

3. ЭКСПРЕСС ВНЕСЕНИЕ ДАННЫХ В БАЗУ ПО ИМЕЮЩЕЙСЯ ОНТОЛОГИИ

При интегрировании данных в единую информационную систему, помимо источников с той или иной формальной структурой данных возможно столкновение с источником без какой-либо формальной структуры. Например, это может быть текст, из которого можно почерпнуть нужные факты, или знания конкретного человека, которые можно внести в информационную систему в виде данных, полученных из рассказа. Возникает проблема экспресс-ввода метаобъектов.

Ввод информации должен быть оперативным, поскольку данных может поступать много в ограниченный промежуток времени. Например, при внесении данных по рассказу в реальном времени (интервью) задержки при вводе информации могут привести к тому, что человек потеряет мысль или же не успеет рассказать всю интересующую информацию, и она останется незафиксированной. Кроме того, поступление данных без формальной структуры либо данных, не укладывающихся в существующие категории, может существенно задержать и затруднить работу операторов системы.

В связи с этим необходимо в кратчайшие сроки создавать новые метаобъекты. Определим, что представляет собой внесение нового метаобъекта в систему. Как правило, в первую очередь вносится имя этого объекта. Отметим, что имя несущественно для некоторых объектов, таких как, например, фотографии или аудиозаписи, поскольку не представляет фактографического интереса и может генерироваться автоматически при внесении в систему. Наиболее важным является установление для нового объекта как можно большего количества ассоциативных связей при внесении в семантическую сеть. Интерес представляют связи, в которых объект состоит с другими объектами, как ранее внесёнными, так и отсутствующими в системе, но фактографически релевантными с учётом позиционирования конкретного архива, музея или библиотеки.

Рассмотрим ввод новых метаобъектов на примере фотодокументов.

В первую очередь необходимо установить, что отражено на фотографии. Отражение — отношении в онтологии, означающее, что некоторый объект фигурирует в некотором документе. Используя стандартный вид архивной карточки фотографии с учётом имеющейся онтологии, необходимо установить следующие факты:

1) когда это происходит (дата);
2) где это происходит: как правило, город (геосистема), где сделана фотография;
3) что здесь (на фотографии) происходит: например мероприятие (оргсистема);
4) кто здесь отражен: персоны, возможно здание организации (оргсистема), возможно какая-нибудь река на заднем плане (геосистема);
5) кто автор фотографии (персона).

Если при этом тот или иной объект не описан в системе, необходимо сразу же внести его в систему и по возможности описать до перехода к следующей фотографии. Если объект уже присутствует, то для фотографии

необходимо установить с ним ассоциативную связь, а не добавлять его дубликат; соответственно, нужен механизм оперативного поиска объектов.

Проблематика быстрого ассоциирования нового объекта с существующими достаточно интересна. Ниже приведены некоторые из проблем и решения, предложенные в разработанной системе.

Отсутствие четкого формального принципа создания названий. Проблема типична для таких объектов, как мероприятия.

1. Давно прошедшие официальные мероприятия, статус которых оператору трудно установить; например, конференция может называться слётом, симпозиумом или конгрессом.

2. Неофициальные мероприятия, названия у которых отсутствует вообще; например, выезд на природу можно назвать пикником или отдыхом на природе. Фактографически такие мероприятия могут быть весьма значимыми, например, если на них присутствовали руководители стран, крупных организаций и т.п. Спецификацию наименований подобных мероприятий должны разрабатывать специалисты, отвечающие за конкретный информационный ресурс.

Со своей стороны, мы можем помочь этому, во-первых, давая возможность специалистам ввести несколько названий этому мероприятию. В нашей онтологии эта возможность предоставляется при помощи конструкции «именование», позволяющей к существующему метаобъекту добавить псевдонимы. При этом при поиске нужного нам мероприятия, поиск будет осуществляться не только по имени, но и по псевдонимам. Во-вторых, конечно, мы можем использовать и стандартный поиск, позволяющий искать по ключевым словам, по дате и другим свойствам объекта. Используя преимущества нашей онтологии, мы:

1) ищем не абстрактный текстовый документ с такими словами, как это делают стандартные поисковые системы, а объект с соответствующими ключевыми словами в определении или соответствующими свойствами;

2) можем осуществлять более гибкий поиск с использованием связей искомого объекта с другими объектами базы данных (например, «человек, работающий в определенной организации»).

Проблема разных названий может встречаться не только у мероприятий, но и у людей, и у организаций. У людей это, например, смена фамилий, либо псевдонимы. У организаций — сокращенные названия, названия на других языках. При помощи конструкции «именование» и соответствующей организации поиска, мы обнаруживаем подход к решению этой проблемы.

Проблема просмотра всей базы для поиска ассоциируемых объектов. Поиск по все базе данных занимает время, особенно, если запрос для этого поиска вовлекает в себя не только ключевые слова и непосредственные свойства, но и связи с другими объектами. Кроме того, часто документы вводятся из одной серии либо одного года, поэтому для упрощения ввода новых документов программа может помогать пользователю, контролируя контекст новых вводимых документов.

Рассмотрим несколько возможных вариантов контекстного контроля. Самое простое — это статические параметры, которые в данный момент присутствуют у всей вводимой серии. Например, это автор документа либо источник поступления. Реже — дата документа, конкретное мероприятие, которое отражено в этой серии документов.

Контекст не является физической папкой для хранения документов. Контекстные «навески» не обязательно соответствуют виртуальным папкам, которые «навешиваются» на серию объектов. Термин «виртуальная папка» часто используется в нынешних структурах хранения данных, однако это только одно направление структуризации, один вид грани на ассоциативном графе. В нашей онтологии для каждого объекта устанавливается соответствующая ассоциативная связь (например, с автором этого документа), либо указывается конкретное свойство (например, дата). Хотя по каким-то темам эти документы все равно могут быть сгруппированы, исходя из предпочтений информационного специалиста, и в этом случае это будут виртуальные папки («коллекции» в онтологии).

Однако несмотря на то, что хранение контекста происходит при помощи модификации свойств или установления связей для каждого объекта, для ускорения процесса ввода мы должны определить контекст сразу для всей вводимой серии документов.

Помимо названных статических параметров, существуют аналогичные динамические параметры, т.е. те, которые очень похожи в этой серии документов, однако полного совпадения между ними нет. Например, это может быть список отраженных персонажей в документе. Если у нас идет серия документов, то персонажи при этом часто повторяются, и мы снова сталкиваемся с понятием контекста. В качестве простейшего варианта решения мы можем сразу вывести список недавно использованных объектов заданного типа, например, список персон, с которыми недавно была установлена ассоциативная связь. При этом, если мы будем выводить их в порядке убывания времени их последнего «использования», то при минимальных затратах это будет существенно увеличивать скорость ассоциирования документов и персон.

Вкратце обозначим более сложные возможности подбора контекстных динамических параметров. Помимо их недавнего использования в процессе ввода, у нас уже имеется семантическая сеть между другими документами и персонами, между организациями и персонами. К тому же у нас есть хронологическое распределение объектов, географическое распределение объектов и т.д. Исходя из этого при помощи более сложных алгоритмов мы можем осуществлять выборку нужного контекста и ощутимо ускорять ввод новых документов, а также подсказывать пользователю возможные варианты.

4. РАСПРЕДЕЛЕННОЕ РЕДАКТИРОВАНИЕ ДАННЫХ

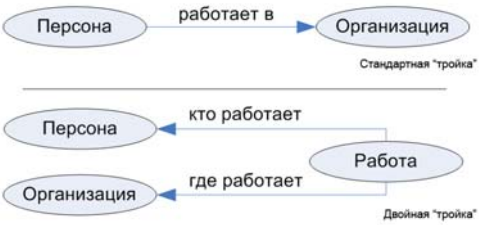
В этом разделе речь идет прежде всего о метаобъектах, которые содержатся в нашей информационной системе. Некоторые из них были внесены в нашу конкретную систему, некоторые могут храниться удаленно (при этом в нужном нам формате). Некоторые могут быть также на другом удаленном сервере и, к тому же, в другом формате, но при этом у нас есть импорт-фильтр для их использования. В общем, исходя из первоначальных источников мы получаем RDF-документы, в которых хранятся метаобъекты. В совокупности эти RDF-документы дают нам единое информационное поле, с которым мы должны работать.

При работе с этими метаобъектами в нашем информационном поле в режиме чтения, если у нас нет проблем с дублированием идентификаторов, вся работа выстраивается достаточно простым способом. Например, мы должны найти объект, чтобы посмотреть его свойства, либо мы должны найти объекты, которые ссылаются на этот объект, чтобы отобразить их свойства. Методология такого поиска объектов, части нашего ассоциативного графа описана, например, в языке поиска SPARQL Query Language for RDF [9].

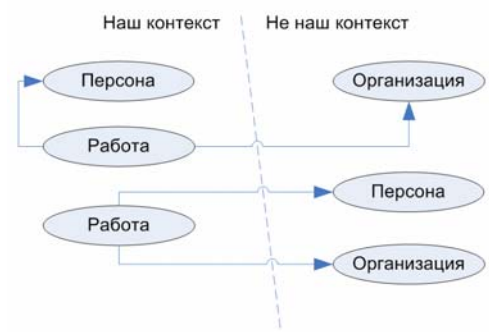
В этом разделе сформулирована проблема, а так же предложен вариант решения этой проблемы в том случае, если нам требуется редактировать информацию в нашем информационном поле, при том, что у нас есть источники, которые находятся не в нашей собственности.

Сначала обозначим некоторую принципиальную для данного раздела особенность нашей онтологии. В стандартной концепции графов RDF ассоциативные связи между объектами обозначаются при помощи «троек» (Triples) RDF. В тройку входит субъект, предикат и объект (см. рисунок, первая часть). В нашей же онтологии базовая схема установления ассоциативной

связи между объектами выглядит следующим образом: ассоциативную связь, которую мы устанавливаем, мы также заводим в качестве метаобъекта в базу данных, и эта связь ссылается на объекты, между которыми мы эту связь устанавливаем (см. рисунок, вторая часть). Фактически одна связь представлена в виде двух «троек».



Теперь определим контекст работы. Зададим простейшую модель, при которой в нашей единой распределенной информационной системе существуют только 2 источника — наш источник, где мы можем редактировать данные, и чужой источник, данные которого мы редактировать не можем, но можем просматривать, а он наши данные ни просматривать, ни редактировать не может. При этом нашей областью просмотра является вся распределенная система, а контекстом редактирования — только наш источник.



Исходя из этой модели и нашей онтологии, мы можем добавлять в наш контекст новые метаобъекты, редактировать их свойства и устанавливать связи между нашими метаобъектами. Кроме того, мы можем устанавливать связи и между объектами разных источников (см. рисунок). При этом, имея областью просмотра все источники, мы получим полную информационную картину. Второй источник полной картины не получит, так как его область просмотра по нашему определению не включает нашего источника, но и противоречий наши добавленные объекты не вызовут.

Так как основу семантической сети составляют скорее связи между объектами, нежели конкретные свойства каждого объекта, то, используя приведенную выше модель, мы достигаем заметного результата, однако полной универсальности данному решению пока недостает.

Например, поскольку мы создаем много новых метаобъектов, а у каждого метаобъекта могут быть несколько свойств, то свойства также начи-

нают играть ощутимую роль. Мы можем захотеть изменить уже имеющееся свойство, например, неправильное название организации, либо добавить новое свойство, например, дату основания.

Кроме того, мы можем посчитать, что какой-то метаобъект неверен, например, человек никогда не работал в такой-то организации, а в базе данных есть соответствующее упоминание. Таким образом, мы должны удалить какой-то метаобъект. Понятно, удалить его из своего контекста не составляет труда, однако это становится проблемой в случае с чужим контекстом.

В случае с главной централизованной базой данных данная проблема решается на уровне полномочий. Пользователю дается право на редактирование определенной области данных; возможно, после редактирования это действие должно быть подтверждено модератором соответствующей области. В этом случае после внесения изменений мы даже можем сохранить предыдущую версию, чтобы, возможно, эти изменения отменить, если нужно. В принципе, мы можем перенести эту схему и на распределенную модель хранения данных. Тогда каждый источник может определить полномочия других пользователей информационной системы и контролировать свой «куст» данных. Недостатком этого подхода является обязательное наличие серверной части, отслеживающей полномочия всех пользователей. Кроме того, свойства этих объектов тогда становятся частью «куста» данных определенного владельца, хотя могут не соответствовать его подходу. В принципе, обозначенные недостатки могут не являться проблемными местами информационной системы.

Рассмотрим случай с отсутствием серверной части, определяющей, какие объекты можно редактировать. Например, чужие «кусты» мы просто получаем при доступе через http-протокол, и там не предусмотрено редактирование.

В этом случае нам все равно требуется внести изменения, однако это надо сделать исключительно в своем контексте редактирования. При этом при визуализации изменения должны корректно показываться, то есть, по крайней мере, в нашей части системы они должны выглядеть именно так, как мы хотим их увидеть.

Итак, чтобы внести изменения в свойства метаобъекта, который находится в удаленном «кусте», предлагается следующий вариант решения. Мы создаем новую сущность у себя в контексте редактирования. Наделяем ее при этом всеми теми же свойствами, что были в удаленном объекте. Теперь этот метаобъект находится в нашем «кусте», и мы можем произвести все необходимые нам изменения. Однако в семантической сети ссылки произ-

водятся все еще на старый объект. В своем контексте редактирования мы, в принципе, можем исправить все ссылки на старый объект на ссылки на новый добавленный объект, однако ссылки на старый объект остаются во всех остальных источниках. Поэтому мы добавляем дополнительное указание для просматривающей системы, что произошла смена идентификатора, и новый объект «заменил» старый. Тогда информационная система должна при получении старого идентификатора автоматически его заменять на новый.

Аналогично мы можем поступать и в случае, когда у нас больше чем 2 источника, и каждый из них областью просмотра имеет все источники, а также может редактировать свойства всех увиденных метаобъектов. В этом случае нам требуется заменить идентификатор столько раз, сколько было совершено этих копирований метаобъектов из одного источника в другой.

При некотором изменении данной схемы мы можем аналогично удалять метаобъекты из чужих «кустов», обозначая в нашем контексте, что этого метаобъекта не существует (для нас).

5. РЕДАКТИРОВАНИЕ ПЕРВИЧНЫХ ОБЪЕКТОВ СОХРАНЕНИЯ

Напомним, что первичные объекты сохранения — это то, что хранится в данном конкретном архиве, музее, библиотеке, т.е. это могут быть фотографии, аудиозаписи, видеодокументы, просто текстовые документы в отсканированном виде и тому подобное. Во-первых, нам надо определиться, в каком случае первичные объекты нуждаются в редактировании, и стоит ли их редактировать вообще. Само действие может быть простейшим, тем не менее, требуется определить, действительно ли редактирование не нанесло ущерб исходным данным.

Простым редактированием могут быть поворот либо вертикальное/горизонтальное отражение документа. Так же это могут быть вырезка белых полей, цветовое улучшение, профессиональное восстановление документа дизайнером и так далее.

Кроме того, для разных интерфейсов нашей информационной системы нам могут потребоваться различные версии одного и того же документа. Например, для их печати нам требуется максимальное качество, для просмотра нам требуется оптимальное соотношение между качеством и временем загрузки. Для веб-интерфейса нам требуются соотношение между качеством, разрешенным владельцем документа для выставления в Интернет,

и размером занимаемой им памяти, а также, как правило, еще «предпросмотровый» вариант этого же документа меньшего размера.

Независимо от того, в каком формате документ попадает в архив, если с ним производить изменения (например, поворот) в архиве, его целесообразно сохранять уже в формате, который исключает потерю данных при сохранении (независимо от того используется компрессия или нет).

Если заменять первичный объект после редактирования, то, возможно, по каким-то параметрам у нас может произойти потеря информации. Поэтому с точки зрения сохранения первичных объектов при более сложных редактированиях, например художественных, лучше добавить дополнительный первичный объект в базу данных, а проблему отождествления решать уже на уровне метаобъектов.

6. РЕЗЕРВНОЕ КОПИРОВАНИЕ И АВТО-ОБНОВЛЕНИЯ

На мировом уровне вопросы резервного копирования решены методологически и технологически во многих аспектах. В данном разделе указаны нюансы применимости этих решений в нашей распределенной модели хранения данных.

Решения должны учитывать следующие особенности. Во-первых, ядром информационной системы в данной модели может служить вовсе не серверная конфигурация компьютера, а также не обязательно наличие источника бесперебойного питания. Во-вторых, возможно отсутствие дополнительной надежной архивной машины (для хранения резервной информации). В-третьих, для увеличения числа машин, являющихся хранителями части информации и входящих в единое информационное поле, одно из требований — минимальная настройка этих машин под данную задачу, т.е. требуется использование наиболее распространенных и встроенных в операционные системы программного обеспечения и интерфейсов программирования приложений (API), например, Microsoft .NET Framework, Java Virtual Machine, Microsoft IIS Server.

Вопросы резервного копирования касаются не только первичных данных и базы метаобъектов, но также и версий программного обеспечения, стилей оформления, структуры данных, различных конфигурационных файлов.

С другой стороны, нам требуется поддержка не только резервного копирования данных, но и публикации новых появившихся первичных объектов или новых версий программного обеспечения.

В плане развития и улучшения фактографической информационной системы мы должны предусмотреть производственную цепочку изменений. Кроме публикации на внешнем сервере (публичный интерфейс), либо на компьютере конечного пользователя нам также требуется возможность тестирования новых версий программного обеспечения. При этом целесообразно использование реальных данных для тестирования. Поэтому требуется несколько серверов (или виртуальных машин), для того чтобы разные данные и разные программные системы не смешивались. Кроме того, необходимо, чтобы данные и системы были как можно более новые, чтобы тестирование проходило в условиях, приближенных к реальности. Для этого должен быть реализован контроль обновления данных и программного обеспечения в имеющихся у нас разных источниках/ресурсах.

При использовании описанных в предыдущих разделах механизмов редактирования первичных объектов и метаобъектов у нас также появляется методологическая возможность не только резервного копирования работающей версии, но и отката системы назад. Для первичных объектов это возможно, так как при их изменении старый объект не удаляется, а все отождествление идет на уровне метаобъектов. Для метаобъектов это возможно при помощи использования описанной реализации концепции распределенного хранения. В определенный момент, когда требуется провести резервное копирование, мы переводим редактируемый контекст в разряд внешних источников. При этом мы все еще можем его дальше редактировать, однако, используя преимущества нашей онтологии и подход, который уже был задан в этой статье, мы не редактируем непосредственно эти данные, они находятся фактически в другом RDF-документе. Вследствие этого мы можем, если нам требуется, совершить откат назад.

Также нас интересует возможность откатов назад и в версиях программного обеспечения. Приемы, которыми этого можно добиться, во многом уже описаны в другой литературе.

7. РЕАЛИЗАЦИЯ

Работа велась в рамках проекта «Электронный фотоархив Сибирского отделения Российской академии наук». В проекте требовалось создать информационную систему для сохранения документов и добавления к ним метаданных. С целью привлечения к проекту дополнительных информационных ресурсов необходима возможность последовательной интеграции данных с установлением распределенных механизмов хранения и доступа.

Подходы, рассмотренные выше, были реализованы в виде модулей, взаимодействие которых определяет информационную систему. Помимо интерфейсов для ввода, редактирования и внутреннего поиска информации, существует и «легкий» публичный интерфейс, доступный по адресу: http://soran1957.ru/

СПИСОК ЛИТЕРАТУРЫ

1. Марчук А.Г. Распределенные электронные архивы, библиотеки и базы данных // Новосибирск, 2004. — 25 с. — (Препр. / ИСИ СО РАН; № 122).
2. Resource Description Framework (RDF) . — http://www.w3.org/RDF/
3. Sesame, open source RDF framework. — http://openRDF.org/
4. Jena — A Semantic Web Framework for Java. — http://jena.sourceforge.net/
5. Berners-Lee Tim, Hendler James, Lassila Ora, The Semantic Web // Scientific American. — 2001. — Vol. 284(5). — P. 34–43.
6. Марчук П.А. Особенности интеграции данных из разных источников // Технологии Microsoft в теории и практики программирования / Конф.-конкурс работ студентов, аспирантов и молодых ученых. Тез. докл. — Новосибирск, 2007 — С. 129–131.
7. The Dublin Core Metadata Initiative (DCMI) . — http://dublincore.org/
8. Web Ontology Language (OWL) . — http://www.w3.org/2004/OWL/
9. SPARQL Query Language For RDF. — http://www.w3.org/TR/rdf-sparql-query/