Реализация некоторых методов обработки сложно организованных данных в электронных библиотеках

© В.В. Рыбинкин
ООО «Бюро Интернет Технологий БИТ»
ryb@2bit.ru

Аннотация

В работе предложен алгоритм и проведена первичная структуризация неоднородных данных сложной структуры, характерных для электронных библиотек. Изложены основные концепции формирования графовой модели данных, рассмотрена организация доступа к данным навигационным путем, выполнена модификация схемы данных в процессе тестовой эксплуатации электронной библиотеки. Изложенные подходы практически реализованы с применением графовой СУБД «Синдбад».

1 Введение

При разработке программного обеспечения электронных библиотек приходится решать множество различных задач: выборка и визуализация данных, интеграция неоднородных, сложно структурированных информационных ресурсов, персонализация пользовательского интерфейса, поддержка многоверсионности хранимых материалов, модификация структуры данных в процессе эксплуатации системы, балансировка нагрузки на сервер и т.д. [4].

Целью данной работы является попытка решения задач по первичной структуризации слабоструктурированной текстовой информации при помощи разработанной в нашей компании СУБД «Синдбад», поддерживающей сетевую модель данных. Одновременно решались задачи организации доступа к данным навигационным путем (без формирования поисковых запросов), реструктуризации БД в процессе эксплуатации. Проведенный в рамках Российского семинара по оценке методов информационного поиска [5] анализ применения подсистемы ввода данных СУБД «Синдбад» для рубрикации нормативно-правовых документов дал обнадеживающие результаты и стимулировал проведение более детальных исследований на той же коллекции документов из электронной библиотеки, предоставленной компанией «Кодекс» [6].

Труды 7ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2005, Ярославль, Россия, 2005.

2 Первичная структуризация коллекции правовых документов

2.1 Технические характеристики

Объем коллекции «Кодекс» составил 66995 нормативно-правовых документов. В качестве платформы был выбран компьютер Pentium с тактовой частотой 266МГц (64 Мб оперативной памяти), операционная система Windows-98. В качестве языка программирования использовался язык C. Компиляция утилит пакетной обработки данных выполнялась с помощью транслятора BC++ v3.1, компиляция серверных скриптов для доступа к базе данных – транслятором MS VC++ v6.0. В качестве клиентской части применялся броузер MS IE 5.00.

2.2 Используемые термины

Граф: графовая база данных.

Узел графа: именованная структура данных.

Имя узла: уникальный идентификатор (ID) узла.

Ребро графа: одно из полей узла типа «ссылка», которое содержит ID узла, связанного с данным.

Измерение: именованная характеристика ребра (тематическая, географическая, алфавитная и т.д.).

Размерность узла: число измерений в узле.

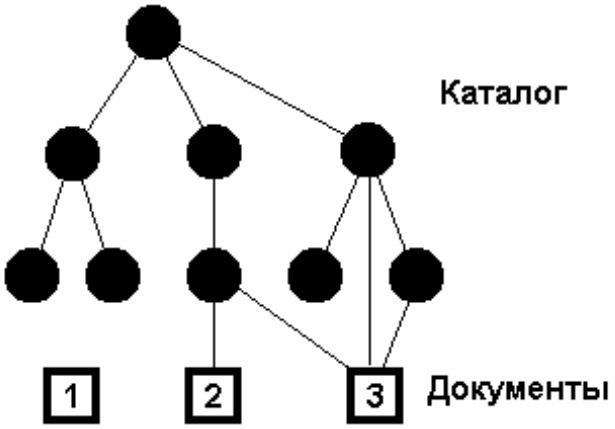
Размерность графа: число измерений в графе.

Многомерный граф: граф с размерностью более единицы, т.е. ребра графа расположены в разных измерениях.

Распределенный граф: граф, узлы которого размещены на нескольких серверах.

Тег узла: набор полей узла графа, содержащих служебную информацию о структуре этого узла.

Рис. 1. Формирование связей документов с рубриками каталога



2.3 Структуризация данных

Исходными данными для импорта являлись иерархический каталог и коллекция нормативно-правовых документов электронной библиотеки, предоставленной компанией «Кодекс». Алгоритм первичной структуризации импортируемых данных и формирования связей между ними по набору ключевых слов был аналогичен алгоритму рубрикации данных [5]. При этом выбор ключевых слов, по которым формировались связи, был достаточно произвольным, т.к. мы не стремились получить базу данных, пригодную для практического применения – это задача для специалистов в области права. Так, элементы рубрики «Природные ресурсы и охрана окружающей среды» должны были обладать набором следующих ключевых слов: ((окружающ & сред) | (ресурс & природ)) & охран

Одновременно с формированием связей по рубрикам иерархического каталога (тематическое измерение) проводилась рубрикация по отдельным ключевым словам (алфавитное измерение), году создания документа (хронологическое измерение), изменениям и дополнениям к существующим документам (версионное измерение).

После проведения первичной структуризации практически все документы получили связи с тематическими рубриками каталога. Нередко связи образовывались сразу с несколькими рубриками, находящимися на разных уровнях иерархии (рис. 1). Так, документ «Методические рекомендации по бухгалтерскому учету затрат, включенных в издержки обращения и производства, и финансовых результатов на предприятиях торговли и общественного питания» приобрел связи сразу с пятью тематическими разделами: «Торговля», «Общественное питание», «Основы организации бухгалтерского учета», «Учет затрат на производство», «Учет финансовых результатов». Документ «В дополнение к письму N 778 от 13.05.1994 г., а также распоряжению N 37 от 02.12.1994 г.», напротив, не был приписан ни к одной тематической рубрике, но получил привязку по хронологическому и версионному измерениям. Некоторые документы, такие как «Ламизил спрей (Тербинафин)» или «О КДП N 3» вообще не получили ни одной связи ни по одному измерению (рис. 1) - всего 3228 документов. Среднее число образованных связей для каждого элемента составило 4.18, максимальное - 21.

Для интеграции с данными иерархического каталога, последний был просто добавлен в общую базу данных как набор текстовых элементов при сохранении имеющихся иерархических связей. Полученная структура данных содержала лишь однонаправленные связи (архивный формат БД). При импорте данных из архивного формата в формат графовой БД программным обеспечением СУБД «Синдбад» выполняется формирование двунаправленных связей, после чего библиотека «Кодекс» образует циклический многомерный граф, имеющий в данном случае четыре измерения: тематическое, хронологическое, алфавитное и версионное.

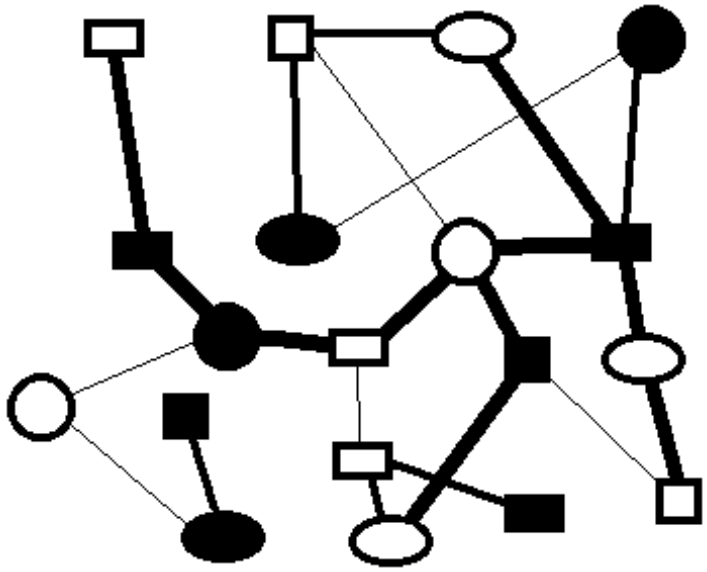
При описании функционирования электронной библиотеки «Кодекс» в формате графовой базы данных, организации доступа к данным, редактирования узлов и ребер, реструктуризации БД в целом, мы будем в необходимых случаях более подробно рассматривать особенности СУБД «Синдбад».

3 Графовая СУБД «Синдбад»

3.1 Общие положения

База данных в формате «Синдбад» в общем случае представляет собой распределенный многомерный циклический смешанный граф с узлами произвольной структуры. В отличие от подхода CODASYL [1], допускаются сложные сети, включающие отношения многие-ко-многим. Еще одно отличие заключается в том, что ребра графа представлены не указателями, а именами, по

Рис. 2. Представление неоднородной распределенной БД в виде графа



которым осуществляется прямой доступ к узлам, поэтому модификация данных, изменение структуры БД, не требуют изменения указателей. Операции манипулирования данными возможны не только над отдельными узлами, но, как и в реляционной модели данных, над группами узлов или графом в целом. Аналогично разработанной специально для полуструктурированных данных СУБД LORE (Lightweigth Object REpository) [3], «Синдбад» не требует априорного знания схемы данных. Как схема данных, так и их структура не являются фиксированными и могут многократно изменяться в процессе эксплуатации. Никаких специальных требований к аппаратуре, клиентскому и серверному программному обеспечению не выдвигается.

3.2 Типы данных

На рис. 2 приведено графическое изображение БД «Синдбад». Форма узлов иллюстрирует различие структур данных элементов БД, цвет узлов обозначает их принадлежность одному из серверов базы данных, толщина ребер обозначает различные измерения графа.

Узел графа (рис. 3) есть структура данных, каждое поле которой является элементом одного из **базовых типов** данных (целое, вещественное, строка, дата, ссылка и т.п.). Эти поля по функциональному назначению объединяются в три группы. **Служебные поля** (ID и тег узла) содержат данные о структуре узла и используются СУБД для идентификации типов данных и обеспечения корректной работы с неоднородными данными. **Информационные поля** содержат данные, предназначенные для конечного пользователя. **Поля окружения** всегда имеют тип «ссылка» и представляют связи текущего узла с другими узлами графа по различным измерениям.

Кроме базовых типов, методы обработки которых (сложение, инверсия, сравнение и т.п.) априори известны СУБД, имеется возможность определения пользователем иных, **реляционных** или **графовых** типов данных как комбинации базовых и определенных ранее типов. В случае электронной библиотеки «Кодекс» после импорта данных информационная часть каждого узла будет содержать лишь одно поле базового типа «строка». Предполагая, что работа с БД может потребовать наличия более разнообразных структур данных, были определены несколько дополнительных типов данных. При этом сами данные в БД пока отсутствовали.

Реляционный тип данных представляет собой именованную группу полей одного узла. Так, тип **FIO** был определен состоящим из трех полей базового типа «строка» **FIO** ФИО (**STRING** Фамилия, **STRING** Имя, **STRING** Отчество). Несколько более сложный реляционный тип **REGION** состоит уже из неоднородных полей: **REGION** REGION (**STRING** NAME, **ID** TYPE). Выбор английского названия типа иллюстрирует тот факт, что это название вообще не предполагается выводить конечному пользователю, оно может совпадать с описанием типа или иметь неопределенное значение. Отметим, что тип региона представляет собой ссылку на узел типа «строка», содержащий название типа (улица, город, страна), но тип элемента по ссылке не указывается, т.к. СУБД самостоятельно определит его по тегу узла. Кроме того, ссылка расположена не в полях окружения, а в информационных полях узла, что дает возможность обращения к этому элементу по имени поля (TYPE). Еще одна ссылка на

Рис. 3. Структура узла графа

родительский регион в описании типа отсутствует, поскольку она находится в полях окружения для обеспечения навигации по новому, географическому измерению.

Еще более сложный, графовый тип данных представляет собой именованную группу полей разных узлов. Так, графовый тип **ADDRESS** описан как группа полей нескольких связанных узлов графа: **ADDRESS** Адрес (**UI32** Индекс, **STRING** Дом, **UI16** Квартира, **ID** REGION, **REGION.NAME** Улица, **REGION.PARENT.NAME** Город, **REGION.PARENT.PARENT.NAME** Страна). Иными словами, данные типа **ADDRESS** расположены в четырех узлах графа, при этом обращение к полям этих узлов осуществляется через одну и ту же ссылку с именем REGION и последующей навигацией по географическому измерению (PARENT). Обратим внимание, что для поля «Дом», в отличие от поля «Квартира», был выбран тип **STRING**, а не **INTEGER**, поскольку номер дома может содержать корпус, строение и т.п.

Все остальные типы данных определены нами как реляционные:

- **PERSON** Person (**ADDRESS** Адрес, **FIO** ФИО, **STRING** Телефон, **STRING** Должность)
- **COMPANY** Организация (**ADDRESS** Адрес, **PERSON** Руководитель, **STRING** Телефон, **STRING** Факс, **STRING** e-mail)
- **DOCUMENT** Документ (**STRING** Название, **ID** TYPE, **STRING** Описание, **DATE** Дата, **PERSON** Автор)

Таким образом, графовый тип **ADDRESS** является элементом реляционных типов **PERSON** и **COMPANY**, т.е. различия между базовыми, реляционными и графовыми типами на уровне пользователя не делается. Обработка вложенных типов данных осуществляется рекурсивно.

3.3 Принципы построения СУБД «Синдбад»

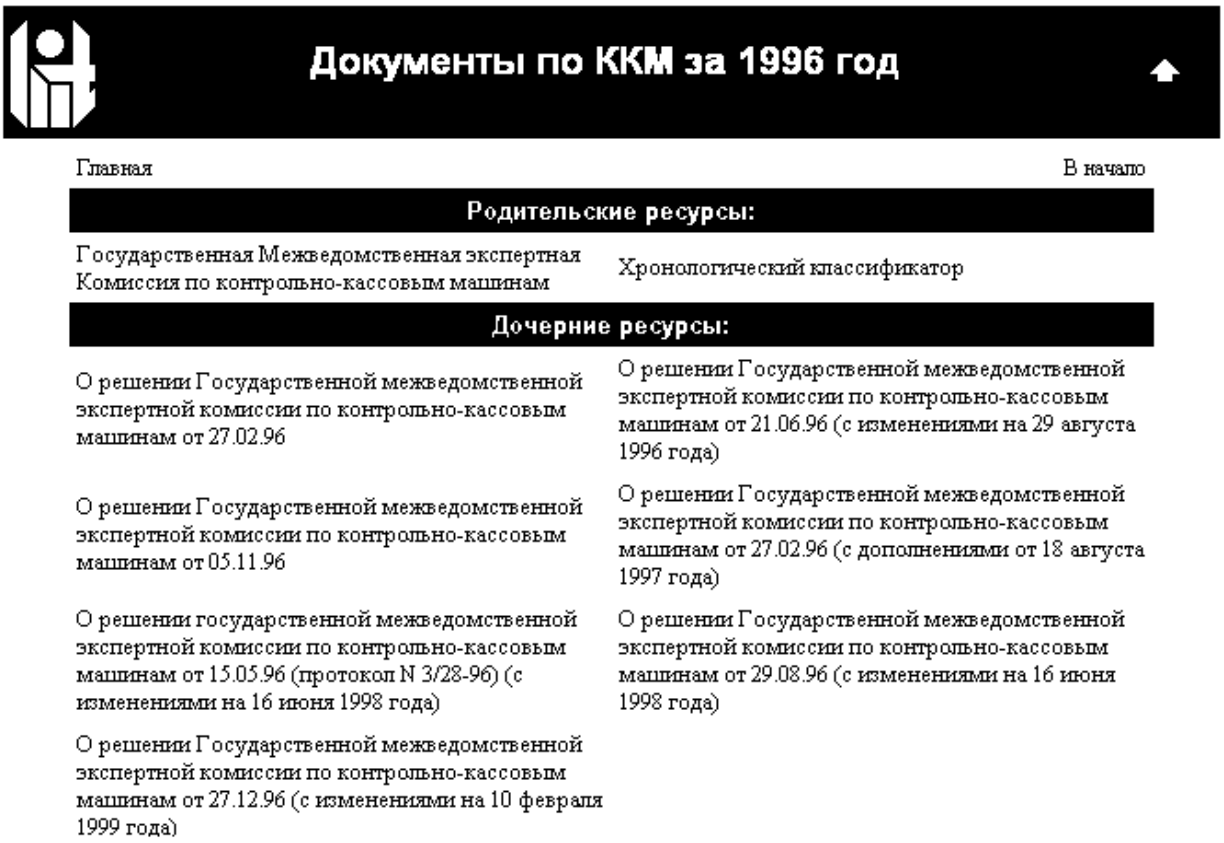
Реализация прямого доступа к данным узлов осуществлена по принципу косвенной адресации. Каждый узел обычно состоит из двух частей: обязательная составляющая постоянной длины служит для определения его физического адреса по принципу индекса. При необходимости, узел имеет составляющую переменной длины, адрес которой указан в одном из полей первой части. Такая организация, помимо прямого доступа, допускает свободное изменение размера узла, в частности, изменение его структуры и количества связей с другими узлами. Этот принцип позволяет работать с полями произвольной длины (именно такие элементы представлены в коллекции).

Для обеспечения возможности работы с разнородной информацией по тем же алгоритмам, что и с однородной, вводится понятие тег узла, что дает возможность определять тип элементов узла, порядок их следования «по месту», считывая и интерпретируя информацию тега. Иначе говоря, данные являются самоописываемыми, включая типы данных, определяемые пользователем. Описание стандартных типов данных хранится в теле СУБД. Типы, определяемые пользователем, хранятся в **паспорте базы данных** - специальном элементе графового типа, о котором будет сказано ниже. Описание незарегистрированных типов данных находится непосредственно в полях тега узла. Теоретически возможно создание базы данных, каждый элемент которой имеет свой уникальный тип, т.е. объем схемы данных соизмерим с объемом самих данных. Этот принцип позволяет работать с данными произвольной структуры.

Использование тега снимает ограничения и на максимальное количество элементов в базе данных, т.к. размер поля типа «ссылка» тоже может быть переменным. В частности, могут быть ссылки на узлы, расположенные на других серверах распределенной базы данных (небольшой объем представленной коллекции не требовал создания распределенной базы данных). Для узлов сложной структуры тег может иметь переменную длину (минимально 1 байт), что определяется установкой его служебного бита (тег тега). Тег не имеет продолжения, если тег его очередного байта сброшен. Этот принцип позволяет сколь угодно сложное описание узла.

Введение понятия графового типа данных является принципиально важным. При обработке элемента графового типа **ADDRESS**, при полностью аналогичной входной форме для пользователя, действия СУБД будут радикально отличаться от обработки его реляционного аналога. Одна команда пользователя «сохранить изменения»

Рис. 4. Представление узла графа вместе с окружением



вызывает, как правило, замену значений сразу в нескольких взаимосвязанных узлах базы данных. Внесение изменений может сводиться к проверке соответствия заносимых данных и данных узлов БД. Так, при заведении одного нового элемента типа **ADDRESS** реально в БД могут быть внесены от одного до четырех узлов, в зависимости от наличия или отсутствия в базе данных узлов, содержащих поля типа **ADDRESS** (Улица, Город, Страна). Отметим, что сам факт графового типа **ADDRESS** уже позволяет выявлять логические ошибки в данных: СУБД не позволит внести неверные данные сообщением, вроде: «В этом городе такой улицы нет». Этот принцип позволяет работать с группами узлов произвольной конфигурации.

Доступ из языков программирования высокого уровня в настоящее время не предусматривается. Концептуально это означает, что интерфейс программиста выполняется на уровне языка команд шаблонов выходных форм, включая команды доступа к элементам базы данных, их редактирования и поиска. В дальнейшем, при разработке полномасштабного сервиса, предполагается поддержку возможностей, специфичных для работы с СУБД, осуществлять библиотечными элементами, а включение языка доступа в язык программирования - средствами препроцессора.

Клиентская часть не является составной частью СУБД «Синдбад». В качестве клиентской части используется стандартный веб-броузер, а в качестве средства обеспечения доступа к СУБД - язык JavaScript. Серверная часть СУБД разработана в двух вариантах: облегченная (на языке Perl как одном из самых популярных платформо-независимых языков) и полная (на языке C), содержащая ряд дополнительных функций, реализация которых невозможна либо слишком трудоемка на интерпретирующмх языках.

4 Создание графовой базы правовых документов

В момент создания, перед импортом данных, в БД заносится единственный элемент - паспорт графа. Это специальный элемент графового типа, который содержит информацию о БД в целом: название базы, описание пользовательских типов данных, признаки неопределенного значения данных и другую служебную информацию, необходимую СУБД для работы с конкретной базой данных и доступную только для системного программного обеспечения СУБД. Паспорт базы является первым элементом БД, поэтому его физический адрес не вычисляется. В настоящее время редактирование паспорта производится при помощи текстового редактора системным администратором и, следовательно, требует

досконального знания его структуры. Необходима доработка системного ПО СУБД для обеспечения возможности редактирования паспорта в диалоговом режиме (такая потребность возникает, например, при определении новых типов данных в процессе эксплуатации системы).

После создания паспорта базы данных, программным обеспечением СУБД «Синдбад» был осуществлен импорт данных электронной коллекции «Кодекс» из архивного формата в формат графовой БД с одновременным формированием двунаправленных связей. Процесс импорта завершает создание графовой базы правовых документов.

5 Работа с графовой базы правовых документов

5.1 Организация доступа к данным

На рис. 4, в виде веб-страницы, показан один из узлов полученного графа вместе с его окружением. Фрагмент статической версии базы данных, содержащий этот узел, представлен на нашем сайте [7].

Все узлы окружения, показанные на рис. 4, представлены в виде гиперссылок, что позволяет осуществлять навигацию по любому измерению графа единообразно, с помощью веб-броузера, в т.ч. навигацию по различным версиям и дополнениям одного документа (версионное измерение). Изображенный узел содержит документы по контрольно-кассовым машинам за 1996 год (хронологическое измерение), являющиеся одновременно дочерними к узлу «Государственная экспертная Комиссия по ККМ» (тематическое измерение), который, в свою очередь, имеет как тематические родительские связи («Бытовое обслуживание населения», «Социальная защита населения», «Общественное питание»), так и связи по алфавитному измерению («ККМ»). Показанный на этом же рисунке узел «О решении Государственной межведомственной экспертной комиссии по контрольно-кассовым машинам от 27.02.96», имеет дочернюю связь по версионному измерению с его продолжением (дополнения от 18 августа 1997 года). Отметим, что как размерность узла, так и количество его ребер могут быть произвольными, в т.ч. нулевыми, если связи узла не заданы либо удалены. Таких улов после импорта данных в БД оказалось 3228.

Использование в качестве клиентской части веб-броузера практически снимает вопросы шрифтового и цветового оформления, удаленного взаимодействия с сервером, представления графической информации, организации многооконного интерфейса, работы с распределенной СУБД. Броузер «автоматически» предоставляет средства навигации в виде гиперссылок, обеспечивает возможность распечатки документов. Добавим сюда привычность и возможность индивидуальной настройки интерфейса пользователя, легкость его перенастройки. Использование фреймовой структуры позволяет организовать итерационный обмен данными с сервером, используя при необходимости невидимые фреймы. Обмен информацией между фреймами осуществляется объектами JavaScript.

Навигация непосредственно по ребрам графа позволяет конечному пользователю получать информацию вообще без формирования поискового запроса. Интересно отметить, что для конечного пользователя работа с многомерным графом оказывается даже проще, поскольку появляется возможность группировки ребер по измерениям и фильтрации измерений по тематике, территориальному расположению, алфавитной близости и т.д. Кроме того, в СУБД «Синдбад» имеются средства стандартного полнотекстового поиска, которые обеспечивают приемлемое время отклика при поиске данных (в данном случае – не более 2 сек). Однако графовая модель позволяет создавать очень гибкие запросы при поиске неоднородных данных, используя преобразования типов и механизм навигационных выражений, близкий к языку запросов Lorel [2],

Пользователь, как правило, ничего не знает о типах запрашиваемых им данных. Более того, данные, имеющие один и тот же семантический смысл, могут быть не только представлены в БД разными типами, но и изменить свой тип после операции редактирования узла или реструктуризации базы данных. Так, смысловой элемент «Адрес» в различных узлах БД может одновременно иметь реляционный, графовый или базовый (STRING) тип данных, поэтому поисковый механизм должен выполнять предварительное преобразование типов.

Графовая модель данных дает возможность использования навигационных выражений в запросах, например: «Найти всю информацию о текущем объекте, которая есть в базе» (**FIND.CURRENT.ALL**). Результаты поиска могут быть представлены пользователю вместе со своими связями, что позволяет производить навигацию по полученному массиву данных, и облегчает их анализ.

5.2 Организация обмена информацией между СУБД и пользователем.

Поскольку конечный пользователь не знает, какой реально тип имеют редактируемые или просматриваемые им данные, на каком сервере распределенной БД они находятся, необходимые преобразования данных выполняют активные объекты клиентской части. Так, поле «Имя» типа данных **FIO** может иметь вид текстового поля и

редактироваться как обычный текст. В реальности это поле может представлять собой ссылку на справочник имен, т.е. перечисляемый тип данных. В этом случае оно может быть представлено пользователям для редактирования различными способами, например, в виде выпадающего списка.

Поскольку в качестве клиентской части используется веб-броузер, необходимая информация передается СУБД через переменные окружения CGI веб-сервера обычным образом. Основная нагрузка по проверке корректности вводимых данных (контроль заполнения полей, соответствие значения поля его типу) приходится на объекты JavaScript клиентской части. Они же формируют параметры запроса к серверу при вводе, редактировании, поиске информации.

Соответствие полей данных, предоставляемых конечному пользователю, реальным типам данных узлов БД определяется шаблонами (templates). Имена шаблонов, если они не указаны явно, выбираются СУБД самостоятельно, по тегу узла. При занесении новых данных тег указывается объектами JavaScript клиентской части, а при операциях редактирования или поиска считывается из базы данных. Создание и редактирование шаблонов возможно при помощи любого текстового редактора. Специально разработанный язык шаблонов позволяет осуществлять обращения к СУБД, формировать внешний вид выходных документов.

Таким образом, именно шаблоны являются посредниками между конечным пользователем и СУБД, поэтому их функциональная нагрузка весьма высока. С помощью команд шаблонов осуществляется доступ к базе данных для получения необходимой пользователю информации. Примеры команд (префиксы **GET** и **CURRENT** не обязательны):

- **GET.PARENT.ID** - дать ID родительского узла;
- **GET.CURRENT.ADDRESS.REGION.NAME** –дать название улицы поля «Адрес» текущего узла;
- **SET.FIO.Имя (Василий)** –записать значение в поле «Имя» текущего узла типа **PERSON**;
- **GET.LIST.SON (Дата, Описание**) – получить список дочерних узлов, при этом выдать только значения полей «Дата» и «Описание».

Командами шаблонов можно определить алгоритм предварительной обработки данных (произвести фильтрацию данных, статистические расчеты и т.п.) – при этом шаблон играет роль программируемого приложения, например:

- **GET.REPORT (MY_SEARCH.TEM, 2003)** - подготовить отчет за 2003 год по образцу, указанному в шаблоне «MY_SEARCH.TEM»;
- **CALCULATE (DOCUMENT, IF ((Дата == NULL) || (Автор == NULL)) THEN (INC_BUFF (Number_Of_Bad_Documents)))** – посчитать количество узлов типа **DOCUMENT**, у которых не указан либо автор, либо дата создания. Результат подсчета сложить со значением ячейки «Number_Of_Bad_Documents», заведенной ранее в теле СУЬД командой шаблона **DEFINE**;
- **TYPE.BUFF (Number_Of_Bad_Documents)** - выдача пользователю значения переменной «Number_Of_Bad_Documents».

Шаблоны участвуют в оформлении выходного документа, определяя количество и вид представления элементов, их взаимное расположение, возможность и способ навигации по связям между узлами графа. Либо непосредственно, формируя документ «на лету», либо через указание активным объектам клиентской части выполнить это формирование на компьютере клиента, осуществляя при необходимости транспортировку таких объектов пользователю.

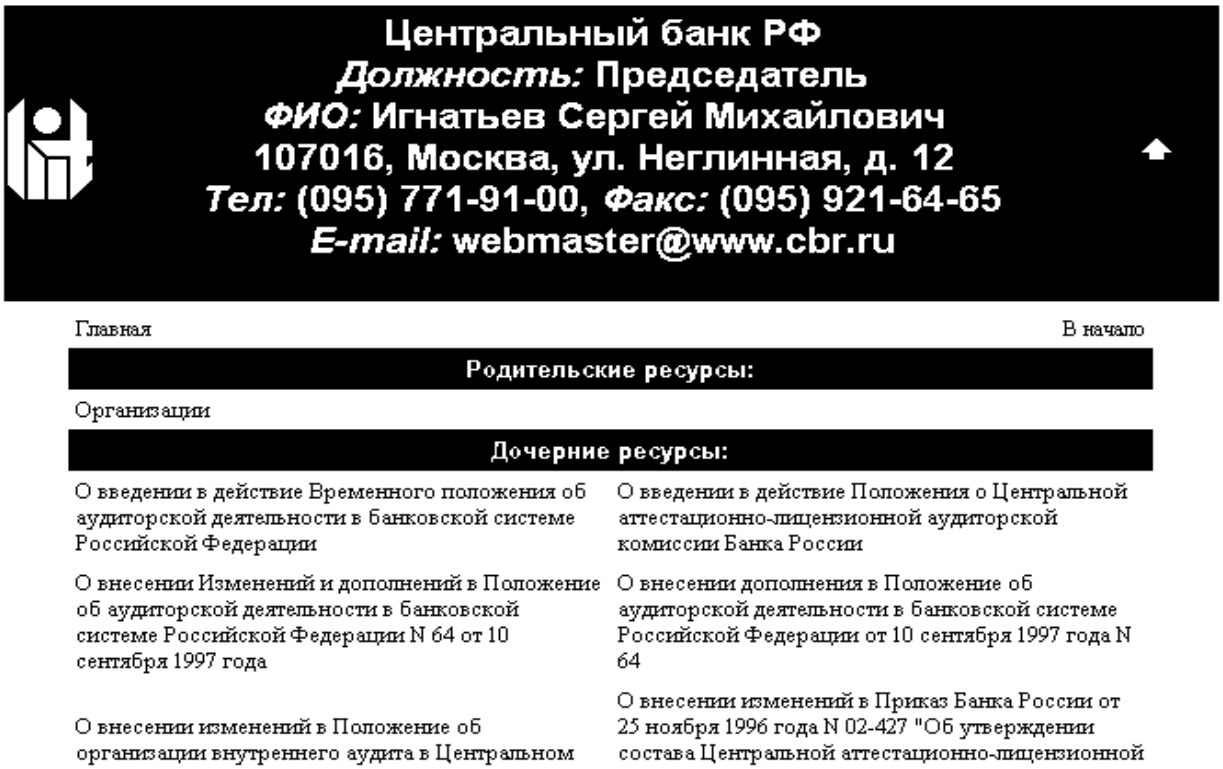
5.3 Редактирование данных

Веб-броузер содержит стандартные встроенные или программируемые инструменты для редактирования текстовой информации на стороне клиента (калькулятор, календарь, выпадающие списки), поэтому редактирование данных не вызывало каких-либо проблем. С точки зрения СУБД добавление или удаление ребер принципиально не отличается от редактирования полей других типов. Однако изменение связей между узлами оказалось весьма трудоемкой операцией (кроме операции удаления ребер). Дело в том, что связываемые узлы могут располагаться на значительном удалении друг от друга, через множество промежуточных ребер. Например, для того, чтобы добавить связь по хронологическому измерению для документа «О решении Государственной межведомственной экспертной комиссии по контрольно-кассовым машинам от 27.02.96», находящимся на шестом уровне тематического классификатора, нужно пройти восемь ребер. Существующий сервис изменения связей между узлами дает возможность лишь получить имя привязываемого узла и отредактировать его как обычное текстовое поле. Доработка технологии изменения связей, исключающей необходимость получения имени узла, должна значительно облегчить процесс ручного редактирования.

5.4 Реструктуризация базы данных

Операции над графом в целом (экспорт, импорт, реструктуризация рубрикация и т.п.) считаются весьма трудоемкими для графовой модели данных. Известно, например, что обход дерева имеет экспоненциальную сложность. Оценить количественно сложность обхода циклического графа вообще вряд ли возможно. Очевидно лишь, что затраты ресурсов при этом будут чрезвычайно

Рис. 5. Представление узла графа сложной структуры



высоки, и практически неприемлемы для графов с количеством узлов порядка нескольких сотен и выше. Эта проблема была устранена разработкой собственного алгоритма обхода графа, имеющего линейную по числу узлов сложность, что позволяет обрабатывать графы с миллионами и даже десятками миллионов узлов на обычном персональном компьютере. Утилиты системного программного обеспечения СУБД «Синдбад» осуществляют обход графа последовательно, узел за узлом, независимо от количества и сложности связей между узлами. Очевидно, что трудоемкость обхода в этом случае оказывается линейной по числу узлов графа. Кроме того, при таком подходе гарантируется доступ даже к тем узлам и группам узлов, которые не имеют связей с остальными узлами графа (в частности, если эти связи повреждены).

Процесс групповой модификации базы данных осуществлялся утилитами пакетной обработки с целью реструктуризация базы данных путем удаления группы узлов БД и последующего импорта нового подмножества данных, а также для оптимизации физической структуры БД («уборка мусора»),

С точки зрения конечного пользователя, операцию удаления узла БД можно понимать в двух смыслах. Операция **безоговорочного** удаления узла вместе со всеми связями доступна лишь привилегированным пользователям уровня администратора базы данных. Гораздо чаще используется **виртуальное** удаление, когда узел остается в базе данных, но становится невидимым для пользователя, причем, только для того окружения, в котором было произведено удаление (по сути, удаляется не узел, а ребро). Рассмотрим действия СУБД при выполнении обеих операций.

Безоговорочно удаляемая группа содержала все дочерние узлы рубрики тематического каталога «Строительство и архитектура» (1651 узел). Для поддержания ссылочной целостности требуется, чтобы при ликвидации любого элемента были удалены и все пути, ведущие к нему (в данном случае необходимо удалить не только 1651 узел, но и 10853 ребра из 235 смежных узлов). Однако, при изменении данных сразу в группе узлов (расположенных, возможно, на нескольких серверах), обеспечение ссылочной целостности потребует неоправданно большого расхода системных ресурсов, и будет служить местом потенциальных ошибок синхронизации. Поэтому подлежащие удалению узлы реально из базы данных не удалялись, их «удаление» осуществлялось установкой в теге элемента флага «удален», который интерпретируется СУБД как указание скрыть такие узлы от пользовательских приложений. Физически узлы будут удалены из БД вместе со всеми своими связями во время операции «уборка мусора», и до этого момента их можно восстановить, сбросив флаг «удален».

Виртуально удаляемая группа узлов содержала все узлы, являющиеся дочерними к узлу

хронологического измерения «1999 год». Всего таких узлов в базе оказалось 1960 (еще 23 узла этой группы были удалены во время операции безоговорочного удаления). При этом 65 узлов не имели связей с другими узлами (например, узел «О дополнении к факсу от 05.07.99 N 1914») и, после удалении ребра на узел «1999 год», стали вообще недоступны навигационным путем. Остальные 1895 узлов остались видимыми. Так, узел «О критериях отбора предприятий, подлежащих приватизации по индивидуальным проектам в 1998-1999 годах (с изменениями на 19 сентября 1997 года)» утратил ребро на узел «1999 год», но остался по-прежнему доступен через узлы «1997 год» и «1998 год».

Импортируемое подмножество данных состояло всего из 15 элементов типа **STRING**, **ADDRESS**, **PERSON** и **COMPANY**. Это подмножество было получено из внешних источников для организаций «Центральный банк», «Министерство юстиции» и «Федеральная налоговая служба». Реструктуризация базы данных проводилась в четыре этапа. На первом этапе данные БД были переведены в архивный формат, с сохранением только однонаправленных связей по каждому измерению графа, при этом элементы, имеющие флаг «удален» (безоговорочно удаленные) и связи на эти элементы из других узлов не сохранялись. Затем к полученному архиву добавили импортируемые данные в том же формате. На третьем этапе была проведена рубрикация данных для привязки элементов БД к импортируемым организациям. Наконец, конвертер из архивного формата обеспечил восстановление двунаправленных связей. При конвертации в архивный формат и обратно была автоматически выполнена операция «уборка мусора». Суммарное время всех четырех операций составило две минуты, практически не создавая помех для работы пользователей. После модификации базы данных вновь стали видимы 6 виртуально удаленных элементов, поскольку они получили новые связи. Так, стал доступен узел «О перечне заболеваний, связанных с выполнением работ по ликвидации последствий аварии на ЧАЭС (Минюст N 1877 31.08.99)», поскольку он имеет связь уже не на «1999 год», а на узел «Министерство юстиции». Фрагмент новой БД, содержащей разнородные узлы, показан на рис. 5.

6 Выводы

- Технология автоматической рубрикации данных может быть применена для первичной структуризации слабоструктурированных данных, характерных для электронных библиотек;
- Использование сетевой модели данных позволяет работать с неоднородными данными произвольной структуры и обеспечивает возможность доступа к данным навигационным путем;
- Реструктуризация БД в процессе эксплуатации может производиться многократно и незаметно для конечных пользователей.

7 Заключение

Предложенные принципы работы с графовой моделью данных могут оказаться полезными при разработке программного обеспечения для электронных библиотек.

Литература

[1] CODASYL DBTG Report. – New York: ACM, 1969.

[2] Sergey Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom and Janet L. Weiner. The Lorel Query Language. In Journal of Digital Libraries, volume 1:1, 1997.

[3] Sergey Abiteboul, Dallan Quass, Roy GoldMan, Jason McHugh and Jennifer Widom. Lore: A DBMS for Semistructured Data. Technical report, Stanford University, 1997.

[4] М.Р. Когаловский. «Энциклопедия технологий баз данных». М. Финансы и статистика, 2002.

[5] В.В. Рыбинкин. «Система рубрикации данных Синдбад». Второй российский семинар по оценке методов информационного поиска, 2004. http://www.romip.narod.ru/romip2004/index.html

[6] Электронная библиотека нормативно-текстовых документов компании «Кодекс» http://www.kodekc.ru

[7] Фрагмент статической версии базы данных нормативно-текстовых документов http://www.2bit.ru/KODEKC/

Realization of some methods of processing of poorly structured data in digital libraries

Vladimir Rybinkin

Bureau of Internet Technoligies BIT Ltd

ryb@2bit.ru

This paper is devoted to the analysis of the problems arising at work of the user with poorly structured information. The basic concepts of formation of structure of graph database are offered, the primary structurization of non-uniform data of complex structure is executed, the organization of access to data without user's query is considered. The scheme of data during test operation of electronic library is changed a few times. The stated approaches are practically realized with the graph DBMS «Sindbad».