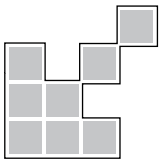

МОДЕЛИРОВАНИЕ В ЗАДАЧАХ ПРОЕКТИРОВАНИЯ И УПРАВЛЕНИЯ

УДК 004.2

Г.П. ТОКМАКОВ

ОНТОЛОГИИ И ИХ ПРИМЕНЕНИЕ ДЛЯ ИНТЕГРАЦИИ ИНФОРМАЦИОННЫХ РЕСУРСОВ

Токмаков Геннадий Петрович, *доктор технических наук, окончил радиотехнический факультет Ульяновского политехнического института, аспирантуру Московского института электронного машиностроения, докторантуру Ульяновского государственного технического университета. Главный научный сотрудник ФНПЦ ОАО «НПО «Марс». Профессор кафедры «Вычислительная техника» Ульяновского государственного технического университета. Имеет монографию, учебные пособия, статьи и изобретения в области разработки моделей данных и систем искусственного интеллекта. [e-mail: mars@mv.ru.].*

Аннотация

В статье рассматриваются вопросы использования средств описания семантики данных с целью интеграции информационных ресурсов, содержащихся в различных базах данных корпоративных автоматизированных систем управления. В первой части статьи проведен обзор современных средств описания синтаксиса и семантики данных. Во второй части статьи предлагается формализация предметной области в виде специализированной онтологии, направленной на описание семантики данных, размещенных в базах данных корпоративной системы управления.

Ключевые слова: единое информационное пространство, семантика, онтология, интеграция данных.

Abstract

The article deals with issues of use of data-semantics description facilities in order to integrate information resources contained in different databases of corporate computer-aided control systems. The first part of the article gives an overview of state-of-the-art description facilities for data semantics and syntax. The second one offers formalization of subject field in the form of special-purpose ontology which is oriented to the description of data semantics, located in databases of the corporate control system.

Key words: common information space, semantics, ontology, data integration.

Введение

В связи с усложнением современного производства предприятия и организации объединяются в более крупные структуры, такие как корпорации. Это объединение сопровождается образованием корпоративной автоматизированной системы (АС), представляющей собой совокупность АС отдельных предприятий и организаций, созданных в разное время, разными коллективами разработчиков и на разной аппаратно-программной платформе.

В сформированной таким образом корпоративной системе каждая входящая в нее АС имеет собственные [1]:

- информационные ресурсы (ИР);
- правила именования и формирования ИР;
- адресные пространства.

В результате, на уровне корпорации складывается технологическая среда реализации про-

цессов управления, для которой характерно:

- отсутствие единого адресования относительно всех ИР, входящих в корпоративную систему АС;
- ограничение доступа должностных лиц к ИР в рамках корпорации;
- отсутствие механизмов связывания ИР различных АС, входящих в корпоративную систему.

Для создания единой технологической среды управления корпорации, лишенной перечисленных недостатков, необходимо реализовать интеграцию данных, когда данные из различных источников, имеющие различные формы представления, могут использоваться в рамках общих процессов распределенной системы [2]. При этом из ИР, хранящихся в отдельных базах данных корпоративной АС, формируется единое информационное пространство (ЕИП), которое обеспечивает оперативный доступ к имеющимся ИР.

Интеграция ИР приводит к тому, что:

- обмен данными осуществляется в рамках единого адресного пространства;
- при решении задач могут быть использованы ИР из различных АС;
- доступ должностных лиц к ИР в рамках корпорации обеспечивается в соответствии с их полномочиями.

1 Обзор средств обмена данными

В настоящее время обмен данными осуществляется в основном синтаксическими способами, требующими согласования структур данных. Это приводит к составлению множества протоколов, в которых оговаривается состав обмениваемых данных, используемая терминология и система классификации данных.

При этом решение вопросов обмена данными носит частный характер, относящийся к конкретным хранилищам данных.

1.1 XML как способ унификации структур

С появлением XML-технологий отмеченная проблема обмена данными была частично решена. XML хранит все данные в виде текста, и поэтому передающие программы должны преобразовать свои данные из двоичного представления в текстовое. При восприятии XML-данных программами происходит обратное преобразование текстового представления в двоичное представление соответствующего типа данных принимающей программы.

Следовательно, XML-формат играет роль «международного» языка, обеспечивающего перевод форматов типов данных разных языков программирования, скажем, с языка Pascal на язык C++.

Именно это свойство XML обеспечивает взаимодействие между программами, реализованными на произвольной аппаратно-программной платформе. Но две обменивающиеся XML-данными стороны могут единообразно понимать и интерпретировать элементы данных только в том случае, если они совместно используют одинаковые определения.

С этой целью используется XML-схема, которая позволяет разработчикам определять структуру документа, используемые типы данных [3]. XML-схема содержит шаблон документа, определяющий тип используемой в документе информации и ее структуру. XML-схемы налагают на структуру документа ряд ограничений:

- определяют особый порядок следования элементов и их вложенность;
- определяют типы данных документа.

Использование XML-схемы существенно облегчает процедуру согласования структур данных, хотя при этом обмен данными между различными АС не выходит за рамки синтаксического способа.

Это объясняется тем, что описание структуры данных выполняется с помощью стандартной спецификации XSD (XML Schema Definition) и размещается в общедоступном для всех XML-документов файле.

Возможностей XML-схемы вполне достаточно для обмена данными между сторонами, которые используют одни и те же структуры обмениваемых данных. В этом случае данные, размещенные в фиксированных позициях, имеют один и тот же смысл для обменивающихся сторон, т. е. могут быть обработаны с помощью одних и тех же программ. Именно это обстоятельство обеспечивает обмен данными в условиях отсутствия поддержки семантики данных.

Однако отсутствие средств описания семантики в XML-схеме серьезно ограничивает выполнение обмена данными в случаях, когда структуры обмениваемых данных различаются. XML — слишком гибкое средство описания данных, и разметка одних и тех же данных может быть реализована различными способами. Например, в Листинге 1 приведены два варианта разметки в XML-формате утверждения «Сергей Васнецов — начальник отдела сбыта».

Приложение, которое использует первый формат, не сможет понять второй и наоборот. Это приводит к тому, что обменивающиеся XML-

```
<Department>
  <name>
    Сбыт
  </name>
  <chief>
    Сергей Васнецов
  </chief>
</Department>
```

```
<Person>
  <Name>
    Сергей Васнецов
  </Name>
  <Post>
    Начальник
  </Post>
  <Department>
    Сбыт
  </Department>
<Person>
```

Листинг 1. Описание одного факта в разных вариантах XML-формата

данными стороны, представляющие одни и те же данные по-разному, не смогут воспользоваться полученными данными.

1.2 RDF – язык описания семантики

Таким образом, при взаимодействии между АС, в которых не согласованы структуры обмениваемых данных, требуется автоматический анализ их содержания, а для этого необходимы средства выражения семантики данных.

Если синтаксис – это свойство, определяющее способ представления информации в сигнале (на носителе), то семантика определяет соответствие сигнала или термина реальному миру. Семантику можно рассматривать как некоторое соглашение между источником и потребителем информации, в соответствии с которым определяется значение каждого используемого термина.

Для описания семантики данных консорциумом W3C был разработан язык RDF (Resource Description Framework) [4]. Основополагающим для RDF является понятие «модели данных», представляемой как набор фактов и семантических связей между ними. Базовая конструкция этой модели данных представляет собой тройку

«ресурс – свойство – значение», где

- ресурс (субъект) обозначает все, что может иметь уникальный идентификатор. Это может быть, например, отдельная таблица;
- именованное свойство (предикат) обозначает некий аспект, характеристику, атрибут или отношение, используемое для описания ресурса;
- значение свойства (объект) определяет допустимые значения, тип ресурсов, к которым оно может быть применено, а также отношения с другими свойствами.

Пример описания фактов на языке RDF приведен в Листинге 2.

RDF-модель представляется XML-элементом, который заключается в тег <rdf:RDF>. Каждый элемент RDF должен содержать объявление

```
<?xml version=«1.0»?>
<rdf:RDF
 xmlns:rdf=«http://www.w3.org/1999/02/22-rdf-ns#»>
 xmlns:db=«http://db_orders.org/tables»>
  <rdf:Description                                  Субъект
   rdf:about=«http://dbOrders.ru/Department»>
    <db:nameDepartment>                             Предикаты
     Сбыт
    </db:nameDepartment >
    <db:Chief>
     Сергей Васнецов
    </db:Chief>
  </rdf:Description>
</rdf:RDF>                                          Объекты
```

Листинг 2. Описание фактов на языке RDF

пространства имен RDF.

В данном примере описывается один ресурс «http://dbOrders.ru/Department» и два его свойства «nameDepartment» и «Chief», имеющие значения «Сбыт» и «Сергей Васнецов» соответственно.

Утверждения, относящиеся к одному ресурсу, группируются с помощью элемента Description из пространства имен RDF. Идентификатор описываемого ресурса URI (Uniform Resource Identifier) указывается в структуре about элемента Description.

RDF дает формализм для аннотирования ресурсов с помощью метаданных и способ его записи в XML, но не дает никакого конкретного смысла элементам словаря, таким как Description или about.

Поэтому в дополнение к модели RDF был разработан механизм RDF Schema, который предоставляет базовую систему типов для моделей RDF (см. Листинг 3).

```
rdfs:Class (Класс)
rdf:Property (Свойство)
rdf:type (Тип)
rdfs:subClassOf (Подкласс)
rdfs:subPropertyOf
rdfs:range (область значений)
rdfs:domain (область определения)
```

Листинг 3. Термины RDF Schema

Эта система типов использует некоторые предопределенные термины, такие как Class, subPropertyOf и subClassOf, для схемы, ориентированной на конкретное приложение. Выражения RDF Schema также являются корректными выражениями RDF (как и выражения XML Schema — корректные выражения XML).

Объекты RDF можно определить как экземпляры одного или нескольких классов с помощью свойства type. Свойство subClassOf позволяет разработчику указывать иерархическую организацию таких классов, а subPropertyOf выполняет то же самое для свойств.

Таким образом, совместное использование с технологией XML модели RDF позволит отразить семантику концептуальных моделей АС, а также избежать изложенных ограничений XML.

1.3 Онтологии как средство формализации предметной области

RDF – это самый низкоуровневый из существующих языков описания семантики, поскольку оперирует лишь понятиями связей примитивных сущностей, например, «Свойство P объекта O равно V». Со временем стало очевидно, что средств XML и RDF для построения непрерывного семантически связанного информационного пространства недостаточно.

Эту проблему можно решить устранением или сведением к минимуму концептуальной и терминологической путаницы и установлением однозначного понимания языка, используемого для формирования требований и спецификаций сложных систем.

Для этого предметная область (ПрО) представляется в виде множества объектов и связей между ними, т. е. онтологий [5]. Онтология – это точная спецификация некоторой предметной области, включающая в себя словарь терминов из этой области и множество логических связей (типа «элемент-класс», «часть-целое»), которая описывает как эти термины соотносятся между собой.

В отличие от обычного словаря для онтологической системы характерно внутреннее единство и логическая непротиворечивость используемых терминов.

Онтология позволяет представить понятия в таком виде, что, с одной стороны, они соответствуют терминам ПрО и предназначены для использования непосредственно конечными пользователями, с другой стороны, – становятся пригодными для машинной обработки. Поэтому их основное назначение – служить посредником между пользователем и распределенной АС, так как они позволяют формализовать договоренности о терминологии между пользователями корпоративной АС.

В центре большинства онтологий находятся классы, которые описывают понятия ПрО. Классы в онтологии – это абстрактные группы, коллекции или наборы объектов. Они могут включать в себя экземпляры, другие классы либо сочетания и того, и другого.

Целесообразно применительно к ПрО рассматривать два вида классов:

- абстрактные классы, соответствующие обобщенным понятиям;
- реальные классы и объекты этих классов.

Если термины, обозначающие реальные классы, служат для ссылки на соответствующие классы ПрО, то абстрактные — предназначены для классификации и навигации по дереву с целью доступа к объектам реальных классов.

Онтология связывает два важных аспекта: во-первых, она определяет формальную семантику информации, обеспечивая обработку этой информации компьютером, во-вторых, определяет семантику реального мира, воспринимаемую человеком. Таким образом, онтология на основе общей терминологии связывает информацию, представленную для машинной обработки, с информацией, представленной для восприятия человеком.

1.4 Язык OWL

Для описания онтологий консорциумом W3C был создан язык онтологии OWL (Web Ontology Language), обеспечивающий запись онтологий в формате XML. Онтология, созданная с помощью OWL, включает описания классов, свойств и их экземпляров. Рассмотрим вкратце устройство языка OWL. Для этого воспользуемся фрагментами файла wine.rdf, описание которого приведено в работе [6], представленной консорциумом W3C.

Пространство имен. Прежде, чем использовать какое-либо множество терминов, необходимо определиться со словарями. Название или ссылка на конкретный словарь терминов (онтологию), как и в случае RDF, записывается в формате URI, в результате чего формируется распределенная база данных доменных имен. Следует отметить, что в этой базе не содержится информация об онтологиях, это просто удобный способ избежать коллизий имен. Объявление онтологий как пространств имен осуществляется в корневом элементе <rdf:RDF> файла онтологии и представлено в Листинге 4.

```
<rdf:RDF
 xmlns=«http://www.site.org/2003/ontology/drinks#»
 xmlns:drinks=«http://www.site.org/2003/ontology/drinks#»
 xmlns:owl=«http://www.w3.org/2002/07/owl#»
 xmlns:rdf=«http://www.w3.org/1999/02/22-rdf-syntax-ns#»
 xmlns:rdfs=«http://www.w3.org/2000/01/rdf-schema#»
 xmlns:xsd=«http://www.w3.org/2000/10/XMLSchema#»
>
```

Листинг 4. Пространства имен, объявленные в корневом теге <rdf:RDF> онтологии

Заголовок онтологии. После определения пространств имен для онтологии вводится заголовок описываемой онтологии, сгруппированный внутри элемента <owl:Ontology>. Также в этот раздел можно включить теги с комментариями и номерами версий. Типовой заголовок онтологии приведен в Листинге 5.

```
<owl:Ontology rdf:about="">
  <rdfs:comment>
    An example OWL ontology
  </rdfs:comment>
  <owl:priorVersion
    rdf:resource="http://www.w3.org/TR/2003/owl/wine"/>
  <owl:imports
    rdf:resource="http://www.w3.org/TR/2004/owl-guide"/>
  <rdfs:label>
    Wine Ontology
  </rdfs:label>
```

Листинг 5. Типовой заголовок онтологии

Внутри элемента <owl:Ontology> содержится вся метаинформация о документе. Атрибут rdf:about описывает имя либо ссылку на онтологию. Обычно этот атрибут имеет пустое значение, при этом названием онтологии служит базовый URI элемента <owl: Ontology>.

Основные элементы языка OWL. Основными элементами языка OWL являются классы, отношения между классами, экземпляры классов и их свойства.

Классы и отношения между классами. Наиболее общие понятия ПрО должны соответствовать классам, которые находятся в корне различных таксономических деревьев. Корневые классы определяются простым объявлением именованного класса.

В онтологии wine.rdf созданы три корневых класса (см. Листинг 6).

```
<owl:Class rdf:ID=«Винодельня»/>
<owl:Class rdf:ID=«Регион»/>
<owl:Class rdf:ID=«ПродуктПитания»/>
```

Листинг 6. Объявление корневых классов онтологии

В этом объявлении просто отмечено, что существуют классы с именами, обозначенными с помощью конструкции 'rdf:ID='. Больше об этих классах пока ничего не известно, единственное, что дает данное определение, так это возможность использования относительного идентификатора, например, #Винодельня.

Фундаментальным инструментом для формирования информационного пространства ПрО является конструкция rdfs: subClassOf, связывающая частный класс с более общим классом (см. Листинг 7).

```
<owl:Class rdf:ID=«Напиток»>
  <rdfs:subClassOf rdf:resource=«#ПродуктПитания»/>
  ...
</owl:Class>
```

Листинг 7. Объявление подкласса

В данном объявлении класс «Напиток» определяется как подкласс «ПродуктПитания».

Таким образом, определение класса состоит из двух частей:

- названия класса, вводимого с помощью конструкции 'rdf:ID=';
- списка ограничений, уточняющих свойства экземпляров класса.

В определении класса и его ограничений, приведенном в Листинге 8, атрибут «lang» определяет поддержку разных языков, а конструкция 'rdfs: label' подобна комментарию и ничего не вносит в интерпретацию онтологии.

```
<owl:Class rdf:ID=«Вино»>
  <rdfs:subClassOf rdf:resource=«&food;Напиток»/>
  <rdfs:label xml:lang=«en»>wine</rdfs:label>
  <rdfs:label xml:lang=«ru»>вино</rdfs:label>
  <rdfs:label xml:lang=«fr»>vin</rdfs:label>
  ...
</owl:Class>
```

Листинг 8. Объявление класса и его ограничений

Экземпляры классов. Определение класса «Вино» все еще неполно, и мы по-прежнему ничего не можем сказать о классе. Но мы можем создавать на основе подобных определений экземпляры классов.

Для наглядной демонстрации этой возможности создадим дополнительный корневой класс «Виноград» и его подкласс «ВинныйВиноград». Для подкласса «ВинныйВиноград» создадим экземпляр «ВиноградКаберне Совиньон», который является конкретным сортом винограда.

Описанное объявление экземпляра класса приведено в Листинге 9.

```
// Определение корневого класса
<owl:Class
  rdf:ID=«Виноград»>
  ...
</owl:Class>

// Определение подкласса корневого класса
<owl:Class
  rdf:ID=«ВинныйВиноград»>
  <rdfs:subClassOf rdf:resource=«&food;Виноград»/>
</owl:Class>

// Определение экземпляра подкласса
<ВинныйВиноград
    rdf:ID=«ВиноградКаберне Совиньон»
/>
```

Листинг 9. Объявление экземпляра класса

Определение свойств. Определение одних только классов без определения их свойств не представляет особого интереса. Только свойства позволяют нам сделать выводы об общих и отличительных фактах относительно экземпляров класса.

Как правило, свойства определяются через домен (domain) и диапазон (range) (см. Листинг 10).

```
<owl:ObjectProperty
  rdf:ID=«сделаноИзВинограда»>
  <rdfs:domain rdf:resource=«#Вино»/>
  <rdfs:range rdf:resource=«#Виноград»/>
</owl:ObjectProperty>
```

Листинг 10. Объявление свойств

Теперь, имея определение свойства, можно расширить определение класса «Вино» (см. Листинг 11). В данном объявлении класса мы добавили свойство «сделаноИзВинограда».

Кроме этого определили кардинальность неименного подкласса, которая определяет набор вещей, по крайней мере, с одним свойством «сделаноИзВинограда».

Как следует из приведенного Листинга, свойства, определенные в OWL, могут иметь диапазон простых типов, определенных в XML Schema datatypes. Ссылки на эти типы осуществляются посредством URI для типов http://www.w3.org/2001/XMLSchema, размещенном в разделе пространства имен (см. Листинг 4).

Итак, мы описали основные компоненты языка онтологий OWL. Следует отметить, что приведенное изложение дает только первое впечатление о языке. Для более подробного знакомства с возможностями OWL следует обратиться к документам, издаваемым консорциумом W3C, в частности, к уже упомянутой работе [6].

```
<owl:Class rdf:ID=«Вино»>
  <rdfs:subClassOf rdf:resource=«&food;Напиток»/>
  <rdfs:label xml:lang=«en»>wine</rdfs:label>
  <rdfs:label xml:lang=«ru»>вино</rdfs:label>
  <rdfs:label xml:lang=«fr»>vin</rdfs:label>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
        rdf:resource=«#сделаноИзВинограда»/>
      <owl:minCardinality rdf:datatype=«&xsd:Integer»>
        1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Листинг 11. Определение класса и его свойства

2 Организация доступа к базам данных как к онтологиям

Во введении к настоящей статье была поставлена задача реализации интеграции информационных ресурсов, размещенных в различных базах данных корпоративной АС, обеспечивающей возможность производить поиск по запросам, задаваемым непосредственно пользователями любой организации или предприятия корпорации.

Но доступ к реляционным базам предполагает знание пользователем структуры базы данных и производится либо через заранее предопределенные формы, либо посредством языка запросов SQL. При этом необходимо формулировать синтаксически правильные и семантически осмысленные запросы на соответствующем диалекте SQL.

В настоящее время формирование таких запросов осуществляется программистом, что определяется следующими причинами:

- разбиение данных по таблицам в различных АС корпорации может быть осуществлено очень большим количеством способов, причем конкретный вариант разбиения конечному пользователю, как правило, неизвестен;
- названия объектов ПрО и их свойств, представленные в реляционных базах данных названиями таблиц, столбцов, назначаются в соответствии с требованиями используемой СУБД и отличаются от употребляемых специалистами ПрО.

2.1 Постановка задачи

Возникает вопрос, а нельзя ли воспользоваться описанным выше языком OWL для разработки онтологии, которая обеспечила бы формулировку семантически правильного запроса к хранилищу данных корпорации в терминах, используемых в данной ПрО?

Отвечая на этот вопрос, необходимо учитывать тот факт, что данный язык был разработан прежде всего для формального описания терминов, используемых в Web-документах. При этом правила, заданные в OWL, могут быть использованы только специальными системами, понимающими язык описания семантики и способными выполнять логические выводы, наподобие процедур, написанных на языке Пролог.

Поэтому при разработке онтологии (системы терминов) для конкретной ПрО необходимо, чтобы она была описана формальным языком, основанном на принципах математической логики.

Только тогда для классов объектов, их свойств и отношений между ними мы сможем сформулировать четкие, детализированные и целостные определения. И только тогда средства обработки онтологии смогут автоматически вывести некоторые заключения, основываясь на принципах математической логики.

С другой стороны, ПрО, описанная в виде реляционных баз данных, представляет собой строго формализованную информацию. Причем в ряде работ, в частности в [7], доказывается, что факты, хранящиеся в реляционных базах данных, являются утверждениями, сформулированными на языке исчисления предикатов первого порядка. Следовательно, данные, содержащиеся в многочисленных базах данных, представляют собой ПрО, описанные формальным языком, основанном на принципах математической логики.

На основании этого описания можно реализовать онтологию, целью которой является обеспечение возможности для пользователей, являющихся специалистами в ПрО, сформулировать семантически правильный запрос с точностью до наименований элементов требуемой информации и их распределения по объектам базы данных, если эти наименования соответствуют терминам, используемым в данной ПрО. Реализация данной идеи позволит обеспечить унифицированный доступ к данным корпоративного хранилища вне зависимости от того, в какой АС эти данные представлены.

Для достижения поставленной цели необходимо решить задачу разработки методов автоматической трансформации запроса, составленного с терминах ПрО, в синтаксически и терминологически корректный запрос на языке SQL кон-

кретной базы данных, предположительно содержащей требуемые сведения, его выполнения и обратной трансформации запроса в термины, понятные пользователю, выдавшему запрос.

С учетом вышеизложенного поставим перед собой задачу: разработать специализированную онтологию для доступа к данным распределенного хранилища, не требующую использования специализированных языков, таких как RDF и OWL. Причем эта онтология, которую мы назовем БД-онтологией, должна обладать такими возможностями языка OWL, как:

- объявление классов и их экземпляров;
- установка на множестве классов родовидовых отношений;
- установка на множестве классов отношений агрегации;
- установка соответствия между классами и их свойствами;
- интерпретация терминов онтологии, заданных на множестве классов и отношений.

2.2 Способ решения задачи

С целью определения способа реализации поставленной задачи воспользуемся тем обстоятельством, что в инженерии знаний под онтологией понимается детальное описание некоторой ПрО, используемое для формального и декларативного определения ее концептуальной схемы. В рамках этого направления онтология реализуется в виде базы данных специального вида, которую можно самостоятельно использовать в рассматриваемой ПрО.

На этой основе можно реализовать механизм доступа к данным распределенного хранилища без использования специализированных языков, таких как RDF и OWL, обеспечивающего работу с реляционными базами данных как с некоторым ограниченным, но достаточно гибким вариантом онтологий. Суть данного способа заключается в том, чтобы:

- сведения об объектах ПрО, их свойствах и взаимосвязях, т. е. онтологии, представить в декларативной форме в базе данных специального вида, называемой базой метаданных;
- разработать программный модуль, осуществляющий преобразование запросов, составленных в терминах наименований объектов, их свойств и взаимосвязей, в запрос, сформулированный в логических терминах наименований таблиц, столбцов и отношений.

Преимущество данного способа реализации онтологии, которую можно назвать декларативной, заключается в том, что для работы с базой метаданных, где хранятся данные онтологии, можно использовать традиционные приложения баз данных, основанные на обычных языках высокого уровня (например, Pascal, Java, C++, C#), которые имеют в своем арсенале богатый набор средств для использования возможностей языка SQL при доступе к данным.

Решение задачи построения БД-онтологии в рамках выбранного способа осуществляется в два этапа:

1. Онтология содержит термины ПрО, используемые пользователями, и термины, введенные при разработке баз данных с учетом требований СУБД, управляемой базой данных некоторой АС корпорации. Поэтому, в первую очередь, необходимо осуществить согласование этих двух терминологических базисов для всех АС корпоративной системы. На этом этапе решаются также вопросы унификации структур данных в базе метаданных и разработки средств их ведения (последний вопрос в данной статье не рассматривается).

2. На втором этапе решаются вопросы, связанные с генерацией исполняемого запроса к базе данных. Здесь необходимо предусмотреть средства ввода данных для интерпретации терминов ПрО, а также разработать программный модуль, реализующий преобразование пользовательских запросов в запросы на языке SQL.

По существу на данном этапе идет речь об отображении агрегации ПрО в пользовательских (или концептуальных) терминах в агрегацию ПрО в логических терминах (или терминах СУБД). Следует отметить, что в предлагаемом подходе должно быть реализовано динамическое отображение базы данных в онтологию. То есть, если данные в базе данных изменяются, то они автоматически отображаются в соответствующих классах онтологии.

2.3 Разработка формальной модели декларативной онтологии

Для решения задачи разработки структур хранения декларативной онтологии воспользуемся традиционным определением онтологии, представляющим ПрО как множество понятий и утверждений об этих понятиях, на основе которых можно строить классы и их экземпляры, свойства классов и отношения между ними.

При этом будем различать БД-онтологии, составленные в соответствии с требованиями СУБД, используемых в АС корпоративной системы, и обобщенную онтологию (далее онтологию), реализованную в терминах ПрО. Сначала разработаем БД-онтологию, а затем при разработке онтологии осуществим согласование терминов этих онтологий.

После согласования терминов решим основную задачу: создание механизма для выражения информационных потребностей в терминах ПрО и их преобразование в запросы на диалектах языка SQL, используемых в СУБД корпоративной системы.

Разработка структур для хранения декларативной онтологии.

Описание БД-онтологии. БД-онтология базируется на следующих исходных правилах, задаваемых реляционной моделью:

- таблицы базы данных являются классами онтологии (каждой таблице БД соответствует свой класс онтологии);

- поля (столбцы) таблиц базы данных являются атрибутами (свойствами) классов;

- записи (строки) в таблицах базы данных являются объектами соответствующих классов;

- значения полей записи (пересечение столбца и строки) в таблице базы данных являются значениями соответствующих свойств объекта онтологии;

- на множестве классов (таблиц) могут быть заданы отношения;

- у всех свойств должны быть даны области определения либо путем отнесения к соответствующему типу данных, либо путем определения домена;

- для атрибутов могут быть заданы ограничения на значения.

Формально описанную БД-онтологию можно представить в виде шестерки:

$$O^{DB} = (C^l, A^l, R^{DB}, T^V, D, F),$$

где $C^l = (c_1^l, ..., c_i^l, ..., c_n^l)$ – множество логических наименований классов (таблиц), описывающих понятия некоторой ПрО;

$A^l = ({}^i a_1^l, ..., {}^i a_j^l, ..., {}^i a_{m(i)}^l), i = 1, ..., n$ – множество логических наименований атрибутов, описывающих свойства классов; кортеж $\left({}_e^i a_1^l, ..., {}_e^i a_j^l, ..., {}_e^i a_{m(i)}^l\right)$, $i = 1, ..., n;$ $e = 1, ..., E(i)$, составленный из значений свойств ${}^i a_j^l$ класса $c_i \in C$, представляет экземпляр этого класса (запись таблицы), а ${}_e^i a_j^l$ – значение свойства ${}^i a_j^l$ для данного экземпляра класса;

R^{DB} – множество отношений, заданных на классах;

T^V – множество типов данных, поддерживаемых СУБД;

D – множество доменов, поддерживаемых СУБД;

F – множество ограничений на значения атрибутов, задаваемых в определении классов (таблиц).

Приведенное формальное описание ПрО представляется в виде модели данных и реализуется в системном каталоге СУБД в виде базы метаданных. СУБД сопоставляет поступивший запрос на языке SQL с данными из базы метаданных и вызывает заданную процедуру (SELECT, INSERT, UPDATE, ...), которая выполняет в базе данных релевантные данному запросу действия.

Разработка онтологии. Компоненты C^l и A^l как элементы БД-онтологии используются при составлении запросов на языке SQL исходя из требований СУБД (например, наименования должны быть составлены только на латинице), что не всегда удобно для конечного пользователя. Поэтому при извлечении информации из реляционных баз данных сторонним пользователем или разработчиком возникают проблемы, связанные с особенностями наименований таблиц и их столбцов в рамках той или иной СУБД.

Эти особенности требуют рассмотрения наименований объектов ПрО как самостоятельных объектов, обладающих собственной структурой и имеющих семантически нагруженные связи. Обеспечение самостоятельности имени означает, что оно не должно зависеть от ограничений, налагаемых СУБД на наименования объектов баз данных.

Назовем такие наименования концептуальными. Если присвоение логических имен осуществляется разработчиком с учетом ограничений СУБД, то концептуальные наименования вводятся на этапе разработки концептуальной модели базы данных без привязки к какой-либо СУБД.

Концептуальная схема не предназначена для машинной обработки, а наименования, введенные в этой схеме, не используются СУБД. Наша задача – сделать концептуальные наименования пригодными для машинной обработки путем разработки онтологии ПрО, формализующей словарь концептуальной схемы.

Введение онтологии стандартизирует процесс именования объектов ПрО, согласно которому при разработке концептуальной схемы должны использоваться исключительно термины из заранее утвержденного словаря онтологии, или, если подходящие наименования в словаре онтологии отсутствуют, то включение в словарь онтологии этих терминов осуществляется строго по согласованию.

Формальная модель онтологии представляется упорядоченной тройкой конечных множеств

$$O = < T, R, I >, \quad (1)$$

где T – термины ПрО, которые описывают онтологию O;

R – отношения между терминами заданной ПрО;

I – функции интерпретации, заданные на терминах и отношениях онтологии O.

Рассмотрение перечисленных множеств онтологии начнем с множества ***терминов***.

С целью обеспечения машинной обработки концептуальных терминов мы должны обеспечить соответствие концептуальных и логических имен, так как запросы, сформулированные в концептуальных терминах, в конечном счете должны преобразоваться в запросы, сформули-

рованные в логических терминах, которыми оперирует СУБД.

Таким образом, в дополнение к терминам БД-онтологии мы получаем два множества концептуальных наименований, сформированных на этапе разработки концептуальной схемы базы данных с учетом ограничений, налагаемых онтологией:

$C^k = (c_1^k, ..., c_i^k, ..., c_n^k)$ – множество концептуальных наименований классов, описывающих понятия некоторой ПрО;

$A^k = ({}^i a_1^k, ..., {}^i a_j^k, ..., {}^i a_{m(i)}^k), i = 1, ..., n$ – множество концептуальных наименований атрибутов, описывающих свойства классов.

Смысл введения этих множеств заключается в том, что наименования, содержащиеся в них, не зависят от ограничений, налагаемых СУБД на имена объектов баз данных, и являются общими для всего корпоративного хранилища, тогда как множества C^l и A^l действительны только в рамках соответствующих АС.

Следует отметить, что в рассматриваемой онтологии множество терминов ПрО, наряду с терминами БД-онтологии, включает в себя термины, обозначающие абстрактные классы. Абстрактные классы не имеют свойств и вводятся для классификации объектов БД-онтологии. Таким образом, с учетом вышеизложенного множество терминов онтологии представляется как

$T = (C^l, C^k, C^a, A^l, A^k),$

где C^l – множество логических терминов, используемых СУБД;

C^k – множество концептуальных терминов ПрО;

C^a – множество терминов, обозначающих абстрактные классы ПрО;

A^l – множество логических терминов атрибутов реальных классов ПрО;

A^k – множество концептуальных терминов атрибутов реальных классов ПрО.

На множестве терминов T заданы ***отношения*** R, составленные из отношений между концептуальными и логическими терминами, родовидовых отношений, отношений «часть-целое» и «класс-атрибут».

Отношения между концептуальными и логическими терминами R^{kl} на множестве T определяются отдельно для терминов, обозначающих классы и их атрибуты. Отношения для заданных классов обозначим символом R_C^{kl} и опишем выражением

$$R_C^{kl} = \left(\frac{c_1^k}{c_1^l}, ..., \frac{c_w^k}{c_w^l}, ..., \frac{c_W^k}{c_W^l} \right), \quad (2)$$

где $\frac{c_w^k}{c_w^l}$ – «связки» концептуальных и логических терминов.

Здесь необходимо отметить, что символ W означает количество абстрактных и реальных классов ПрО и его значение получается путем добавления некоторого числа абстрактных классов к количеству реальных классов, определенных на основе классов в БД-онтологии, поэтому $W \succ n$.

Отношения между концептуальными и логическими терминами R^{kl} на множестве T определяются только для терминов, обозначающих реальные классы. Но с целью *унификации структур*, в выражении (2) используются одинаковые конструкции для терминов всех типов классов. Для абстрактных классов логические термины отсутствуют, поэтому для них логические термины в «связке» не представлены, и в соответствующей позиции структуры элемента онтологии хранится «пустое значение», т. е. $c_w^l = NULL$. Поэтому в данных случаях «связки» представлены как

$c_w^k / NULL \in R_C^{kl}$.

Отношение R^{kl} на множестве терминов атрибутов классов определяет пространство терминов атрибутов

$$R_A^{kl} = \left| \frac{{}^i a_1^k}{{}^i a_1^l}, ..., \frac{{}^i a_{j(i)}^k}{{}^i a_{j(i)}^l}, ..., \frac{{}^i a_{m(i)}^k}{{}^i a_{m(i)}^l} \right|,$$

где $i = 1, ..., n;$

$\frac{{}^i a_{j(i)}^k}{{}^i a_{j(i)}^l}$, $j(i) = 1, ..., m(i)$ – «связки» концептуальных и логических терминов атрибутов классов ПрО.

Множество R_A^{kl} по мощности не отличается от множеств A^l или $A,^k$ так как абстрактные классы атрибутов не имеют, и их добавление в онтологию не сказывается на количестве атрибутов, введенных в БД-онтологии.

Родовидовые отношения R^{rv} определяются только на терминах абстрактных классов путем определения фиксированной системы классификационных признаков. Система классификационных признаков, распределенная по заданным позициям, определяет код абстрактного класса. Значения этих признаков фактически являются

терминами соответствующих классов и определяют систему абстрактных классов, заданных на множестве C^a.

Таким образом, абстрактные классы, а также родовидовые взаимосвязи между ними, определяются кодом, который в определенных позициях содержит значения соответствующих признаков классификации.

Классификационные признаки вводятся независимо от форм организации и представления данных в БД корпоративных АС. Поэтому система абстрактных классов, формируемая с ее помощью, инвариантна относительно схем баз данных АС корпорации и является своего рода надстройкой над схемами баз данных корпоративного хранилища.

Множество терминов, на котором последовательно определены отношения между концептуальными и логическими терминами R^{kl} и родовидовые отношения ${}^{rv}R$ можно описать выражением:

$$ {}^{rv}R_C^{kl} = \left(\frac{c_1^k}{c_1^l} | K_1, \ ..., \ \frac{c_w^k}{c_w^l} | K_w, \ ..., \ \frac{c_W^k}{c_W^l} | K_W \right), $$

где $\frac{c_w^k}{c_w^l}$ – «связки» концептуальных и логических терминов;

K_w – код абстрактного класса, составленный из классификационных признаков.

Классификационные признаки задают единую, унифицированную, предварительно определенную классификацию терминов ПрО. С их помощью определяется наиболее стабильная часть структуры ПрО, мало подверженная изменениям с течением времени. Эта структура содержит фиксированное число уровней в соответствии с количеством классификационных признаков.

Для заполнения описанной структуры реальными классами, соответствующими классам, заданным в БД-онтологии, классификационный код необходимо дополнить кодом вложенности, позволяющим расширить дерево, построенное на базе классификационных признаков, произвольным количеством уровней. Применение такого комбинированного способа кодирования классов ПрО для представления родовидовых взаимосвязей и отношений типа «часть-целое» описано в работе [8].

Дополнение кода класса кодом вложенности позволяет расширить конечные узлы отношения ${}^{rv}R_C^{kl}$ иерархическими деревьями с произвольным числом уровней вложенности и реальными классами, детализирующими соответствующие абстрактные классы. Данное дополнение можно трактовать как введение отношения «часть-целое» ${}_{ip}R$ на отношении ${}^{rv}R_C^{kl}$.

Множество терминов, на котором в дополнение к вышеописанным применено еще и отношение *«часть-целое»*, ${}_{ip}R$ описывается выражением:

$$ {}^{rv}_{ip}R_C^{kl} = \left(\frac{c_1^k}{c_1^l} | \frac{K_1}{L_1}, \ ..., \ \frac{c_w^k}{c_w^l} | \frac{K_w}{L_w}, \ ..., \ \frac{c_W^k}{c_W^l} | \frac{K_W}{L_W} \right), \quad (3) $$

где $\frac{c_w^k}{c_w^l}$ – «связки» концептуальных и логических терминов;

K_w – код абстрактного класса, составленный из классификационных признаков;

L_w – указатель на родительский узел, определяющий уровень вложенности элемента.

Отношения «класс-атрибут» ${}^{ca}R$ определяются между терминами реальных классов и терминами их атрибутов. Эти отношения представлены в БД-онтологии с помощью верхних левых индексов i, однако необходимо переопределить эти отношения и на уровне онтологии с тем, чтобы получить возможность согласования терминов не только для классов, но и для их атрибутов.

Для этого необходимо связать кортежи атрибутов из отношения R_A^{kl} с соответствующими классами из отношения ${}^{rv}_{ip}R_C^{kl}$. Это можно реализовать путем дополнения кортежей отношения R_A^{kl} ссылкой F_i на *i*-й класс в отношении ${}^{rv}_{ip}R_C^{kl}$. В результате получаем новое отношение, описываемое выражением:

$$ {}^{ca}R_A^{kl} = \left| \left(\frac{{}^i a_1^k}{{}^i a_1^l}, \ ..., \ \frac{{}^i a_{j(i)}^k}{{}^i a_{j(i)}^l}, \ ..., \ \frac{{}^i a_{m(i)}^k}{{}^i a_{m(i)}^l} \right) F_i \right|, \quad (4) $$

где $i = 1, ..., n$;

$\frac{{}^i a_{j(i)}^k}{{}^i a_{j(i)}^l}$, $j(i) = 1, ..., m(i)$ – «связки» концептуальных и логических терминов атрибутов классов ПрО;

F_i – ссылка на *i*-й класс в отношении ${}^{rv}_{ip}R_C^{kl}$.

Разработка механизма преобразования пользовательского запроса в исполняемые SQL-запросы. Итак, мы определили термины онтологии и отношения, заданные на множестве терминов. При этом мы решили также задачу согласования терминологических базисов, используемых в ПрО и реляционной базе данных. Теперь мы должны решить задачу генерации исполняемого запроса к базе данных по запросу,

сформулированному в терминах ПрО. Для этого, как уже было отмечено выше, необходимо определить ***интерпретацию*** I терминов ПрО.

Онтология O, описываемая выражением (1), агрегирует ПрО иначе, чем определено в БД-онтологии. В качестве примера возьмем фрагмент схемы базы данных предприятий корпораций, состоящей из таблиц «Предприятия», «Подразделения» и «Служащие», между которыми установлены соответствующие связи.

Подобная классификация неудобна для человеческого восприятия. Человеку привычнее воспринимать этот фрагмент ПрО, когда он разбит на классы: «Предприятия корпорации», «Подразделения предприятия корпорации», «Служащие предприятия корпорации», «Служащие подразделения предприятия корпорации».

Термины онтологии определяют именно подобного рода агрегацию ПрО. Поэтому, когда речь идет о заполнении абстрактных классов конкретными, имеются в виду не классы (таблицы), определенные в рамках БД-онтологии, а классы, сформированные на их основе, но в соответствии с пользовательскими представлениями о ПрО.

При этом мы должны отобразить термины онтологии на термины БД-онтологии таким образом, чтобы перераспределить экземпляры классов БД-онтологии на классы онтологии. Это и будет определять интерпретацию терминов онтологии на БД-онтологии.

Задать интерпретацию терминов – это значит перевести запрос, составленный из терминов ПрО, в запрос, сформулированный в логических терминах СУБД, т. е. на языке SQL. При выполнении такого преобразования необходимо решить, на каких реальных классах (таблицах) выполняется операция, какие атрибуты и экземпляры классов задействованы в операции. Но эта информация не содержится в исходном запросе, сформулированном в концептуальных терминах.

Эта информация частично содержится в БД-онтологии (т. е. в модели данных или системном каталоге СУБД), а частично должна быть добавлена в интерпретирующую функцию, заданную для каждого термина онтологии, и которую нам еще предстоит определить.

Интерпретирующая функция должна иметь в качестве входных данных сведения о классах, их атрибутах, а также ограничениях, обеспечивающих выборку экземпляров класса.

В рамках предлагаемого подхода терминам онтологии сопоставляются соответствующие сведения из базы метаданных онтологии, тем самым снабжая интерпретирующую функцию всеми необходимыми данными для формирования запроса в терминах SQL.

В общем виде входные данные интерпретирующей функции, обозначенные термином INP, можно представить выражением:

$$INP = \begin{vmatrix} c_1^k & \dots & c_p^k & \\ {}^1a_1^k & \dots & {}^pa_{q(p)}^k & \\ {}^1_ua_1^k & Op_1^k & v_1^k & {}^1LOp_1^k \\ \dots & \dots & \dots & \dots \\ {}^{p-1}_ua_{s(p-1)}^k & Op_{s(p-1)}^k & v_{s(p-1)}^k & {}^{p-1}LOp_{s(p-1)}^k \\ {}^p_ua_{s(p)}^k & Op_{s(p)}^k & v_{s(p)}^k & \end{vmatrix}, \quad (5)$$

где INP – обозначение входных данных интерпретирующей функции;

$c_1^k, \dots, c_p^k$ – концептуальные термины классов, над которыми выполняется операция;

${}^1a_1^k, \dots, {}^pa_{q(p)}^k$ – концептуальные термины для обозначения атрибутов классов $c_1^k, \dots, c_p^k$, участвующих в операции;

${}^1_ua_1^k, \dots, {}^p_ua_{s(p)}^k$ – концептуальные термины для обозначения атрибутов классов $c_1^k, \dots, c_p^k$, используемых для задания ограничений;

$Op_1^k, \dots, Op_{s(p)}^k$ – концептуальные термины для обозначения операций проверки ограничений;

$v_1^k, \dots, v_{s(p)}^k$ – значения для проверки ограничений;

${}^1LOp_1^k, \dots, {}^{p-1}LOp_{s(p-1)}^k$ – концептуальные термины для обозначения логических операций для объединения операций $Op_1^k, \dots, Op_{s(p)}^k$ в одну общую операцию.

Описанная конструкция формируется для каждого термина онтологии и активизируется всякий раз, когда возникает необходимость в интерпретации соответствующего термина. Эта конструкция относится к декларативной онтологии, и ее можно рассматривать как способ реализации отношения «часть-целое» на БД-онтологии.

Назначение интерпретирующей функции заключается в преобразовании описанных выше данных в запросы на языке SQL. Для реализации интерпретирующей функции необходимо разработать программный модуль, который, используя входные данные INP, должен преобразовать пользовательский запрос в исполняемое SQL-предложение, обозначаемое термином OUT.

Конструкции запросов для операций языка SQL (таких, как SELECT, INSERT, UPDATE, DELETE) различаются. Поэтому для каждого типа операции можно разработать обобщенное представление, позиции которого будут заполняться с помощью интерпретирующей функции.

В качестве примера рассмотрим обобщенное

представление запроса $SELECT$ в логических терминах СУБД, которое формально можно представить как

$$OUT = \begin{vmatrix} {}^{1}a_1^l & \dots & {}^{p}a_{q(p)}^l & \\ c_1^l & \dots & c_p^l & \\ {}^{1}_{u}a_1^l & Op_1^l & v_1^l & {}^{1}LOp_1^l \\ \dots & \dots & \dots & \dots \\ {}^{p-1}_{u}a_{s(p-1)}^l & Op_{s(p-1)}^l & v_{s(p-1)}^l & {}^{p-1}LOp_{s(p-1)}^l \\ {}^{p}_{u}a_{s(p)}^l & Op_{s(p)}^l & v_{s(p)}^l & \end{vmatrix}, \quad (6)$$

где OUT– обозначение выходного выражения интерпретирующей функции;

${}^{1}a_1^l, \dots, {}^{p}a_{q(p)}^l$ – логические термины атрибутов классов $c_1^l, \dots, c_p^l$, участвующих в операции;

$c_1^l, \dots, c_p^l$ – логические термины классов, над которыми выполняется операция;

${}^{1}_{u}a_1^l, \dots, {}^{p}_{u}a_{s(p)}^l$ – логические термины атрибутов классов $c_1^l, \dots, c_p^l$, используемых для задания ограничений;

$Op_1^l, \dots, Op_{s(p)}^l$ – логические термины для обозначения операций проверки ограничений;

$v_1^l, \dots, v_{s(p)}^l$ – значения для проверки ограничений;

${}^{1}LOp_1^l, \dots, {}^{p-1}LOp_{s(p-1)}^l$ – логические термины для обозначения операций объединения отдельных операций $Op_1^l, \dots, Op_{s(p)}^l$ в одно общее выражение.

Таким образом, интерпретирующую функцию I можно представить как отображение, осуществляющее формирование конструкции OUT по данным, извлекаемым из конструкции INP, что можно записать как

$$I : INP \to OUT. \quad (7)$$

При реализации данного отображения выполняются следующие действия:

- извлекаются концептуальные термины заданного узла онтологии согласно выражению (5);
- в соответствии с выражениями (3, 4) этим терминам сопоставляются их логические аналоги;
- затем логические термины подставляются в позиции выражения (6).

Итак, мы определили все компоненты онтологии, представленной выше в виде упорядоченной тройки конечных множеств, описываемых выражением (1). В следующем разделе наметим подходы реализации описанной онтологии и ее использования при доступе к данным корпоративного хранилища.

2.4 Схема реализации и использования декларативной онтологии

При описании способа решения задачи, поставленной в данной статье, было определено, что онтологию для решения проблемы интеграции данных, содержащихся в базах данных, управляемых различными СУБД, целесообразно представить преимущественно декларативно, в отличие от онтологий, реализуемых с помощью языка OWL.

С этой точки зрения компоненты T и R представляют декларативную часть онтологии, а компонент I – процедурную.

С учетом данной формальной модели декларативная составляющая онтологии представляется выражениями (3, 4, 5), а процедурная – выражением (7), в котором описывается процедура формирования выражения SQL-запроса (6) по данным из выражения (5).

В соответствии с вышеизложенным онтологию ПрО, состоящую из декларативной и процедурной составляющих, можно представить в виде схемы, приведенной на рисунке 1.

Причем декларативная онтология является инвариантной для всех АС корпоративной системы, тогда как процедурная онтология должна учитывать особенности SQL-запросов, используемых СУБД.

Как следует из рисунка 1, пользовательский запрос, сформулированный в терминах ПрО, на-

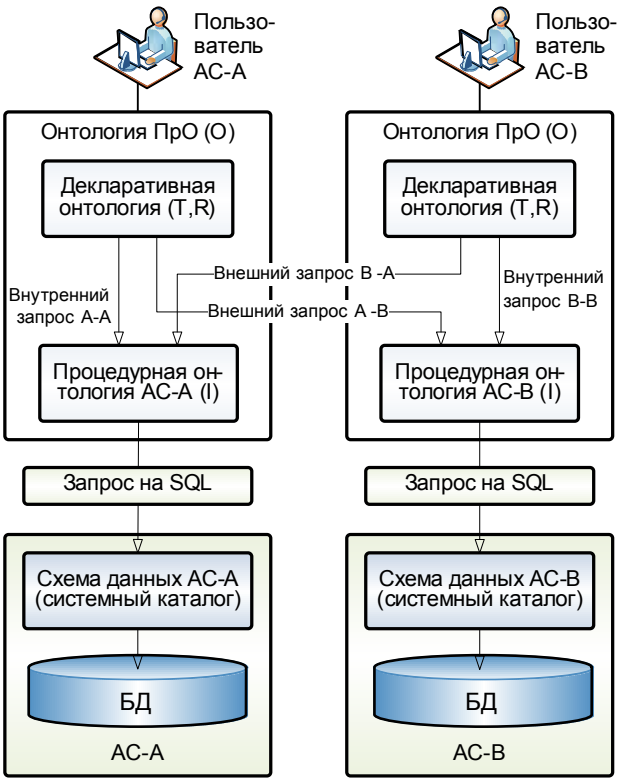

Рис. 1. Структура онтологии и ее использование при доступе к локальным и удаленным данным корпоративного хранилища

правляется на вход процедурной онтологии той АС, данные которой запрашиваются. При этом обеспечивается формирование SQL-запроса в терминах используемой в данной АС СУБД.

Описанный способ доступа к распределенным данным можно рассматривать как интеграцию данных, в соответствии с которой доступ к внешним источникам данных осуществляется посредством единого интерфейса в рамках единой модели данных. При этом, с точки зрения пользователя, обращение к внешним источникам ничем не отличается от обращений к единому источнику данных.

Заключение

Анализ средств, обеспечивающих обмен распределенными данными, показывает, что использование для описания семантики данных языков RDF и OWL приводит к необходимости использования специализированных систем, понимающих эти языки и способных выполнять логические выводы на основе конструкций OWL.

В качестве альтернативы использования языка OWL в статье предлагается:

- использовать декларативную онтологию, для хранения терминов и отношений которой разрабатываются соответствующие конструкции базы метаданных;

- разработать механизм преобразования пользовательских запросов, сформулированных в терминах онтологии, в исполняемые запросы на языке SQL используемой СУБД для последующей реализации на традиционном языке программирования высокого уровня.

Разработка такой онтологии обеспечивает прозрачный доступ к базам данных корпоративного хранилища за счет использования при кодировании экранных форм стандартизированных концептуальных терминов онтологии. На локальной АС эти термины «переводятся» в логические, и на их основе с помощью механизма преобразования пользовательского запроса может быть построен исполняемый SQL-запрос.

Данная статья не претендует на полное и подробное описание вопросов использования онтологии для решения проблемы интеграции данных. В ней не рассмотрены такие важные вопросы, как разработка языка описания декларативной онтологии, ввод данных онтологии, редактирование и ведение онтологии в ходе ее эксплуатации, отображение результатов интерпретации терминов онтологии и т. д.

Поэтому данную статью можно рассматривать как начало цикла публикаций в данном журнале по новой тематике, связанной с использованием онтологий в реальных разработках.

СПИСОК ЛИТЕРАТУРЫ

1. Единое информационно-функциональное пространство ВМФ: от идеи до реализации / под общ. ред. В.И. Кидалова. – СПб. : Ника, 2003.

2. Токмаков Г. П. Интеграция информационных ресурсов: эволюция, механизмы и формальная модель / Г. П. Токмаков // Автоматизация процессов управления. – 2005. – № 1(5). – С. 33–39.

3. Сандра Э. XML : справочник / Э. Сандра. – СПб. : Питер, 1999.

4. Андон Ф. И. Semantic Web как новая модель информационного пространства Интернет / Ф. И. Андон, И. Ю. Гришанова, В. А. Резниченко // Проблеми програмування. – 2008. – № 2–3. – С. 417–430.

5. Вагин В. Н. Разработка метода интеграции информационных систем на основе метамоделирования и онтологии предметной области / В. Н. Вагин, И. С. Михайлов // Программные продукты и системы. – 2008. – № 1.

6. OWL, язык Web-онтологий. Краткий обзор // Рекомендации W3C от 10 февраля 2004 г. – Режим доступа: http://www.w3.org./TR/owl-features/.

7. Плесневич Г. С. Логические модели / Г. С. Плесневич // Искусственный интеллект. В 3 кн. Кн. 2. Модели и методы : справочник / под ред. Д. А. Поспелова — М. : Радио и связь, 1990. — 304 с.

8. Захарьев А. А. Концептуальные вопросы построения информационного обеспечения ведения обстановки в АСУ ВМФ / А. А. Захарьев, А. Ф. Зальмарсон // Автоматизация процессов управления. – 2006. – № 2(8). – С. 118–131.