Архивная фактографическая система

© Марчук А.Г., Марчук П.А.

Институт систем информатики им. А.П. Ершова СО РАН, НГУ, Новосибирск
mag@iis.nsk.su, peter@iis.nsk.su

Аннотация

В Институте систем информатики СО РАН создана и используется архивная система, построенная на фактографических принципах [1, 2]. Особенностями системы являются: распределенная база данных, реализованная в виде документов RDF, специальные средства поддержки распределенного редактирования и динамической синхронизации; использование формальных спецификаций схемы данных и ряда свойств данных через описания OWL, что обеспечивает сменность или расширяемость системы структуризации и наборов метаинформационных полей; предложена и реализована техника кассет, группирующих документные массивы и обеспечивающая экономный доступ к документам, перемещаемость опубликованных документов и общую схему публикации и архивации документов и базы данных; имеется совместимость с базами данных класса Semantic Web, реализованными средствами RDF и OWL, имеется возможность организации синхронизации с базами данных, реализованными средствами реляционных таблиц. Данный доклад описывает реализацию сформированного подхода и является фактически продолжением указанных работ. Работа выполнена при поддержки гранта РГНФ 08-03-12125в, программы РАН р 2/12.

1 Архитектура системы

Рассмотрим программные компоненты архивной системы и их взаимодействие Архивная система создана как модульная открытая система с возможностью гибкого конфигурирования и настроек на потребности групп пользователей. Объединяющие такой подход компоненты это ядро системы и базовая онтология. Формально говоря, базовая онтология не является «встроенной» в примененный подход, «встроенными» являются некоторые принципы формирования базовой онтологии. Это дает возможность подставить в архивную систему другую онтологию (построенную с учетом указанных принципов) и получить архивную систему с другой структуризацией данных. При этом, если подставленная онтология совсем другая, то мы утеряем возможность интеграции данных, основанных на этих разных онтологиях. Рекомендуемый способ изменения онтологии – расширение базовой онтологии до требуемой полноты описания конкретной предметной области. Тогда срез данных, соответствующих базовой системе понятий и отношений, будет основой для интеграции данных, построенных на разных онтологиях.

К другим компонентам текущей реализации архивной системы относятся:

- Универсальный редактор базы данных, выполненный в виде Web-приложения;
- Публичный интерфейс информационного массива «Фотоархив СО РАН»;
- Система обработки, архивации и публикации первичных документов;
- Универсальное фактографическое приложение «Фактограф», ориентированное на персональную и коллективную работу с фотодокументами и базой данных;
- Системный интегратор, позволяющий динамически синхронизовать изменения, порождаемые в разных активных моделях;
- Набор утилит, предназначенный для анализа и восстановления данных, импорта внешних массивов данных, резервного копирования и обеспечения целостности данных.

Общая архитектура архивной системы представлена на рисунке 1.

Архивная система реализуется как распределенная информационная система, в которой распределены данные (распределенная база данных), сервера обработки и доступа, а также абоненты системы. Распределенная база данных выполнена в виде опубликованных RDF/OWL файлов, клиентская часть может состоять из универсальных клиентов (браузеры) и специализированных клиентских приложений. Сервера обработки и доступа выполняют предписанные действия на собранных из элементов распределенной базы моделях. Модель представляет собой внутреннее представление

Труды 11[й] Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2009, Петрозаводск, Россия, 2009.

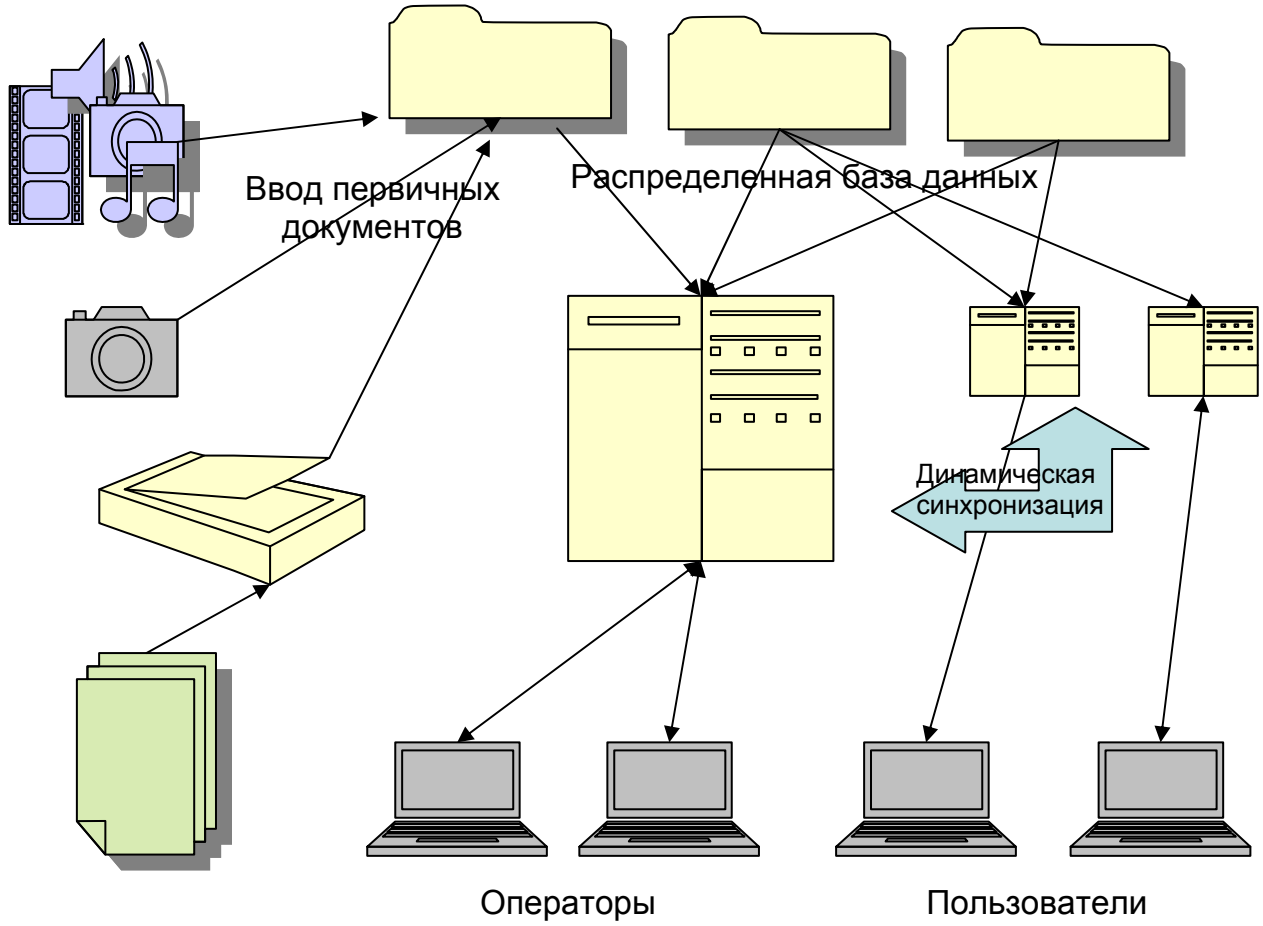

Рис. 1. Архитектура архивной системы

семантической сети и формируется из подмножества загруженных RDF-документов. Поскольку при функционировании системы отдельные модели могут изменяться, например средствами редактирования, необходимо производить динамическую синхронизацию, как с файлами документов, так и моделей между собой.

Отдельным слоем архивной фактографической системы является работа с документами, являющимися элементами базы данных. К документам относятся публикации, фотографии (фотодокументы), отсканированные образы страниц, аудио-видео файлы и т.д. Особенностью таких документов является то, что они фигурируют в базе данных в двух вариантах. С одной стороны – документы представлены в базе данных своими метаинформационными записями. С другой стороны, каждый документ это и конкретный информационный контент, например файл имиджа фотографии, который специальным образом структурируется, группируется и публикуется. Для этого была применена технология кассет данных.

Детали устройства и реализации отдельных слоев архивной фактографической системы будут изложены в следующих разделах.

2 Структура данных, онтология

Центральным местом структуризации данных в системе является онтология. Онтология задается в соответствии со спецификациями OWL Light [3] одним или несколькими OWL-документами. В настоящее время используется базовая онтология неспецифических сущностей, в некоторых случаях она несколько расширена для соответствия конкретной предметной области. Онтология определяет набор классов сущностей и отношений между ними, для реализации атрибутированных отношений используются классы псевдосущностей [2]. Например, отношение «работа» между персоной и организацией, нуждается в использовании атрибутов таких как: дата начала, дата завершения, должность и др. Соответственно, в онтологии введен класс participation. Этот класс не определяет объектов, а через объектные свойства participant и in-org порождает сложную связь между персонами и организациями. Такой подход делает нелогичным зафиксированный в стандарте OWL идентификатор thing в качестве корневого для всех классов, в нашей онтологии в качестве идентификатора корневого класса используется entity (сущность).

С некоторыми упрощениями, приведем основную часть используемой онтологии.

Спецификации в формализме OWL довольно громоздки и не достаточно наглядны, поэтому представлением в виде дерева, отражающего наследование классов. На диаграммах простые идентификаторы – идентификаторы классов, идентификаторы, следующие за прямоугольником – идентификаторы свойств. Причем если далее идут скобки, то это объектные свойства (ObjectProperty) и в скобках отмечен связываемый класс (range).

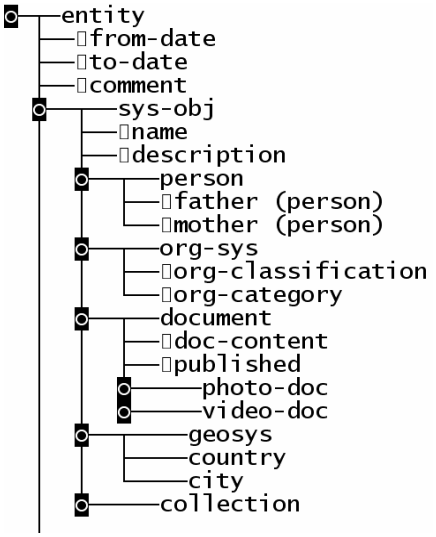

Рис. 2. Система объектов

На рисунке 2 приведены основные используемые в онтологии классы объектов. Для выделения свойства «объектности» класса, введен абстрактный класс системного объекта (sys-obj). Определено пять классов: персоны, организационные системы, документы, географические системы и коллекции. Для этих классов введен минимальный набор полей (DatatypeProperty) и объектных ссылок (ObjectProperty), используется наследование свойств.

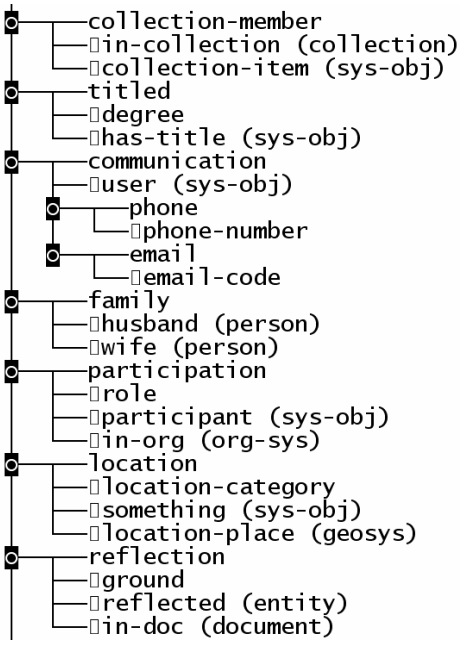

Рис. 3. Сложные (составные) отношения

Рисунок 3 представляет собой продолжение рисунка 2, на нем приведена система составных отношений между введенными объектами. Например, отношение collection-member устанавливает ассоциацию между объектом класса collection и системным объектом. Имеются не только бинарные отношения, но и унарные: titled, communication.

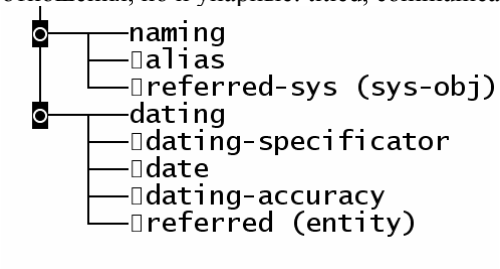

Рис. 4. Специальные отношения

Специальные унарные отношения представлены на рисунке 4, они реализуют более развернутым образом такие важные поля как name, from-date, to-date. Это дает возможность множественного указания имен и дат, а также записать уточняющую информацию. Например, девушка Иванова вышла замуж и сменила фамилию на Петрова. Соответственно, в базе данных была запись:

```
<person rdf:about="ivanova_4133">
   <name>Иванова</name>
</person>
```

Традиционным методом приведения базы данных в соответствие, является ее редактирование. Однако, если изменить значение name, то утеряется предыдущее значение, что является нарушением фактографических принципов. Применяя специальное отношение naming, мы можем отредактировать запись так, чтобы предыдущее значение не терялось, а было правильно оформлено. Для данного примера вместо одной записи появится две:

```
<person rdf:about="ivanova_4133">
   <name>Петрова</name>
</person>
<naming rdf:about="nmg28738">
   <alias> Иванова </alias>
   <referred-sys rdf:resource="ivanova_4133"/>
   <from-date>(дата рождения)</from-date>
   <to-date>(дата смены фамилии)</to-date>
</naming>
```

Аналогично, датирование помогает преодолевать проблемы связанные с неточностью знаний о дате и наличия нескольких версий дат.

3 Программное ядро системы

В архивной фактографической системе выделен общий модуль ядра системы, работающий с моделью данных. Этот модуль используется во всех программных решениях, использующих модель данных для своей работы. Такое инкапсулирование работы с данными позволяет оставлять возможность изменения отдельных технических решений,

связанных с форматами, пространствами имен, структурой и смыслом атрибутов документов и др. Кроме того, ядро системы сопряжено со слоем динамической синхронизации (см. далее).

В отличие от многих систем, работающих с RDF-данными, было принято решение не использовать реляционную СУБД в качестве движка, а создать прямую программную реализацию действий и доступов, соответствующую базовой модели RDF. Это значит, что объектами модели являются сущности и отношения, порождающие в совокупности семантическую сеть – ориентированный граф, состоящий из узлов и направленных дуг. Такое построение пока рассчитано только на полную загрузку модели в оперативную память исполняющего компьютера. Основанием к тому является быстрый рост объемов памяти не только на серверных, но и на пользовательских компьютерах. Оценка показывает, что подобное сетевое построение для 1 миллиарда фактов потребует порядка 100 Гб оперативной памяти, что не выглядит невыполнимым уже сейчас. При этом, баз данных общего назначения, содержащих миллиард фактов пока не создано, обычные объемы – сотни тысяч и единицы миллионов фактов.

Модуль ядра загружает RDF и OWL файлы, при этом используется технология кассет, см. ниже. Далее строится модель данных и модель онтологии. К модели данных и модели онтологии определен ряд методов доступа и манипуляций. Логически, модель данных представляется ориентированным графом, причем методы доступа позволяют «ходить» как по стрелкам (ссылкам), так и против. Еще одной особенностью представления модели является то, что сохраняется исходная структура документов. Это означает, что каждая информационная запись приписана к тому документу, в котором она находилась в файле RDF-документа. Имеется набор методов редактирования модели. При редактировании, сделанные изменения периодически записываются в RDF-документы, доступные компьютеру для записи. Если документ не доступен по записи, то посылается сообщение, в общем случае удаленному, сервису синхронизации детали работы которого будут изложены позже.

Еще один слой ядра поддерживает многоабонентскую работу. Это используется например в Web-приложениях, работающих в конфигурациях архивной фактографической системы. Также имеется менеджер документов и менеджер проектов.

Ядро поддерживает протоколирование (Log) выполненных действий, соответственно есть возможность реализации откатов и восстановления данных в случае порчи по непредвиденным обстоятельствам. Ядро снабжено Web-сервисом, работающем в логике синхронизации (см. далее) распределенной системы активных моделей.

Класс работы с онтологией расширяет класс работы с RDF-моделью специальными методами доступа к модели и ее различных представлений.

4 Публичный и операторский интерфейсы

В качестве основных средств работы, фактографическая система снабжена Web-приложениями, представляющими публичный пользовательский (front-end) и операторский (back-end) интерфейсы. Публичный интерфейс предназначен для доступа пользователей к информационным ресурсам системы. В общем случае, для достижения этой цели, для каждого большого проекта нужно разрабатывать свой вариант публичного интерфейса. Для проекта «Фотоархив СО РАН», такой интерфейс был создан и эксплуатируется с мая 2007 года. Скриншот типовой страницы сайта и некоторые детали его устройства имеются в [1].

Для наполнения фотоархива данными и их редактирования, был создан операторский интерфейс. Это Web-приложение, совместно с графическим обликом страниц, было спроектировано как универсальное решение, настраивающееся на предметную область через формальные спецификации – онтологию. Как универсальный компонент, операторский интерфейс был опробован и использован в целом ряде проектов, выполненных в ИСИ.

Операторский интерфейс построен на основе представления множеств записей в виде таблицы. Пусть нашей задачей является изображение нескольких однотипных записей, каждая из которых имеет вид (используется графовая модель RDF), изображенный на рисунке 5:

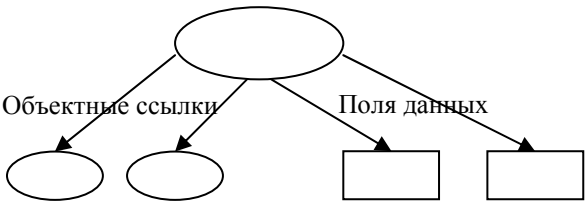

Рис. 5. Представление отдельной записи в виде фрагмента RDF-графа

При наличии заданной онтологии, однотипность означает единый набор возможных или имеющихся полей-данных и прямых ссылок. Сопоставим рассматриваемым записям таблицу, строками которой будут информационные поля записей, а колонками – поля и прямые ссылки. Поля-данные, помещаются в ячейки таблицы непосредственно, прямые ссылки логично поместить через их inline-

представление, например в виде гиперссылки.

Метка типа				
Метка поля 1	Метка поля 2	…	Метка прямой ссылки 1	…
Значение поля 1	Значение поля 2	…	Inline-представление ссылки 1	…
…				

Таблица 1

Обозначим (формулы приводятся в формализме XPath):

$type – тип элементов набора,

$type_id – идентификатор этого типа,

$items – набор записей этого типа,

$ontology – множество записей определения онтологии.

Дополнительно вычислим два промежуточных множества:

$dprops = $ontology/DatatypeProperty[domain/@rdf:resource=$type_id] – описание полей-данных,

$oprops = $ontology/ObjectProperty[domain/@rdf:resource=$type_id] – описание полей-ссылок.

Теперь рассмотрим каноническое представление информационной единицы (записи) в которой к записи добавляются через объектные ссылки еще другие элементы, оно изображено на рисунке 6.

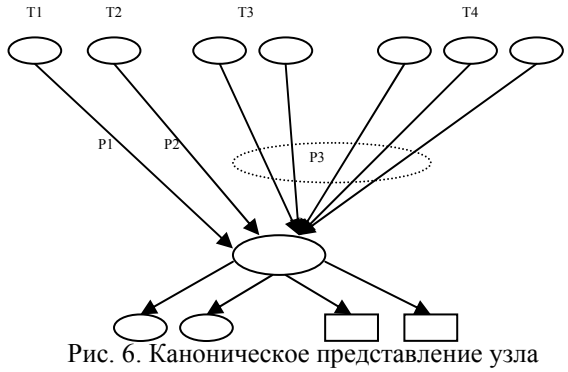

Рис. 6. Каноническое представление узла семантической сети RDF, соответствующего информационной единице (записи)

Легко видеть, что, если отсутствуют объектные ссылки узлов самих на себя, то любой узел семантической сети RDF представляется в представленном на рисунке 6 каноническом виде: в центре располагается опорная сущность, внизу имеются поля-данные (DatatypeProperty) и прямые объектные ссылки (ObjectProperty), сверху изображены объектные ссылки (отношения) других информационных единиц, ссылающиеся на данную. Обратные отношения для элемента сгруппированы по типу отношения и по типу класса элементов, ссылающихся на данный.

Получившиеся в результате разбиения подграфы укладываются в модифицированный вид рассмотренной таблицы (Таблица 2).

Где в первую строку перед меткой типа добавлено название общего для группы объектного свойства. Соответственно, переменная $oprop обозначает это свойство (ObjectProperty). Кроме того, для обратных отношений для колонок должно выполняться следующее условие:

$oprops[i] != $oprop.

Это означает, что не нужно выдавать информацию о записи «из которой» мы пришли к построению таблицы.

Имеющиеся в таблицы выражения теперь надо рассматривать в контексте групп записей, к которым таблица относится. Обозначим функцию, строящую таблицу по указанным формулам:

ViewTable(prop_id, type_id, items).

Если проанализировать каноническое представление информационной единицы, то в контексте вышесказанного можно увидеть, что целостное информационное представление информационной единицы состоит из N + 1 таблицы, где N – количество вариантов свойство-тип для обратных отношений. Первая таблица является вырожденной, имеющей одно и только одно описание информационной единицы – собственно базовое описание. И для этой таблицы не определено значение prop_id. Остальные таблицы могут изображать ноль или более элементов определенного типа, ссылающихся на базовую единицу через указанное отношение.

Это представление опишем формулами. Пусть имеется идентификатор item_id записи, которую целостно мы хотим изобразить. И пусть data_base есть множество записей нашей базы данных. Тогда первая таблица строится в виде:

```
$items = $data_base/*[@rdf:about=$item_id]
ViewTable(null, name($items), $items)
```

А остальные порождаются в цикле:

```
$range_set = $ontology/ObjectProperty[
    range/@rdf:resource=name($items)]
$inverse_links =
    $data_base/*/*[@rdf:resource=$item_id]
Foreach $inverse_prop in $range_set do
{
    $inverse_type_set =
    $inverse_prop/domain/@rdf:resource
    Foreach $inverse_type_id in
    $inverse_type_set do
{
       ViewTable( name($inverse_prop),
    $inverse_type_id,
    $inverse_links[name()=name($inverse_prop)
    and name(parent::*)=$inverse_type_id])
}
}
```

В этом псевдокоде не учтено наследование свойств для классов. Это сделано для того, чтобы не загромождать выражений и не меняет ситуации по существу.

По предложенной схеме был выполнен ряд разработок. В частности, рассматриваемый операторский интерфейс редактирования базы данных, используемый в проекте «Фотоархив СО РАН» имеет практически именно предложенный вид).

$oprop/label : $type/label				
$dprops[1]/label	$dprops[2]/label	…	$oprops[1]/label	…
$items[1]/*[name()= $dprops[1]]/text()	$items[1]/*[name()= $dprops[2]]/text()	…	ViewItemInline($items[1]/*[name()=$oprops[1]]/@rdf:resource)	…
…				

Таблица 2.

5 Приложение Fact-o-graph для ведения малых (фото) документных архивов

Средства, представленные в предыдущем разделе, для пользователя достаточно громоздки для простых применений. Они требуют наличия Web-сервера, интерфейс универсален, а потому не достаточно лаконичен, средства работы с фотодокументами отделены от операторского интерфейса и составляют целую технологическую цепочку. В связи с этим, для простых проектов и каждодневного использования, было спроектировано приложение Fact-o-graph.

Фактограф работает на том же поле данных, что и другие компоненты архивной системы и можно его использовать для доступа и редактирования данных и в больших проектах и совместно с интерфейсами, описанными в предыдущем разделе. Собственно это определяется тем, что ядро (см. ранее) фактографа то же самое, что и в интерфейсах. Это несколько «утяжеляет» использование приложения на пользовательском компьютере, поскольку на нем формируется полная модель данных проекта, но в большинстве случаев, это не существенно.

Рассмотрим модель данных и их распределения, использованную при создании фактографа.

База данных состоит из отдельных файлов – документов RDF. Все документы RDF соответствуют единой спецификации – базовой онтологии. Некоторые подмножества документов могут иметь расширенную спецификацию. Онтология и ее расширения задаются средствами OWL-описаний. У каждого документа есть владелец и только владелец может вносить в документ изменения. Модель устройства и функционирования такой базы данных рассмотрена в работе [2]. Документы базы данных могут быть общими и приватными. Общие документы публикуются в открытом для внешнего интернетовского доступа месте, приватные – сохраняются на компьютере пользователя. В базу данных включены также произвольные файлы документов: фотодокументы, видео, аудио, doc, txt, pdf, rtf, html и прочие текстовые и гипертекстовые документы. Все документы, включая RDF-документы, группируются в кассеты (подробнее об этом – см. далее). Собственно признак общности или приватности относится к кассетам, а не к отдельным документам.

Интерфейс фактографа сформирован на основе достаточно традиционного трех-панельного интерфейса, применяемого например в Windows Explorer. Две панели свойств располагаются в левой части плоскости окна приложения, одна – в правой. Левая верхняя панель дает информацию о «рассматриваемой» записи, на правой панели располагается список простых и сложных отношений между записью и другими записями. Левая нижняя панель активируется когда выделяется конкретное отношение для выдачи свойств этого отношения. Список реализует традиционный формы представления списка и его элементов в виде таблицы, с большими или маленькими иконками, в виде детальной композиции.

Эта же форма интерфейса сохраняется для случая поисковых запросов. При этом, левая верхняя панель используется для формулирования запроса (задания имени объекта и некоторых его свойств), в правой панели мы получаем результаты поиска в виде списка найденных записей. При выделении элемента списка, расширенная информация о нем появляется в виде списка свойств в левой нижней панели.

Также как и для Windows Explorer, можно породить несколько экземпляров окна приложения, реализованы операции drag and drop. Интерфейс системы интуитивно понятен пользователю, легко осваивается и удобен в применении.

Существенным слоем фактографа является работа с документами. Любой файл из системной области Windows можно поместить в архив, заполнить его учетную карточку, связать его с элементами базы данных. При этом файл копируется в активную кассету, при необходимости производится его предобработка для обеспечения большей удобности работы с файлом в Интернете. Собственно архив представляет собой совокупность кассет, содержащих документы и часть базы данных. Обеспечение дальнейшего жизненного цикла кассет (сохранение, резервное копирование, общее администрирование) возлагается на произвольную систему сохранения данных (хранилище данных).

На рисунке 8 приведен скриншот окон фактографа с некоторыми элементами визуального интерфейса.

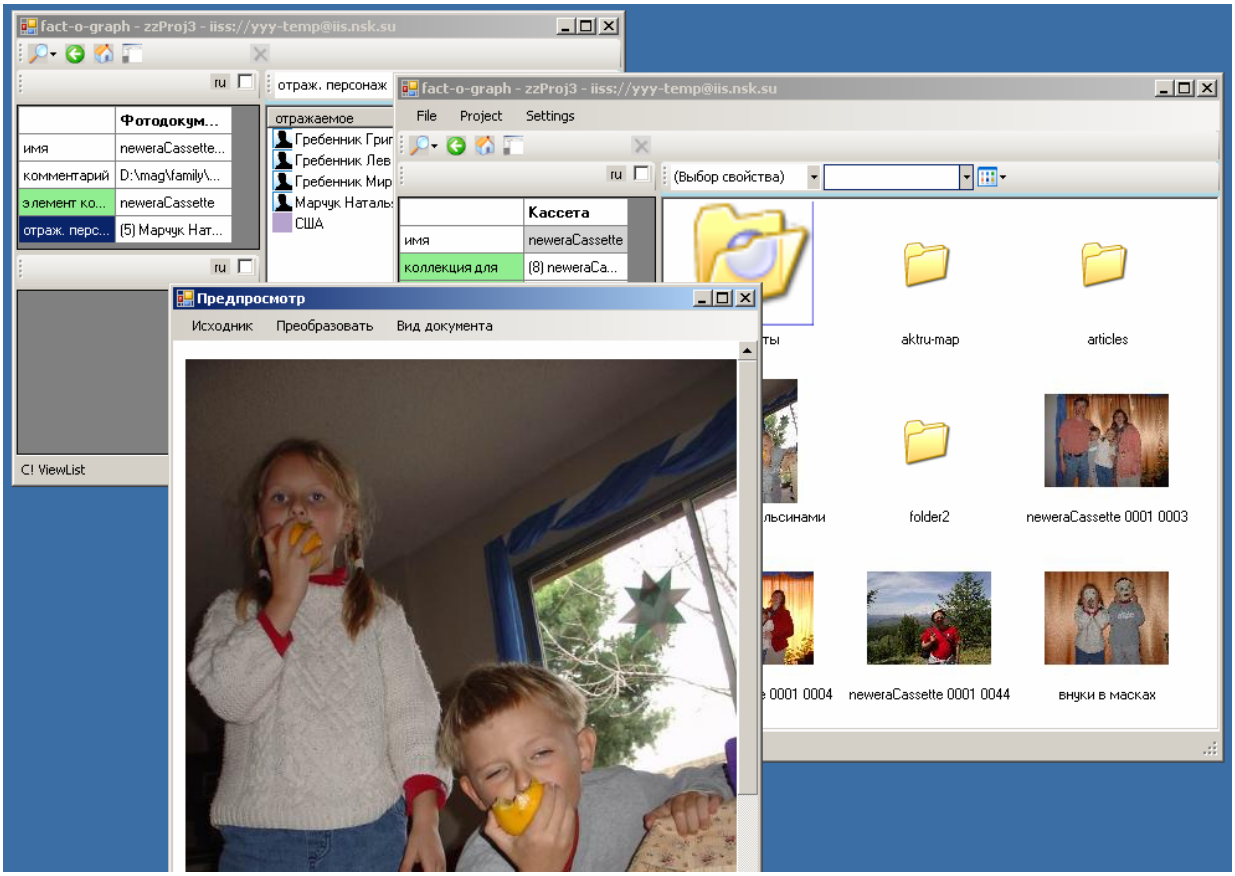

Рисунок 8

6 Обеспечение синхронизации редактирования в распределенной конфигурации

Как уже указывалось, в конкретной конфигурации одновременно могут быть запущены различные серверные и пользовательские приложения, разворачивающие и поддерживающие модель данных. Модель данных формируется из подмножества RDF-документов, соответственно одни и те же RDF-документы могут одновременно использоваться в разных местах. Это одновременное использование означает то, что при запуске системы документы используются приложениями для формирования фрагментов базы данных. Если документ не изменяется, то мы можем быть уверенными, что этот фрагмент идентичен для разных приложений. Теперь предположим, какой-то RDF-документ редактируется его владельцем. Ядро приложения будет обеспечивать синхронизацию фрагмента с документом путем записи новых состояний фрагмента в файл. Но теперь фрагмент перестанет быть идентичным для разных приложений. Производить слишком частую перезагрузку данных – решение неприемлемое, поэтому нужен другой механизм для обеспечения идентичности моделей.

Анализ показал, что мгновенная синхронизация, т.е. синхронизация с блокировкой процессов («остановка» времени), не требуется для рассматриваемых задач. Вполне достаточными и приемлемыми являются задержки в синхронизации на минуты и десятки минут для передачи изменений и одной модели в другие. В принципе, таймер для выполнения синхронизационных действий можно установить на малые интервалы времени, тогда можно реализовывать интерактивные сценарии типа обмена сообщениями и коллективных обсуждений.

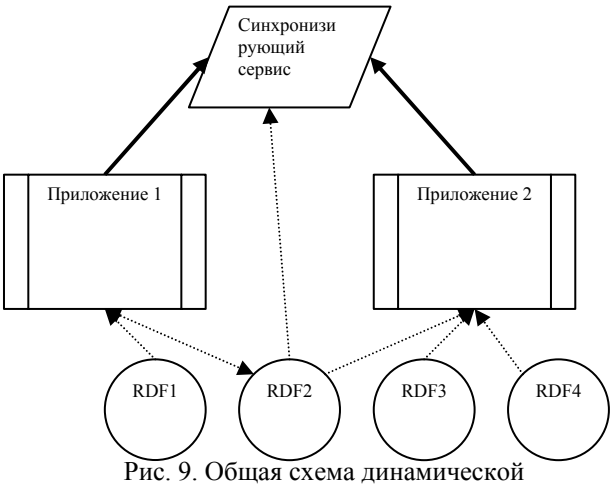

Рис. 9. Общая схема динамической синхронизации редактируемых моделей

Рассмотрим механизм динамической синхронизации моделей, содержащих одинаковые сегменты данных. На рисунке 9 изображены два

приложения, в которых имеется общий RDF-документ (RDF 2), причем приложение 1 этот документ может изменять. Пунктирными стрелками изображены связи приложений с файлами документов. Приложения читают файлы документов при инициации модели, владелец документа изменяет в приложении соответствующий образ документа и ядро приложения (через какое-то время) записывает образ в файл. Как уже указывалось, такая схема осуществляет слишком «медленную» синхронизацию – модели получают изменения только при загрузке. Поэтому вводится еще один активный участник – синхронизирующий сервис.

Сервис также, при инициации, строит у себя модель данных. Теперь пусть приложение 1 внесло изменения в образ документа 2. Это изменение сразу же пересылается «наверх» - синхронизирующему сервису. Посылается, естественно, не весь измененный документ, а только выполненное изменение. Сервис, в свою очередь, фиксирует произведенные изменения в своей модели, проставляя отметки времени на этих изменениях. Приложение 2, по таймеру или по другому «толчку», делает запрос сервису. Смысл запроса следующий: дай мне все изменения документа RDF 2, произошедшие с момента времени t. Где t, условно говоря (опустим детали) – время предыдущей синхронизации по данному документу. Синхронизирующий сервис посылает известные (пришедшие) ему изменения в данном документе, а приложение 2 фиксирует эти изменения в своей модели.

Реальный алгоритм синхронизации несколько сложней, но принципиально – он такой. Теперь достроим сервис возможностью посылать (транслировать) пришедшие от приложений изменения вышестоящему сервису и транслировать вышестоящему сервису запросы по тем документам, которые сервис «не обслуживает», соответственно возвращать ответ от вышестоящего сервиса своему клиенту. После такой достройки, можно создавать гибкие иерархические синхронизирующие конфигурации сервисов и приложений.

Данная схема синхронизации была реализована в архивной фактографической системе и используется в конфигурациях, где применяется приложение Fact-o-graph.

7 Технология кассет данных

Рассмотрим запись, соответствующую некоторому документу. Важнейшей составляющей записи является указание на собственно файл документа. Запись могла бы выглядеть следующим образом:

```
<document rdf:about="идентификатор
    документа">
    <coordinate>URL-документа</coordinate>
</document>
```

В этом варианте указания на файл документа, мы жестко фиксируем его координату и не позволяем альтернатив. Кроме того, все наши документы должны быть опубликованы и доступны только через серверные средства. Покажем как подход, изложенный в статье [2] может быть реализован для порождения более гибкого решения данной задачи.

Не умаляя общности, предположим, что все документы, зарегистрированные в базе данных, располагаются группами, которые мы назовем кассетами. Для каждой кассеты мы зададим ее уникальное логическое имя. А координата файла будет формироваться как конкатенация (в расширенном случае – как некоторая функция) расположения кассеты и относительного местоположения (адреса) файла в кассете.

```
file_coord = cassette_location +
    file_relative_position
```

Теперь, если мы заменим задание координаты в документе, на пару: имя кассеты, относительный адрес файла,

```
<document rdf:about="идентификатор
    документа">
    <cassette-name>имя кассеты</cassette-
    name>
    <file-relative-position>позиция
    файла</file-relative-position>
</document>
```

то мы легко вычислим текущую координату файла, имея информацию о текущей координате кассеты. Поэтому, в структуру настройки модели должны быть включены записи вида:

```
<cassette name="имя
    кассеты">cassette_location</cassette>
```

задающие таблицу текущего соответствия имен кассет их расположению.

Теперь любой, присутствующий в базе данных документ доступен из приложения, поддерживающего некоторую модель, если для этой модели имеется описание документа и если для кассеты этого документа зафиксирована координата.

Связь базы данных с пространством публикации документов будет не полной, если не дать механизма фиксации локализации кассет. В этом плане, проблему представляет множественность вариантов изображения координат файлов для разных случаев расположения файлов. Действительно, файлы могут располагаться в регулярном интернетовском пространстве, могут оказаться в локальном дисковом пространстве конкретного компьютера или на оптических дисках. Возможно даже инкапсулированное существование файлов с организацией доступа через серверный запрос. Кроме того, естественно, что к одному и тому же файлу могут вести различные адресные пути.

Рассмотрим задачу конфигурирования приложения. Точнее, нас будет интересовать только та часть, конфигурирования, которая относится к модели. Модели надо указать несколько моментов: какие RDF-документы должны быть загружены в базу данных модели, какие и как определены кассеты. Естественно, что максимально большую

часть конфигурационной информации надо погрузить в базу данных.

Во-первых, нам нужна декларация того, что существует некоторый RDF-документ, во-вторых, нам требуется путь доступа к нему, в-третьих, нужно указание необходимости или целесообразности загрузки данного документа в модель. Последнее вряд ли относится к содержимому базы данных, но остальные указания, можно превратить в запись в базе данных вида:

```
<RDF-doc rdf:about="идентификатор RDF-
    документа">
   Поля и ссылки элемента
</RDF-doc>
```

Если мы будем считать RDF-документы подклассом документов, то техника указания места публикации сохранится та же. Таким образом, если заложить такую особенность документов, как инициирование загрузки этого документа при его упоминании в базе данных, то можно считав один документ, всю остальную совокупность загружаемых документов определить по базе данных.

Тут есть одна проблема. При распределенном редактировании, мы позволяем отдельным записям «мигрировать» из одних документов в другие, это может породить нестабильный характер загрузки модели и ее неработоспособность. Другой проблемой является то, что не все RDF-документы могут оказаться совместимыми. Это, конечно, нарушает принципы единой распределенной базы данных, но практически может оказаться целесообразным. Соответственно, желательно прямо указывать конфигурационные наборы документов, требуемы для достижения тех или иных целей.

Кассеты и работа с ними, в достаточно целостном виде реализована и использована в приложении Fact-o-graph.

8 Заключение

Рассмотренная система компонентов и технологических решений обладает хорошей гибкостью и достаточно удобна для практического применения. Комбинируя компоненты в конфигурации, можно порождать проектные решения в спектре от индивидуальной малой архивной системы до большой системы, интегрирующей ряд проектов, имеющей протяженные пространственные и временные рамки. С помощью архивной фактографической системы или ее элементов были выполнены следующие проекты: Фотоархив СО РАН, Информационная система кафедры программирования ММФ НГУ, База данных Летних школ юных программистов, Хроника Сибирского отделения, Исторический портал ММФ, некоторые другие.

Литература

[1] Марчук А.Г., Марчук П.А. Платформа интеграции электронных архивов // Электронные библиотеки: перспективные методы и технологии, электронные коллекции. Труды девятой всероссийской конференции. Переславль, 2007, с. 89-94.

[2] Марчук А.Г. О распределенных фактографических системах // Электронные библиотеки: перспективные методы и технологии, электронные коллекции. Труды десятой Всероссийской конференции, Дубна, 2008, с. 93-102.

[3] Web Ontology Language (OWL). — http://www.w3.org/2004/OWL

Archival Factographic System

A.G. Marchuk, P.A. Marchuk

New information system for archives was build in the Institute of Informatics Systems. System is based on factographic principles, its main features are: distributed database formed as a set of RDF-documents, special realization for distributed editing and dynamic synchronization, use of formal data specifications by OWL descriptions. System provides flexibility of data structuring, possibility of implementation of different ontologies, particularly, different metainformation set of fields. Several projects were fulfilled using proposed information system and proposed technilogies.