2003 - Том 6 - Выпуск 3

RDFS как основа среды разработки цифровых библиотек и Web-порталов

Бездушный А.А.*, Бездушный А.Н.**, Нестеренко А.К.**, Серебряков В.А.**, Сысоев Т.М.*

***МФТИ, **ВЦ РАН**

В работе рассматриваются основные цели, понятия и технологии Semantic Web, анализируются перспективы их поддержки. Приводится сопоставление парадигмы Semantic Web с традиционными парадигмами программирования. Описывается место RDF(S) в новой версии системы ИСИР [ISIR], опирающейся на открытые стандарты W3C: Semantic Web, XML технологии, использующей open-source Java решения. Рассматриваются совокупность расширений RDFS, обусловленных требованиями системы.

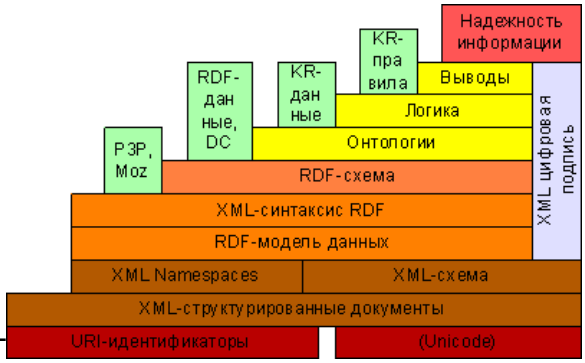
Semantic Web – XML, RDF, RDFS, …

“Semantic Web – это расширение Web, в котором информации придается определенная семантика, позволяя людям и машинам работать вместе“ – примерно такое определение дают своим работам члены W3C Semantic Web Activity [SW]. Целью этого проекта является внедрение в Web таких технологий, которые позволят существенно повысить уровень интеграции информации, обеспечить развитую машинную обработку данных, дадут возможность выдавать более адекватные ответы на поисковые запросы и т.д.

Текущее состояние Web характеризуется слабой структурированностью данных, низким уровнем их взаимосвязи. Распространение XML-технологий дает возможность структурировать информацию, обеспечить синтаксическую интероперабельность приложений. Следующей этап развития Web ориентируется на RDF-технологии, которые вносят в Web возможность семантической интероперабельности. Semantic Web является логическим продолжением развития Web – от гипертекстовых страниц к XML-данным, от XML – к машинной интерпретации данных и объединению разбросанной в Web информации.

Semantic Web базируется на модели данных Resource Description Framework (RDF), позволяющей объединять информацию из различных источников.

Второй базовый компонент Semantic Web – это RDF/XML-синтаксис, который дает возможность представить RDF-данные в XML- форме. Следующий уровень в пирамиде технологий Semantic Web занимает язык RDF Schema (RDFS) – язык описания словарей RDF-терминов (классов и свойств Web-ресурсов). В отличие от XML-схем, которые описывают структуру XML-документов, ограничивают их содержание, RDF-схемы позволяют определять семантику данных, основанных на XML представлении. RDFS служит фундаментом для более богатых языков описания моделей предметных областей (языков описания онтологий), которые позволяют адаптировать к Web системы математической логики и обеспечить семантическую обработку данных.



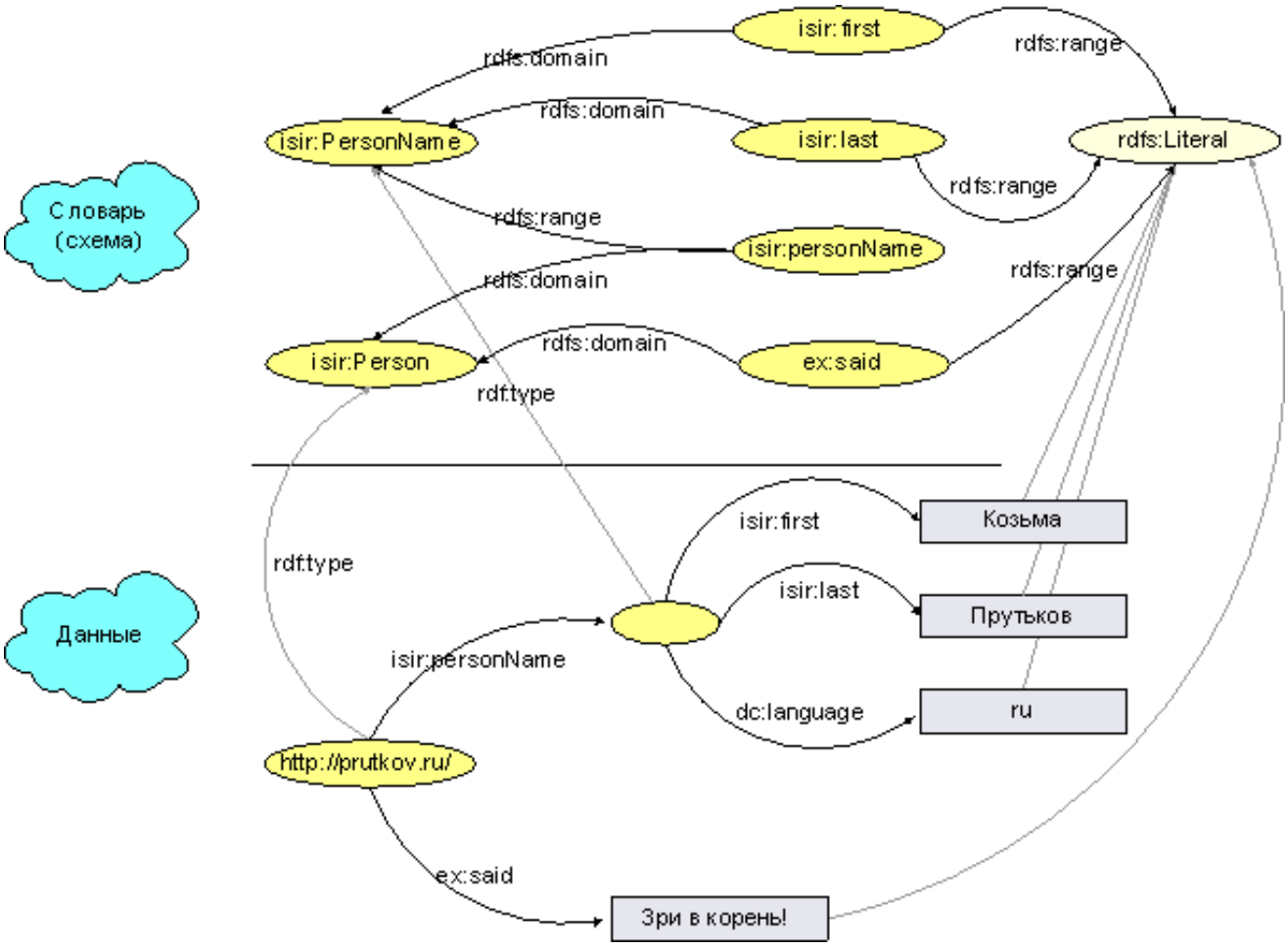
RDF модель данных, составляющая основу методики Semantic Web, является представителем семейства реляционных моделей данных, специфика которой состоит том, что ресурсы и свойства идентифицируются с помощью глобальных идентификаторов - URI. RDF описывает предметную область в терминах ресурсов, свойств ресурсов и значений свойств. RDF-данные можно расценивать как совокупность утверждений – субъект, предикат (свойство) и объект утверждения, представлять в виде направленного графа, образуемый такими утверждениями.

RDF/XML-синтаксис позволяет записать граф в последовательной форме, пригодной для обмена данными. Этот синтаксис достаточно гибок – он допускает различные формы записи одного и того же графа, различные сокращенные формы.

RDFS предоставляет механизм для определения необходимой совокупности типов ресурсов и свойств. RDFS вводит такие понятия, как классы, подклассы, свойства и подсвойства, дает возможность накладывать на них ограничения. RDFS позволяет определить классы ресурсов и свойства как элементы словаря типов данных, и специфицировать, какие свойства, с какими классами могут быть использованы. RDFS выражает эти словари средствами RDF, предоставляя набор предопределенных ресурсов и свойств с обозначенной для них смысловой нагрузкой, которые могут быть использованы для описания новых RDF-словарей.

Таким образом, любое RDFS-описание представляет собой «обычные» RDF-данные – данные о классах и свойствах. RDFS позволяет определить уникальные (идентифицируемые URI) классы ресурсов, представляющие концептуальную модель конкретной предметной области, и уникальные (идентифицируемые URI) свойства, интересующие нас в этой области. Принадлежность ресурса к конкретному классу задается с помощью свойства rdf:type, представляемого в

графе дугой от экземпляра к классу. Описываемые в словаре классы сами являются экземплярами предопределенного класса rdfs:Class, свойства же являются экземплярами класса rdf:Property. RDFS позволяет указать, каким классам присущи заданные свойства, и ресурсы какого класса могут появиться в качестве значения заданного свойства. Эта информация указывается в словаре с помощью свойств rdfs:domain и rdfs:range соответственно. RDFS позволяет связать классы (rdfs:Class) отношениями множественного наследования (rdfs:subClassOf). В RDFS-модели, как и в обычном объектном подходе, классам свойственен полиморфизм. То есть, экземпляр подкласса всегда может сыграть роль экземпляра своего суперкласса, и появиться как субъект или объект свойства, для которого в качестве соответственно range или domain был указан суперкласс. Свойства также могут быть связаны отношениями множественного наследования (rdfs:subPropertyOf). Наследование свойства означает более узкую специализацию этого свойства, уточнение смысла и сужение границ использования. Ниже приведен пример двух RDF-графов, один из которых соответствуют RDF-схеме некоторой предметной области, а второй - конкретным данным.



С введением механизмов определения словарей, деятельность Semantic Web выходит на новый уровень. На данный момент различные организации по стандартизации предлагают стандартные словари для описания ряда предметных областей. Использование таких публичных словарей (или сопоставление с ними) позволяет «незнакомым» приложениям обмениваться информацией друг с другом. В качестве примера таких инициатив стандартизации можно упомянуть инициативу Dublin Core [DC], предоставляющую минимальный набор свойств для

идентификации ресурсов Web, Publishing Requirements for Industry Standard Metadata (PRISM), определяющую словарь метаданных для издательских организаций, Electric Power Research Institute Common Information Model (CIM/XML), указывающую общую семантику для энергетических систем, RDF Site Summary(RSS) для описания каналов новостей Web-порталов и многие другие инициативы.

Язык RDFS предоставляет лишь базовые возможности для описания схем данных предметных областей, но он легко может быть расширен дополнительными примитивами моделирования, более детально и специализировано описывающими нужные аспекты классов и свойств. Механизм расширения внутренне присущ RDFS, поскольку для описания схем используется модель данных RDF, которая позволяет расширить описание любых ресурсов дополнительной информацией. Предопределенный словарь «мета-типов» RDFS также может быть расширен под нужды приложения, благодаря чему появляется возможность добавлять в язык новые примитивы.

Расширяемость позволяет RDFS стать фундаментом для более богатых языков концептуального моделирования – языков описания web-онтологий предметных областей. Цель таких языков – указать дополнительную машинно-интерпретируемую семантику ресурсов, то есть сделать машинное представление данных более похожим на положение вещей в реальном мире. Использование богатых языков концептуального моделирования позволит адаптировать к Web большое количество наработок в области систем инженерии знаний и баз знаний. Привлечение к Web систем логики и искусственного интеллекта составляет вершину «пирамиды Semantic Web», обеспечивая адекватный поиск информации и ее машинную интерпретацию.

Первыми предложениями по описанию онтологий на базе RDFS были DAML-ONT (DARPA Agent Markup Language) [DAML] и European Commission OIL (Ontology Inference Layer) [OIL]. На базе этих двух предложений возникло совместное решение – DAML+OIL [DAML+OIL], которое привело к созданию в рамках инициативы Semantic Web отдельной группы, ответственной за пересмотр этого решения и стандартизацию языка описания Web-онтологий (OWL – Web Ontology Language) [OWL].

Однако ориентированность языков описания онтологий на системы математической логики делает их слишком тяжеловесными для огромного количества приложений, которым достаточно простого языка описания словарей – RDFS. И это правильно, каждая ступень в пирамиде Semantic Web – это ступень, на которой многие приложения могут остановиться, согласно своим собственным требованиям к данным и их использованию.

Сопоставление Semantic Web с другими парадигмами

Система типов RDFS похожа на многие общепринятые системы типов, как в ER-моделировании, объектно-ориентированном программировании и UML, и т.п. Инициатива Semantic Web не ставит перед собой цели создать новую модель данных, напротив, она ориентируется на интеграцию различных моделей данных с

целью получения информации из соответствующих источников. RDFS отличается от этих стандартных систем типов в нескольких существенных аспектах, которые являются следствием глобализации и децентрализации информационной системы, к которой мы приходим, «выходя» в Web из установленных моделью данных рамок. В каком-то смысле RDF(S) есть адаптация этих моделей к Web. Рассмотрим сопоставление примитивов RDFS и модели данных объектно-ориентированного программирования (согласно [RDF Premier]).

Один из архитектурных принципов Web состоит в том, что *кто угодно может расширить описание существующих ресурсов* [TBL-98], то есть «*кто угодно может сказать, что угодно, о чем угодно*». Это означает, что отношение между двумя объектами может храниться *отдельно* от любой другой информации об этих объектах. Это сильно отличается от того, к чему мы привыкли в обычных объектно-ориентированных системах, в которых считается, что информация об объекте хранится *внутри* объекта: определение класса объекта подразумевает указание места хранения его свойств. Такое отличие является следствием децентрализации и адаптации к положению вещей в реальном мире. Например, один человек может определить автомобиль, как нечто, имеющее колеса, вес и размер, но не предвидеть цвет. Это не остановит другого человека от утверждения, что его машина – красная, используя некоторый словарь цветов.

Из этого архитектурного принципа Web следует основное отличие парадигмы RDFS от объектной парадигмы – это ее *свойство-центричность*. Свойства (отношения, предикаты) в RDFS являются объектами первого уровня, как и классы: они идентифицируются URI и определяются независимо от классов, тогда как в объектной и ER парадигмах свойства (атрибуты) указываются в «теле» класса, смысл свойств с одинаковыми названиями в разных классах может быть различен. Впрочем, такой подход уже использовался, например, в X.500, LDAP, где свойства и их характеристики описываются отдельно от класса, а потом «привязываются» к нужным классам. Он оправдывает себя в системах, ориентированных именно на хранение разнообразной слабоструктурированной информации.

Вместо того, чтобы описывать *классы в терминах свойств* (структуры), имеющихся у него, как это делается в объектно-ориентированных системах, RDFS описывает *свойства в терминах классов*, к которым они применимы, указывая *rdfs:domain* (область применения свойства) и *rdfs:range* (область значений свойства). Различие между этими подходами может показаться только синтаксическим, но на самом деле есть существенная разница, которая связана как раз с глобализацией информационной системы при адаптации ее к Web, где «кто угодно может сказать, что угодно, о чем угодно». Например, если кем-то определен класс *ex:Book* со свойством *ex:author*, принимающим значения типа *ex:Person*, то это не запрещает другим разработчикам придать классу *ex:Book* дополнительное свойство *my:publisher*, достаточно лишь указать этот класс в *rdfs:domain* нового свойства *my:publisher*. Это не требует переопределения класса, причем создатели класса могут быть в неведении данного факта. В то же время в ООП потребовалось бы переопределить и перекомпилировать класс.

Кроме того, RDFS вообще не требует, чтобы у свойства была задана область применения – свойство *без domain* может быть использовано для описания любого

ресурса, независимо от его класса. Определение свойства без указания области применения позволяет использовать его в будущем в ситуациях, которые не могли быть предвидены в момент разработки схемы. Именно так поступает Dublin Core, предоставляя словарь стандартных свойств, пригодных для описания любого Web-ресурса, для которого они окажутся полезными.

Другое важное отличие в семантике RDFS-описаний – это то, что они носят *описательный*, а не *«предписывающий»* характер, то есть, они могут использоваться не для того, чтобы наложить ограничения на применение свойств, а просто чтобы предоставить дополнительную информацию приложению, обрабатывающему эти данные. Если ОО язык программирования объявляет класс *Book* с атрибутом *author* типа *Person*, это обычно интерпретируется как набор ограничений (условий применения). ОО язык не позволит создать экземпляр класса *Book* без атрибута *author* или указать в качестве значения *author* объект, не являющийся экземпляром *Person.* Наконец, он не позволит создать экземпляр *Book* с каким-то другим атрибутом.

RDF Schema, напротив, предоставляет информацию о схеме как дополнительное описание ресурсов, но не указывает, как это описание должны *использоваться* приложениями. Приложение вольно по своему усмотрению считать RDF-данные соответствующими схеме или нет, если в описании отсутствует некоторое свойство, требуемое схемой, либо присутствуют свойства, не указанные в схеме. Одно приложение может интерпретировать RDFS-описания как шаблон для генерации данных, и проверять соответствие данных областям значений свойств, то есть интерпретировать описания схемы так же, как они интерпретируются в ОО языке программирования. Другое приложение может интерпретировать RDFS-описания как дополнительную информацию о данных, которые оно получает. Например, если оно получит RDF-данные с указанием свойства *ex:author*, содержащим значение без указания типа, то может заключить на основе описания схемы, что это значение является *ex:Person*. Третье приложение может получить данные, в которых свойство *ex:author* содержит ресурс типа *ex:Student*, и использовать информацию схемы как базис для предупреждения, что данные могут содержать ошибку. Хотя, возможно, где-то существует RDFS-описание, решающее эту проблему, например, указывающее, что *ex:Student* подкласс *ex:Person*.

Итак, RDFS утверждения всегда описательны. Они могут, конечно, интерпретироваться как «предписывающие» (ограничения), но только если приложение желает их так интерпретировать. Все, что делает RDFS-описание – это предоставляет приложениям дополнительную информацию «для размышления».

Интеграция в Semantic Web систем баз знаний, математической логики и инженерии знаний в состоянии принести двойную прибыль. Во-первых, для механизмов Web (таких как поиск информации) становится доступным большое количество баз знаний, созданных в области медицины, биологической химии и пр. С другой стороны, появляется возможность адаптации к Web самих технологий, наработанных в области математической логики и инженерии знаний, что позволит поисковым системам и программным агентам самостоятельно

анализировать предоставленную информацию.

Однако прямой перенос этих технологий невозможен в силу глобальности и децентрализации, которую мы наблюдаем в Web и не наблюдаем в имеющихся системах инженерии знаний. Многие решения (например, [KIF]) в силу своей концептуальной и физической централизации требуют глобальной целостности ссылок, исходят из предположения об «ограниченности мира». Это же было и с гипертекстовыми системами 70-90 годов, до появления Web. Обобщение KIF для Web должно быть во многом аналогично тому, как были обобщены первые гипертекстовые системы – следует «заменить» локальные идентификаторы на URI и убрать требование глобальной целостности. Кроме того, адаптированные к Semantic Web системы логики должны быть хорошо расширяемы и приспособлены к различными типами «противоречивости» данных.

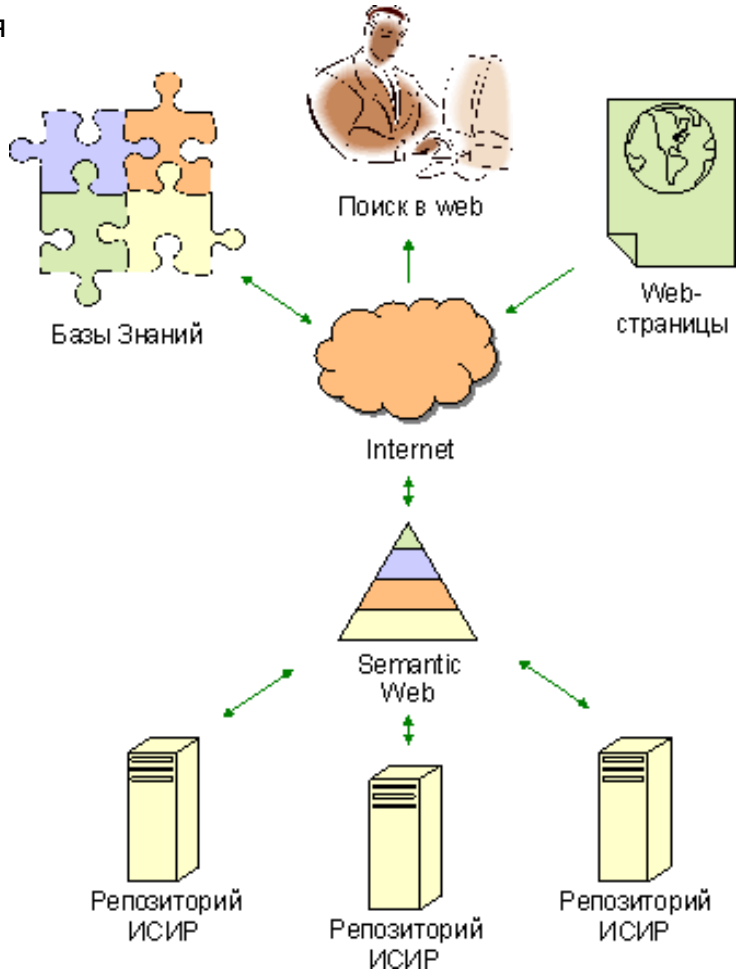
Semantic Web – это то, что мы получим, применив ту же глобализацию к представлению знаний, какая изначально была применена к гипертексту. Достаточно убрать централизованные понятия абсолютной истины, абсолютного знания, полной доказуемости, и посмотреть, что мы можем сделать с ограниченным знанием.

Сопоставление решений ИСИР с задачами Semantic Web

ИСИР-технологии [ISIR] ориентированы на формирование единой информационной среды из разнородных и распределенных источников информации, называемых репозиториями, содержащих ресурсы в реляционных и объектных базах данных, LDAP-каталогах, XML и RDF хранилищах, Z39.50-системах и т.п. ИСИР предоставляет ряд служб по поддержке репозиториев, например, репликацию и обмен данными, индексирование и поиск ресурсов, технологию построения Web-порталов для доступа к данным и манипулирования ими. В плане интеграции разнородных информационных источников ИСИР пытается в меру своих возможностей двигаться в направлении целей, намеченных Semantic Web, используя его технологии. Semantic Web предлагает стандартный механизм и унифицированную модель данных для интеграции информационных источников Web, и ожидает от них поддержки этого механизма. ИСИР предоставляет некоторое техническое решение для построения таких информационных источников, для организации их взаимодействия, как между собой, так и с другими независимыми источниками. ИСИР стремится к реализации идей Semantic Web в своей более узкой и более специализированной области – объектно-ориентированные информационные порталы и цифровые библиотеки.

ИСИР-решения могут работать в рамках Semantic Web. С одной стороны, ИСИР-репозитории служат источниками RDF-знаний для сторонних программных агентов Semantic Web, таких как поисковые системы. С другой стороны, Semantic Web предоставляет ИСИР-репозиториям возможность «добывать» информацию из сторонних источников - доступных в Web тезаурусов и классификаторов, баз знаний и пр. систем, с которыми может быть связана информация в информационном портале или цифровой библиотеке. Использование RDF для обмена данными позволяет обмениваться информацией как между репозиториями ИСИР, так и со сторонними заинтересованными системами.

Основой Java-архитектуры ИСИР является объектно-ориентированный подход к представлению данных. Для каждого типа хранилища поддерживается механизм отображения объектной модели данных во внутреннюю модель (реляционную, LDAP...). ИСИР-репозиторий не универсален – он не может, и не должен хранить, «что угодно». Каждый репозиторий способен хранить данные, соответствующие жесткой замкнутой объектной схеме, описывающей допустимые классы, их свойства, которой сопоставлена система хранимых Java-классов и, например, система таблиц в реляционной БД. Такая объектная схема соответствует традиционной парадигме объектно-ориентированного программирования и объектных баз данных, то есть исходит из понятия «самодостаточности» схемы. Это естественно, так как пока мы находимся в контексте одного репозитория, нет никакой необходимости в «децентрализации» схемы, более того, как правило, модель данных расположенного под объектным уровнем хранилища требует замкнутости схем (реляционная, объектная БД, LDAP...).



Когда же мы выходим на Web-уровень и задаемся задачей интеграции разнородных репозиториев в Web, мы сталкиваемся с «глобализацией» и «децентрализацией» информации, и логично воспользоваться в этой области парадигмой Semantic Web и языком RDFS. Элементы схемы репозитория становятся глобально-идентифицируемыми (с помощью URI), схемы разных репозиториев перестают быть независимыми. Например, может оказаться так, что некоторые репозитории хранят в себе объекты одного и того же класса, но разные наборы свойств, в соответствии с теми аспектами этого класса, которые им нужны в их специфической предметной области. Может выясниться, что репозитории хранят схожие (но не одинаковые) классы, и при их интеграции часть информации из одного репозитория может использоваться противоположными, например, для пополнения имеющейся у них информации или для установления требуемых взаимосвязей между ресурсами.

Возникает естественное желание как-то интегрировать эти две аспекта в рамках одной схемы данных. С этой целью в ИСИР сформировано *расширение* языка RDFS,

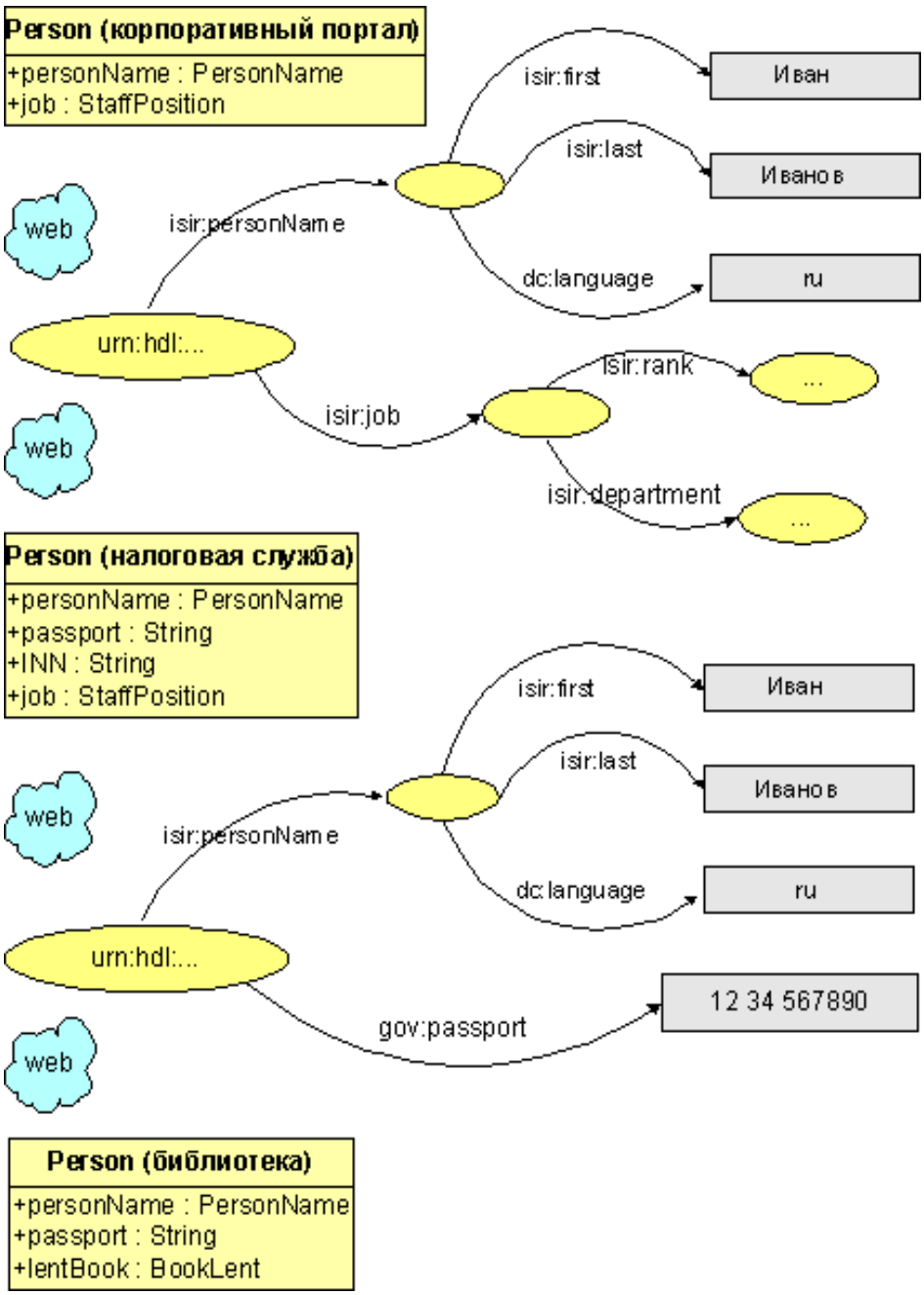
пригодное для описания локального и глобального аспектов объектной схемы данных. Мы используем понятия RDFS-классов и свойств, идентифицируемых URI, понятия наследования классов и свойств, *rdfs:range* свойства, но добавляем к этим примитивам моделирования возможность четко описать класс в терминах его атрибутов, как это делается в ОО языках программирования. С этой целью введено специальное свойство *schema:property* - отношение, связывающее класс с его свойствами.

Каждый репозиторий рассматривает свою схему ресурсов как замкнутое описание, по которому можно получить, например, соответствующее ODL-описание или сгенерировать систему bean-подобных Java-классов. Он интерпретирует *rdfs:range* как ограничение на тип значений свойства, которому должны удовлетворять хранимые данные, *schema:property* - как ограничение, четко задающее все присутствующие в классе атрибуты (которым будут соответствовать get/set методы в java-классах, поля в реляционной БД и т.п.). Репозиторий интерпретирует свою схему ресурсов в рамках традиционной объектной парадигмы, как требуется ИСИР при хранении данных. Это не противоречит идеям Semantic Web, поскольку приложения могут интерпретировать RDFS-описания по собственному усмотрению.

С другой стороны, при обмене данными между репозиториями и интеграции их информации, RDFS-описание объектной схемы каждого репозитория расценивается с позиции децентрализации, распределенности информации в Web. При загрузке сторонних RDF-данных, из них отбирается только та информация, которую репозиторий способен сопровождать согласно своей замкнутой схеме и своим ограничениям. При формировании результатов поисковых запросов поисковый сервис может возвращать пользователям описания ресурсов, интегрирующие свойства разных репозиториев, а, возможно, к тому же отфильтрованные в соответствии с предпочитаемой пользователем схемой ресурсов.

Рассмотрим пример совместной работы трех информационных источников – корпоративного портала некоторой организации, базы данных налоговой службы и портала библиотеки. Схема объектных данных каждого из этих ИСИР-репозиториев содержит класс isir:Person, представляющий нужную в репозитории информацию о человеке. Пространство имен ‘isir’ соответствует базовой схеме метаданных ИСИР, находящей применение в большинстве приложений. В той же базовой схеме определено свойство isir:personName, указывающее ФИО человека на нужном языке.

Однако, каждому репозиторию нужна более специфичная информация о данной личности – корпоративный портал содержит данные о должности человека и контактной информации, которая может быть полезна пользователю портала. Налоговая служба заинтересована, помимо этого, в паспортных данных этого лица, его ИНН и сведениях о доходах. Библиотечная система хранит лишь паспортные данные и информацию о взятых книгах. Соответственно, каждый репозиторий специализирует класс isir:Person под свои нужды, и связывает с ним только те свойства, которые представляют интерес в данной предметной области.



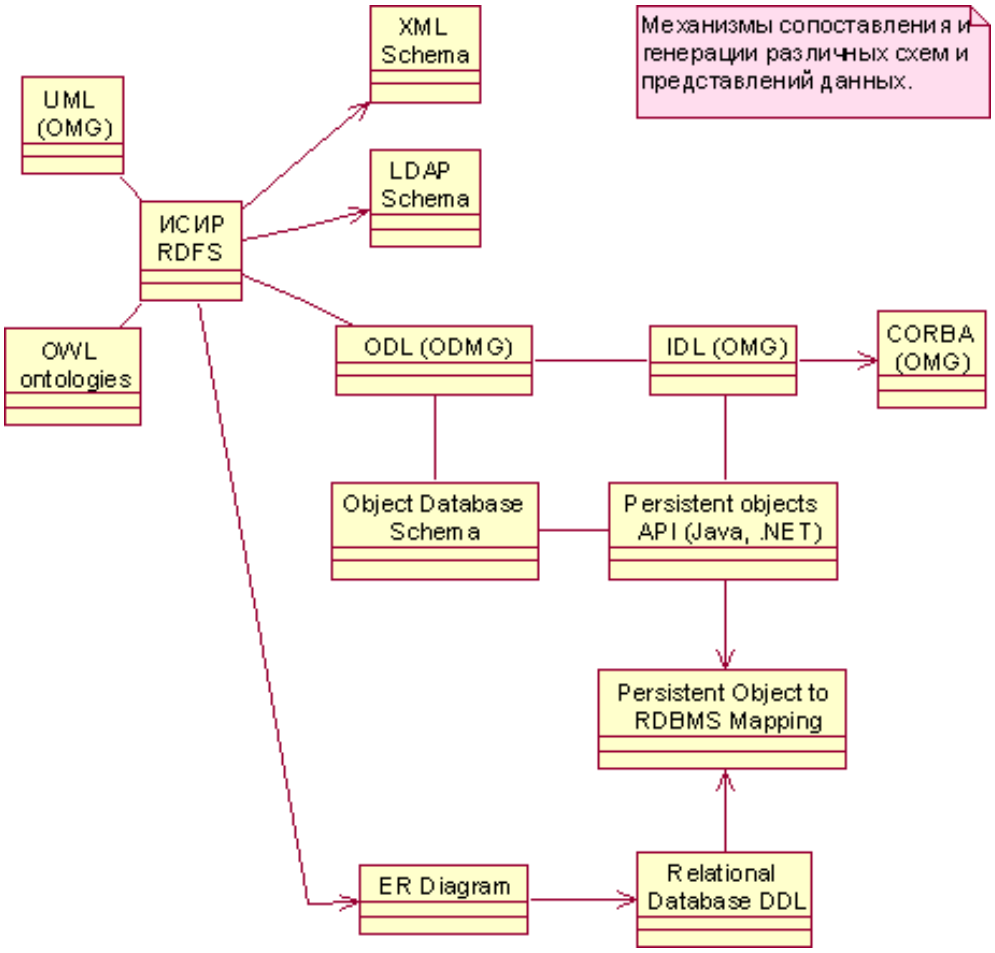
Таким образом, структура java-класса Person различна для разных репозиториев (на рисунке эти различные «взгляды» на isir:Person изображены в стиле UML). Тем не менее, часть информации в этих источниках пересекается, и один репозиторий может реплицировать нужную информацию из другого. Так, налоговая служба может позаимствовать информацию о месте работы непосредственно из корпоративного портала организации-работодателя, а библиотечная система – паспортные данные из налоговой. Репозитории обмениваются информацией в RDF/XML-представлении, что обеспечивает интероперабельность не только между собой, но и со сторонними системами Semantic Web.

Роль RDFS в архитектуре ИСИР

ИСИР использует RDFS как фундамент для описания объектных схем за простую объектную модель, удобство расширения новыми примитивами и адаптированность к Web, а следовательно и к обмену RDF и XML данными.

Онтологические языки, в частности OWL, слишком сложны в силу своей ориентации на системы логики, и не имеет смысла использовать их лишь для описания объектной схемы репозитория и простых ограничений на нее. Но многие онтологии могут быть легко адаптированы под требования ИСИР путем их упрощения, что позволяет создавать цифровые библиотеки, хранящие информацию в соответствии с системой классов и свойств данной онтологии.

Приоритетное положение RDFS в ИСИР не мешает изначально описать объектную схему репозитория на ODL [ODMG], смоделировать на UML [UML], либо получить по существующей системе Java-классов. ИСИР предоставляет некоторые механизмы по отображению таких описаний объектной схемы друг в друга и их генерации.

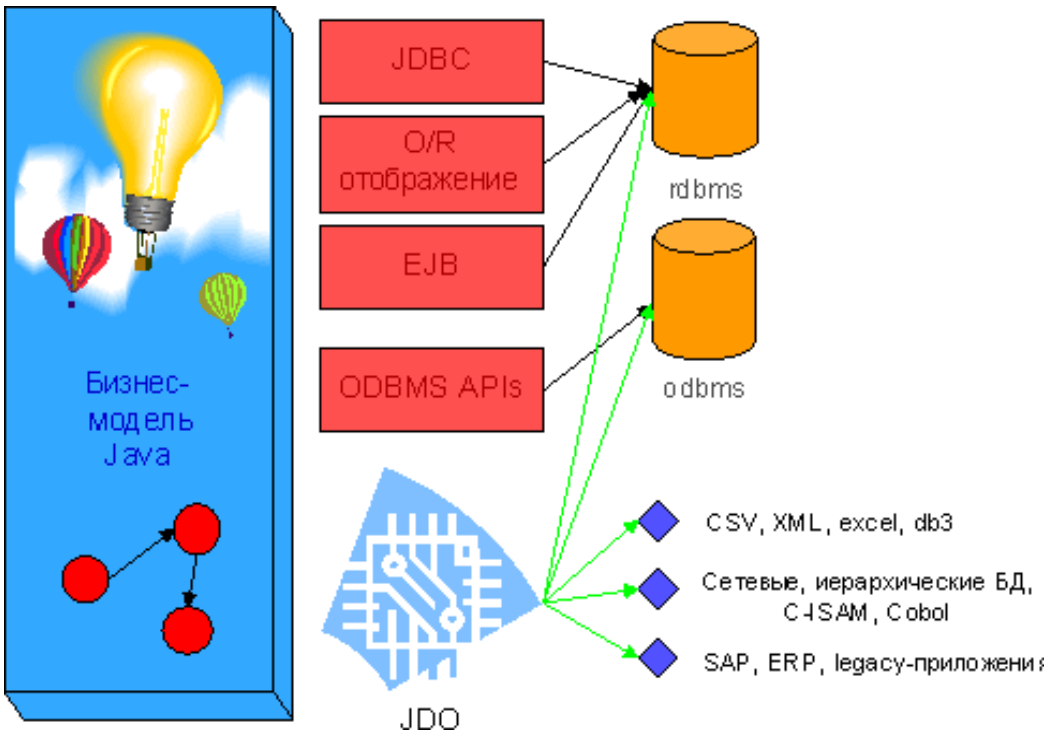


В частности, генератор Java-классов позволяет получить по ИСИР RDFS-описанию исходный код bean-подобных классов, которые будут использоваться репозиторным сервисом и прикладным кодом портала. В эти классы вручную может быть заложена любая бизнес-логика, заменяющая или дополняющая исходное поведение. При изменении схемы (например, добавлении свойств) будет произведена инкрементная перегенерация классов – внесенные в код изменения будут сохранены. Таким образом, не ограничивая функциональных возможностей системы, RDFS позволяет автоматизировать большинство операций.

Генератор реляционной БД позволяет получить по ИСИР RDFS-описанию SQL DDL-скрипт для создания таблиц, в которых будут храниться данные, и описание объектно-реляционного отображения. Процесс генерации является

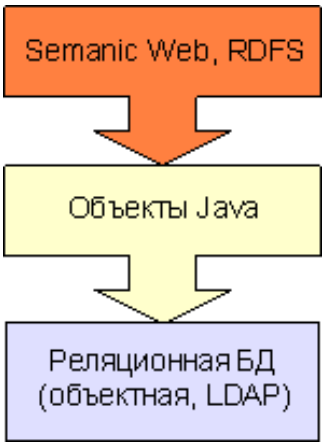
настраиваемым, и дизайнер может явно указать, какие примитивы реляционной БД необходимо использовать для заданных примитивов объектной схемы в том или ином случае, например, какое решение применить для поддержки наследования классов. Настройки генератора указываются в RDF-формате вместе с объектной схемой, благодаря расширяемости RDFS. Помимо разработки новых хранилищ, ИСИР-технологии могут быть применены к унаследованной базе данных, таким образом открывая к ней доступ ведущим Web-технологиям и интегрируя ее в Semantic Web.

Основой Java-архитектуры ИСИР является представление хранимых в репозитории данных в виде java-объектов. Репозиторный сервис ИСИР обеспечивает прозрачное отображение «хранимых» java-объектов, соответствующих объектной схеме репозитория, в данные хранилища конкретного вида, например, в записи таблиц реляционной БД. Основным внешним интерфейсом репозиторного сервиса является подмножество ODMG-интерфейса, позволяющее работать с хранимыми объектами, для осуществлять поисковые запросы и управлять транзакциями. Репозиторный сервис фактически превращает низлежащее хранилище данных (реляционную БД, LDAP..) в объектную базу данных. В дополнение к этому репозиторный сервис предоставляет различные услуги по управлению хранимыми объектами, такие как разграничение прав доступа, аудит, поддержка версий и пр. Та или иная реализация репозиторного сервиса выбирается в соответствие с типом хранилища. На данный момент такой механизм имеется для реляционных БД, на основе JNDI реализуется механизм для LDAP каталогов. Sun Microsystems занимается стандартизацией механизмов прозрачного хранения java-объектов в различных базах данных и иных системах - Java Data Objects Specification. На рынке уже доступно большое количество полных или частичных реализаций этой спецификации, любая из которых может подключена в Репозиторный Сервис ИСИР.



ИСИР-технологии не поддерживают средства для хранения произвольных RDF-

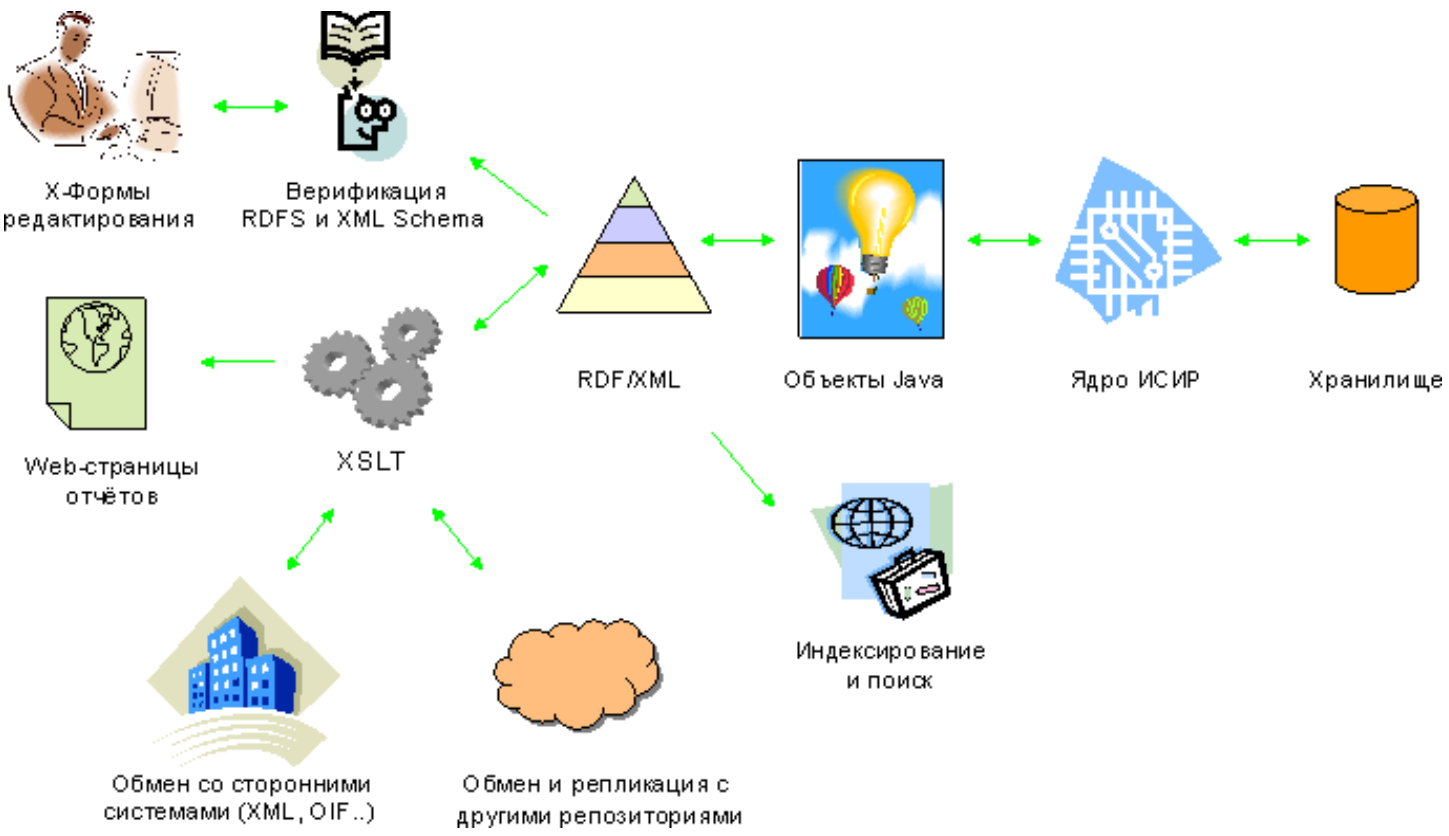
данных. Каждый репозиторий жестко ограничен своей схемой предметной области, которая выражается не только в структуре хранимых Java-классов, но и в схеме скрытого под ним хранилища, например в структуре таблиц реляционной БД. Существуют продукты, нацеленные именно на хранение произвольного RDF (см. [SW RDBMS], [KAON], [RDFSUITE]...), которые хранят в БД RDF-тройки (утверждения), и позволяют делать RDF-специфичные запросы к этому множеству троек. Эти системы следуют абсолютно иным целям, чем ИСИР, мало эффективны в работе с большими объемами данных, более приближены к системам баз знаний, нежели к цифровым библиотекам. При необходимости данные ИСИР-репозитория посредством их RDF-представления могут быть перегружены в подобную систему.



Итак, ИСИР сопоставляет модель данных хранилища и модель данных Semantic Web через прослойку представления объектов языка Java, таким образом открывая Semantic Web доступ к хранимой информации. Когда мы находимся на web-уровне, мы исходим из принципов глобализации и децентрализации Semantic Web и используем RDFS для отражения объектной схемы интересующей нас предметной области. Когда же мы ограничиваем свою область видимости конкретным репозиториям, мы используем замкнутую объектную схему, соответствующую традиционной объектной парадигме, которая может быть выражена в терминах ODL, UML, или расширенного RDFS ИСИР. Этой схеме соответствует набор java-классов, отражающих предметную область. И наконец, Репозиторный Сервис ИСИР позволяет организовать отображение объектов этих классов в данные конкретного используемого хранилища, в соответствии со специфицированным механизмом отображения объектной модели данных в модель данных хранилища.

Архитектура ИСИР повсеместно применяет XML-технологии (XSLT, SOAP, XForms и пр.) и использует XML для представления хранимых объектных данных в сериализованном виде. Для XML-сериализации объектов используется сокращенный RDF/XML-синтаксис, который, с одной стороны, позволяет использовать XML Schema и воспользоваться всеми удобствами XML-технологий. С другой стороны, при обмене такой информацией для интеграции данных

различных репозиториев ИСИР, мы пользуемся всеми преимуществами Semantic Web. Структура такого (RDF/XML) файла четко соответствует объектной схеме репозитория ИСИР, объекты которого представлены в XML. Таким образом, RDF-файл несет в себе семантику сериализованных данных, а не только данные и их структуру. Получатель сможет правильно сопоставить сериализованные данные с собственной системой классов и свойств, и адекватно их обработать. Это как раз то преимущество, которое несет Semantic Web – семантическая интероперабельность.



Примитивы моделирования ИСИР RDFS

ИСИР расширяет язык RDFS возможностью четкого указания допустимых атрибутов класса (в соответствие с традиционной парадигмой ООП), возможностью выделения линии одиночного наследования в наследовании классов (понятие «абстрактных классов») и некоторыми простыми ограничениями на множество значений свойства. С этой целью используются следующие метаклассы:

- schema:Class и schema:Property – подклассы соответственно rdfs:Class и rdf:Property. С этими метаклассами связаны дополнительные свойства.
- schema:AbstractClass – подкласс schema:Class, позволяющий выделить линию одиночного наследования во множественном наследовании классов rdfs:subClassOf. Понятие «абстрактного класса» используется при сопоставлении схемы с теми языками программирования, в которых запрещено множественное наследование, в частности Java. При генерации Java-кода, каждому schema:Class соответствует класс Java, а каждому schema:AbstractClass – Java-интерфейс. Соответственно, генератор не

допускает наследования schema:Class от более чем одного не-абстрактного класса.

- schema:property – свойство класса schema:Class, сопоставляющее со schema:Class набор допустимых атрибутов schema:Property, таким образом определяя структуру хранимых Java-классов репозитория (согласно традиционной парадигме ООП).
- schema:inverseOf – свойство schema:Property, задающее «обратное отношение» к данному свойству. Например, “isir:job” имеет обратное отношение “isir:hired”, а “isir:author” -обратное отношение “isir:authorOf”. Аналогичное свойство имеется в языках описания онтологий – DAML+OIL, OWL.
- schema:minCardinality, schema:maxCardinality – минимальная и максимальная мощность свойства, то есть ограничения на минимальное и максимальное количество значений этого свойства. В DAML+OIL и OWL эти ограничения указываются в виде локальных ограничений на свойство в классе.

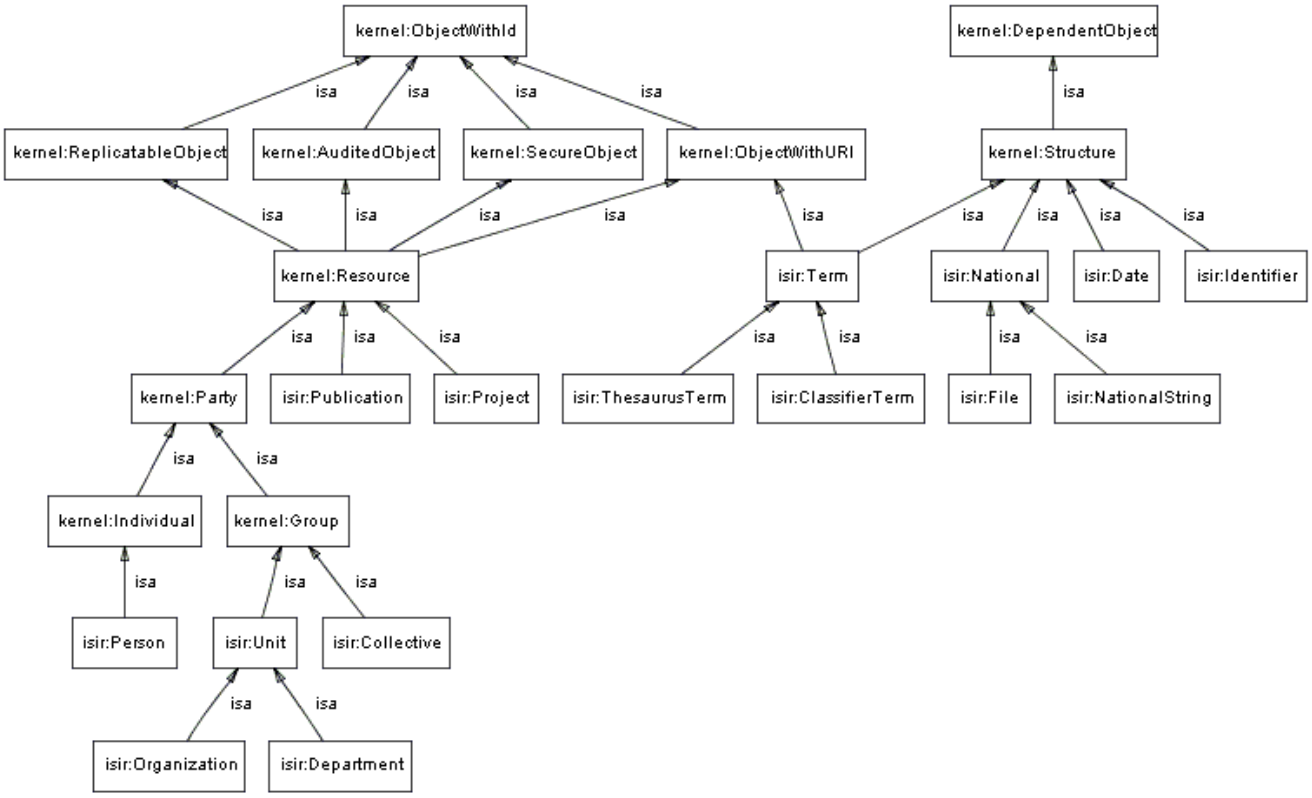
  Ядро ИСИР предоставляет различные встроенные услуги по управлению хранимыми объектами, такие как автоматическая проверка прав доступа, аудит и пр. Для того, чтобы к хранимому классу была подключена некоторая услуга, он должен расширить соответствующий предопределенный «абстрактный класс». Так, для подключения системы безопасности и возможности конфигурации персональных прав доступа к объектам класса, достаточно сделать класс наследником kernel:SecureObject. Благодаря использованию множественного наследования, к классу можно подключить несколько услуг одновременно (например, «зависимый объект» и персональные права доступа). К RDFS-классам представляющим услуги ядра относятся следующие:

  - kernel:ObjectWithId – такой объект имеет уникальный в пределах хранилища числовой идентификатор - id. Большинство сервисов ядра, такие как security, audit и пр., нуждаются в том, чтобы объект обладал таким id.
  - kernel:SecureObject – такой объект обладает персональными правами доступа – Access Control List, указывающий, каким пользователям какие привилегии над этим объектом даны. Достаточно наследовать свой класс от SecureObject – и он попадет под политику безопасности ядра.
  - kernel:AuditedObject – наследование от этого «абстрактного класса» подключает автоматический аудит изменений объекта.
  - kernel:DependentObject – экземпляры подклассов этого «абстрактного класса» расцениваются ядром как зависимые объекты. Такой объект имеет неявное обязательное поле scope – «контекст», связывающее его с объемлющим хранимым объектом. Зависимый объект не может существовать без контекста. При создании зависимый объект должен быть проассоциирован с допустимым объектом-контекстом, иначе он не получит статус хранимого. При удалении объекта-контекста автоматически удаляется и зависимый объект. Ассоциация происходит при

присвоении зависимого объекта некоторому свойству объекта-контекста, принимающему только одно значение, либо при добавлении его в коллекцию значений многозначного свойства объекта-контекста. Свойства, проводящие ассоциацию зависимого и объемлющего объектов, называются ИСИР-атрибутами (schema:Attribute). На тип объекта-контекста не накладывается ограничений – он и сам может быть зависимым объектом от некоторого третьего хранимого объекта. Для чтения, модификации, удаления зависимого объекта необходимо иметь аналогичное право для объекта-контекста (чтение контекста для чтения зависимого, модификация контекста для модификации зависимого, модификация или удаление контекста для удаления зависимого). При изменении зависимого объекта меняется и дата модификации объекта-контекста.

- kernel:ObjectWithURI – такой объект может обладать URI – строкой, однозначно идентифицирующей его в контексте данного хранилища, а иногда (в зависимости от схемы URI) и в глобальном аспекте. URI используется в RDF-сериализации для того, чтобы ссылаться на этот объект. Lookup Plugin в Persistence Service позволяет быстро найти объект по URI в обход обработки OQL-запроса.

На следующей схеме показана взаимосвязь перечисленных системных классов, обеспечивающих поддержку технологий ИСИР, и прикладных классов системы ИСИР РАН, реализованной средствами этих технологий.



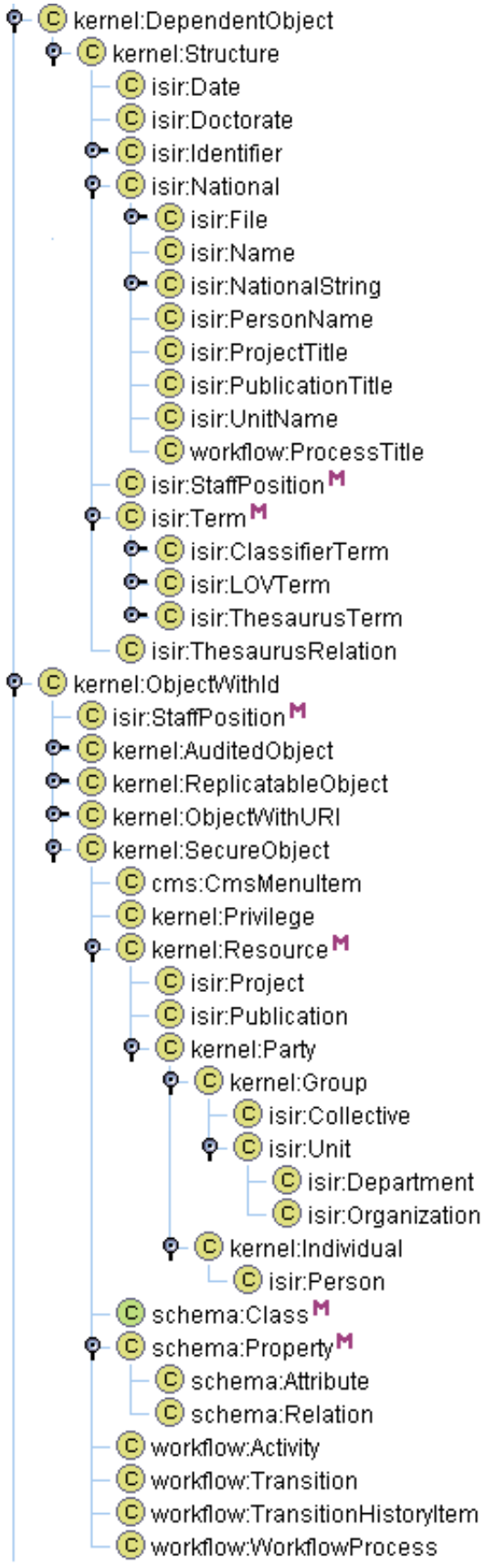
Помимо указанных абстрактных классов, вводится два типичных базовых класса, удобных для использования в большинстве информационных

систем – это «ИСИР-ресурс» и «структура».

- kernel:Resource - логическая единица хранения, условная единица информации, наделенная «самостоятельностью», «индивидуальностью» в том смысле, что она может существовать вне зависимости от других информационных объектов, представлять первичный интерес пользователей системы. Каждый ресурс относится к некоторому типу ресурсов, который определяет некоторую совокупность сведений, внутренне присущих ресурсам этого типа, но не рассматривающих взаимосвязи с другими ресурсами. Ресурс обладает глобально уникальным идентификатором, относительно которого известен и поддерживается механизм разрешения идентификатора в протокол и адрес доступа к ресурсу, например, Handle System.
- kernel:Structure - структурное значение («зависимый объект»), существующее только в контексте логических единиц хранения (ресурсов). Простой подкласс kernel:DependentObject.

При описании RDFS-подсхемы прикладных классов для системы ИСИР РАН был сформирован набор базовых прикладных классов и свойств, характерных для многих информационных систем. На рисунке показан фрагмент иерархии классов ИСИР РАН, визуализируемый редактором онтологий Protege (http://protege.stanford.edu/). Например, класс core:File предназначен для отражения длинной текстовой или бинарной информации. Конкретный способ хранения этой информации зависит от конфигурации системы и указанного «режима». Файл может представлять собой как web-ссылку (URL), по которой может быть получено его содержимое, так и CLOB или BLOB в реляционной БД (когда хранилищем является RDBMS), либо файл локальной файловой системы. С файлом может быть связана различная метаинформация, такая как размер, MIME-тип и пр. Описанная функциональность класса File обеспечивается за счет скрытой в нем прикладной логики. «Файловая» информация широко используется в конкретных прикладных системах

(документы). Имеются компоненты, которые обеспечивают загрузку файлов на сервер, организацию работы с zip-архивами файлов, например, для хранения html-документа с картинками. Эти системы классов, возможно, их функциональная поддержка из ИСИР РАН могут быть использованы при реализации системах для других предметных областей.



Литература

[SW] Semantic Web Activity. // http://www.w3.org/2001/sw

[RDF Primer] RDF Primer. W3C Working Draft. // http://www.w3.org/TR/rdf-primer

[RDF/XML] RDF/XML Syntax Specification (Revised). W3C Working Draft. //

http://www.w3.org/TR/rdf-syntax-grammar/
[RDFS] Resource Description Framework (RDF) Schema Specification // http://www.w3.org/TR/2000/CR-rdf-schema-20000327
[TBL-98] Tim Berners-Lee. What the Semantic Web can represent., 1998 // http://www.w3.org/DesignIssues/RDFnot.html
[OWL] OWL Web Ontology Language 1.0 Reference. W3C Working Draft. // http://www.w3.org/TR/owl-ref/
[DAML] DAML Language. // http://www.daml.org/about.html
[DAML+OIL] DAML+OIL (March 2001) Reference Description. // http://www.daml.org/2001/03/daml+oil-index.html
[OIL] Ontology Inference Layer. // http://www.ontoknowledge.com/oil
[DC] Dublin Core Activity. // http://dublincore.org/
[RSS] Beged-Dov G., Brickley D., Dornfest R., Davis I., Dodds L., Eisenzopf J., Galbraith D., RDF Site Summary (RSS) 1.0
[SW RDBMS] Dave Beckett, Jan Grant. Semantic Web Scalability and Storage: Mapping Semantic Web Data with RDBMSes. // http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report
[KAON] The Karlsruhe Ontology and Semantic Web Tool Suite (KAON). // http://kaon.semanticWeb.org/
[RDFSUITE] RDFSutie Project. // http://139.91.183.30:9090/RDF/
[OMG] Object Management Group. // http://www.omg.org/
[ODMG] Object Database Management Group. // http://www.odmg.org/
[KIF] Knowledge Interchange Format, Genesereth M. // draft proposed American National Standard NCITS.T2/98-004. http://logic.stanford.edu/kif/dpans.html
[UML] OMG UML Resource Page. // http://www.omg.org/uml/
[ISIR] Бездушный А.Н., Жижченко А.Б., Кулагин М.В., Серебряков В.А., Интегрированная система информационных ресурсов РАН и технология разработки цифровых библиотек. // Программирование V 26, N 4, 2000, pp. 177-185.

Об авторах

Бездушный Алексей Анатольевич - студент Московского Физико-Технического института. Сфера деятельности: программирование, Java, распределенные системы, RDFS, Semantic Web, онтологии, Web-порталы, базы данных, цифровые библиотеки.
Тел. (095)1355471
E-mail: alix@7ka.mipt.ru

Бездушный Анатолий Николаевич - кандидат физико-математических наук, с.н.с. Вычислительного Центра имени А. А. Дородницына РАН. Сфера деятельности: системное программирование, параллельные вычисления, сети, базы данных,распределенные системы, информационно-поисковые технологии, цифровые библиотеки, Web-порталы.
Тел. (095)1355471(*4216)
E-mail: bezdushn@ccas.ru

Нестеренко Алексей Константинович - аспирант Вычислительного Центра имени А. А. Дородницына РАН. Сфера деятельности: программирование, Java, управление потоками работ, базы данных, Web-порталы, цифровые библиотеки.
Тел. (095)1355471
E-mail: alexn@ccas.ru

Серебряков Владимир Алексеевич - с.н.с., зав.отделом Вычислительного Центра имени А. А. Дородницына РАН. Сфера деятельности: системное программирование, параллельные вычисления, сети, базы данных, распределенные системы, информационно-поисковые технологии, цифровые библиотеки, Web-порталы.
Тел. (095)1355471(*4220)
E-mail: serebr@ccas.ru

Сысоев Тимофей Михайлович - аспирант Вычислительного Центра имени А. А. Дородницына РАН. Сфера деятельности: программирование, Java, распределенные системы, атрибутно-полнотекстовый и распределенный поиск, Web-порталы, базы данных, цифровые библиотеки.
Тел. (095)1355471
E-mail: tim@7ka.mipt.ru